ARRAYCOMM	**i-BURST Modem Driver Writer's Guide**	

Originator: Middleton/Merrill	**Create Date: 2/2/2001**	**Control Dept.: Engineering**

i-BURST

Modem Driver Writer's Guide

Version 2.1

Printed: June 8, 2006

1. Overview

This document attempts to provide the essential information necessary to write a device driver to allow a host computer to drive an iBurst modem as designed by the ArrayComm Engineering Department. To the extent that the modem design has been modified by ArrayComm's customers, this document no longer applies. This document is publicly released in the hope that it will provide useful reference material. In no event is ArrayComm liable for any consequences as a result of the use of this information. ArrayComm makes no guarantee of as to correctness of this information and offers no promise of support.

2. Modem Interface/Architecture

This document covers two generations of iBurst modem architectures:

- The ut02 hardware/software architecture (based on the asic01 iBurst chip and other chips).
- The ut04 hardware/software architecture (based on the asic02 iBurst chip and other chips).

Each of these architectures supports two possible interfaces (not both at the same time):

- The PCMCIA interface consists of a 4 Kbyte shared memory located in the ASIC and read/writable using byte or larger access from either the modem or from the host; plus an interrupt line from modem to host.
- The USB full speed (12 Mbit/second) interface uses bulk data pipes to transmit data as a sequence of 64 byte USB packets.

Each such interface connects a modem (also referred to as a **UT** or a target) with a host computer (also referred to as a **PC** or a host), and transfers packets over an emulated Ethernet connection. Two Ethernet addresses are thus required. The modem provides the "target end" Ethernet address to the host. Flipping the least bit of the last byte of the Ethernet address forms the “host end” Ethernet address.

The interface also uses out of band control depending on the interface type as explained below.

NOTE: All multi-byte integer data is big-endian (most significant byte first) except where noted.

3. Packet Format

Ethernet-like packets are exchanged between the modem and the host, but to reduce waste of modem bandwidth (and to incorporate additional information), the packets use a form of compression. The source and destination addresses are removed since the source address is always known and the destination address can only be either broadcast or modem Ethernet address. A bit is used to reconstruct whether the destination had broadcast and modem address. The priority bits of 802.1p/Q packets are used to indicate packet by packet priority. There is a standard 6 byte header followed by a payload of 0 to 1500 bytes. The payload and the EthernetType is the same for real Ethernet packets. Like real Ethernet packets, the MTU (maximum payload length) is 1500 bytes.

Packet Composition

Packet Field	Length in Bytes
HeaderWord1	2
SequenceByte	1
Complement	1
EthernetType	2
Payload	0 to 1500 bytes
Padding	(See text)

3.1. HeaderWord1 (2 bytes)

The packet begins with the two-byte big-endian HeaderWord1. The bit definitions are:

HeaderWord1

Bit Field Name	Bit Position
Extension	15
Priority	14:12
Broadcast	11
PacketLength	10:0

- If Extension is set, the remainder of the packet is determined by additional header bytes; no such header bytes are currently defined, this is for future extension. Any packets with this bit set should be thrown away.
- The Priority bits are a direct mapping of the 8021.p/Q scheme with 3 bits providing class or quality of service. The 12 VLAN bits and format control bit are dropped. Set the priority bits to zero unless there is a clear reason to set otherwise.
- The Broadcast bit compresses the first twelve bytes of the Ethernet packet (the source and destination addresses) into a single bit, and must be reconstructed by the receiver. For this compression, the source address is always that of the sender, and the destination address is reconstructed according to the value of bit Broadcast: if this bit is set, the destination is the broadcast address, else the destination is the assigned Ethernet address of the receiver.

- The PacketLength field contains the actual size in bytes of the entire packet, including the header, but excluding any padding. Therefore it must be at least 6, the length of the header itself, and must not exceed 1506, the length of the header (6) plus the maximum payload data size (1500).
- Pad bytes are allowed (and in some cases required) at the end of the packet. The amount of padding allowed or required depends upon the interface type but in any case must not exceed 64 bytes.

