PC CARD STANDARD

Volume 4

Metaformat Specification

PCMCIA
JEIDA

PCMCIA (Personal Computer Memory Card International Association)
2635 North First Street, Suite 209
San Jose, CA 95134 USA
+1-408-433-2273
+1-408-433-9558 (Fax)

JEIDA (Japan Electronic Industry Development Association)
Kikai Shinko Kaikan, 3-5-8, Shibakoen
Minato-ku, Tokyo 105, JAPAN
+81-3-3433-1923
+81-3-3433-6350 (Fax)

Document No. 0299-04-2000

First Printing, February 1999

CONTENTS

TABLES

1. INTRODUCTION

1.1 Purpose

This specification provides information necessary for implementing the Card Information Structure (CIS), or Metaformat, on PC Cards and for interpreting the metaformat for the purposes of configuring and utilizing PC Cards.

1.2 Scope

It is recommended that all aspects of a PC Card which can be described in CIS be so described. Full disclosure of a PC Card's characteristics in the CIS is a basic component of compatibility and interchangeability.

1.3 Related Documents

PC Card Standard Release 7.0 (February 1999), PCMCIA /JEIDA
Volume 1. ***Overview and Glossary***
Volume 2. ***Electrical Specification***
Volume 3. ***Physical Specification***
Volume 4. ***Metaformat Specification***
Volume 5. ***Card Services Specification***
Volume 6. ***Socket Services Specification***
Volume 7. ***Media Storage Formats Specification***
Volume 8. ***PC Card ATA Specification***
Volume 9. ***XIP Specification***
Volume 10. ***Guidelines***
Volume 11: ***PC Card Host System Specification***

IEEE 1394-1995 Specification, IEEE

2. OVERVIEW

2.1 Metaformat Overview

Metaformat goals include the ability to handle numerous, somewhat incompatible data-recording formats and data organizations. As is done with networking standards, the Metaformat is a hierarchy of layers. Each layer has a number, which increases as the level of abstraction gets higher. Below the Metaformat is the physical layer, the electrical and physical interface characteristics of PC Cards. (See the ***Physical Specification*** and the ***Electrical Specification***.)

The Metaformat layers are:

1. The Basic Compatibility Layer — specifies a minimal level of card-data organization. Tuples at this level provide fundamental information about the PC Card including supported configurations, manufacturer, and individual device characteristics, such as size, speed, and programming information.
2. The Data Recording Format Layer — includes tuples which describe partitioning information and provide card initialization information.
3. The Data Organization Layer — currently includes a single tuple, CISTPL_ORG, which specifies the partition organization (for example, the file system) in use in a partition described by Data Recording Format Layer tuple(s).
4. The System-Specific Layer — includes the special purpose tuple, CISTPL_SPCL, and the range of vendor-unique tuple codes. The special purpose tuple provides a mechanism for documenting the format and interpretation of special tuple usage within the PC Card Standard. The format and interpretation of any tuple in the vendor-unique range is not documented within the Standard.

2.2 Metaformat Requirements

The PC Card Standard has the following Card Information Structure (CIS) requirements:

- All PC Cards shall have a CIS that describes the functionality and characteristics of the card.
- The CIS of a 16-bit PC Card shall be readable whenever the card is powered, the card is asserting **READY** and the card has been reset by the host after power-up in accordance with the PC Card Standard. This includes after the PC Card is configured and when the PwrDwn bit is set in the Card Configuration and Status Register. (See the ***Electrical Specification***.)
- The CIS of a CardBus PC Card shall be readable whenever the card is powered and the power up process has been completed. (See the ***Electrical Specification***.)
- All linear memory PC Cards shall describe how they are partitioned, even if the entire PC Card is used as a single partition. (See the ***Media Storage Formats Specification***.)
- All ATA PC Cards shall be formatted with a Master Boot Record (MBR) in the first physical sector of the media. The MBR shall contain a partition table describing how the media is partitioned. (See the ***Media Storage Formats Specification***.)

2.3 Metaformat Architecture

2.3.1 Basic Tuple Format and Tuple Chain Structure

The Card Information Structure is one or more chains (or linked lists) of data blocks or tuples. Longlink and linktarget tuples are used to connect chains (See ***3.1 Control Tuples***.) All tuples have the format shown below.

Table 2-1 Basic Tuple Format

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE Tuple code: CISTPL_xxx.							
1	TPL_LINK Offset to next tuple in chain. This is the number of bytes in the tuple body. (n)							
2··(n + 2)	The tuple body. (n bytes)							

Byte 0 of each tuple contains a tuple code. A tuple code of FFH is a special mark indicating that there are no more tuples in the chain. Byte 1 of each tuple contains a link to the next tuple in the chain. If the link field is zero, then the tuple body is empty. If the link field of a 16-bit PC Card contains FFH, then this tuple is the last tuple in its chain.

There are two ways of marking the end of a tuple chain for 16-bit PC Cards: a tuple code of FFH, or a tuple link of FFH. There is only one way of marking the end of the tuple chain for CardBus PC Cards: A tuple code of FFH. (See also ***2.3.8 16-bit PC Card Tuple Chain Processing*** and ***2.3.9 CardBus PC Card Tuple Chain Processing***.)

The use of an FFH link value is allowed in 16-bit PC Cards for backward compatibility, but it is recommended to use the End of Chain tuple. System software must use the link field to validate tuples. No 16-bit PC Card tuple can be longer than 257 bytes: 1 byte *TPL_CODE* + 1 byte *TPL_LINK* + FFH byte tuple body (and this 257 byte tuple ends the chain). No CardBus PC Card tuple can be longer than 256 bytes: 1 byte *TPL_CODE* + 1 byte *TPL_LINK* + FEH byte tuple body. Some tuples provide a termination or stop byte that marks the end of the tuple. In this case, the tuple can effectively be shorter than the value implied by its link field. However, software must not scan beyond the implied length of the tuple, even if a termination byte has not been seen.

2.3.2 Byte Order Within Tuples

Within tuples, all multi-byte numeric data shall be recorded in little-endian order. That is, the least-significant byte of a data item shall be stored in the first byte of a given field.

Within tuples, all character data shall be stored in the natural order. That is, the first character of the field shall be stored in the first byte of the field. Fixed-length character fields shall be padded with null characters, if necessary.

2.3.3 Byte Order on Wide Cards

If a card has a data path wider than 8-bits, one must assign a byte order to the data path. This applies to all CardBus PC Card CIS's and to those fields within a 16-bit PC Card CIS that are recorded in Common Memory space. At present, Attribute Memory Space is byte-wide only; only the even bytes are present. This standard requires that the low-order byte of word 0 be used to record byte 0 of the CIS. Ascending bytes of each word shall be used to sequentially record bytes from the CIS. When the first word is filled, the same process shall be repeated on subsequent words until the entire CIS is recorded. On x86 architecture machines, this byte order is equivalent to the native order; other machines may need to reorder the bytes when reading or writing the CIS.

The basic compatibility layer does not impose any particular byte order on non-header portions of the card. However, some data-format layers impose further requirements.

2.3.4 16-bit PC Card Metaformat in Attribute Memory Space

16-bit PC Cards have two address spaces: Attribute Memory space and Common Memory space. The electrical specification for 16-bit PC Cards requires that information be placed only in even-byte addresses of Attribute Memory space. The contents of odd-byte addresses of Attribute Memory space are not defined.

For simplicity, this specification describes the tuples of the Metaformat as if the bytes of each tuple were recorded consecutively. When a tuple is recorded in Common Memory space of a 16-bit PC Card, the bytes will indeed be recorded consecutively. However, when a tuple is recorded in Attribute Memory space, the data will be recorded in even bytes only.

Link fields of tuples stored in Attribute Memory space are handled as follows. If only the even bytes are read, and the tuples are packed into consecutive bytes in system memory, the link fields shall be set appropriately for byte addressing. This means that the link-field values are conceptually the same whether a tuple resides in Common or in Attribute memory. However, this does mean that if Attribute Memory is directly addressed, the offset to the next tuple in Attribute Memory is two times the link field.

2.3.5 16-bit PC Card Metaformat in Common Memory Space

For cost reasons, many 16-bit PC Cards, such as ROM cards, will not implement a separate Attribute Memory space. On these cards, regardless of the state of the **REG#** line, memory cycles always access Common Memory. These cards provide an Attribute Memory-style CIS starting at byte zero of the card, and recorded in even bytes only. If, for space reasons, the manufacturer wants to switch to a Common Memory-style CIS (packed into ascending bytes), a long link to Common Memory shall be embedded in the CIS. The Common Memory CIS may be stored immediately following the Attribute Memory CIS.

It is important to distinguish between Attribute Memory *space* and Attribute Memory. All 16-bit PC Cards will have Attribute Memory space, accessed by asserting the **REG#** pin. In addition, some 16-bit PC Cards will have a distinct physical Attribute Memory. In this case, the contents of location 0 in Attribute Memory space will be different and distinct from the contents of location 0 in Common Memory space. However, some 16-bit PC Cards will not have Attribute Memory distinct from Common Memory. Here, memory-read operations from a given location in Attribute Memory space will return the same data as read operations from the same location in Common Memory space. Data accessed from Attribute Memory space must be stored in the even bytes only, even if Attribute Memory is not distinct from Common Memory. Regardless of the presence or absence of Attribute Memory, the CIS for 16-bit PC Cards always begins at location 0 of Attribute Memory space.

This standard allows attribute information to be stored both in Attribute Memory and Common-Memory space. Tuples stored in Common Memory space are recorded in sequential bytes. Both the card's even and the odd bytes are used to record data.

Note: The use of odd bytes to represent tuple data is controlled by the logical-address space in which the tuple resides, not by the type of memory actually used to record the tuple. If the tuple is intended to be accessed in Attribute Memory space, it must be stored only in the even bytes. If it is intended to be accessed in Common Memory space, it must be stored in both even and odd bytes following a longlink target.

2.3.6 16-bit PC Card Metaformat for Multiple Function Cards

Multiple function 16-bit PC Cards shall contain multiple Card Information Structures (CIS). The first or global CIS on a 16-bit PC Card shall identify the card as containing multiple functions by the presence of a *CISTPL_LONGLINK_MFC* tuple. The 16-bit PC Card shall also contain a separate function-specific CIS for each set of Configuration Registers on the card.

The starting location of each function-specific CIS is given in a single CISTPL_LONGLINK_MFC tuple in the global CIS. Each function-specific CIS begins with a CISTPL_LINKTARGET tuple.

The global CIS on a multiple function 16-bit PC Card shall contain the following tuples.

Note: A CISTPL_FUNCID with a TPLFID_FUNCTION field reset to zero (0) shall not be placed in the CIS of a Multiple Function PC Card. This tuple is reserved for vendor-specific multiple function PC Cards that do not follow the multiple function PC Card definitions in this specification.

Table 2-2 Global CIS for Multiple Function PC Cards

Tuple	Code	Presence
CISTPL_DEVICE	01H	Mandatory (if PC Card has 5 volt key)
CISTPL_EXTDEVICE	09H	Mandatory (if PC Card has > 64 MBytes of common memory)
CISTPL_DEVICE_OC	1CH	Recommended
CISTPL_LONGLINK_MFC	06H	Mandatory
CISTPL_VERS_1	15H	Mandatory
CISTPL_MANFID	20H	Mandatory
CISTPL_END	FFH	Mandatory

There is a function-specific Card Information Structure for each function on a Multiple Function PC Card. The following tuples are contained in each function-specific CIS.

Table 2-3 Function-specific CIS for Multiple Function PC Cards

Tuple	Code	Presence
CISTPL_LINKTARGET	13H	Mandatory
CISTPL_CONFIG	1AH	Mandatory
CISTPL_CFTABLE_ENTRY	1BH	Mandatory
CISTPL_FUNCID	21H	Recommended
CISTPL_FUNCE	22H	Recommended
CISTPL_END	FFH	Mandatory

2.3.7 CardBus PC Card Metaformat

There is one CIS per card function, which may consist of multiple tuple chains in different spaces. Tuple chains may be located in any of the card space with the exception of I/O space. All tuple chains must start with a CISTPL_LINKTARGET tuple aligned on a four word boundary, but subsequent tuples within a chain need not be so aligned. The beginning of the function's CIS is pointed to by the **CIS Pointer** in the function's configuration space header. If the function's CIS does not complete in the current chain, the location of the next tuple chain is indicated by a CISTPL_LONGLINK_CB. Tuple chains can be located in the following card spaces:

- **Configuration Space**. Tuple chains in configuration space may only be placed in the device dependent region.
- **Memory Space**. Tuple chains may be located anywhere within memory space.
- **Expansion ROM**. Tuple chains may appear in any of the images in an expansion ROM, but no single chain shall span multiple images. The formats of the CIS Pointer and the *Address-Space-Offset* field of the CISTPL_LONGLINK_CB allow specifying a twenty-eight bit offset from the base of any of the first sixteen images in the expansion ROM. The chain need not be located in the image on which the offset is based, allowing chains to be placed in images beyond the first sixteen.

Multi-function CardBus PC Cards have an independent configuration space and CIS for each function. Requiring a CIS for each function allows each function to be generically described and helps in delivering such things as function specific executables in the expansion ROM associated with the function. This also allows functions to be individually manufactured, independent of their eventual placement on a CardBus PC Card.

2.3.8 16-bit PC Card Tuple Chain Processing

The information block must be located such that it can be easily found by low-level software. This Standard requires that the primary CIS of a 16-bit PC Card be recorded in Attribute Memory starting at address zero (00H).

The first tuple in the primary CIS chain of a 16-bit PC Card with a 5 volt key must be either a CISTPL_DEVICE (tuple code 01H), a CISTPL_NULL (tuple code 00H), or an CISTPL_END (tuple code FFH, see ***3.1.2 CISTPL_END: The End Of Chain Tuple*** for processing). The CISTPL_DEVICE (tuple code 01H) must be the first non-control tuple found when traversing the chain(s).

It is recommended that 16-bit PC Cards using a low voltage key begin the primary CIS chain with a CISTPL_LINKTARGET (tuple code 13H). Low voltage keyed cards shall omit the CISTPL_DEVICE (tuple code 01H) or shall include a CISTPL_DEVICE with NULL *Device Info* fields.

Cards with > 64 MBytes of Common Memory must contain the tuple CISTPL_EXTDEVICE (tuple code 09H).

For flexibility, the CIS of a 16-bit PC Card can be extended into Common Memory. To facilitate automatic identification of "blank" cards, Attribute Memory can be read-only memory.

It is expected that the CIS will be written once when the card is manufactured (or formatted) and then infrequently updated. On any PC Card that expects or requires the CIS to be erased (for example a Flash or EEPROM technology card that erases the CIS area when it is reorganized), it is suggested that problems of process interruption and disaster recovery be addressed. These issues are beyond the scope of the Standard.

Note that most implementations will be limited to reading cards of a specific format, or at most, of a few different formats. Thus, many combinations of values available in the tuples will be nonportable. It is recommended that implementers restrict themselves to the suggested low-level formats defined in the Media Storage Formats Specification.

2.3.9 CardBus PC Card Tuple Chain Processing

The recommended method of traversing a CardBus PC Card CIS is to use the Card Services **GetFirstTuple/GetNextTuple** interface. When a client does this, Card Services will decode any long link tuples for the client. The beginning of each function's CIS is indicated by the **CIS Pointer** in that function's configuration space header. The *Address Space Indicator* field indicates in which space the CIS begins and the *Address Space Offset* field gives the offset into that space for the memory spaces and the expansion ROM space. For the configuration space, the *Address Space Offset* field gives the absolute address in device-dependent configuration space. (See also the ***Electrical Specification*** and ***3.1.6 CISTPL_LONGLINK_CB: The CardBus PC Card LongLink Tuple***.)

Each tuple chain on a CardBus PC Card must begin with a CISTPL_LINKTARGET (tuple code 13H). It may contain one CISTPL_LONGLINK_CB (tuple code 05H) to another tuple chain in the current space or another space. The CISTPL_LONGLINK_CB tuple allows the placement of tuple chains in configuration space, memory space, or the expansion ROM.

Traversing a CardBus PC Card's CIS(s) is the same as traversing that of a 16-bit PC Card, with the following exceptions:

1. There is a separate CIS for each card function.
2. For a given function, the beginning of the CIS is pointed to by the CIS Pointer.
3. All tuple chains, including the first one, begin with a CISTPL_LINKTARGET tuple.
4. The encoding of the CISTPL_LONGLINK_CB address matches that of the CIS Pointer.

The client requests tuples from each function's CIS by specifying the logical function number (0 through 7).

2.3.10 Tuple Processing Recommendations

This standard requires that system software be carefully coded in order to prevent incompatibilities from one system to another. The following are some specific recommendations.

The routine that reads a given tuple should be coded to start by examining the tuple code. If the tuple code is not recognized by the routine (e.g. if the code is vendor specific or represents an extension under a future standard), then the tuple should be ignored. If the code is not recognized, it is safe to read the code byte and the link byte. However, other bytes within the tuple may represent active registers.

2.3.10.1 Tuple Code Known

If the tuple code is known, and if the tuple does not contain active registers (which is the case for all standard tuples), then the routine should copy bytes into a buffer in main storage. Bytes should be copied from the code byte up to the last byte before the next tuple. If the link field is FFH (which also means end-of-chain and is only allowed for a 16-bit PC Card) then a maximum of 257 bytes — the code byte, the link byte and as many as 255 bytes of tuple data — should be copied from the card to the main store.

2.3.10.2 Processing Longlink Tuple

When processing a longlink tuple (CISTPL_LONGLINK_A, _C, _CB), software should merely record the target address and address space. The software should not validate the target address, nor should it immediately begin processing of tuples from the target address. Similarly, when a no-link tuple (CISTPL_NO_LINK) is found, that fact should be recorded for later use.

Longlink and no-link tuples should be processed *after* reaching the end of the tuple chain. At that time, if a longlink is to be processed, software should validate the target address (by checking for a CISTPL_LINKTARGET tuple) and begin processing the target chain if it appears to be valid.

2.3.10.3 Longlink Pointing to Invalid Tuple Chain

A longlink that points to an invalid tuple chain should not usually cause any diagnostic messages to be displayed to the user. This situation may result from an uninitialized card, from a card which was initialized for some unanticipated use, or from corrupted data. Since only the corrupted data case merits a diagnostic message, it is better to assume either that the card is uninitialized, or that it is initialized in some non-conforming way.

2.3.11 16-bit PC Card CIS with Indirect Access PC Card Memory

16-bit PC Cards with very limited Attribute and Common Memory spaces can utilize an optional, Common Memory register-based indirect access mechanism. (See the ***Electrical Specification***.) Using this indirect access method these cards provide full disclosure of their capabilities and attributes by extending their CIS into the indirect space(s). In all cases, a minimal legal CIS must be placed in the standard Attribute and Common Memory spaces.

The CISTPL_INDIRECT tuple is used to indicate the presence of indirect access registers. After processing tuple chains in the Attribute and Common memory spaces, tuple processing follows the implied link indicated by the CISTPL_INDIRECT tuple to the indirect Attribute and indirect Common spaces. Once tuple processing begins in the indirect spaces there is no return link to the direct access spaces. This means that all longlink tuples placed in the indirect spaces refer to the indirect spaces, i.e. a CISTPL_LONGLINK_A refers to indirect Attribute memory.

Tuple processing in the indirect spaces follows the previously stated rules for processing tuples on all 16-bit PC Cards. For example, the implied link goes first to the indirect Attribute space where a CISTPL_LINKTARGET should be located at indirect Attribute address zero. There is an implied link to indirect Common address zero if there is no tuple chain at indirect Attribute address zero.

An example of a minimal, but legally sufficient, Card Information Structure is shown in the table below.

Table 2-4: Minimal CIS for Indirect Memory Access PC Cards

Address Space / Offset	Tuple	Values
Attribute / 00H	CISTPL_DEVICE	01H 02H 00H FFH
Attribute / 08H	CISTPL_INDIRECT	03H 00H
Attribute / 0CH	CISTPL_END	FFH
Common / 00H	CISTPL_END	FFH

2.4 Metaformat Summary

2.4.1 Metaformat Layer Hierarchy

1. **The Basic Compatibility Layer** — Fundamental information about the card's physical devices and configurations.
2. **The Data Recording Format Layer** — Specifies how data on a card which is used for data storage is organized at the lowest level.
3. **The Data Organization Layer** — Specifies how data on a card which is used for data storage is logically organized.
4. **The System-Specific Layer** — Includes tuples that by their nature are specific to a particular operating environment or vendor implementation.

2.4.2 Tuple Code Summary

The following table provides a summary of all curent tuples and an indicator of the types of cards or the classes of functions where they are used.

The columns are:

Tuple Code	the numeric code assigned to the tuple.
Name	the name which uniquely identifies the tuple.
Description	a brief statement of the tuple's intended use.
First Pub	First Published - reference indicates when this tuple definition was first published.
CardBus	notes for CardBus PC Cards.
16-bit PC Card	
all	notes applying to 16-bit PC Cards in general — NA here applies also to all other 16-bit PC Card columns.
MEM	Memory - 16-bit PC Cards providing either disk-like or memory-like data storage.
CFG	Configurable - 16-bit PC Cards having configuration and status registers.
MFC	Multiple Function PC Card - notes for Multiple Function 16-bit PC Cards.
5 Volt Key	notes for 16-bit PC Cards which operate at 5 Volts **Vcc**.
Low Volt Key	Low Voltage - notes for 16-bit PC Cards which operate at 3.3 Volts or X.X Volts **Vcc**.

Column Entries:

R — Recommended
M — Mandatory
NA — Never, Not Applicable
S — Single Instance Only per Function-Chain; or Main-Chain in 16-bit MFC; or per region/partition definition; or base address register description
1 — first published PCMCIA 1.0 / JEIDA 4.0
2 — first published PCMCIA 2.0 / JEIDA 4.1
3 — first published PCMCIA 2.1 / JEIDA 4.2
4 — first published PC Card Standard, February 1995
5 — recommended for use as appropriate
6 — required in cards capable of operating at voltages other than 5 V **Vcc**
7 — recommended for use when applicable values (IDs and extensions) exist
8 — not recommended, requires NULL *Device Info* fields

9 — first published PC Card Standard, November 1995
10 — first published PC Card Standard March 1997
11 — first published PC Card Standard February 1999
12 — mandatory for cards with >64 Mbytes of Common Memory

Table 2-5 Tuple Summary Table

Tuple Code	Tuple Name	Description	1st Pub	Card Bus PC Card	16-bit PC Card					
					all	MEM	CFG	MFC	5 Volt Key	Low Volt Key
Layer 1: Basic Compatibility Tuples										
Control Tuples										
10H	CISTPL_CHECKSUM	Checksum control.	1							
FFH	CISTPL_END	The end-of-chain tuple.	1	M, 4	R, 3					
03H	CISTPL_INDIRECT	Indirect Access PC Card Memory	9	S	S					
13H	CISTPL_LINKTARGET	Link-target-control.	1	M,4,S	S			M, 4		R, 4
11H	CISTPL_LONGLINK_A	Longlink to Attribute Memory.	1	NA	S					
12H	CISTPL_LONGLINK_C	Longlink to Common Memory.	1	NA	S					
02H	CISTPL_LONGLINK_CB	Longlink to next chain on a CardBus PC Card.	4		NA					
06H	CISTPL_LONGLINK_MFC	Longlink to function specific chain(s) on a Multiple Function PC Card.	4	NA	S			M, 4		
14H	CISTPL_NO_LINK	No-link to Common Memory.	1	NA	S					
00H	CISTPL_NULL	Null tuple - ignore.	1							
Basic Compatibility Tuples										
16H	CISTPL_ALTSTR	Alternate-language-string.	1							
01H	CISTPL_DEVICE	Common Memory device information.	1	NA	S				M,3	8
17H	CISTPL_DEVICE_A	Attribute Memory device information.	1	NA	S				R,3	8
1DH	CISTPL_DEVICE_OA	Other operating conditions device information for Attribute Memory.	2	NA					5	5
1CH	CISTPL_DEVICE_OC	Other operating conditions device information for Common Memory.	2	M, 4					5,6	M,4, 6
1EH	CISTPL_DEVICEGEO	Device geometry information for Common Memory devices.	3	5,S	S	5				
1FH	CISTPL_DEVICEGEO_A	Device geometry information for Attribute Memory devices.	3	5,S	S	5				
09H	CISTPL_EXTDEVICE	Extended Common Memory device information	11	NA		12				
22H	CISTPL_FUNCE	Function Extensions.	3	7			7			
21H	CISTPL_FUNCID	Function class identification.	3	7,S	S		7			
19H	CISTPL_JEDEC_A	JEDEC programming information for Attribute Memory.	1	NA	5,S					
18H	CISTPL_JEDEC_C	JEDEC programming information for Common Memory.	1	5,S	5,S					
20H	CISTPL_MANFID	Manufacturer Identification string.	3	M,4,S	R,3, M,4, S					
15H	CISTPL_VERS_1	Level 1 version/product-information.	1	M,4,S	M,4, S					

Table 2-5 Tuple Summary Table - Continued

Tuple			1st	Card	16-bit PC Card					
Code	**Tuple Name**	**Description**	**Pub**	**Bus PC Card**	**all**	**MEM**	**CFG**	**MFC**	**5 Volt Key**	**Low Volt Key**
Layer 1: Basic Compatibility Tuples										
Configuration Tuples										
07H	CISTPL_BAR	Base Address Register definition tuple for a CardBus PC Card.	4	M,4,S	NA					
1BH	CISTPL_CFTABLE_ENTRY	A Configuration-Table-Entry.	2	NA			R,3, M,4	M,4		
05H	CISTPL_CFTABLE_ENTRY_CB	A Configuration-Table-Entry for a CardBus PC Card function.	4	M,4	NA					
1AH	CISTPL_CONFIG	Configuration tuple for a 16-bit PC Card.	4	NA	S		R,3, M,4	M,4		
04H	CISTPL_CONFIG_CB	Configuration tuple for a CardBus PC Card function.	4	M,4,S	NA					
08H	CISTPL_PWR_MGMNT	Function state save/restore definition.	10	NA	S					
Layer 2: Data Recording Format Tuples										
Card Information Tuples										
45 H	CISTPL_BATTERY	Battery replacement date.	1	S	S					
44H	CISTPL_DATE	Card initialization date.	1	S	S					
40H	CISTPL_VERS_2	Level-2 version tuple.	1	S	S					
Data Recording Format Tuples										
43H	CISTPL_BYTEORDER	Byte ordering for disk-like partitions.	1	S	S	R,3				
41H	CISTPL_FORMAT	Data recording format for Common Memory	1	5,S	5,S					
47H	CISTPL_FORMAT_A	Data recording format for Attribute Memory	4	NA	5,S					
42H	CISTPL_GEOMETRY	Partition geometry.	1	5,S	5,S					
23H	CISTPL_SWIL	Software interleaving.	3	S	S					
Layer 3: Data Organization Tuples										
46H	CISTPL_ORG	Partition organization.	1	5,S	5,S					
Layer 4: System-Specific Standard Tuples										
90H	CISTPL_SPCL	Special Purpose	4							
80H··8FH		Vendor unique tuples	1							
Available Tuple Codes										
0AH··0FH		Reserved for future Layer 1 tuples.								
24H··3FH		Reserved for future Layer 2 tuples.								
48H··7FH		Reserved for future Layer 3 tuples.								
91H··FEH		Reserved for future Layer 4 tuples.								

Table 2-6 Tuple Summary Table (Numerically by Tuple Code)

Code	Name	Description	Metaformat Layer
00H	CISTPL_NULL	Null tuple - ignore.	Layer 1: Control
01H	CISTPL_DEVICE	Common Memory device information.	Layer 1: Basic Compatibility
02H	CISTPL_LONGLINK_CB	Longlink to next chain on a CardBus PC Card.	Layer 1: Control
03H	CISTPL_INDIRECT	Indirect Access PC Card Memory.	Layer 1: Control
04H	CISTPL_CONFIG_CB	Configuration tuple for a CardBus PC Card function.	Layer 1: Configuration
05H	CISTPL_CFTABLE_ENTRY_CB	A Configuration-Table-Entry for a CardBus PC Card function.	Layer 1: Configuration
06H	CISTPL_LONGLINK_MFC	Longlink to next chain on a Multiple Function card.	Layer 1: Control
07H	CISTPL_BAR	Base Address Register definition tuple for a CardBus PC Card.	Layer 1: Configuration
08H	CISTPL_PWR_MGMNT	Function state save/restore definition.	Layer 1: Configuration
09H	CISTPL_EXTDEVICE	Extended Common Memory device information	Layer 1: Basic Compatibility
0AH··0FH		Reserved for future Layer 1 tuples.	
10H	CISTPL_CHECKSUM	Checksum control.	Layer 1: Control
11H	CISTPL_LONGLINK_A	Longlink to Attribute Memory.	Layer 1: Control
12H	CISTPL_LONGLINK_C	Longlink to Common Memory.	Layer 1: Control
13H	CISTPL_LINKTARGET	Link-target-control.	Layer 1: Control
14H	CISTPL_NO_LINK	No-link to Common Memory.	Layer 1: Control
15H	CISTPL_VERS_1	Level 1 version/product-information.	Layer 1: Basic Compatibility
16H	CISTPL_ALTSTR	Alternate-language-string.	Layer 1: Basic Compatibility
17H	CISTPL_DEVICE_A	Attribute Memory device information.	Layer 1: Basic Compatibility
18H	CISTPL_JEDEC_C	JEDEC programming information for Common Memory.	Layer 1: Basic Compatibility
19H	CISTPL_JEDEC_A	JEDEC programming information for Attribute Memory.	Layer 1: Basic Compatibility
1AH	CISTPL_CONFIG	Configuration tuple for a 16-bit PC Card.	Layer 1: Configuration
1BH	CISTPL_CFTABLE_ENTRY	A Configuration-Table-Entry.	Layer 1: Configuration
1CH	CISTPL_DEVICE_OC	Other operating conditions device information for Common Memory.	Layer 1: Basic Compatibility
1DH	CISTPL_DEVICE_OA	Other operating conditions device information for Attribute Memory.	Layer 1: Basic Compatibility
1EH	CISTPL_DEVICEGEO	Device geometry information for Common Memory devices.	Layer 1: Basic Compatibility
1FH	CISTPL_DEVICEGEO_A	Device geometry information for Attribute Memory devices.	Layer 1: Basic Compatibility
20H	CISTPL_MANFID	Manufacturer Identification string.	Layer 1: Basic Compatibility
21H	CISTPL_FUNCID	Function class identification.	Layer 1: Basic Compatibility
22H	CISTPL_FUNCE	Function Extensions.	Layer 1: Basic Compatibility
23H	CISTPL_SWIL	Software interleaving.	Layer 2: Data Recording Format
24H··3FH		Reserved for future Layer 2 tuples.	
40H	CISTPL_VERS_2	Level-2 version tuple.	Layer 2: Card Information
41H	CISTPL_FORMAT	Data recording format for Common Memory	Layer 2: Data Recording Format

Tuple Summary Table (Numerically by Tuple Code) - continued

Code	Name	Description	Metaformat Layer
42H	CISTPL_GEOMETRY	Partition geometry.	Layer 2: Data Recording Format
43H	CISTPL_BYTEORDER	Byte ordering for disk-like partitions.	Layer 2: Data Recording Format
44H	CISTPL_DATE	Card initialization date.	Layer 2: Card Information
45H	CISTPL_BATTERY	Battery replacement date.	Layer 2: Card Information
46H	CISTPL_ORG	Partition organization.	Layer 3: Data Organization
47H	CISTPL_FORMAT_A	Data recording format for Attribute Memory	Layer 2: Data Recording Format
80H··8FH		Vendor unique tuples	Layer 4: System-Specific Standard
90H	CISTPL_SPCL	Special Purpose	Layer 4: System-Specific Standard
90H··FEH		Reserved for future Layer 4 tuples.	
FFH	CISTPL_END	The end-of-chain tuple.	Layer 1: Control

2.5 Special Considerations

2.5.1 Vendor-Specific Information

Vendor-specific information allows card and software vendors to implement proprietary functions while remaining within the general framework of this Standard.

Vendor-specific information is of two kinds:

- Vendor-specific fields are areas reserved in the data structures for free use by vendors. These fields have no meaning to the standard software.
- Vendor-specific codes are encoding values reserved to represent non-standard values in standard fields. In the absence of other information, standard software must interpret vendor-specific codes as meaning "the information in this field is not specified."

The card-manufacturer field in the CIS gives (knowledgeable) system software enough information to interpret vendor-specific fields and code values in the card Physical-Description tuples.

Similarly, the *OEM* and *INFO* fields in the CISTPL_VERS_2 tuple give (knowledgeable) system software enough information to interpret vendor-specific fields and code values in the card Logical-Format tuples.

In general, a system will not be able to interpret all possible vendor-specific fields or code values.

2.5.2 System Rejection of Unsupported Cards

This standard requires the following behavior when a system encounters an unrecognized vendor-specific field:

- If the unrecognized field itself is vendor-specific, the system shall ignore that field.

If a standard field contains an unrecognized vendor-specific code, the system must refuse to perform any operation that requires the information encoded in that field.

3. BASIC COMPATIBILITY (LAYER 1)

This layer is the cornerstone of the Standard. All PC Cards shall have at least a rudimentary Card Information Structure.

The CIS on 16-bit PC Cards must start at address zero of the card's Attribute Memory space.

Each function of a CardBus PC Card has a CIS Pointer field in configuration space which points to a CISTPL_LINKTARGET tuple which begins the CIS for that function. (See also the ***Electrical Specification***.)

3.1 Control Tuples

Multiple instances of a tuple are allowed unless otherwise specified.

3.1.1 CISTPL_CHECKSUM: The Checksum Tuple

For additional reliability, the CIS can contain one or more checksum tuples. This tuple has three fields:

- The relative address of the block of CIS memory to be checked;
- The length of the block of CIS memory to be checked; and
- The expected checksum.

The checksum algorithm is a straight modulo-256 sum. Relative addressing is used to make the CIS, as a whole, position independent. The checksum tuple can only validate memory in its own address space.

Table 3-1 CISTPL_CHECKSUM: Checksum Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_CHECKSUM (10H)							
1	TPL_LINK Link to next tuple (at least 5).							
2··3	TPLCKS_ADDR Offset to region to be checksummed, stored with LSB first.							
4··5	TPLCKS_LEN Length of region to be checksummed, given with LSB first.							
6	TPLCKS_CS The checksum of the region.							

The checksum is calculated by summing the bytes of the selected region using modulo 256. The result must match the value stored in byte 6 of the checksum tuple.

TPLCKS_ADDR contains the offset, relative to the start address of this tuple, of the region to be checksummed. The address is a signed, 2-byte integer. Negative values indicate locations prior to the checksum tuple; positive values indicate locations after the checksum tuple. The exact interpretation depends on the address space containing the tuple.

TPLCKS_LEN contains the number of bytes to be checksummed. The number is expressed as an unsigned, 2-byte integer.

If the tuple appears in the Common Memory space of a 16-bit PC Card or in any of the spaces of a CardBus PC Card, the checksum is calculated in the obvious way. The contents of *TPLCKS_ADDR* (as a signed integer) are added to the base address of the tuple, yielding the target address. Starting at the target address, the algebraic sum is calculated of all the bytes included in the range. Then, the low-order 8-bits of this sum are compared to the value stored in *TPLCKS_CS*. If identical, the region of tuple memory covered by the checksum passes the checksum test.

If the tuple appears in the Attribute Memory space of a 16-bit PC Card, the checksum operation is a bit more complicated. Again, the data structures are recorded in such a way as to minimize the differences between Attribute space representation and Common Memory representation. To form the target address, 2 * *offset* is added to the base byte target address of the tuple. Then, the algebraic sum is formed of the even bytes in the address range [i.e., *target*, *target* + 2 * *length-1*], ignoring the odd bytes. The low-order 8-bits of this sum is then compared to the value stored in *TPLCKS_CS*.