3.2. SequenceByte

The SequenceByte is the low eight bits of the sequence number of this packet. Sequence numbers are distinct for the two directions. Sequence numbers begin with any integer and increment by 1 for every packet sent in the particular direction.

3.3. Complement

The Complement must be the 1's complement of the second byte of the header, i.e. the low 8 bits of the PacketLength. Received packets for which this is not true should be discarded. This is intended for detection of corrupted packets.

3.4. Ethernet Packet Type (2 bytes)

Ethernet Packet Type is the same as the packet type field in the standard Ethernet packet header. It is a two-byte big-endian integer value. In addition, special packets have one of the following values for the Ethernet packet type:

Special EthernetPacketType Values

Value Name	Value (in hex)
ControlStatus	AC00
Loopback	AC02
Configuration	AC03

3.5. Payload

The payload depends upon the EthernetPacketType value.

3.5.1. Network Data Packet

If the packet is used for transmitting user data, then the payload of a packet will be a PPPoE packet (i.e. Packet Type of 0x8863 and 0x8864) for direct hand-off to the iBurst protocol stack (on the modem end) or the PPPoE software on the host end. Other standard Ethernet protocols such as ARP and IP may be supported depending on the modem firmware version and configuration, for example to serve status web pages.

3.5.2. ControlStatus Packet

The payload of a packet for control and status purposes (i.e. Packet Type of 0xAC00) will contain fields of Nbytes, ControlStatus, and Payload. As always, all integers are big-endian.

Header Field	Length in bytes
Nbytes	4
ControlStatus Type	4
Control Payload	N

- Nbytes is the number of bytes in the ControlStatus packet including the Nbytes field
- ControlStatus is the specific request or response being made.
- Control Payload is the optional field containing data specific to a particular control or status transaction (described later in this document).

3.5.3. Loopback Packet

Packet type 0xAC02 (Loopback) is used for transmitting arbitrary chunks (0 to MTU bytes) of arbitrary data between the modem and host and back. The loopback packet is useful for development and manufacturing support only. The modem may, if so configured, send back a loopback packet with the same payload for every loopback packet received.

3.5.4. Configuration Packet

Packet type 0xAC03 (Configuration) is used for transmitting configuration data between modem and host. This is required once at the beginning of a connection for USB, and is contained within the shared memory for PCMCIA. The payload is the 2-byte configuration data as defined below.

4. Configuration Data Format

Configuration data must be sent by the host to the modem at the beginning of each connection session for the USB interface; the same data appears in the shared memory for the PCMCIA interface.

Configuration data consists of two bytes as follows:

Configuration Data Composition

Packet Field	Length in Bytes
SequenceByte	1
ConfigurationByte	1

The SequenceByte is the low 8 bits of the sequence number of the most recently queued packet, where packet sequence numbers may start at any integer and increment by one for each packet queued.

The ConfigurationByte should be set to the value 2 for networking only operation or to the value 6 if Control/Status packets (see below) will also be used.

5. ControlStatus Packet Format

Control and status packets may be optionally enabled (see Configuration Data) so that the modem may provide information to host and to allow firmware flashing. The following describes the payload of a packet with the ControlStatus ethernet packet type (see above).

5.1. RequestStatus

RequestStatus is sent by the host to the modem to request that the modem perform certain immediate actions.

ControlStatus type: 0x01

Payload containing one of the following request types:

- 0x01: Status1Request (4 bytes)
  - Requests the modem to send status1 now
- 0x02: Status2Request (4 bytes)
  - Requests the modem to send status2 now
- 0x03: Status3Request (4 bytes)
  - Requests the modem to send status3 now

Sample RequestStatus

Nbytes	RequestStatus	Status1Request
12	0x01	0x01

5.2. SetStatus

SetParameter is sent by the host to the modem to request that the modem set a parameter (which may be lost on reboot).

ControlStatus type: 0x02

Payload containing one of the following:

- 0x01: SetStatusInterval (4 bytes)
  - Controls the interval by which the modem automatically sends dynamic status.
  - Fields
    - Status type (4 bytes)
      - 0x01 – Status1
      - 0x02 – Status2
      - 0x03 – Status3
    - NMSec – send status every NMSec msecs; 0 turns off (4 bytes)
      - For Status1 maximum interval is 1hour.

Sample SetStatus

Nbytes	SetStatus	Set Status Interval	Status Type	Nmsec
20	0x02	0x01	0x01	0x64

5.3. Status1

Status1 is used by the modem to send status information to the host. This is intended for communicating periodic and dynamically changing data, which will be sent to the host on a regular basis.

ControlStatus type: 0x05

Payload containing all of the following fields, ALL LITTLE-ENDIAN:

- Signal Strength (4 bytes)
  - This value is based on the current BCH or TCH if stream(s) are active.
  - -1 specifies Invalid data; Range is 0 – 100 where 0 is no signal and 100 is modem's signal limit.
- Uplink Bytes (4 bytes) – Number of bytes Sent. Uplink bit rate should be calculated from Uplink Bytes and TimeInMSec.
- Downlink Bytes (4 bytes) – Number of bytes received. Downlink bit rate should be calculated from Downlink Bytes and TimeInMSec.
- TimeInMSecs (4 bytes) – The amount of time (in milli sec's) Uplink Bytes and Downlink Bytes are captured.
- Cumulative Uplink Bytes (4 bytes) – Cumulative number of bytes sent from Modem.
- Cumulative Downlink Bytes (4 bytes) – Cumulative number of bytes received by Modem.
- Status Flag (4 bytes) – Each bit represents different status information, right now only one bit is used but it will be extended in future depending on the requirements.
  - Bit 0 – 0: No BS detected; 1: BSs detected
- Status Valid (4 bytes) – Each bit indicates whether the corresponding bit in Status Flag is valid or not. Status Flag value should be taken into consideration only if the valid bit is set.
  - 1 – Valid data; 0 – Invalid data
- Number of TCH Bursts Successfully Received (4 bytes)
  - The number of TCH bursts across all modClasses successfully received since the last report.
  - Used to calculate FER
- Number of TCH Bursts Attempted Received (4 bytes)
  - The number of TCH bursts across all modClasses attempted to be received since the last report.
  - Used to calculate FER

    - Calculation: FER = 100*(1 – (# Success / # Attempted))%
- Signal to Interference plus Noise Ratio – SINR (4 bytes)
    - This value is based on the recently received standard downlink bursts of CM, AA, and TCH. It does not apply to bursts for which SINR is not available including F, T, B, and PCH.
    - This value is provided as a signed value which has been left shifted 4 (X.4). For example, a SINR value of 2.5 will be reported as 40.
- Base station color code – BSCC (4 bytes)
    - This value reflects current base station color code with which the modem is uplink aligned (i.e. CR/CM exchange complete, but not registered).
    - It contains the 6 bit BSCC allowing valid values from value 0 to 63. A value of 255 represents an invalid BSCC indicating the modem isn't uplink aligned with any base station.
- Base station identity (8 bytes)
    - It contains 48 bits of base station ID. A value of –1 represents an invalid base station ID indicating the modem isn't registered with a base station. But, the modem could be uplink aligned and so have a valid BSCC field.
    - The first 4 bytes are LSW (least significant word). The last 4 bytes are MSW (most significant word).