3.1.2 CISTPL_END: The End Of Chain Tuple

The end of chain tuple marks the end of a tuple chain. It has a non-standard form and consists solely of the code byte. All CardBus PC Card tuple chains are required to be terminated with a CISTPL_END tuple. No other method of terminating a tuple chain is permissible for CardBus PC Cards.

Table 3-2 CISTPL_END: End of Chain Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_END (FFH): end of this tuple chain.						

Upon encountering this tuple, system software shall take one of the following actions.

- If a longlink tuple was encountered previously in this chain, continue tuple processing at the location specified in the longlink tuple.
- If processing a tuple chain (other than the primary CIS tuple chain of a 16-bit PC Card), and no longlink tuple was seen in this chain, then no tuples remain to be processed.
- If processing the primary CIS tuple chain of a 16-bit PC Card (the chain starting at address 0 in Attribute Memory space), and a CISTPL_NO_LINK tuple was encountered previously in this chain, no tuples remain to be processed.
- If processing the primary CIS tuple chain of a 16-bit PC Card (the chain starting at address 0 in Attribute Memory space), and neither a longlink nor a no-link tuple were seen in this chain, then continue tuple processing as if a longlink to address 0 of Common Memory space were encountered. For validation of the implied longlink to Common Memory, as with an explicit CISTPL_LONGLINK_C tuple, the tuple chain in Common Memory must begin with a valid CISTPL_LINKTARGET tuple.

3.1.3 CISTPL_INDIRECT: Indirect Access PC Card Memory

16-bit PC Cards providing indirect access to memory spaces through registers placed in Common Memory shall indicate the presence of these registers with this tuple. (See the *Electrical Specification*.) When processing software encounters this tuple in either Attribute or Common Memory it shall assume the presence of an additional chain beginning at address zero (0) of either the indirect Attribute or the indirect Common Memory space. Tuple processing in the indirect spaces follows processing of all tuples in the direct spaces. All tuple chains in indirect spaces begin with CISTPL_LINKTARGET tuples.

Table 3-3: CISTPL_INDIRECT: Indirect Access PC Card Memory

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_INDIRECT (03H)							
1	TPL_LINK Link to next tuple (may be zero).							

Note: The body of this tuple is always empty.

This tuple chain shall appear at most once in a Card Information Structure and always in a tuple chain present in direct accessed Attribute or Common Memory space.

3.1.4 CISTPL_LINKTARGET: The LinkTarget Tuple

The linktarget tuple is used for robustness. Every longlink tuple must point to a valid linktarget tuple. The linktarget tuple has one field— the string "CIS". It is recommended that the link field of the link target point to the next byte after the CISTPL_LINKTARGET tuple. Processing software is required to check that the linktarget tuple is correct before deciding to process the tuple chain at the new target address. This tuple must be at the beginning of every tuple chain present on a CardBus PC Card and the beginning of any secondary chain(s) on a 16-bit PC Card. It is recommended that 16-bit PC Cards using a low voltage key begin their primary CIS chain with this tuple.

Table 3-4 CISTPL_LINKTARGET: Link Target Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_LINKTARGET (13H)							
1	TPL_LINK Link to next tuple (at least 3).							
2	TPLTG_TAG "C" (43H)							
3	"I" (49H)							
4	"S" (53H)							

3.1.5 CISTPL_LONGLINK_A, CISTPL_LONGLINK_C: The 16-bit PC Card LongLink Tuples

A given tuple chain for a 16-bit PC Card shall contain at most one longlink tuple (_A, _C or _MFC). The longlink tuples are used to jump beyond the limits of the 1-byte link field, from one tuple chain to another. The target tuple chain may be in Attribute Memory or Common Memory space, as indicated by the tuple code.

WARNING

The 16-bit PC Card longlink tuples (_A, _C and _MFC) are not applicable to, and must never be used by, CardBus PC Cards.

Table 3-5 CISTPL_LONGLINK_A and CISTPL_LONGLINK_C: 16-bit PC Card LongLink Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE LongLink tuple code (CISTPL_LONGLINK_A, 11H; or CISTPL_LONGLINK_C, 12H)							
1	TPL_LINK Link to next tuple (at least 4).							
2··5	TPLL_ADDR Target address; stored as an unsigned long, low-order byte first.							

Note: A given tuple chain shall contain at most one longlink tuple (_A, _C, _CB or _MFC).

The tuple code byte selects the new address space. For example, CISTPL_LONGLINK_A indicates that the target is in Attribute Memory space; CISTPL_LONGLINK_C indicates Common Memory space.

A given tuple chain shall contain at most one longlink tuple. The longlink tuple need not appear as the last tuple in a given chain because all remaining tuples in the current chain will be processed before the link is honored.

Software shall verify that the longlink tuple points to a linktarget tuple before processing the target chain. Because a longlink tuple may point to uninitialized RAM, it is important that software simply reject target tuple chains that do not begin with a linktarget tuple.

The *TPLL_ADDR* field of a CISTPL_LONGLINK_A tuple is interpreted in the same manner as its *TPL_LINK* field, only even bytes in the address range are counted. To compute the address of the linktarget in the card's Attribute Memory space, multiply the value in the *TPLL_ADDR* field by two.

3.1.6 CISTPL_LONGLINK_CB: The CardBus PC Card LongLink Tuple

This gives the location of a tuple chain in a function's address space, whether device dependent configuration space, memory space, or the expansion ROM. This tuple may also occur in tuple chains in any of these spaces, as well.

Table 3-6 CISTPL_LONGLINK_CB: The CardBus PC Card LongLink Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_LONGLINK_CB (02H)						
1	TPL_LINK Link to next tuple (at least 4)							
2-5	TPLL_ADDR	Indicates the location of a tuple chain in a function's address space. The encoding is the same as for the CIS Pointer. (See the ***Electrical Specification***.)						

Note: A given tuple chain shall contain at most one longlink tuple (_A, _C, _CB or _MFC).

TPLL_ADDR points to a tuple chain in one of the following spaces:

- configuration space - must begin in device dependent configuration space at or after location 40H.
- memory space - may be in any of the (up to) six (6) spaces. Note that this cannot be an I/O space.
- Expansion ROM space - may be in any of the images.

The encoding for this pointer is given below:

***TPLL_ADDR* field**

DWORD	31	…	28	27	…	8	7	…	3	2	1	0
	Address Space Offset									Address Space Indicator		
Config Space	Reserved, must be 0						offset within Config Space			replace with 0s		
Memory Space	offset within Base Address Register memory space									replace with 0s		
Exp ROM	image number			offset from image base						replace with 0s		

The *Address Space Indicator* field indicates in which of this function's address spaces the tuple chain is located. The encoding for this field is given below:

***Address Space Indicator* field**

Value	Meaning
0	Tuple chain is in device dependent configuration space.
1-6	The tuple chain is in the memory address space governed by one of the six Base Address Registers. For example, if the value is 2, then the tuple chain is in the memory address space governed by Base Address Register 2.
7	The tuple chain is in the Expansion ROM space.

The offset into the address space indicated by the *Address Space Indicator* field is given by the 32 bit value where the three low order bits, 0 through 2, are zero and the high order bits, 3 through 31, are given by the *Address Space Offset* field. An address at which a tuple chain begins will have an 8 byte alignment. Appropriate values for the various spaces are given below:

***Address Space Offset* field**

Address Space Indicator	**Space Type**	**Address Space Offset**
0	configuration space	$40_H \le$ value $\le F8_H$. The address in device dependent configuration space at which the tuple chain starts.
x; $1 \le x \le 6$	memory space	$0 \le$ value $\le FFFFFFF8_H$. This is the offset into the memory address space governed by Base Address Register x. Adding this value to the value in the Base Address Register gives the location of the start of the tuple chain.
7	expansion ROM	$0 \le$ image $\le F_H$, $0 \le$ value $\le 0FFFFFF8_H$. This is the offset into the expansion ROM address space governed by the Expansion ROM Base Register. The image number is in the uppermost nibble of the Address Space Offset. The value consists of the remaining bytes. The image is the image number used as the location reference for the tuple chain. The value is the offset of the tuple chain from the base of that image. Adding this offset value plus the starting offset of the image to the value in the Expansion ROM Base Register gives the location of the start of the tuple chain.

3.1.7 CISTPL_LONGLINK_MFC: The Multiple Function 16-bit PC Card Link Tuple

This longlink tuple indicates the number of sets of configuration registers on a Multiple Function 16-bit PC Card. This tuple does not apply to and must never be used by a CardBus PC Card. This tuple also describes the location of the each function-specific CIS describing the function provided by a specific set of configuration registers.

When a *TPLMFC_TASn* field is 00H, indicating a longlink to Attribute Memory space, the associated *TPLMFC_ADDRn* field is interpreted in the same manner as its *TPL_LINK* field in a tuple placed in Attribute Memory space, only even bytes in the address range are counted. To compute the address of the linktarget in the card's Attribute Memory space, multiply the value in the *TPLMFC_ADDRn* field by two.

Table 3-7 CISTPL_LONGLINK_MFC: The Multiple Function 16-bit PC Card Link Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_LONGLINK_MFC (06H)							
1	TPL_LINK Link to next tuple (at least 6, 1 plus 5 bytes per function)							
2	TPLMFC_NUM Number of sets of configuration registers for individual functions							
3	TPLMFC_TAS1 CIS Target address space for first function on the 16-bit PC Card (00 = Attribute, 01 = Common)							
4‥7	TPLMFC_ADDR1 Target address, stored as an unsigned long, low order byte first							
8	TPLMFC_TAS2 CIS Target address space for second function on the 16-bit PC Card							
9‥12	TPLMFC_ADDR2 Target address, stored as an unsigned long, low order byte first							
13‥n	Additional TPLMFC_TASn and TPLMFC_ADDRn entries for any additional functions on the 16-bit PC Card. If there are only two (2) sets of configuration registers, only the prior fields shall be present.							

Note: A given tuple chain shall contain at most one longlink tuple (_A, _C, _CB or _MFC).

3.1.8 CISTPL_NO_LINK: The No-Link Tuple

To save Attribute Memory space on 16-bit PC Cards, processing software shall assume the presence of a CISTPL_LONGLINK_C tuple to address 0 of Common Memory as part of the primary tuple chain. The primary tuple chain starts at address 0 of Attribute Memory space. This assumption can be overridden either by placing an explicit longlink tuple (CISTPL_LONGLINK_A or CISTPL_LONGLINK_C) or a no-link tuple (CISTPL_NO_LINK) in the primary tuple chain.

Table 3-8 CISTPL_NO_LINK: The No-Link Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_NO_LINK (14H)						
1	TPL_LINK Link to next tuple (may be zero).							

Note: The body of this tuple is always empty.

A given tuple chain shall contain at most one no-link tuple. No-link tuples and longlink tuples are mutually exclusive. A given chain may contain either a no-link tuple, or a longlink tuple, but not both.

3.1.9 CISTPL_NULL: The Null Tuple

The null tuple is simply a place holder. It has a non-standard form and consists solely of the code byte.

Table 3-9 CISTPL_NULL: The Null Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_NULL (00H): ignore this tuple.							

Note: Software shall ignore these tuples. The next tuple begins at the next byte in sequence.

3.2 Basic Compatibility Tuples

Multiple instances of a tuple are NOT allowed unless otherwise specified.

3.2.1 CISTPL_ALTSTR: The Alternate Language String Tuple

Several tuples contain character strings which are intended to be displayed to the user only under certain circumstances. Some international applications need the ability to store strings for a number of different languages. Rather than having various languages used in the tuples, this standard provides alternate string tuples. Strings in the primary tuples are always recorded in ISO 646 IRV code using characters in the range 20H··7EH. Multiple instances of this tuple are allowed; each associates with most recent (previous) "NON-ALT STRING" tuple. Tuple codes affected are 15H (CISTPL_VERS_1) and 40H (CISTPL_VERS_2).

Alternate string tuples contain two kinds of information:

- A code representing the language (an ISO-standard escape sequence), and
- A series of strings.

These strings are to be substituted for the primary strings when operating in a different language environment.

Table 3-10 CISTPL_ALTSTR: The Alternate Language String Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_ALTSTR (16H)							
1	TPL_LINK Link to next tuple (at least p-1).							
2··(m-1)	TPLALTSTR_ESC ISO-standard escape sequence to select the character set for these strings. Indicates which character set is associated with these strings. The leading ESCAPE is not recorded. Terminated by a NULL (00H). A special escape sequence denotes the PC Extended ASCII character set.							
m··(n-1)	Alternate string 1 translation for first string in most recent non-ALTSTR tuple. Terminated by 00H.							
n··(o-1)	Alternate string 2 translation for second string in most recent non-ALTSTR tuple. Terminated by 00H.							
…	Etc.							
p	FFH - marks end of strings.							

3.2.2 CISTPL_DEVICE, CISTPL_DEVICE_A: The 5 volt Device Information Tuples

The 5 volt device information tuples contain information about the card's devices. The tuples contain: device speed, device size, device type, and address space layout information for either Attribute Memory, CISTPL_DEVICE_A, or Common Memory, CISTPL_DEVICE, space. PC Cards with a 5 volt key shall present a device information tuple for Common Memory space, CISTPL_DEVICE, as the first non-control tuple in Attribute Memory. (See ***2.3.8 16-bit PC Card Tuple Chain Processing***). It is recommended that the device information tuple for Attribute Memory, CISTPL_DEVICE_A, be provided.

16-bit PC Cards that operate at both 5 volts **V**CC and 3.3 volts **V**CC shall use the Device Info fields in CISTPL_DEVICE and CISTPL_DEVICE_A to describe the characteristics when operating at 5 volts **V**CC. One or more Other Conditions Tuples, CISTPL_DEVICE_OC and CISTPL_DEVICE_OA, shall exist to describe the 3.3 volts **V**CC operating characteristics. Whenever both CISTPL_DEVICE and CISTPL_DEVICE_OC tuples are present, there must be a one-to-one correspondence between their Device Info entries. This one-to-one correspondence shall also exist between Device Info entries in the CISTPL_DEVICE_A and CISTPL_DEVICE_OA tuple(s). Null devices are counted in the matching process.

Table 3-11 CISTPL_DEVICE, CISTPL_DEVICE_A: The 5 volt Device information Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_DEVICE (01H) or CISTPL_DEVICE_A (17H)							
1	TPL_LINK Link to next tuple (at least m-1)							
n	Device Info 1 (2 or more bytes)							
	Device Info 2 (2 or more bytes)							
…	(etc.) Device Info n (2 or more bytes)							
m	FFH (marks end of *Device Info* field)							

The tuple code CISTPL_DEVICE indicates that this tuple describes Common Memory space. The code CISTPL_DEVICE_A indicates that this tuple describes Attribute Memory space.

See section ***3.2.2.1 Device Info Fields***, for the *Device Info* field definition.

3.2.2.1 Device Info Fields (For Tuples CISTPL_DEVICE, CISTPL_DEVICE_A)

The device information tuples are composed of a sequence of *Device Info* fields. Each field is further composed of two variable length byte sequences — the *Device ID* and the *Device Size*. Each *Device Info* field defines the characteristics of a group of addresses in the appropriate memory space.

Device Info **fields**

Byte	7	6	5	4	3	2	1	0
n	Device ID	one or more bytes of Device ID fields						
n + m	Device Size	one byte indicating the device size						
	# of address units - 1					Size Code		

3.2.2.1.1 Device ID Fields

The device ID indicates the device type and the access time for a block of memory.

Device ID **fields**

Byte	7	6	5	4	3	2	1	0
n	Device Type Code				WPS	Address Space Indicator or Device Speed		
n + 1	Extended Device Speed	If Device Speed Code equals 7_H, otherwise omitted.						
	EXT	Speed Mantissa				Speed Exponent		
(n+2)··(o-1)	Additional Extended Device Speed	While EXT of previous byte is set.						
	EXT	Reserved for future use.						
o	Extended Device Type	If Device Type Code equals E_H, otherwise omitted.						
	EXT	Extended Device Type Code						
(o+1)··	Additional Extended Device Type	While EXT of previous byte is set.						
	EXT	Additional Extended Device Type Code						

3.2.2.1.1.1 Device Type Code Field

The *Device Type Code* field in bits 4 through 7 of byte 0 of the *Device ID* sequence indicates the device type. The following codes are defined:

***Device Type Code* field**

Code	Name	Meaning
0	DTYPE_NULL	No device. Generally used to designate a hole in the address space. If used for 16-bit PC Cards, the Device Speed field should be set to 0H.
1	DTYPE_ROM	Masked ROM
2	DTYPE_OTPROM	One Time Programmable ROM
3	DTYPE_EPROM	UV EPROM
4	DTYPE_EEPROM	EEPROM
5	DTYPE_FLASH	Flash EPROM
6	DTYPE_SRAM	Static RAM
7	DTYPE_DRAM	Dynamic RAM
8··CH		Reserved
DH	DTYPE_FUNCSPEC	Function-specific memory address range. Includes memory-mapped I/O registers, dual-ported memory, communication buffers, etc., which are not intended to be used as general-purpose memory. Accesses to function-specific address ranges may be configuration dependent
EH	DTYPE_EXTEND	Extended type follows.
FH		Reserved

Note: Device Type Codes are used to describe only devices which are fixed in their memory address, not dynamically relocatable devices. Relocatable devices are described by the configuration tuples.

If there are no devices in a 16-bit PC Card's common memory space, the Device Type Code shall be DTYPE_NULL. If the devices at offset zero in the 16-bit PC Card's common memory space are intended for function specific use, such as memory mapped I/O registers or communications buffers, the Device Type Code shall be DTYPE_FUNCSPEC.

The extended device type, if specified, is reserved for future use. Bit 7, if set, indicates that the next byte is also an extended type byte. The chain of extended type bytes can continue indefinitely. The end is marked by an extended device type byte with bit 7 reset.

3.2.2.1.1.2 WPS Field

The *WPS* bit indicates whether the **Write Protect Switch** is in control of the device(s) in this address range. When the *WPS* bit is reset (0), the Write Protect Switch and **WP** signal indicate whether or not the device(s) is(are) writable. When the *WPS* bit is set (1), the device is always writable unless the device code is DTYPE_ROM, in which case this address range is never writable. (See the ***Electrical Specification***.)