5.4. Status2

Status2 is used by the modem to send information to the host related to battery and temperature. The measurements will depend on hardware support. The measurements will be uncompensated with the assumption that the host side will perform any calculations necessary on the raw values.
ControlStatus type: 0x06
Payload containing all of the following fields, ALL LITTLE-ENDIAN:

- Battery temperature (4 bytes)
    - Is reported in Kelvin; -1 is invalid data.
- Battery voltage (4 byte)
    - Is reported in milli volts; -1 is invalid data.
- Battery current draw (4 bytes)
    - Is reported in milliamps; -1 is invalid data.
- modem Operating Temperature (4 bytes)
    - Is reported in Kelvin; -1 is invalid data.
- Interface current draw (4 bytes)
    - Is reported in milliamps; -1 is invalid data.
- Battery charge percentage (4 bytes)
    - Range is 0 – 100; -1 is invalid data.
- Status Flag Bits (4 bytes) - Each bit represents different status information. Right now only few bits are used but it will be extended in future depending on the requirements.

    - Bit 0 – 0: USB voltage is not present; 1- USB voltage is present.
    - Bit 1 – 0: No fault in Charger; 1 – Charger fault.
    - Bit 2 – 0: Battery is OK; 1 – Battery Dead.
    - Bit 3 – 0: Reset switch is not pushed; 1 – Reset switch pushed
    - Bit 4 – 0: External 12VDC is not present; 1 – External 12VDC present.
    - Bit 5 – 0: No Fan Failure; 1- Fan Failure.
    - Bit 6 – 0: On Off switch not pressed; 1 – On Off switch pressed.
- Status Flag Valid (4 bytes) - Each bit indicates whether the corresponding bit in Status Flag is valid or not. Status Flag value should be taken into consideration only if the valid bit is set.
    - 1 – Valid data; 0 – Invalid data

5.5. Status3

Status3 is used by the modem to send information to the host which is relatively static in nature.
ControlStatus type: 0x07
Payload containing all of the following fields (all integers LITTLE-ENDIAN):

- Software release name or version (12 bytes)
    - Current running image release name, such as 0901A008 for trial releases or 1002 for commercial releases.
- Protocol version (12 bytes)
    - iBurst protocol version
- Hardware version (12 bytes)
    - HwModel in upper 4 bytes. This field is composed of RadioModel and RadioRev from flash config in lower 16 bits. The upper 16 bits are currently unused and set to 0.
    - HwId in lower 8 bytes. This field is directly from flash config.
- Ethernet address (6 bytes)
- Device name (80 bytes)
    - Length includes a null byte for termination with null padding at the end
    - e.g. “Product name xyz with iBurst technology\0\0\0\0…”
- Flash Boot release name (12 bytes)
    - Flash boot image release name
- Flash App0 release name (12 bytes)
    - Flash application0 image release name
- Flash App1 release name (12 bytes)
    - Flash application1 image release name

6. Control Packet Usage

6.1. Status Reporting

All status APIs are unacknowledged and neither side has any guarantee that the other side has received the packet successfully.

- The RequestStatus API can be used by the host to immediately request a status report from the modem.
- The Status1, Status2, Status3 APIs are sent to the host either based on a RequestStatus or when the StatusInterval has expired if that field is non-zero value. The default values are defined in modem software which reports Status1 every 1 sec, Status2 every 2 seconds, Status3 every 3 seconds.
- The SetParameter API allows the host to set the StatusInterval by which the modem sends the StatusX APIs. In the future, the power consumption setting of the modem can be controlled via this API as well.

7. General Host Driver Responsibilities

7.1. Header Compression

The driver must convert between full Ethernet format and the compressed format. When header compression is used, the compressed header indicates true length of the packet, but padding may be present at the end depending upon the link type. The padding of received packets must be removed before passing them on. The driver must generate and fill in sequence numbers for compressed headers of transmitted packets.

7.2. Configuration Mechanism

USB drivers must send a configuration packet at the beginning of each connection session. For PCMCIA, this information is installed in the shared memory header.

7.3. Flow Control

Like any Ethernet type device, flow control is not required or even entirely possible. However, it is probably a good idea if the driver avoids excessive queueing of packets to be transmitted, either by using available back pressure mechanisms if any, or by throwing packets away..