3.2.2.1.1.3 Address Space Indicator Field (CardBus PC Card only)

The *Address Space Indicator* field indicates the memory or Expansion ROM Base Address Register associated with the devices described in the CISTPL_DEVICE_OC tuple. The Address Space Indicator field used here has the same format and interpretation as the field used in the CIS Pointer and the CISTPL_LONGLINK_CB tuple. An *Address Space Indicator* value of zero (0) is illegal since it would indicates that the device(s) is(are) in CardBus PC Card Configuration Space. (See ***3.1.6 CISTPL_LONGLINK_CB: The CardBus PC Card LongLink Tuple***.)

3.2.2.1.1.4 Device Speed Field (16-bit PC Card only)

If the device speed/type byte is 00H, there is no device at this address. If the device size information is valid, the address range shall be treated as a NULL device.

Bits 0 through 2, and up to one or two additional bytes, represent the speed of the devices associated with this part of the address space. The speed value indicates the external card access time to this address range (See the ***Electrical Specification***.) The device speed field contains one of the values in the table below.

Device Speed Codes

Code	Name	Meaning
0	DSPEED_NULL	Use when device type = NULL
1	DSPEED_250NS	250 nsec
2	DSPEED_200NS	200 nsec
3	DSPEED_150NS	150 nsec
4	DSPEED_100NS	100 nsec
5··6		(Reserved)
7	DSPEED_EXT	Use extended speed byte.

Note: The values given in the *Device Speed* fields of CISTPL_DEVICE and CISTPL_DEVICE_A tuples must define the worst case cycle time required by a 16-bit PC Card without the use of the **WAIT#** signal. This is required by PCMCIA 1.0 / JEIDA 4.0 host platforms that pre-date the definition of the **WAIT#** signal. 16-bit PC Cards that support the use of the **WAIT#** signal shall include the CISTPL_DEVICE_OC tuple to define 5 volt device speed with the **WAIT#** signal honored.

If the extended speed byte is zero, then the byte should be ignored.

The EXT bit, if set, indicates that an additional extended speed byte follows. The meaning of that byte is not presently defined. However, the string of extended speed bytes may be arbitrarily long. It extends through (and includes) the first byte with bit 7 reset.

The extended device speed *mantissa* and *exponent* specify the speed of the device, as follows:

Extended Device Speed Codes

Mantissa		Exponent part	
Code	**Meaning**	**Code**	**Meaning**
0H	Reserved	0H	1 ns
1H	1.0	1H	10 ns
2H	1.2	2H	100 ns
3H	1.3	3H	1 µs
4H	1.5	4H	10 µs
5H	2.0	5H	100 µs
6H	2.5	6H	1 ms
7H	3.0	7H	10 ms
8H	3.5		
9H	4.0		
AH	4.5		
BH	5.0		
CH	5.5		
DH	6.0		
EH	7.0		
FH	8.0		

3.2.2.1.2 The Device Size Byte (For Tuples CISTPL_DEVICE, CISTPL_DEVICE_A)

Within the device information tuple fields, following the device speed/type information, is the *Device Size* byte. The indicated size describes the total memory address range of the specified device type.

Device Size Codes

Code	Units	Max Size
0	512 bytes	16 KBytes
1	2 KBytes	64 KBytes
2	8 KBytes	256 KBytes
3	32 KBytes	1 MByte
4	128 KBytes	4 MBytes
5	512 KBytes	16 MBytes
6	2 MBytes	64 MBytes
7	Reserved	Reserved

Bits 3 through 7 represent the number of address units. A code of zero indicates 1 unit; a code of 1 indicates 2 units; and so on.

For a device size > 64 MBytes the device size byte shall indicate a size of 64 MBytes. The tuple CISTPL_EXTDEVICE shall indicate the exact divice size when device size exceeds 64 MBytes.

If the device size byte is FFH, then this entry should be treated as an end marker for the device information tuple. The device type and speed information encoded for this entry should be ignored.

3.2.3 CISTPL_DEVICE_OC, CISTPL_DEVICE_OA: The Other Conditions Device information Tuples

The Other Conditions device information tuples contain information about the card's devices under a set of operating conditions. The tuples are identical in format to the device information Common and Attribute Memory tuples except that an *Other-Conditions-Info* field precedes the first *Device Info* field. (See ***3.2.2 CISTPL_DEVICE, CISTPL_DEVICE_A: The 5 volt Device Information Tuples***.) There may be several CISTPL_DEVICE_OC and CISTPL_DEVICE_OA tuples in the CIS — one to describe the card under each set of alternative operating conditions.

There shall be a one-to-one correspondence between entries of each related tuple that describes operating characteristics. If a CISTPL_DEVICE tuple is present, there must be a one-to-one correspondence between entries in the CISTPL_DEVICE tuple and each CISTPL_DEVICE_OC tuple. All CISTPL_DEVICE_OC tuples have corresponding entries. If a CISTPL_DEVICE_A tuple is present, there must be a one-to-one correspondence between entries in the CISTPL_DEVICE_A tuple and each CISTPL_DEVICE_OA tuple. All CISTPL_DEVICE_OA tuples have corresponding entries. Null devices are counted in the matching process.

CardBus PC Cards shall use the CSTPL_DEVICE_OC tuple to describe memory space devices.

Table 3-12 CISTPL_DEVICE_OC, CISTPL_DEVICE_OA: The Other Conditions Device information Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_DEVICE_OC (1CH) and CISTPL_DEVICE_OA (1DH).						
1	TPL_LINK	Link to next tuple (m-1, 3 or more).						
2	Other Conditions Info	1 or more bytes						
n	Device Info 1	2 or more bytes						
	Device Info 2	2 or more bytes						
…	Device Info n	(etc.) 2 or more bytes						
m		FFH (marks end of Device Info fields).						

3.2.3.1 Other Conditions Info Field

The *Other-Conditions-Info* field is a sequence of one or more bytes each with up to seven condition fields and one extension bit. The condition fields indicate the set of defined conditions which apply to the device information in the tuple. There are two condition fields defined — the *VccUsed* and the *MWAIT* fields. All undefined fields are reserved for future standardization and must be zero. Bit 7 of each byte is the extension bit which indicates that another byte of conditions follows the present byte.

***Other-Conditions-Info* field**

7	6	5	4	3	2	1	0
EXT	Reserved, must be 0.				VccUsed		MWAIT
EXT	Additional Bytes	Each is present only if EXT bit is set in previous byte					

The *MWAIT* field is ignored by CardBus PC Cards.

When the *MWAIT* field is set (1), the timings given are intended for PCMCIA 2.0 / JEIDA 4.1 and later systems which will observe the **WAIT#** signal. Note that the card is not required to assert the

WAIT# signal during operation. Host systems which do not support the **WAIT#** signal, PCMCIA 1.0 / JEIDA 4.0 and earlier, shall not use timing data provided in a tuple with the *MWAIT* field set (1).

When the *MWAIT* field is reset (0), the timings given represent worst case cycle times and shall be used by host systems which do not support the **WAIT#** signal. Note that the *VccUsed* field shall not indicate 5 V **VCC** operation because this condition, MWAIT reset (0) and 5 V **VCC**, is indicated using the CISTPL_DEVICE and CISTPL_DEVICE_A tuples

The *VccUsed* field indicates a **VCC** voltage at which the PC Card can be operated. A PC Card capable of operating at more than one **VCC** voltage shall have a separate instance of the Other Conditions tuple, CISTPL_DEVICE_OC or CISTPL_DEVICE_OA, for each such operating voltage.

***VccUsed* field**

Value	Meaning
00B	5 volt **Vcc** operation.
01B	3.3 volt **Vcc** operation.
10B	Reserved for X.X volt **Vcc** operation.
11B	Reserved for CardBus PC Card Y.Y volt **Vcc** operation.

The ***Device Info*** fields for the CISTPL_DEVICE_OC and CISTPL_DEVICE_OA tuples have the same definition and format as the CISTPL_DEVICE and CISTPL_DEVICE_A tuples. The fields are defined in section ***3.2.2.1 Device Info Fields (For Tuples CISTPL_DEVICE, CISTPL_DEVICE_A)***.

3.2.2.1Device Info Fields

3.2.4 CISTPL_DEVICEGEO, CISTPL_DEVICEGEO_A: Device Geometry Tuples

The device geometry info tuple CISTPL_DEVICEGEO is required for certain memory technologies (e.g. Flash Memory, EEPROM) or card integrated memory subsystems. While conceptually similar to a DOS disk geometry tuple (CISTPL_GEOMETRY), it is not a format dependent property; this deals with the fixed architecture of the memory device(s) or subsystem(s). Rather than being specific to a partition within a larger device type, this applies to the entire address range of that device type. Accordingly, each *Device Geometry Info* entry in the CISTPL_DEVICEGEO tuple must have a corresponding *Device Info* entry, including NULL device space, in CISTPL_DEVICE or CISTPL_DEVICE_OC tuples when the CISTPL_DEVICEGEO tuple is employed. This one-to-one relationship also applies to device geometry entries in the CISTPL_DEVICEGEO_A tuple and device info entries in CISTPL_DEVICE_A or CISTPL_DEVICE_OA tuples.

Table 3-13 CISTPL_DEVICEGEO and CISTPL_DEVICEGEO_A: Device Geometry Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_DEVICEGEO (1EH) and CISTPL_DEVICEGEO_A (1FH)						
1	TPL_LINK	Link to next tuple (at least 6 * k).						
2··7		Device Geometry Info for Device 1.						
8··13		Device Geometry Info for Device 2.						
		(etc.)						
··((6 * k)+1)		Device Geometry Info for Device k.						

***Device Geometry Info* field**

Byte	7	6	5	4	3	2	1	0
+0	DGTPL_BUS	Value = n, where card interface width = $2^{(n-1)}$ bytes. n = 2 for 16-bit PC Cards. n = 3 for CardBus PC Cards. The value n = 00H is not allowed.						
+1	DGTPL_EBS	Value = n, where memory array/subsystem's physical memory segments have a minimum erase block size of $2^{(n-1)}$ address increments of DGTPL_BUS-wide accesses. The value n = 00H is not allowed.						
+2	DGTPL_RBS	Value = n, where memory array/subsystem's physical memory segments have a minimum read block size of $2^{(n-1)}$ address increments of DGTPL_BUS-wide accesses. The value n = 00H is not allowed.						
+3	DGTPL_WBS	Value = n, where memory array/subsystem's physical memory segments have a minimum write block size of $2^{(n-1)}$ address increments of DGTPL_BUS-wide accesses. The value n = 00H is not allowed.						
+4	DGTPL_PART	Value = p, where memory array/subsystem's physical memory segments can have partitions subdividing the arrays in minimum granularity of $2^{(p-1)}$ number of erase blocks. P = 1 where array partitioning at erase block boundaries is allowed. The value p = 00H is not allowed.						
+5	DGTPL_HWIL	Value = q, where card architectures employ a multiple of $2^{(q-1)}$ times interleaving of the entire memory arrays or subsystems with the above characteristics. Non-interleaved cards have values of q = 1. The value q = 00H is not allowed.						

If the *Device Geometry Info* field is used, the entire six byte entry must be filled out for each entry in the corresponding CISTPL_DEVICE, CISTPL_DEVICE_OC, CISTPL_DEVICE_OA or CISTPL_DEVICE_A, even for devices without any special geometry (e.g. ROM, RAM), so that a 1:1 correspondence between *Device Info* and *Device Geometry Info* fields of the described regions exists. The table length is 6 * k + 2 bytes, where k = the number of devices described. Bit 7 of

DGTPL_HWIL, the last (sixth) byte of each detailed *Device Geometry Info* field, is reserved for future use and must be reset (0). When set (1) this bit indicates that an extended device geometry information structure follows.

The *DGTPL_BUS* tuple has a value of n = 2 for 16 bit bus width of 16-bit PC Cards regardless of word or byte wide operation (byte wide operation is logically low and high byte sequencing of a fundamental 16 bit wide card internal bus) and n = 3 for the 32 bit bus width of CardBus PC Cards. The purpose of this entry is to accommodate possible wider width cards of the future and/or to allow file systems to use this tuple structure in non-card environments (e.g. system resident memory arrays using the same file system).

Note: The device geometry info is normalized to byte equivalent geometry to simplify the context for the O/S or driver level software which utilizes it. Subsystems using internal bus widths less than 8 bits wide must employ low-level drivers which accept 8 bit minimum data from the higher levels.

The *DGTPL_EBS*, *DGTPL_RBS*, and *DGTPL_WBS* (address increment or bus operation based values) are multiplicative of the *DGTPL_BUS* entry (denoting bus width) to define the non-interleaved physical memory erase, read, and write block sizes in bytes, respectively. The *DGTPL_HWIL* value for cards employing hardware interleaved (i.e., "banks" of) memory arrays or subsystems (where *DGTPL_HWIL* - 2) is multiplicative of the resulting non-interleaved erase, read, and write geometry. The product of these three geometry info layers yields the resulting card level minimum physical block geometry.

DGTPL_PART is a special partitioning info field based on physically distinct segments of the memory array(s) such that data are not affected by read/write/erase operations in adjacent partitions. It is multiplicative of the resulting erase geometry (only) and defines the minimum partition size allowed for that card. *DGTPL_PART: p* = 1 where array partitioning at erase block boundaries is allowed (i.e., there are no special partitioning requirements).

Examples:

1. A particular non-interleaved (*DGTPL_HWIL*: q = 1) 16-bit PC Card based memory array (*DGTPL_BUS*: n = 2) employs a new EEPROM type of byte-wide memory device which erases in 4K bytes (*DGTPL_EBS*: n = 13) and writes in 256 byte pages (*DGTPL_WBS*: n = 9). It has no special partitioning requirements (*DGTPL_PART*: p = 1). The resulting physical geometry is:

		Bus	x Block	x Interleave		
ERASE Geometry	=	2 x	4096	x 1		= 8192 bytes
READ Geometry	=	2 x		1	x 1	= 2 bytes
WRITE Geometry	=	2 x		256	x 1	= 512 bytes
PARTITION Boundary	=	ERASE Geometry			x 1	= 8192 bytes

2. A particular four layer interleaved (*DGTPL_HWIL*: q = 3) 16-bit PC Card based memory array (*DGTPL_BUS*: n = 2) employs a new type of word wide flash memory device with built in byte/word mode operation. Internal components erase in blocks that are 64 KBytes, or 32 KBytes times their 16 bit bus (*DGTPL_EBS*: n = 16) and write in single words (*DGTPL_WBS*: n = 1). It requires partitioning in erase block groups of four (*DGTPL_PART*: p = 3). The resulting physical geometry is:

		Bus	x Block	x Interleave		
ERASE Geometry	=	2 x	32,768	x 4		= 262,144 bytes
READ Geometry	=	2 x	1	x 4		= 8 bytes
WRITE Geometry	=	2 x	1	x 4		= 8 bytes
PARTITION Boundary	=	ERASE Geometry			x 4	= 1,024 KBytes

3.2.5 CISTPL_EXTDEVICE: The Extended Common Memory Device Information Tuple

The extended common memory device information tuple contains information about the card's devices which reside in common memory. The CISTPL_EXTDEVICE tuple contains device speed, device size, device type, address space layout, address extension size, address extension register location and memory page size information for Common Memory space. The tuple (illustrated in ***Table 3-14***) is primarily intended to be used with PC Cards containing more than 64 MBytes of common memory.

When this tuple is present and recognized by the system software, the data provided by the tuple shall override the ***Device Info*** field data presented by the CISTPL_DEVICE and CISTPL_DEVICE_OC tuples; however, tuple CISTPL_EXTDEVICE should present the same ***Device ID*** field information in the same sequential order as what the tuple CISTPL_DEVICE and/or tuple CISTPL_DEVICE_OC provides. Even though the CISTPL_EXTDEVICE tuple overrides the ***Device Info*** parts of a CISTPL_DEVICE_OC tuple, the ***Other Conditions Info*** part of the CISTPL_DEVICE_OC tuple shall still apply.

To maintain backward compatibility as much as possible with host systems which do not recognize the CISTPL_EXTDEVICE tuple, tuples CISTPL_DEVICE and CISTPL_DEVICE_OC should be encoded to describe the first (non-extended) 64 MByte region of common memory for a PC Card. For each device type the CISTPL_DEVICE and/or CISTPL_DEVICE_OC tuple(s) would encode a maximum device size of 64 MBytes. Then, all host systems would recognize the CISTPL_DEVICE and/or CISTPL_DEVICE_OC tuple(s) and, using the tuple information, be able to access the 1st 64 MBytes of the card's memory. If a host system supported the CISTPL_EXTDEVICE tuple, the host system would process the information presented by the CISTPL_EXTDEVICE tuple and ignore the information contained in the CISTPL_DEVICE and CISTPL_DEVICE_OC tuples.

Table 3-14 CISTPL_EXTDEVICE: The Extended Common Memory Device information Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_EXTDEVICE (09H)							
1	TPL_LINK Link to next tuple (at least m-1)							
2	Memory Paging Info (1 or more bytes)							
n	Device Info 1 (2 or more bytes)							
	Device Info 2 (2 or more bytes)							
…	(etc.) Device Info n (2 or more bytes)							
m	FFH (marks end of *Device Info* field)							

3.2.5.1 Memory Paging Info Field

The *Memory Paging-Info* field is a sequence of one or more bytes. Each byte consists of a set of fields with up to seven bits of paging information and one extension bit. The fields indicate defined paging conditions which apply to the device information in the tuple.

All undefined fields in the *Memory Paging-Info* byte sequence are reserved for future standardization and shall be zero. Bit 7 of each byte is defined as the extension bit which indicates that another byte of paging information follows the present byte. For the first byte of the byte sequence there are 3 paging condition fields — the *Page Size, Page Address Location* and the *Number of Address Extension Bits* fields. Other bytes in the byte sequence are undefined.

The 3 fields for the single defined byte are defined as follows:

***Memory-Paging-Info* field**

7	6	5	4	3	2	1	0
EXT	Number of Address Extension Bits - 1				Page Address Location	Page Size	
EXT	Additional Bytes Each is present only if EXT bit is set in previous byte						

The *Page Size* field indicates the page size for Common Memory.