8. Hardware Specific Interfaces

8.1. USB Interface

The modem is compatible with USB 1.1 and 2.0 full speed (12 Mbit/second) standards. The modem provides the standard control end points plus IN and OUT bulk data endpoints using 64 byte USB packet size.

The end of transfer of an Ethernet or special packet is marked by a short (less than 64 byte) USB packet. Should the transfer be an exact multiple of 64 bytes, an additional USB packet with at least one byte of padding must be added, to avoid problems with zero-length USB packets. In general, up to 4 bytes of padding may be added to any packet, so long as zero-length USB packets do not result.

Only one USB configuration (in the USB sense of the word) is defined.

The following endpoints are used for ASIC01/UT02:

- Endpoint 0 used in the standard way. The endpoint size is 8.
- Endpoint 1 is the bulk OUT endpoint. The endpoint size is 64.
- Endpoint 2 is the bulk IN endpoint. The endpoint size is 64.

The following endpoints are used for ASIC02/UT04:

- Endpoint 0 used in the standard way. The endpoint size is 64..
- Endpoint 2 is the bulk IN endpoint. The endpoint size is 64.
- Endpoint 3 is the bulk OUT endpoint. The endpoint size is 64.

The modem replies to the minimum required standard configuration (SETUP) requests.
The particular values, including Vendor ID and Model Number depend upon the modem.
The driver may need to be updated to reflect supported vendor ids and model numbers as new modem models are added. Ideally the driver will allow these parameters to be easily configured when the driver is loaded or the operating system brought up. The ArrayComm vendor ID is 3348 decimal, and the Model Number used by ArrayComm is 9 (for all models that follow this interface). These values may be used by other vendors who are so licensed by ArrayComm.

A single vendor-specific setup transaction (code 99) must be used at host driver connect time to obtain the Ethernet addresses to use. The 8 bytes of reply data will contain:

- The first byte will be the total length of the reply data (8).
- The second byte will be 99 for ut02/asic01 and 77 for ut04/asic02.
- The 6-byte SerialNumber doubles as the Ethernet (MAC) address of the modem end of the simulated Ethernet. The host end of the simulated Ethernet has an implied Ethernet address formed by flipping the LSB of the last byte.

The host sends at the beginning of connection (redundantly is harmless) a configuration packet as described above.

8.2. PCMCIA Interface

The PCMCIA hardware implements a shared memory along with the ability for the modem to interrupt the host. The shared memory is 4 Kbytes addressable by byte or 16-bit word (according to the PCMCIA standard). A second 4 Kbytes of address space is required for a memory-mapped control register, depending upon the modem architecture. The layout of the shared memory is determined by software as follows and is essentially the same for all models; the model information is only important for interrupt control. Following is a typical card information structure:

8.2.1. PCMCIA Card Information Structure (CIS)

Tuple	Tuple Data/Sub-fields
CISTPL_DEVICE	Common Memory device is 8kb static RAM interface @ 100ns access time.
CISTPL_VERS_1	Compliant with V8.0 of PCMCIA Specification Manufacturer name: “ArrayComm, Inc.” Product name string: “iBurst UT PCMCIA B”
CISTPL_DEVICE_A	Attribute Memory device is 512 b static RAM interface @ 100ns
CISTPL_MANFID	Arraycomm Manufacturer ID: 0x2e3 Manufacturer Information (product revision): “B”
CISTPL_FUNCID	Card function ID = 6: Network Adapter
CISTPL_FUNCE	A 6-byte LAN ‘Node Identification’ (LAN_NID) field is defined to store the modem Ethernet address.
CISTPL_CONFIG	A single Configuration Entry is defined; Configuration Register base address is 0x200 Four Configuration Registers are present:
CISTPL_CFTABLE_ENTRY	The Configuration Entry defines: • Function requires Memory+I/O mode.