The *Page Address Location* bit applies only for 64 MByte page size and denotes which Function Configuration Register provides a set of address extension bits for accessing > 64 MBytes of Common Memory. The *Page Address Location* bit indicates the choice of either the Configuration Option Register or Address Extension Register 0 as the source for the address extension bits.

For a 32 MByte page size Address Extension Registers 0 and 1 provide the address extension bits for addressing a 32 MByte memory page, while for a 16 MByte page size Address Extension Registers 0, 1, 2 and 3 contain the address extension bits to address a 16 MByte memory page.

The *Memory Paging-Info* field's *Page Size* and *Page Address Location* sub-fields indicate the page size and Function Configuration Registers as follows:

Memory-Paging-Info Field Bits			**Description**	
Page Address Location	**Page Size**			
Bit 2*	**Bit 1**	**Bit 0**	**Selected Page Size**	**Function Configuration Register(s) providing Address Extension**
0	0	0	64 MBytes	Address Extension Register 0
1	0	0	64 MBytes	Configuration Option Register
	0	1	32 MBytes	Address Extension Registers 0 and 1
	1	0	16 MBytes	Address Extension Registers 0, 1, 2 and 3
	1	1	Reserved	—

* Bit 2 is defined only for the condition where bits 1 and 0 both equal `0'. For all other conditions bit 2 is reserved and must = `0'.

For the Function Configuration Register(s) providing an address extension the ***Number of Address Extension Bits - 1*** field indicates the actual number of bits in the address extension provided by the register(s). The number of actual address extension bits is represented as a binary value equal to the number of actual address extension bits minus 1; thus, 0 would imply that an address extension consisted a single address extension bit.

3.2.5.2 Device Info fields (for tuple CISTPL_EXTDEVICE)

The device information tuples are composed of a sequence of *Device Info* fields. Each field is further composed of two variable length byte sequences — the *Device ID* and the *Device Size*. Each *Device Info* field defines the characteristics of a group of addresses in the appropriate memory space.

For a device memory size ≤ 64 MBytes the following definition holds:

Device Info **fields**

Byte	7	6	5	4	3	2	1	0
n	Device ID one or more bytes of Device ID fields							
n + m	Device Size one byte indicating the device size for a ≤ 64 MByte device memory space							
	# of address units - 1					Size Code		

For a device memory size > 64 MBytes the following definition holds:

Device Info **fields**

Byte	7	6	5	4	3	2	1	0
n	Device ID one or more (m) bytes of Device ID fields							
n + m	Device Size Extender Indicates > 64 MBytes of device memory space							
	0	0	0	0	ExtBytes	1	1	1
	ExtBytes = 0 signifies that 1 Device Extended Size byte and then the Device Final Size byte follow extender byte. ExtBytes = 1 signifies that 2 Device Extended Size bytes and then the Device Final Size byte follow extender byte. All other bit combinations of bits 7 through 0 indicate that this byte is the single Device Size byte defining a ≤ 64 MByte device memory space (i.e., is not extender byte) and that no further device size bytes follow.							
n+m+1	P7	P6	P5	P4	P3	P2	P1	P0
	Device Extended Size byte #1 Bits 0 through 7 of P (`64 MByte page' multiplier lower byte)							
n+m+2	P15	P14	P13	P12	P11	P10	P9	P8
	Device Extended Size byte #2 Bits 8 through 15 of P (`64 MByte page' multiplier upper byte)							
n + m + (s-1)	Device Final Size Last of three or four bytes (s). This byte codes an amount of device memory up to a maximum of 64 MBytes.							
	# of address units - 1					Size Code		

3.2.5.2.1 Device ID fields

Same as Section ***3.2.2.1.1 Device ID Fields***.

3.2.5.2.2 The Device Size Byte(s) (For Tuple CISTPL_EXTDEVICE)

As outlined in Section 3.2.4, the device size information depends on whether or not the amount of common memory for the device being described is greater than 64 MBytes. For a ≤ 64 MByte device memory size a single Device Size Byte provides the information. However, for a > 64 MByte device memory size, 3 or 4 bytes contain the device size information.

3.2.5.2.2.1 Device Memory Size ≤ 64 MBytes

Within the device information tuple fields, following the device speed/type information, is the *Device Size* byte. The indicated size describes the total memory address range of the device type specified with the device speed/type information. The *Device Size* byte consists of 2 fields, the # *of Address Units - 1* field (bits 7 - 3 of the *Device Size* byte) and the Size Code field (bits 2 - 0 of the *Device Size* byte).

For ≤ 64 MByte memory device size the # *of Address Units - 1* field represents the number of memory units specified by the *Size Code* field. A code of zero indicates 1 unit, a code of 1 indicates 2 units, and so on (for device size codes 0 through 6).

When the device size code = 7 (and reserved bits 7 through 4 of the *Device Size* byte = 0), then a > 64 MByte device memory size is indicated (see following section).

The *Size Code* field ranges in value from 0 to 7 with codes 0 to 6 used to determine ≤ 64 MByte device memory size The *Size Code* field is defined as follows:

Device Size Codes

Code	Units	Max Size
0	512 bytes	16 Kbytes
1	2 Kbytes	64 Kbytes
2	8 Kbytes	256 Kbytes
3	32 Kbytes	1 MByte
4	128 Kbytes	4 MBytes
5	512 Kbytes	16 Mbytes
6	2 MBytes	64 Mbytes
7*		
ExtBytes = 0	Identifies byte as Device Size Extender byte for > 64 MByte device memory size and indicates that 1 Device Extended Size byte and the Device Final Size byte follow next in the tuple sequence (see following section)	
ExtBytes = 1	Identifies byte as Device Size Extender byte for > 64 MByte device memory size and indicates that 2 Device Extended Size bytes and the Device Final Size byte follow next in the tuple sequence (see following section)	
Device Size [7::4]	Reserved (= 0)	

* Code 7 is defined for a memory device size > 64 MBytes. For a code 7 the *Device Size* byte is redefined to be the *Device Size Extender* byte, the first of a sequence of bytes defining > 64 MByte device size. The *Device Size Extender* byte contains the *ExtBytes* field which is not defined for the *Device Size* byte.

If the *Device Size* byte is FFH, then this entry should be treated as an end marker for the device information tuple. The device type and speed information encoded for this entry should be ignored.

3.2.5.2.2.2 Device Memory Size > 64 MBytes

Card and Socket Services software support is specified for PC Card common memory spaces not exceeding 4 Gigabytes. The size definition which follows allows for PC Cards with more than 4 Gigabytes of common memory; the larger memory provisions can be utilized with Memory Technology Driver (MTD) software which bypasses Card and Socket Services software to directly drive a PC Card.

Within the device information tuple fields, following the device speed/type information, is a sequence of bytes for defining a > 64 MByte device memory size. The byte sequence describes the total memory address range of the device type specified with the device speed/type information. For > 64 MByte device memory spaces the device size information is contained in the following 4 bytes:

1. Device Size Extender Byte
2. Device Extended Size Byte 1
3. Device Extended Size Byte 2 (for device memory size > 16 Gigabytes) and
4. Device Final Size Byte

Bits 0 - 2 of the *Device Size Extender* byte must all = 1 in order to indicate device size extension beyond 64 MBytes. If bits 0 -2 do not all = 1, then they represent the *Size Code* field for a device memory size < 64 MBytes (see the previous section).

If the *Device Size Extender* byte is FFH, then this entry should be treated as an end marker for the device information tuple. The device type and speed information encoded for this entry should be ignored.

When the single-bit *ExtBytes* field (bit 3) of the *Device Size Extender* byte = 0, a single *Device Extended Size* byte and then a *Device Final Size* byte follow next in line after the *Device Size Extender* byte.

Device memory size is calculated using the single *Device Extended Size* byte and the *Device Final Size* byte as follows:

(P)* 64 MBytes + Device Final Size

where ,

P = the binary-encoded value provided by the *Device Extended Size* byte
and
Device Final Size = the device size coded in the *Device Final Size* byte.

The *Device Final Size* byte which follows the *Device Extended Size* byte is formatted identical to the single *Device Size* Byte used for < 64 MByte device memory sizes. Thus, 64 MBytes, obtained with a device size code of 6, is the maximum value which can be indicated for Device Final Size. The *Device Final Size* byte should not contain a device size code of 7 (where the device size codes are defined in the table of the previous section).

When the single-bit *ExtBytes* field (bit 3) of the *Device Size Extender* byte = 1, the *Device Size Extender* byte indicates that 2 *Device Extended Size* bytes and then a *Device Final Size* byte follow next in line after the *Device Size Extender* byte. The order and format of the 2 *Device Extended Size* bytes are illustrated in section 3.2.4.

Device memory size is calculated using the 2 *Device Extended Size* bytes and the *Device Final Size* byte as follows:

(P)* 64 MBytes + Device Final Size

where:

P = the binary-encoded value provided by the *Device Extended Size* bytes
and
Device Final Size = the device size coded in the *Device Final Size* byte.

The *Device Final Size* byte which follows the 2 *Device Extended Size* bytes is formatted identical to the single *Device Size* Byte used for < 64 MByte device memory sizes. Thus, 64 MBytes, obtained with a device size code of 6, is the maximum value which can be indicated for Device Final Size. The *Device Final Size* byte should not contain a device size code of 7 (where the device size codes are defined in the table of the previous section).

Regardless of the device size stipulated by the *Device Extended Size* byte(s) immediately preceding the *Device Final Size* byte, if the *Device Final Size* byte is FFH, then this entry should be treated as an end marker for the device information tuple. The device type and speed information encoded for this entry should be ignored.

3.2.5.3 Tuple Examples for Memory Device Size > 64 MBytes

If 16-bit PC Card Common Memory consists of 128 MBytes of Flash memory arranged in 64 MByte pages and the page address is provided by the COR, then a software header block might be laid out as follows:

```
/byte[0]:      (CISTPL_DEVICE_OC,      /*1CH*/
/byte[1]:      /*link*/    4,
/byte[2] /*other conditions -- 3.3 volt Vcc operation */       1<<1,
/byte[3]:      /*type/speed*/          DTYPE_FLASH | DSPEED_200NS,
/byte[4]:      /*device size –– units/code – 64 MBytes*/  (1Fh<<3) | 6,
/byte[5]:      /*end of tuple*/  FFh
      );
/byte[6]:      (CISTPL_EXTDEVICE,      /*09H*/
/byte[7]:      /*link*/    6,
/byte[8]:      /*number of address extension bits/page address
location/page size*/ 1<<2 ,
/byte[9]:      /*type/speed*/          DTYPE_FLASH | DSPEED_200NS,
/byte[10]:     /*device size extender –– 1 device extended size byte*/
      7,
/byte[11]:     /*device extended size byte*/ 1,
/byte[12]:     /*device final size byte –– units/code – 64 MBytes*/
      (1Fh<<3) | 6,
/byte[13]:     /*end of tuple*/  FFh
      );
/byte[14]:           (CISTPL_CONFIG,   /*1AH*/
/byte[15]:           /*link*/    6,
/byte[16]:           /*field sizes -- TPCC_RFSZ/TPCC_RMSZ/TPCC_RASZ*/
      1,
/byte[17]:           /*TPCC_LAST*/     0,
/byte[18]:           /*TPCC_RADR byte 0*/    00h,
/byte[19]:           /*TPCC_RADR byte 1*/    40h,
/byte[20]:           /*TPCC_RMSK*/     01h,
/byte[21]:           /* end of tuple*/ FFh
      );
```

If a 16-bit PC Card contains a 576 Mbyte Common Memory area consisting of 128 MBytes of ROM memory, followed by a 128 MByte empty (null) area, followed by 320 MBytes of Flash memory, then a software header block might be laid out as follows using 16 MByte pages:

```
/byte[0]:      (CISTPL_DEVICE,   /*01H*/
/byte[1]:      /*link*/    7,
/byte[2]:      /*type/speed */   DTYPE_ROM| DSPEED_100NS,
/byte[3]:      /*device size –– units/code - 64 MBytes*/  ((1Fh<<3) | 6)
/byte[4]:      /*type/speed –– no device present*/  DTYPE_NULL |
DSPEED_NULL,
/byte[5]:      /*device size –– units/code - 64 MBytes*/  ((1Fh<<3) | 6)
/byte[6]:      /*type/speed*/          DTYPE_FLASH | DSPEED_200NS,
/byte[7]:      /*device size –– units/code - 64 MBytes*/  ((1Fh<<3) | 6)
/byte[8]:      /*end of tuple*/  FFh
      );
/byte[9]:      (CISTPL_EXTDEVICE,            /*09H*/
/byte[10]:     /*link*/    Eh,
/byte[11]:     /*number of address extension bits/page address
location/page size*/ ((6<<3) | 2)
/byte[12]:     /*type/speed*/          DTYPE_ROM | DSPEED_100NS,
/byte[13]:     /*device size extender –– 1 device extended size byte */
      7,
/byte[14]:     /*device extended size byte*/ 01h,
/byte[15]:     /*device final size byte –– units/code - 64 MBytes */
      ((1Fh<<3) | 6)
/byte[16]:     /*type/speed –– no device present */ DTYPE_NULL |
DSPEED_NULL,
/byte[17]:     /*device size extender –– 1 device extended size byte*/
      7,
/byte[18]:     /*device extended size byte*/ 01h,
/byte[19]:     /*device final size byte –– units/code - 64 MBytes */
      ((1Fh<<3) | 6)
/byte[20]:     /*type/speed*/          DTYPE_FLASH | DSPEED_200NS,
/byte[21]:     /*device size extender –– 1 device extended size byte*/
      7,
/byte[22]:     /*device extended size byte*/ 04h,
/byte[23]:     /*device final size byte –– units/code - 64 MBytes */
      ((1Fh<<3) | 6)
/byte[24]:     /*end of tuple*/  FFh
         );
/byte[25]:           (CISTPL_CONFIG,   /*1AH*/
/byte[26]:           /*link*/    8,
/byte[27]:           /*field sizes -- TPCC_RFSZ/TPCC_RMSZ/TPCC_RASZ*/
      2<<2 | 1,
/byte[28]:           /*TPCC_LAST*/     0,
/byte[29]:           /*TPCC_RADR byte 0*/    00h,
/byte[30]:           /*TPCC_RADR byte 1*/    40h,
/byte[31]:           /*TPCC_RMSK byte 0*/    00h,
/byte[32]:           /*TPCC_RMSK byte 1*/    54h,
/byte[33]:           /*TPCC_RMSK byte 2*/    01h,
/byte[34]:           /* end of tuple*/ FFh
      );
```

3.2.6 CISTPL_FUNCE: Function Extension Tuple

The function extension tuple contains information about a specific PC Card function. The information provided is determined by the function identification tuple, CISTPL_FUNCID, that is being extended. Each function has a defined set of extension tuples. They are described in subsections following this one.

Table 3-15 CISTPL_FUNCE: Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK Link to next tuple (see following sections)							
2	TPLFE_TYPE	Type of extended data (see following sections)						
3··n	TPLFE_DATA	Function information (see following sections)						

The *TPLFE_TYPE* field identifies the type of extended information provided about a function by this tuple. This information varies depending on the function being extended.

The *TPLFE_DATA* field is the specific information being provided by the extended function. The content of this field varies depending on the function being extended and the *TPLFE_TYPE* field.

Each of the function classes identified in the PC Card Functions table may be extended. Function extension tuples are optional. If present, they relate to the function identification tuple preceding them in the Card Information Structure.

Actual card identification extensions are to be determined by working groups concerned with specific PC Card function classes. The following subsections describe specific extension tuples. Multiple Function 16-bit PC Card function identification tuples are not extendible. The subsections are numbered corresponding to the function code being extended. If a particular function may be extended with multiple types of extended data, a further division is provided for each type of extension data within the subsection.

3.2.6.1 Function Extension Tuples for Serial Ports

The tuples defined in this document contain information which describe the operational features of a modem. Modems are identified using a function ID of a Serial Port and defined further using function extension tuples to describe specific capabilities. The structure of a function extension tuples is determined by a code appearing in the *TPLFE_TYPE* field. Function extension tuples are intended for informational use and not configuration control. Only those features commonly available and of particular interest to application software are included.

The *TPLFE_TYPE* field presented below, distinguishes each successive CISTPL_FUNCE serial port/modem extension tuple. Since all function extension tuples contain the same code (22H), both the position and the value of the *TPLFE_TYPE* field must be used to identify the functionality described by a particular extension tuple. Consequently, the serial port/modem extension tuples defined in this document must always appear immediately after the CISTPL_FUNCID tuple defined for serial port devices.

The codes defined for the *TPLFE_TYPE* field are presented below. Please note that separate codes are provided to describe the modem and serial port interfaces for individual modem services. Separate codes are also provided to describe the modem and serial port interfaces common to all modem services.

***TPLFE_TYPE*: Serial Port Extension Tuple Type Codes**

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				PC Card Subfunction Id			

***PC Card Subfunction Id:* Identifies a category of services provided by the modem and ISDN Terminal Adapter**

Code	Description
0	Identifies the extension tuple as a description of a serial port interface.
1	Identifies the extension tuple as a description of the capabilities of the modem interface, **common to all modem and ISDN Terminal Adapter** services.
2	Identifies the extension tuple as a description of **data** modem services.
3	Identifies the extension tuple as a description of **facsimile** modem services.
4	Identifies the extension tuple as a description of **voice** encoding services.
5	Identifies the extension tuple as a description of the capabilities of the **data** modem interface.
6	Identifies the extension tuple as a description of the capabilities of the **facsimile** modem interface.
7	Identifies the extension tuple as a description of the capabilities of the **voice** modem interface.
8	Identifies the extension tuple as a description of a serial port interface for **data** modem services.
9	Identifies the extension tuple as a description of a serial port interface for **facsimile** modem services.
10	Identifies the extension tuple as a description of a serial port interface for **voice** modem services.
11	Identifies the extension tuple as a description of **ISDN Terminal Adapter** services.
12	Identifies the extension tuple as a description of the capabilities of **ISDN Terminal Adapter** services.
13	Identifies the extension tuple as a description of a serial port interface for **ISDN Terminal Adapter** services.
14··15	Reserved for future standardization.

***PC Card Subfunction Descriptor*: Identifies a sub-category of services provided by the modem**

Code	Description
1··15	Identifies the extension tuple as a description of the EIA/TIA Service Class specified in the numeric value of the code.