	• 5V is required • Any interrupt may be used. • 8 kb of Common Memory defined, accessed from 0 using any host address The card configuration option register should be set to: • ASIC01: 0xC5 • ASIC02: 0xC7
CISTPL_NO_LINK	End of tuple data.

Table 1 PCMCIA CIS definitions

8.2.2. PCMCIA Memory Mapping and Control

For both ut02 and ut04 the driver must use a 8Kbyte window of shared address space with the modem. The first 4 Kbytes is occupied by the shared memory, and the memory-mapped control register occupies the first 16-bit location in the second 4Kbytes.

The use of the shared memory is discussed below.

Only one use should be made of the memory mapped control register. A value of 1 should be written using a 16 bit word write in order to acknowledge a pending interrupt from the modem to the host.

8.2.3. PCMCIA Ethernet Address

The permanent Ethernet address is stored in the CIS, as part of the Network CISTPL_FUNCE tuple data. The session Ethernet address may be changed at any memory protocol reset (see below). If desired, the CIS Ethernet address may be ignored, keeping in mind that the Ethernet address in shared memory is not valid until a protocol reset had finished.

8.2.4. PCMCIA Common Memory Interface

The Common Memory is logically broken into a sequence of 32-byte chunks, numbered from zero. The first chunk (chunk 0) is always the header (defined below). The subsequent chunks are allocated by the header (or may be unallocated) to define two ring buffers: one for modem-to-host and one for host-to-modem data transfer.

In addition to chunk numbers, ring buffer indices are used for operations within the ring buffers. Ring buffer indices are a chunk number relative to the start of the ring buffer, but do not wrap back to zero until twice the number of chunks in the ring buffer is reached. Thus, every chunk in the ring buffer can be described by two different ring index values.

8.2.5. PCMCIA Common Memory Header

The 32-byte header at the beginning of common memory contains the following fields (each of 1-byte size):

Field Name	Offset (hex)	Description	Owner
UT Magic1	00	Zero or Magic Value 1	UT
UT Magci2	01	Zero or Magic Value 2	UT
UT ResetSequence	02	UT reset sequence number	UT
UT ResetFeedback	03	Copy of PCs reset sequence number	UT
UT WriteIndex	04	UT-to-PC Ring buffer write index	UT
UT ReadIndex	05	PC-to-UT Ring buffer ring index	UT
UT PacketSequence	06	Configuration byte 1 (Packet sequence)	UT
UT StatusByte	07	Configuration byte 2 (Status)	UT
PC Magic1	08	Zero or Magic Value 1	PC
PC Magic2	09	Zero or Magic Value 2	PC
PC ResetSequence	0A	PC reset sequence number	PC
PC ResetFeedback	0B	Copy of UT's reset sequence number	PC
PC WriteIndex	0C	PC-to-UT Ring buffer write index	PC
PC ReadIndex	0D	UT-to-PC Ring buffer read index	PC
PC PacketSequence	0E	Configuration byte 1 (Packet sequence)	PC
PC StatusByte	0F	Configuration byte 2 (Status)	PC
UT RingChunkStart	10	Chunk number of first chunk of UT-to-PC ring.	UT
UT RingNChunks	11	Number of chunks in UT-to-PC ring.	UT
PC RingChunkStart	12	Chunk number of first chunk of PC-to-UT ring.	UT
PC RingNChunks	13	Number of chunks in PC-to-UT ring.	UT
Hardware Interface Type	14	Interrupt interface and memory size indicator.	UT
Reserved	15..19	Set to zero.	UT
UT EthernetAddress	1A..AF	6-byte Ethernet address of UT	UT

Fields 0x08 - 0x0f are written only by the host; all other fields of the header are written only by the modem. The modem-to-host ring buffer is written only by the modem. The host-to-modem ring buffer is written only by the host, but only after it's location has been defined by the appropriate header fields in the presence of valid modem magic numbers and matching modem reset feedback.