3.2.6.1.1 Serial Port Interface Function Extension

This tuple indicates the type and capabilities of the serial port interface used in communicating with the modem. A specific code in the *TPLFE_TYPE* field is defined to describe the serial port interface to all modem services. Additional codes are also defined to describe the serial port interface for each available modem service (data, facsimile, voice, and/or ISDN terminal adapter). Please refer to the ***Serial Port Extension Tuple Type Codes*** Table for a list of all *TPLFE_TYPE* field codes.

The UART is identified with a generic reference to the de-facto standard it conforms to (i.e., 16450, 16550) not, however, a specific product identification. The UART capabilities field is provided to describe the asynchronous data formats supported in UART implementations which vary from the standard. When describing a de-facto standard UART the contents of this field may be set to zero.

The structure of the tuple and the codes currently defined for each field are presented below.

Table 3-16 CISTPL_FUNCE: Serial Port Interface Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (04H minimum)						
2	TPLFE_TYPE	Type of extended data (00H, 08H, 09H, 0AH, or 0DH)						
3	TPLFE_UA	Identification of the type of UART in use.						
4··5	TPLFE_UC	Identification of the UART capabilities.						

***TPLFE_UA*: UART Identification Field**

7	6	5	4	3	2	1	0
RFU, set to zero.			interface				

interface: The type of the UART is described using the codes defined below	
Code	**Description**
0	Indicates that an 8250 UART is present.
1	Indicates that a 16450 UART is present.
2	Indicates that a 16550 UART is present.
3	Indicates that an 8251 USART is present.
4	Indicates that an 8530 SCC is present.
5	Indicates that an 85230 ESCC is present.
6··31	Reserved for future standardization.

***TPLFE_UC*: UART Capabilities Field**

7	6	5	4	3	2	1	0
RFU, set to zero.				even parity	odd parity	mark parity	space parity
RFU, set to zero	2 stop bits	1.5 stop bits	1 stop bit	8 bit char	7 bit char	6 bit char	5 bit char

Field	Description
space parity	The protocol where the parity bit is always reset.
mark parity	The protocol where the parity bit is always set.
odd parity	The protocol where the parity bit is generated to create an odd number of set bits.
even parity	The protocol where the parity bit is generated to create an even number of set bits.
5 bit char	The encoding of data where each serial data unit contains 5 data bits.
6 bit char	The encoding of data where each serial data unit contains 6 data bits.
7 bit char	The encoding of data where each serial data unit contains 7 data bits.
8 bit char	The encoding of data where each serial data unit contains 8 data bits.
1 stop bit:	The use of one (1) stop bit encoded in each serial data unit.
1.5 stop bit	The use of one and one-half (1.5) stop bits encoded in each serial data unit.
2 stop bit	The use of two (2) stop bits encoded in each serial data unit.

When the field is set (1) the specified capability is supported, when the field is reset (0) the specified capability is not supported.

3.2.6.1.2 Function Extension Tuple for Modem and ISDN Interface

This structure describes the characteristics of the modem interface when considered separately from the UART. This includes an indication of the types of flow control supported, the size of the command buffer, DCE to DTE buffer, and the DTE to DCE buffer.

A specific code in the *TPLFE_TYPE* field is defined to describe the modem interface to all modem services. Additional codes are also defined to describe the modem interface for each available modem service (data, facsimile, voice, and/or ISDN Terminal Adapter). Please refer to the Serial Port Extension Tuple Type Codes table for a list of all *TPLFE_TYPE* field codes.

Table 3-17 CISTPL_FUNCE: Modem and ISDN Interface Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (09H minimum)						
2	TPLFE_TYPE	Type of extended data (01H, 05H, 06H, 07H, or 0CH)						
3	TPLFE_FC	Supported flow control methods.						
4	TPLFE_CB	Size in bytes of the DCE command buffer.						
5··7	TPLFE_EB	Size in bytes of the DCE to DTE buffer.						
8··10	TPLFE_TB	Size in bytes of the DTE to DCE buffer.						

***TPLFE_FC*: Flow Control Methods**

7	6	5	4	3	2	1	0
RFU, set to zero.			transparent	rx h/w flow ctrl	tx h/w flow ctrl	rx xon/xoff	tx xon/xoff

Field	Description
tx xon/xoff	DTE to DCE transmit flow control using DC1/DC3 for XON/XOFF
rx xon/xoff	DCE to DTE receive flow control using DC1/DC3 for XON/XOFF
tx h/w flow ctrl	DTE to DCE transmit flow control (CTS)
rx h/w flow ctrl	DCE to DTE receive flow control (RTS)
transparent	DCE to DCE flow control

When the field is set (1) the specified flow control method is supported, when the field is reset (0) the specified flow control method is not supported.

***TPLFE_CB*: DCE Command Buffer**

7	6	5	4	3	2	1	0
size of DCE command buffer							

size of DCE command buffer

Code	Description
v/4 - 1	Indicates the number of bytes within the command line buffer. To compute the actual size of the command buffer, the value of this field (represented in the formula as n) is increased by 1, then multiplied by 4 ((n+1)* 4). A command buffer of 256 bytes is represented by a field value of 3FH (63).

***TPLFE_EB*: Modem DCE to DTE Buffer Field**

7	6	5	4	3	2	1	0
size of DCE-DTE buffer — LSB of LSW							
size of DCE-DTE buffer — MSB of LSW							
size of DCE-DTE buffer — LSB of MSW							

size of DCE to DTE buffer

Code	Description
0	Indicates that no DCE to DTE buffer is present.
1··16777215	Indicates that a DCE to DTE buffer is available and its size in bytes.

The largest possible size in bytes of the modem's DCE to DTE buffer used in buffering received data, not including the UART FIFO buffer. The actual number of bytes in the buffer appears in this field to allow a more precise definition of its size.

***TPLFE_TB*: Modem DTE to DCE Buffer Field**

7	6	5	4	3	2	1	0
size of DTE-DCE buffer — LSB of LSW							
size of DTE-DCE buffer — MSB of LSW							
size of DTE-DCE buffer — LSB of MSW							

size of DTE to DCE buffer

Code	Description
0	Indicates that no DTE to DCE buffer is present.
1··16777215	Indicates that a DTE to DCE buffer is available and its size in bytes.

The largest possible size in bytes of the modem's DTE to DCE buffer used in buffering transmitted data, not including the UART FIFO buffer. The actual number of bytes in the buffer appears in this field to allow a more precise definition f its size.

3.2.6.1.3 Function Extension Tuple for Data Modem

This structure describes the capabilities of the data modem. This includes the level of error correction and data compression supported, the highest possible data rate supported in the DTE to UART interface of the data modem, the command set in use for DTE to DCE control and configuration (e.g. AT command set), the ITU-T(CCITT) standard and non ITU-T(CCITT) standard modulations supported, the standardized data encryption methods supported, and the ITU-T(CCITT) defined country code of the target market The escape codes supported for the return to command mode are also described. Features which cannot be included in the categories reserved for the capabilities mentioned above are included in the *TPLFE_EF* field, the "Miscellaneous End User Feature Selection" field.

The ITU-T(CCITT) country code field defines the countries targeted for distribution of the data modem. This field has been specified to support a maximum value of 254 (255 or FFH is reserved as a country code list terminator, see below) in accordance with the ITU-T(CCITT) standard T.35 which has assigned each member country or area a unique 8 bit value. Please refer to T.35 Annex A for a list of country or area codes currently assigned.

To specify support of multiple countries, the country code field *TPLFE_CD* was positioned as the last field in the Data Modem function extension tuple. Additional country codes may be specified by increasing the value of the *TPL_LINK* field by one for each additional country specified. A value of FFH is used to indicate the end of the country code list if it does not extend to the end of the tuple.

Any additional fields which follow this value must be represented by an appropriate increase in the value of the *TPL_LINK* field and are considered a description of manufacturer specific capabilities.

Table 3-18 CISTPL_FUNCE: Data Modem Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (0CH minimum)						
2	TPLFE_TYPE	Type of extended data (02H, see Serial Port Extension Tuple Type Codes)						
3··4	TPLFE_UD	The highest possible bit rate in the DTE to UART interface.						
5··6	TPLFE_MS	Modulation standards supported						
7	TPLFE_EM	Error correction/detection protocols and non ITU-T(CCITT) modulation schemes supported						
8	TPLFE_DC	Data compression protocols supported						
9	TPLFE_CM	Command protocols supported						
10	TPLFE_EX	Indication of the escape mechanisms supported						
11	TPLFE_DY	Standardized data encryption						
12	TPLFE_EF	Miscellaneous end user feature selection						
13··n	TPLFE_CD	The ITU-T(CCITT) defined country code (ITU-T(CCITT) T.35/T.35 Annex A)						

***TPLFE_UD*: UART Interface Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					DTE to UART maximum data rate MSB		
DTE to UART maximum data rate - LSB							

DTE to UART maximum data rate

Code	Description
v/75	Indicates the highest DTE to UART bit rate supported. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600H (1536).

***TPLFE_MS*: Modulation Standards Field**

7	6	5	4	3	2	1	0
V.26bis	V.26	V.22bis	Bell 212A	V.22A&B	V.23	V.21	Bell 103
RFU, set to zero.		V.90	V.34	V.32bis	V.32	V.29	V.27bis

Field	Description
Bell 103:	300 bps duplex modem standardized for use in the GSTN.
V.21:	300 bps duplex modem standardized for use in the GSTN.
V.23:	600/1200 baud modem standardized for use in the GSTN.
V.22A&B:	1200 bps duplex modem standardized for use in the GSTN and for telephone-type leased circuits.
Bell 212A:	2400 bps duplex modem standardized for use in the GSTN.
V.22bis:	2400 bps duplex modem using the frequency-division technique standardized for use on the GSTN and on point-to-point two-wire leased telephone-type circuits.
V.26:	2400 bps modem standardized for use on 4-wire leased telephone-type circuits.
V.26bis:	2400 bps duplex modem standardized for use in the GSTN, specifying two alternate encoding schemes (A or B).
V.27bis:	4800/2400 bps modem with automatic equalizer standardized for use on leased telephone-type circuits.
V.29:	9600/7200/4800 bps modem standardized for use on point-to-point 4-wire leased telephone-type circuits
V.32:	A family of two-wire, duplex modems operating at data rates of up to 9600 bps for use on the GSTN or on lease telephone-type circuits.
V.32bis:	A family of two-wire, duplex modems operating at data rates of up 14400 bps for use on the GSTN or on lease telephone-type circuits.
V.34:	A modem operating at data signaling rates of up to 33,600 bps for use on the GSTN and on leased point-to-point 2-wire telephone-type circuits.
V.90	A modem operating at data signaling rates of up to 56,000 bps (downstream) and 33,600 bps (upstream) for use on the GSTN when connecting to a digital switched network.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_EM*: Error Correction/Detection Protocols and Non ITU-T(CCITT) Modulation Schemes**

7	6	5	4	3	2	1	0
RFU, set to zero.						V.42 / LAPM	MNP 2-4

Field	Description
MNP 2-4:	Error correction/detection protocols progressively defined in MNP levels 2, 3, and 4.
V.42/LAPM:	Error correction/detection protocol defined by the ITU-T(CCITT).

When the field is set (1) the specified error correction/detection services and/or non ITU-T(CCITT) modulation schemes is supported, when the field is reset (0) the specified error correction/detection services and/or non ITU-T(CCITT) modulation schemes is not supported.

***TPLFE_DC*: Data Compression Protocols**

7	6	5	4	3	2	1	0
RFU, set to zero.						MNP 5	V.42bis

Field	Description
V.42bis	Data compression protocol defined by ITU-T(CCITT), requiring additional support of V.42.
MNP 5	Data compression protocol requiring the additional support of MNP 2, 3, or 4.

When the field is set (1) the specified compression protocol is supported, when the field is reset (0) the specified compression protocol is not supported.

***TPLFE_CM*: Command Protocols**

7	6	5	4	3	2	1	0
RFU, set to zero.	DMCL	V.25A	V.25bis	MNP AT	AT3	AT2	AT1

Field	Description
AT1	The “Action” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
AT2	The “ACE/DTE Interface Parameters” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
AT3	The “ACE Parameters” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
MNP AT	The command mode convention available for DTE to DCE control and configuration of the available MNP protocols.
V.25bis	A specification of the formatting and response of DCE commands used in automatic calling procedures over the 100-series interchange circuits.
V.25A	A specification on test facilities relating to the implementation of the V.25bis automatic calling procedure.
DMCL	A command mode convention available for DTE to DCE control and configuration.

When the field is set (1) the specified command protocol is fully supported, when the field is reset (0) the specified command protocol is not fully supported.

***TPLFE_EX*: Escape Mechanisms Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					User Defined Escape Char.	+++	BREAK

Field	Description
BREAK	The standardized BREAK represented by a sequence of 0 bits of a specified length.
+++	A sequence of characters used to return the modem to command mode.
User Defined Escape Char.	Indicates support for a user defined escape character.

When the field is set (1) the specified escape mechanism is supported, when the field is reset (0) the specified escape mechanism is not available to return the modem to command mode.

***TPLFE_DY*: Standardized Data Encryption**

7	6	5	4	3	2	1	0
RFU, set to zero.							

***TPLFE_EF*: Miscellaneous End User Feature Selection**

7	6	5	4	3	2	1	0
RFU, set to zero.							Caller Id

Field	Description
Caller Id	A protocol which defines the encoding of specific information to allow the answering station to identify the calling station. When the field is set (1) caller id decoding services are provided. When the field is reset (0) caller id decoding services are not provided.

***TPLFE_CD*: ITU-T(CCITT) Country Code**

7	6	5	4	3	2	1	0
country code							

Field	Description
country code	ITU-T(CCITT) numeric code assigned to the country targeted for distribution of this modem (Annex A, Recommendation T.35). A value of 0··254 is the assigned ITU-T(CCITT) country code. A value of 255 indicates the end of the variable length country code list if it does not extend to the end of the tuple. The fields which follow this field when a value of 255 is present represent manufacturer specific capabilities.

3.2.6.1.4 Function Extension Tuple for Document Facsimile

The Document Facsimile function extension tuple describes the capabilities of the facsimile modem. This includes the highest possible data rate in the DTE to UART interface of the facsimile modem, the ITU-T(CCITT) modulation standards supported, the ITU-T(CCITT) defined country code of the target market, the standardized data encryption methods supported, and the various document facsimile features supported.

A separate tuple shall be used to describe the capabilities of each supported Service Class. Thus a facsimile modem supporting the EIA/TIA-578 Service Class 1 standard and the draft standard Service Class 2 must include two separate extension tuples. The PC Card Subfunction Descriptor represented in the least significant nibble of the *TPLFE_TYPE* field would vary with each tuple.

The document facsimile services of Error Correction Mode, Voice Request, Polling, File Transfer, Password, Selective Polling, Character Mode, and Copy Quality were not included in this tuple. Many have not yet achieved wide spread use and are still in the midst of the ITU-T(CCITT) approval process. Others require control and configuration through the use of manufacturer specific AT commands and are consequently not within the scope of this document.

The ITU-T(CCITT) country code field defines the country targeted for distribution of the facsimile modem. This field has been specified to support a maximum value of 254 (255 or FFH is reserved as a country code list terminator, see below) in accordance with the ITU-T(CCITT) standard T.35 which has assigned each member country or area a unique 8 bit value. Please refer to T.35 Annex A for a list of country or area codes currently assigned.

To specify support of multiple countries, the country code field *TPLFE_CF* is positioned as the last field in the Document Facsimile function extension tuple. Additional country codes may be specified by increasing the value of the *TPL_LINK* field by one for each additional country specified. A value of FFH indicates the end of the country code list if it does not extend to the end of the tuple. Any additional fields which follow this value must be represented by an appropriate increase in the value of the *TPL_LINK* field and are considered a description of manufacturer specific capabilities.

Table 3-19 CISTPL_FUNCE: TPCID_FF: Document Facsimile Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (8 minimum)						
2	TPLFE_TYPE	Type of extended facsimile features (13H, 23H or 33H, see Serial Port Extension Tuple Type Codes)						
3··4	TPLFE_UF	The highest possible data rate in the DTE to UART interface.						
5	TPLFE_FM	The supported ITU-T(CCITT) modulation standards						
6	TPLFE_FY	Standardized data encryption						
7··8	TPLFE_FS	The document facsimile feature selection						
9··n	TPLFE_CF	The ITU-T(CCITT) defined country code (ITU-T(CCITT) T.35/T.35 Annex A)						

***TPLFE_TYPE*: Extended Facsimile Features Type**

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				PC Card Subfunction Id			

Field	Description
PC Card Subfunction Id	A value of 03 indicates that document facsimile services are described in this tuple.
PC Card Subfunction Descriptor	Indicates the EIA/TIA Service Class described in this tuple. Legal values are 01··03, i.e., a value of 1 indicates support of EIA/TIA-578 Service Class 1. A separate extension tuple is required for each Service Class described.

***TPLFE_UF*: UART Interface Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					DTE to UART maximum data rate — MSB		
DTE to UART maximum data rate — LSB							

Field	Description
DTE to UART maximum data rate	The highest bit rate supported for communications with the DTE (v) in increments of 75 bps, where the field value is computed by v/75. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600H (1536).

***TPLFE_FM*: ITU-T(CCITT) Modulation Standards**

7	6	5	4	3	2	1	0
RFU, set to zero.		V.34	V.33	V.17	V.29	V.27ter	V.21-C2

Field	Description
V.21-C2	300 bps duplex modem standardized for universal use in the GSTN
V.27ter	4800/2400 bps modem standardized for use in the GSTN
V.29	9600/7200/4800 bps modem standardized for use on point-to-point 4-wire leased telephone-type circuits
V.17	14.4-K/12000/9600/7200 bps modem using trellis-coded modulation (TCM) standardized for use in the GSTN
V.33	14.4-K/12000/9600/7200 bps modem using trellis-coded modulation (TCM) standardized for use on point-to-point 4-wire leased telephone-type circuits
V.34	A modem operating at data signaling rates of up to 33,600 bps for use on the GSTN.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_FY*: Standardized Data Encryption**

7	6	5	4	3	2	1	0
RFU, set to zero.							

***TPLFE_FS*: Document Facsimile Feature Selection**

7	6	5	4	3	2	1	0
password	file transfer	polling	voice request	error correct mode	T.6	T.4	T.3
RFU, set to zero.							

Field	Description
T.3	Standardization of group 2 facsimile apparatus for document transmission.
T.4	Facsimile coding schemes and coding control functions for group 3 facsimile apparatus.
T.6	Facsimile coding schemes and coding control functions for group 4 facsimile apparatus.
error correct mode	A mode of operation allowing error correction of document facsimile page data.
voice request	Interruption of the transmission of a facsimile image to allow voice contact with the remote station.
polling	A request to receive or send a facsimile regardless of which station is considered the originator.
file transfer	ITU-T(CCITT) defined procedure for transferring files between PC based facsimile DCE 's.
password	A mode of operation allowing secured facsimile transmissions.

When the field is set (1) the feature is supported in the specified Service Class, when the field is reset (0) the feature is not supported the specified Service Class.

***TPLFE_CF*: ITU-T(CCITT) Country Code**

7	6	5	4	3	2	1	0
country code							

Field	Description
country code	ITU-T(CCITT) numeric code assigned to the country targeted for distribution of this modem (Annex A, Recommendation T.35). A value of 0··254 is the assigned ITU-T(CCITT) country code. A value of 255 indicates the end of the variable length country code list if it does not extend to the end of the tuple. The fields which follow this field when a value of 255 is present represent manufacturer specific capabilities.

3.2.6.1.5 Function Extension Tuple for Voice Function

This structure describes the capabilities of a modem equipped to process digitally encoded voice data. The highest possible data rate in the DTE to UART interface of the voice modem is described. The PC Card Subfunction Descriptor specifies the number of the Service Class used to activate the voice command mode.

The sample rate, size, and compression methods supported are represented in a series of three separate tuples. Each tuple is repeated for the supported sample rates, sizes, and compression methods. These fields serve only to described the range of supported options not, however, the various combinations of each. As extended voice AT commands are defined by the EIA/TIA and the operation of a voice equipped modem is standardized this tuple will be appropriately updated.

Table 3-20 CISTPL_FUNCE: Voice Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (06H minimum)						
2	TPLFE_TYPE	Type of extended data (84H assuming Service Class 8, see Serial Port Extension Tuple Type Codes)						
3··4	TPLFE_UV	The highest possible data rate in the DTE to UART interface.						
5··	TPLFE_SR	A variable length list of the sample rates supported.						
··	TPLFE_SS	A variable length list of the sample sizes supported.						
··n	TPLFE_SC	A variable length list of the EIA/TIA Compression Method Identifiers. It is not yet certain if the EIA/TIA will maintain the assignment of these numeric id's. Consequently, until the outcome is determined, if manufacturer specific data is present at the end of the tuple, this field shall remain set to zero.						

***TPLFE_TYPE*: Type of CISTPL_FUNCE Extended Data**

7	6	5	4	3	2	1	0
PC Card Subfunction Descriptor				PC Card Subfunction Id			

Field	Description
PC Card Subfunction Id	A value of 04 indicates the presence of PCM voice encoding capabilities. See Serial Port Extension Tuple Type Codes for a list of all Subfunction Codes defined.
PC Card Subfunction Descriptor	The numeric value of the Service Class reserved for the definition of the extended voice AT commands (currently being standardized). Defined values are in the range 4··15. The actual value used by the manufacturer shall be indicated in this field. A value of zero (0) indicates that voice encoding services are provided using a protocol other than that defined in the applicable Service Class.

***TPLFE_UV*: UART Interface Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					DTE to UART maximum data rate — MSB		
DTE to UART maximum data rate — LSB							

Field	Description
DTE to UART maximum data rate	The highest bit rate supported for communications with the DTE (v) in increments of 75 bps, where the field value is computed by v/75. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600H (1536).

***TPLFE_SR*: Sample Rate Field**

7	6	5	4	3	2	1	0
RFU, set to zero.	Sample Rate in KHz						
RFU, set to zero.	Sample Rate in KHz/100						

Field	Description
Sample Rate in KHz	The voice sampling rate supported in increments of a 1000 Hz. Legal values are in the range 1··99. A value of 0BH (11) indicates an 11 KHz sample rate. A value of zero (0) indicates the end of the variable length list of supported sample rates. The next byte represents the beginning of the variable length list of supported sample sizes.
Sample Rate in KHz/100	The voice sampling rate supported in units of 1/100 KHz. The fractional component of 11.025 KHz is represented in this field as 02H. The final sample rate is derived by adding the fractional component in this field to the whole number value in the field *Sample Rate in KHz*. A value of 11.025 KHz would be derived by adding 0BH KHz or 11 KHz to 02H/100 KHz or 0.02 KHz for a value of 11.02 KHz in four significant digits. Legal values are in the range 1··99.

***TPLFE_SS*: Sample Size Field**

7	6	5	4	3	2	1	0
RFU, set to zero.				Sample Size in Bits			
RFU, set to zero.				Sample Size in Bits/10			

Field	Description
Sample Size in Bits	The sample size supported in units of 1 bit. Legal values are in the range 1··15. The integer portion of 2.66 bits is represented in this field as 02H. A value of zero (0) indicates the end of the variable length list of supported sample sizes. The next byte represents the beginning of the variable length list of supported data compression methods.
Sample Size in Bits/10	The sample size supported in units of 1/10 bits. Values in the range 0··9 indicate the fractional component of the supported sample size in units of 1/10 bits. The final sample size is derived by adding the fractional component in this field to the whole number value in the field *Sample Size in Bits*. A sample size of 2.66 bits would be derived by adding 02H or 2 bits from *Sample Size in Bits* to 06H or 0.6 bits from *Sample Size in Bits/10* for a value of 2.6 bits in 2 significant digits.

***TPLFE_SC*: Voice Compression Methods**

7	6	5	4	3	2	1	0
Voice Compression Method Id (RFU, set to zero.)							

Voice Compression Method Id

Value	Description
1··255	Indicates the EIA/TIA assigned Compression Method Identifier associated with the supported compression method.
0	Indicates the end of the variable length list of supported Compression Method Identifiers. The next byte represents the beginning of fields containing manufacturer specific data. This byte need not be present if the list extends to the end of the tuple.

Assuming the EIA/TIA presides over the assignment of id's to the many compression methods currently available, this field represents the EIA/TIA assigned Compression Method Identifier.

3.2.6.1.6 Function Extension Tuple for ISDN Terminal Adapter

This structure describes the capabilities of ISDN Terminal Adapter. This includes the highest possible data rate in the DTE to UART interface of ISDN Terminal Adapter, Physical Layer protocols supported, Data Link Layer protocols supported, Network Layer protocols supported, Rate Adaption protocol supported, other services supported, the command set in use for DTE to DCE control and configuration(e.g.AT command set), the ITU-T(CCITT) standard defined country code of the target market. The escape codes supported for return to command mode are also described.

The ITU-T(CCITT) country code field defines the countries targeted for distribution of ISDN Terminal Adapter. This field has been specified to support a maximum value of 254 (255 or FFH is reserved as a country code list terminator, see below) in accordance with the ITU-T(CCITT) standard T.35 which has assigned each member country or area a unique 8 bit value. Please refer to T.35 Annex A for a list of country or area codes currently assigned.

To specify support of multiple countries, the country code field *TPLFE_CD* was positioned as the last field in ISDN Terminal Adapter function extension tuple. Additional country codes may be specified by increasing the value of the *TPL_LINK* field by one for each additional country specified. A value of FFH is used to indicate the end of the country code list if it does not extend to the end of the tuple. Any additional fields which follow this value must be represented by an appropriate increase in the value of the *TPL_LINK* field and are considered a description of manufacturer specific capabilities.

Table 3-21: CISTPL_FUNCE: ISDN Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (10H minimum)							
2	TPLFE_TYPE Type of extended data (0BH, see Serial Port Extension Tuple Type Codes)							
3	TPLFE_LAYER1 Physical layer protocols supported							
4	TPLFE_LAYER2 Data Link layer protocols supported							
5	TPLFE_LAYER3 Network layer protocols supported							
6	TPLFE_RA Rate adoption protocols supported							
7	TPLFE_SP Speech Standards Field							
8	TPLFE_DT Data Transfer Standards Field							
9	TPLFE_AD Audio Standards Field							
10	TPLFE_VD Video Standards Field							
11	TPLFE_FE Miscellaneous Feature Selection							
12..13	TPLFE_UD The highest possible data rate in the DTE to UART interface.							
14	TPLFE_DC Data compression protocols supported							
15	TPLFE_CM Command protocols supported							
16	TPLFE_EX Indication of the escape mechanisms supported							
17--n	TPLFE_CD The ITU-T(CCITT) defined country code (ITU-T(CCITT) T.35/T.35 Annex A)							

***TPLFE_LAYER1*: Layer 1 protocols (Physical Layer)**

7	6	5	4	3	2	1	0
RFU, set to zero.				NT1	S/T	I.431	I.430

Field	Description
I.430:	Basic user-network Interface - Layer 1 specification.
I.431:	Primary rate user-network Interface - Layer 1 specification.
S/T	S/T(Local Bus/Terminal) Interface Connector.
NT1	NT1(Network Termination 1).

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_LAYER2*: Layer 2 protocols (Data Link Layer)**

7	6	5	4	3	2	1	0
RFU, set to zero.						Q.922	Q.921

Field	Description
Q.921:	ISDN user-network interface - Data link layer specification.
Q.922:	ISDN Data Link Layer specification for Frame mode Bearer services.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_LAYER3*: Layer 3 protocols (Network Layer)**

7	6	5	4	3	2	1	0
RFU, set to zero.						Q.933	Q.931

Field	Description
Q.931:	ISDN user-network interface layer 3 specification for basic call control.
Q.933:	ISDN Signalling specification for Frame mode basic call control.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_RA*: Rate Adaption protocols**

7	6	5	4	3	2	1	0
RFU, set to zero.						I.465	I.463

Field	Description
I.463:	Support of data terminal equipments(DTEs) with V-series type interfaces by an ISDN. (V.110)
I.465:	Support by an ISDN of data terminal equipment with V-series type interface with provision for statistical multiplexing. (V.120)

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_SP*: Speech Standards Field**

7	6	5	4	3	2	1	0
RFU, set to zero.						G.721	G.711

Field	Description
G.711:	Pulse mode modulation(PCM) of voice frequencies.
G.721:	32Kbit/s adaptive differcential pulse code modulation (ADPCM).

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_DT*: Data Transfer Standards Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					MLPPP	PPP	X.75

Field	Description
X.75:	Packet-switched signaling system between public networks providing data services.
PPP:	Piont-to-Point Protocol.
MLPPP	MultiLink Piont-to-Point Protocol.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_AD*: Audio Standards Field**

7	6	5	4	3	2	1	0
RFU, set to zero.						G.725	G.722

Field	Description
G.722:	7 kHz audio-coding within 64 kbit/s.
G.725:	System aspects for the use of 7 kHz audio codec within 64 kbit/s.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_VD*: Video Standards Field**

7	6	5	4	3	2	1	0
RFU, set to zero.						H.320	H.261

Field	Description
H.261:	Codec for audiovisual services at n x 384 kbit/s.
H.320:	Narrow band visual telephone systems and terminal equipment.

When the field is set (1) the specified standard is supported, when the field is reset (0) the specified standard is not supported.

***TPLFE_EF*: Miscellaneous Feature Selection**

7	6	5	4	3	2	1	0
RFU, set to zero.	Aux. Analog Port.				G4 FAX	Circuit mode	Packet mode

Field	Description		
Packet mode	Packet mode		
Circuit mode	Circuit mode		
G4 FAX	Group 4 Facsimile service		
Aux. Analog Port.	Number of supported Analog Ports.		
	0 : no ports supported	6 : six ports supported	12 : twelve ports supported.
	1 : one port supported	7 : seven ports supported	13 : thirteen ports supported
	2 : two ports supported	8 : eight ports supported	14 : fourteen ports supported
	3 : three ports supported	9 : nine ports supported	15 : fifteen ports supported
	4 : four ports supported	10 : ten ports supported	
	5 : five ports supported	11 : eleven ports supported	

When the field is set (1) the specified command protocol is fully supported, when the field is reset (0) the specified command protocol is not fully supported.

***TPLFE_UD*: UART Interface Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					DTE to UART maximum data rate — MSB		
DTE to UART maximum data rate — LSB							

DTE to UART maximum data rate

Code	Description
v/75	Indicates the highest DTE to UART bit rate supported. To compute the actual bit rate, the value of this field (represented in the formula as n) is multiplied by 75 (n * 75). A bit rate of 115,200 would be represented by a field value of 600H (1536).

***TPLFE_DC*: Data Compression Protocols**

7	6	5	4	3	2	1	0
RFU, set to zero.							V.42bis

Field	Description
V.42bis	Data compression protocol defined by ITU-T(CCITT),

When the field is set (1) the specified compression protocol is supported, when the field is reset (0) the specified compression protocol is not supported.

***TPLFE_CM*: Command Protocols**

7	6	5	4	3	2	1	0
RFU, set to zero.	DMCL	V.25A	V.25bis	RFU, set to zero.	AT3	AT2	AT1

Field	Description
AT1	The “Action” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
AT2	The “ACE/DTE Interface Parameters” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
AT3	The “ACE Parameters” AT command group as defined in the Automatic Calling Equipment (ACE) command standard ANSI/EIA/TIA 602.
V.25bis	A specification of the formatting and response of DCE commands used in automatic calling procedures over the 100-series interchange circuits.
V.25A	A specification on test facilities relating to the implementation of the V.25bis automatic calling procedure.
DMCL	A command mode convention available for DTE to DCE control and configuration.

When the field is set (1) the specified command protocol is fully supported, when the field is reset (0) the specified command protocol is not fully supported.

***TPLFE_EX*: Escape Mechanisms Field**

7	6	5	4	3	2	1	0
RFU, set to zero.					User Defined Escape Char.	+++	BREAK

Field	Description
BREAK	The standardized BREAK represented by a sequence of 0 bits of a specified length.
+++	A sequence of characters used to return the modem to command mode.
User Defined Escape Char.	Indicates support for a user defined escape character.

When the field is set (1) the specified escape mechanism is supported, when the field is reset (0) the specified escape mechanism is not available to return the modem to command mode.

***TPLFE_CD*: ITU-T(CCITT) Country Code**

7	6	5	4	3	2	1	0
country code							

Field	Description
country code	ITU-T(CCITT) numeric code assigned to the country targeted for distribution of this modem (Annex A, Recommendation T.35). A value of 0··254 is the assigned ITU-T(CCITT) country code. A value of 255 indicates the end of the variable length country code list if it does not extend to the end of the tuple. The fields which follow this field when a value of 255 is present represent manufacturer specific capabilities.

3.2.6.2 Function Extension Tuple for Disk

See also the ***PC Card ATA Specification*** for Disk Device function extension tuples. Additional function extension tuples may be added in the future for PC Card ATA and other disk interfaces.

Table 3-22 CISTPL_FUNCE: Disk Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (02H minimum)						
2	TPLFE_TYPE	Extension type: Disk Device Interface tuple (01H)						
3	TPLFE_DATA	Interface type: PC Card-ATA Interface (01H)						

Table 3-23 CISTPL_FUNCE: Combined PC Card ATA Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (03H minimum)						
2	TPLFE_TYPE	Extension type: Basic PC Card ATA Interface tuple (02H)						
3	TPLFE_DATA							
	RFU	RFU	RFU	D	U	S	V	
4	TPLFE_DATA							
	RFU	I	E	N	P3	P2	P1	P0

Data Fields for PC Card ATA Function Extension Tuple

Field	Description
V	**V$_{PP}$[2::1]** Power 0 Not Required 1 Required for Media Modification Accesses 2 Required for all Media Accesses 3 Required Continuously
S	Silicon 0 Rotating Device 1 Silicon Device
U	Unique Drive Identifier 0 Identify Drive Model / Serial Number may not be unique 1 Identify Drive Model / Serial Number is guaranteed unique
D	Dual Drive 0 single drive on card. 1 two drives on card.
P0	Sleep — Low Power Mode 0 Sleep Mode Not Supported 1 Sleep Mode Supported
P1	Standby — Low Power Mode 0 Standby Mode Not Supported 1 Standby Mode Supported
P2	Idle — Low Power Mode 0 Idle Mode Not Supported 1 Idle Mode Supported
P3	Auto — Low Power Mode 0 Low Power Mode Use Required to Minimize Power 1 Drive Automatically Minimizes Power. No need for host to actively power manage.
N	3F7/377 Register Inhibit Available 0 All Primary and Secondary I/O Addressing Modes include ports 3F7$_H$ or 377$_H$. 1 Some Primary or Secondary I/O Addressing Modes exclude 3F7$_H$ and / or 377$_H$ for floppy interference avoidance.
E	Index Emulated 0 Index Bit is Not Emulated 1 Index Bit is Supported or Emulated
I	**IOIS16#** on Twin Card 0 **IOIS16#** use is Unspecified on Twin-Card Configurations 1 **IOIS16#** is asserted only for Data Register on Twin-Card Configurations
RFU	Reserved (set to zero).