Magic value 1 is 0xAC. Magic value 2 is 0x02. The appropriate magic number bytes are set to these values only when as described below.

Reset sequence numbers are arbitrary byte values that are advanced to indicate that a reset has occurred. Reset feedback numbers are copies of the peer's reset sequence number. This is described in detail below.

Ring buffer indices have a range from zero to one less than twice the number of chunks in the ring buffer. The chunk addressed by a ring buffer index is the ring buffer index modulo the number of chunks in the ring buffer, plus the start chunk number for the ring buffer. Thus every chunk in a ring buffer is identified by two possible values of index. This redundancy allows distinguishing an empty ring buffer from a full ring buffer, and also provides a sanity check on the validity of the indices, since the difference between the read and write indices (modulo number of indices) must never exceed the number of chunks in the ring buffer.

Configuration bytes (packet sequence byte and status byte) are as defined generically for all hardware types.

Hardware interface types are defined previously according to their affect on interrupt interface. The memory interface of the two types is identical except for the amount of memory; driver code to support both can map memory sufficient for the larger.

8.2.6. PCMCIA Memory Protocol Operations

The following memory protocol operations are defined:

- Reset
- Header corrupt reset
- Peer reset detect, wait and acknowledge
- Transmit all or part of packet
- Receive all or part of packet

The host driver must perform all applicable operations either on a continual polled basis every 4 ms or more; or whenever interrupted by the modem, supplemented by polling every 250 ms or more often.

8.2.7. PCMCIA Memory Protocol Reset Operation

On reboot by either side, that side must perform a reset operation. In addition, a reset operation must be performed by the side that detects a corruption problem. Additionally,

either side may perform a reset at its discretion, although this should not be a common occurrence due to the obvious impact on performance.

A reset operation by the host driver consist of the following steps:

- Set both host magic bytes to zero (set the second one to zero before first one, or both at same time).
- Set both host ring buffer indices to zero (any partially transmitted or received packets must be discarded).
- Set host reset sequence equal to modem reset feedback plus 1 (plus 7 for first reset after reboot), modulo 256.
- Set host reset feedback equal to modem reset sequence.
- Set both host magic bytes to correct values (set the first one first, or set both at same time).
- Wait for peer reset and/or peer reset acknowledgement to finish.

A reset operation by the modem is almost identical except that after clearing the magic numbers, the modem waits one second and then reinitializes all parts of the header not owned by the host, before continuing with the above list of operations. It is assumed that the host driver will not spend longer than 1 second in any transmit or receive operation; this allows the possibility that the modem may redefine the ring buffer layout during reset operation.

8.2.8. PCMCIA Memory Protocol Corruption Detect Operation

Immediately before any transmit or receive operation, the driver must perform header corruption detection and reset detection. Header corruption is inferred if the bytes under control of our side have been modified from the values we last placed in them. Header corruption is also inferred if both sides appear to be reset but, for either ring buffer, the difference between ring buffer indices (modulo the number of indices) is greater than the number of chunks in the ring buffer. In these cases we must perform a reset operation. Do not perform a reset operation in response to reset operation of other side.

8.2.9. PCMCIA Memory Protocol Peer Reset and Acknowledge Detect Operation

Immediately before any transmit or receive operation, the driver (for either side) must perform peer-reset detection. A reset by the other side is detected by finding invalid magic numbers owned by the peer or by observing changes to the sequence and feedback numbers owned by the peer. This also includes the case where we have finished resetting but the peer has not acknowledged it yet (peer reset feedback not equal to our reset sequence byte). Following detect, wait and acknowledge by the host of modem reset, the host must re-read all header fields. The modem may redefine the ring buffer layout and reconfigure the Ethernet address at any reset.

Whenever a peer reset is detected, or failure to acknowledge our reset is detected, our driver must discard partially transmitted or received packets, set its ring buffer indices to zero, and wait until the other side's magic numbers are correct and until their feedback number is equal to our reset sequence number. Then acknowledge the peer reset by setting our feedback number equal to their reset sequence number.

8.2.10. PCMCIA Memory Protocol Receive Operation

A receive operation is performed by the host driver by comparing the <UT WriteIndex> with the <PC ReadIndex>. The difference between the indices (modulo 2 times ring buffer size in chunks) is the number of chunks we may receive. A difference value larger than ring buffer size is cause for reset. If we have room, we may copy the chunks to our local memory and advance the read index (wrapping at 2 times ring buffer size). The chunk number of each chunk is computed as the ring index modulo ring buffer size, plus the start chunk number. Following reset, the first chunk we receive will be the first (perhaps only) chunk of a new packet.

Any sign of corruption in the packet header is cause for reset. Otherwise, the number of chunks is determined from the <PacketLength> field of the header, rounded up to the nearest chunk if not exactly a multiple of chunks. Following each packet (and pad up to chunk boundary) will be another packet. Frequently it is not possible for a packet to fit entirely into the ring buffer, so the driver must maintain a state machine, unloading parts of a packet at a time so that there will be room for the rest of it. In any event, the driver should update the read index so that the modem may write more. On reset or reset acknowledge, any partially received packet must be thrown away as discussed above.

8.2.11. PCMCIA Memory Protocol T ransmit Operation

A transmit operation is performed by the host driver by comparing the <PC WriteIndex> with the <UT ReadIndex>. The difference between the indices (modulo 2 times ring buffer size in chunks) is the number of chunks unread in the ring buffer; the number of writeable chunks is the number of other chunks in the ring buffer. If data is available for transmit, the driver may copy in chunks of data to the available chunks in sequence, making sure to write the <PC WriteIndex> when done.

If a packet is not an even multiple of the chunk size, the driver must pad it to the next multiple of the chunk size. Since the ring buffer may not be large enough to hold an entire packet, the driver must maintain a state machine, loading parts of the packet as room is made. In any event, the driver should update the write index so that the modem may read out more. On reset or reset acknowledge, any partially transmitted packet must be thrown away, as discussed above.

8.2.12. PCMCIA Memory Protocol Configuration Information

The PCMCIA interface does not use an explicit configuration event. The configuration information of both sides is recorded in the header. The driver should update the host side configuration data and make the modem side configuration data available where needed at every interrupt from modem or other polling time and possibly between operations as well. Configuration packets are not expected to be sent by the modem and if received should be ignored.

Appendix ASignal Strength Computation

The signal strength percentage provided in status1 is computed from a calibrated DSSI value. The DSSI is a function of RSSI and SNR and is essentially the desired signal strength in dBm. The iBurst specification provides signal power at which a given burst type should be received by a modem. So, the signal strength percentage used in status1 is principally a mapping of the DSSI to what TCH modClass can theoretically be received. It is not based on what actual modClass or modClasses are actually being received since that would be problematic when we not receiving TCH at all and are only receiving B-burst and CM bursts (i.e. when the modem doesn't have any streams up). The actual mapping of DSSI to signal strength is as follows:

DSSI in dBm	Theoretical Max TCH ModClass	Signal Strength Percentage
<=-109	None (B-burst)	0
-109 < x <= -107	None (B-burst)	5
-107 < x <= -106	0	10
-106 < x <= -104	1	20
-104 < x <= -101	2	30
-101 < x <= -99	3	40
-99 < x <= -97	4	50
-97 < x <= -95	5	60
-95 < x <= -93	6	70
-93 < x <= -91	7	80
-91 < x <= -90	8	90
-89	8	92
-88	8	94
-87	8	96
-86	8	98
>=-85	8	100

The signal strength percentage is based on an average of the last 16 DSSI values received. When there are active streams open, then new DSSI values will be quickly received (every 5ms). But, when no active streams are open, then the DSSI values will be driven much more slowly (every few seconds), so the signal strength percentage will change much more slowly.