Table 3-24 CISTPL_FUNCE: Dual-Drive Card PC Card ATA Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK	Link to next tuple (03H minimum)						
2	TPLFE_TYPE	Extension type: Basic PC Card ATA Interface tuple (02H for first drive and 03H for additional drive on card)						
3	TPLFE_DATA							
	RFU	RFU	RFU	D	U	S	V	
4	TPLFE_DATA							
	RFU	I	E	N	P3	P2	P1	P0

3.2.6.3 Function Extension Tuple for LAN

The tuples defined here contain information about the operational features of LAN cards. LAN cards are identified by a function ID of LAN and are further defined using function extension tuples to describe specific capabilities. Function extension tuples are optional for LAN cards. The general form of the LAN function extension tuple is:

Table 3-25 CISTPL_FUNCE: LAN Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FUNCE (22H)						
1	TPL_LINK Link to next tuple (m-1 minimum)							
2	TPLFE_TYPE	LAN type / feature code						
3··m	TPLFE_DATA	Content and format-dependent on LAN_FEATURE						

Field LAN_FEATURE is a code indicating the feature being described. The structure and content of the DATA field varies according to LAN_FEATURE. A CIS may contain zero, one or more of these tuples. If the LAN card doesn't support a feature it will lack the corresponding Extension Tuple. If it supports more than one variation of a particular feature, it can have multiple extension tuples with the same LAN_FEATURE value, each describing a different variant of the same feature. Defined LAN_FEATURE values are:

***TPLFE_TYPE*: Values for LAN Functions**

Value	Description	
1	LAN_TECH	Technology type, e.g. ethernet, token ring, etc.
2	LAN_SPEED	Raw bit rate
3	LAN_MEDIA	Cable or transmission media
4	LAN_NID	Node identification
5	LAN_CONN	Connector standard

3.2.6.3.1 LAN_TECH Function Extension Tuple

Table 3-26 CISTPL_FUNCE: LAN_TECH Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (02H minimum)							
2	TPLFE_TYPE LAN_TECH (01H)							
3	LAN_TECH_CODE Network technology code							

LAN_TECH_CODE **Values**

Value	
1	Arcnet
2	Ethernet
3	Token Ring
4	Local Talk
5	FDDI/CDDI
6	ATM
7	Wireless

3.2.6.3.2 LAN_SPEED Function Extension Tuple

Table 3-27 CISTPL_FUNCE: LAN_SPEED Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (02H minimum)							
2	TPLFE_TYPE LAN_SPEED (02H)							
3	LAN_SPEED_VALUE Raw bit rate							

LAN_SPEED_VALUE is a 32 bit integer containing the raw network bit rate.

3.2.6.3.3 LAN_MEDIA Function Extension Tuple

Table 3-28 CISTPL_FUNCE: LAN_MEDIA Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (02H minimum)							
2	TPLFE_TYPE LAN_TECH (03H)							
3	LAN_MEDIA_CODE Cable or transmission media							

***LAN_MEDIA_CODE* Values**

Value	Description
1	Unshielded twisted pair
2	Shielded twisted pair
3	Thin coax
4	Thick coax
5	Fiber
6	Spread spectrum radio 902··928 MHz
7	Spread spectrum radio 2.4 GHz
8	Spread spectrum radio 5.4 GHz
9	Diffuse infra red
10	Point to point infra red

3.2.6.3.4 LAN_NID Function Extension Tuple

Table 3-29 CISTPL_FUNCE: LAN_NID Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (02H minimum)							
2	TPLFE_TYPE LAN_NID (04H)							
3	LAN_NID_SZ Number of bytes in LAN node ID							
4··m	LAN_NID_CODE Binary value of LAN node ID							

3.2.6.3.5 LAN_CONN Function Extension Tuple

Table 3-30 CISTPL_FUNCE: LAN_CONN Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (02H minimum)							
2	TPLFE_TYPE LAN_CONN (05H)							
3	LAN_CONN_CODE Connector standard compliance code							

***LAN_CONN_CODE* Values**

Value	Description
0	Open connector standard
1	Closed connector standard

3.2.6.4 Function Extension Tuple for ISDN

The tuples defined here contain information about the operational features of ISDN cards. ISDN cards are identified by a function ID of ISDN and are further defined using function extension tuples to describe specific capabilities. Function extension tuples are optional for ISDN cards. The general form of the ISDN function extension tuple is:

Table 3-31: CISTPL_FUNCE: ISDN Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (m-1 minimum)							
2	TPLFE_TYPE ISDN type / feature code							
3··m	TPLFE_DATA Content and format-dependent on ISDN FEATURE							

Field ISDN_FEATURE is a code indicating the feature being described. The structure and content of the DATA field varies according to ISDN_FEATURE. A CIS may contain zero, one or more of these tuples. If the ISDN card doesn't support a feature it will lack the corresponding Extension Tuple. If it supports more than one variation of a particular feature, it can have multiple extension tuples with the same ISDN_FEATURE value, each describing a different variant of the same feature. Defined ISDN_FEATURE values are:

***TPLFE_TYPE*: Values for ISDN Functions**

Value	Description
81H	ISDN_TECH Technology type

3.2.6.4.1 ISDN_TECH Function Extension Tuple

Function extension tuples are the same as for a serial type ISDN card. See ***Table 3-21: CISTPL_FUNCE: ISDN Function Extension Tuple***.

Note The following fields on ***Table 3-21: CISTPL_FUNCE: ISDN Function Extension Tuple*** are not used for a network adapter type ISDN card and all fields should be set to zero:

TPLFE_UD, TPLFE_DC, TPLFE_CM, TPLFE_EX

3.2.6.5 Function Extension Tuple for Serial I/O Bus Adapter

The tuples defined here contain information about the operational features of Serial I/O Bus Adapter. Serial I/O Bus Adapters are identified by a function ID of 0BH in TPLFID_FUNCTION code and are further defined using function extension tuples to describe specific capabilities. The general form of the Serial I/O Bus Adapter function extension tuple is:

Table 3-32: CISTPL_FUNCE: 1394 Serial I/O Bus Adapter Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (05)							
2	TPLFE_TYPE 1394_TYPE (01H)							
3	TPLFE_ADAPTER 1394 Host Adapter type							
4	TPLFE_POWER Power management data							
5..6	TPLFE_SPEED Bus speed, Max raw bit rate							

TPLFE_SPEED is a 16 bit integer containing the raw bus Mb/second rate.

The *TPLFE_ADAPTER* field identifies the type of IEEE-1394 host adapter implemented in the PC Card. This data is used by the host to select and load the appropriate driver.

The defined *TPLFFE_ADAPTER* codes are listed below

***TPLFE_ADAPTER* Values**

Value	Description
00	IEEE-1394/OHCI
01	IEEE-1394/Pele
02-FFH	Reserved

The *TPLFE_POWER* field describes the bus power requirements of the IEEE-1394 device. The device may either source or sink power. The POWER_CLASS values associated with each code can be found in the ***IEEE 1394-1995 Specification*** in Section ***4.3.4.1 (Self-ID packet)***.

***TPLFE_POWER* Values**

Value	Description
0-7 (bit 0-2)	Power class code of 1394
(bit 3-7)	Reserved (0)

Table 3-33: CISTPL_FUNCE: USB Serial I/O Bus Adapter Function Extension Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_FUNCE (22H)							
1	TPL_LINK Link to next tuple (05)							
2	TPLFE_TYPE USB_TYPE (02H)							
3	TPLFE_ADAPTER USB Host Adapter type							
4	TPLFE_POWER Power management data							
5..6	TPLFE_SPEED Bus speed, Max raw bit rate							

TPLFE_SPEED is a 16 bit integer containing the raw bus Mb/second rate.

The *TPLFE_ADAPTER* field identifies the type of USB host adapter implemented in the PC Card. This data is used by the host to select and load the appropriate driver.

The defined *TPLFFE_ADAPTER* codes are listed below

***TPLFE_ADAPTER* Values**

Value	Description
00	USB/OHCI
01	USB/Universal
02-FFH	Reserved

The *TPLFE_POWER* field describes the power source ability of the USB device. A Self Powered device will neither require nor supply power on the USB bus. A Bus Powered device will draw it's power from the USB bus within the constraints of the USB specification. Both of these power codes are valid for USB functions only, not for a USB root or hub. A Low Power device can be a root device or a hub. A Low Power device is capable of supplying only 100 mA of current per port. A High Power device is either a root or hub capable of supplying up to 500 mA per port

The defined *TPLFFE_POWER* codes are listed below

***TPLFE_POWER* Values**

Value	Description
0	Self Powered
1	Bus Powered
2	USB low power source (100 mA)
3	USB high power source (500 mA)
4-FFH	Reserved

3.2.7 CISTPL_FUNCID: Function Identification Tuple

The function identification tuple contains information about the functionality provided by a PC Card. Information is also provided to enable system utilities to decide if the PC Card should be configured during system initialization. If additional function specific information is available, one or more function extension tuples follow this tuple.

Multiple Function 16-bit PC Cards primary CIS may include a CISTPL_FUNCID tuple in each function-specific secondary CIS. The *TPLFID_FUNCTION* code of zero (0) for multi-function is only used for 16-bit PC Cards with more than one function that do not follow this standard. In other words, a *TPLFID_FUNCTION* code of zero (0) identifies a 16-bit PC Card that provides multiple functions using a vendor specific implementation.

CardBus PC Cards have one to eight functions, each function is uniquely identified and contains a separate CIS. Each CardBus PC Card function CIS shall contain a CISTPL_FUNCID tuple describing that function.

Table 3-34 CISTPL_FUNCID: Function Identification Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_FUNCID (21H)					
1	TPL_LINK		Link to next tuple (at least 2)					
2	TPLFID_FUNCTION		IC Card function code					
3	TPLFID_SYSINIT		System initialization bit mask					

The *TPLFID_FUNCTION* field identifies the function provided by the PC Card.

As noted above, additional information about a specific function may be available in function extension tuples. These extension tuples follow their related function identification tuple.

The defined *TPLFID_FUNCTION* codes are listed below.

***TPLFID_FUNCTION* field codes**

Code	Name	Meaning
0	Multi-Function	16-bit PC Card using a vendor specific implementation with more than one function and only one (1) set of configuration registers
1	Memory	Memory Card (RAM, ROM, EPROM, flash, etc.)
2	Serial Port	Serial I/O port, includes modem cards
3	Parallel Port	Parallel printer port, may be bi-directional
4	Fixed Disk	Fixed drive, may be silicon may be removable
5	Video Adapter	Video interface, extension tuples identify type and resolutions
6	Network Adapter	Network adapter
7	AIMS	Auto Incrementing Mass Storage card
8	SCSI	SCSI bridge
9	Security	Security services, extension tuples identify type and capabilities.
AH	Instrument	Instrumentation Cards
BH	Serial I/O Bus Adapter	High speed Serial Bus Adapter (e.g. IEEE-1394, USB)
0CH··FDH	Reserved	Unused in this release. Reserved for future use
FEH	Vendor-Specific	Unique to specific vendor
FFH	Do Not Use	Invalid code.

The *TPLFID_SYSINIT* is a bit-mapped field. It allows PC Cards with standard system resources to be installed during system initialization. The bit definitions are described in the following table: the *TPLFID_SYSINIT* field.

***TPLFID_SYSINIT* field**

7	6	5	4	3	2	1	0
RFU, must be zero (0).						ROM	POST

The *POST* bit indicates that the Power-On Self Test routines may attempt to configure the PC Card during system initialization.

The *ROM* bit indicates the PC Card contains a system expansion ROM which may be installed during system initialization. Information describing the ROM to be installed is in the *TPCE_FS* Feature Selection Byte and related Memory Space Description Structure in a 16-bit PC Card CISTPL_CFTABLE_ENTRY tuple or CardBus PC Card CISTPL_CFTABLE_ENTRY_CB tuple.

Only the specific function requiring POST or BOOT processing should have the appropriate bits set in the System Initialization Byte.

3.2.8 CISTPL_JEDEC_C, CISTPL_JEDEC_A: The JEDEC Identifier Tuples

This optional tuple is provided for cards containing programmable devices. It provides an array of *k* entries, where *k* is the number of distinct entries in the device information tuple (codes 01H or 17H). There is a one-to-one correspondence between JEDEC identifier entries in this tuple and device information entries in the device information tuple.

Table 3-35 CISTPL_JEDEC_C and CISTPL_JEDEC _A: JEDEC Identifier Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_JEDEC_C (18H) and CISTPL_JEDEC _A (19H).						
1	TPL_LINK	Link to next tuple (at least m-1).						
2··3		JEDEC identifier for first device info entry.						
4··5		JEDEC identifier for second device info entry (if needed).						
6··m		JEDEC identifiers for remaining device info entries (if needed).						

The *TPL_CODE* field indicates to which device information tuple(s) the JEDEC identifier tuple corresponds. If the value is 18H. CISTPL_JEDEC_C, the tuple corresponds to the CISTPL_DEVICE or CSTPL_DEVICE_OC tuples. CardBus PC Cards use CISTPL_JEDEC_C tuples to indicate JEDEC identifiers which correspond to CISTPL_DEVICE_OC tuples. If the value is 19H, CISTPL_JEDEC_A, the tuple corresponds to the CISTPL_DEVICE_A or CISTPL_DEVICE_OA tuples (16-bit PC Cards only).

JEDEC identifiers consist of two bytes. The first byte is the device manufacturer ID, as assigned by JEDEC committee JC-42.4. This byte, if valid, must always have an odd number of bits set true. The MSB (bit 7) of the byte is used as a parity bit to ensure that this constraint is met. The values of 0 and FFH are illegal JEDEC identifiers (due to their even parity). The value 0H indicates "no JEDEC identifier for this device."

The second byte contains manufacturer specific information representing device type, programming algorithm, and the like. If the manufacturer ID is 00H, then this byte is reserved and should be set to zero.

JEDEC identifiers will only be provided for device info entries that indicate some kind of programmable device. For all other entries, the corresponding JEDEC identifier field shall be absent, i.e., set to 00H.

Examples:

If a 16-bit PC Card consists of four "Company X" 27C512 devices, whose JEDEC identifier is (hypothetically) manufacturer: 40H, device ID: 15H, then the header block might be laid out as follows:

```
/byte[0]:      (CISTPL_DEVICE,
/byte[1]:      /*link*/         3,
/byte[2]:      /*type/speed*/    DTYPE_OTPROM | DSPEED_250NS,
/byte[3]:      /*size: units/code*/    (1<<3) | 4)
/byte[4]:      /*end of tuple*/        FFh
         );

/byte[5]:      (CISTPL_JEDEC_C,        /*18H*/
/byte[6]:      /*link*/                0FFh, /*(end of chain)*/
/byte[7]:      /*manufacturer ID*/     40h,
/byte[8]:      /*mfr's info*/          15h,
         );
```

Note: In the above example, the size byte indicates that 2 x 128K bytes are available, for a total of 256K. Programming software would use the JEDEC information to determine that in fact 4, 64K x 8 devices are present. Since 16-bit PC Cards are always 16 bits wide, programming software could therefore deduce the organization of such a card.

The following tuples might be used to describe a card with 256K of OTPROM and 16 KBytes of RAM. In this example, RAM occupies locations [0 - 03FFFH], and ROM occupies locations [20000H - 5FFFFH] (for ease of decoding).

```
/byte[0]:      (CISTPL_DEVICE,
/byte[1]:      /*link*/         7,
/byte[2]:      /*type/speed*/    DTYPE_SRAM | DSPEED_100NS,
/byte[3]:      /*size: units/code*/    ((1<<3) | 2)
/byte[4]:      /*type/speed*/    DTYPE_NULL | DSPEED_NONE,
/byte[5]:      /*size: units/code*/    ((0Dh<<3) | 2)
/byte[6]:      /*type/speed*/    DTYPE_OTPROM | DSPEED_250NS,
/byte[7]:      /*size: units/code*/    ((1<<3) | 4)
/byte[8]:      /*end of tuple*/        FFh
         );

/byte[9]:      (CISTPL_JEDEC_C,
/byte[10]:     /*link*/                0FFh, /*(end of chain)*/
/byte[11]:     /*RAM: no code*/        0,
/byte[12]:     /*no info*/             0,
/byte[13]:     /*hole: no code*/       0,
/byte[14]:     /*no info*/             0,
/byte[15]:     /*manufacturer ID*/     40h,
/byte[16]:     /*mfr's info*/          15h,
         );
```

3.2.9 CISTPL_MANFID: Manufacturer Identification Tuple

The manufacturer identification tuple contains information about the manufacturer of a PC Card. Two types of information are provided: the PC Card's manufacturer and a manufacturer card number. Only one manufacturer identification tuple may be present in the card information structure.

Table 3-36 CISTPL_MANFID: Manufacturer Identification Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_MANFID (20H)						
1	TPL_LINK	Link to next tuple (at least 4)						
2··3	TPLMID_MANF	PC Card manufacturer code						
4··5	TPLMID_CARD	manufacturer information (Part Number and/or Revision)						

The *TPLMID_MANF* field identifies the PC Card's manufacturer. New codes are assigned by both PCMCIA and JEIDA. The first 256 identifiers (0000H through 00FFH) are reserved for manufacturers who have JEDEC IDs assigned by JEDEC Publication 106. Manufacturers with JEDEC IDs may use their eight bit JEDEC manufacturer code as the least significant eight bits of their PC Card manufacturer code. In this case, the most significant eight bits must be zero (0). For example, if a JEDEC manufacturer code is 89H, their PC Card manufacturer code is 0089H.

The *TPLMID_CARD* field is reserved for use by the PC Card's manufacturer. It is anticipated that the field will be used to store card identifier and revision information.

3.2.10 CISTPL_VERS_1: The Level 1 Version / Product Information Tuple

This tuple contains Level 1 version compliance and card manufacturer information. It applies to Level 1 tuples only. It should appear only once in each partition descriptor chain.

Table 3-37 CISTPL_VERS_1: Level 1 Version / Product Information Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_VERS_1 (15H).							
1	TPL_LINK Link to next tuple (at least m-1).							
2	TPLLV1_MAJOR Major version number.							
3	TPLLV1_MINOR Minor version number.							
4	TPLLV1_INFO Product information string: name of the manufacturer, terminated by NULL (00H). Name of product, terminated by NULL (00H). Additional product information, in text; terminated by NULL (00H). Suggested use: lot number. Additional product information, in text; terminated by NULL (00H). Suggested use: programming conditions.							
m	FFH: marks end of chain.							

Indicates compliance with the following Standard	***TPLLV1_MAJOR***	***TPLLV1_MINOR***
PC Card Standard, February 1999 (Release 7.0)	7 (07H)	0 (00H)
PC Card Standard, April 1998 (Release 6.1)	6 (06H)	1 (01H)
PC Card Standard, March 1997 (Release 6.0)	6 (06H)	0 (00H)
PC Card Standard, May 1996 (Release 5.2)	5 (05H)	2 (02H)
PC Card Standard, November 1995 (Release 5.1)	5 (05H)	1 (01H)
PC Card Standard, February 1995 (Release 5.0)	5 (05H)	0 (00H)
PCMCIA 2.1 / JEIDA 4.2	4 (04H)	1 (01H)*
PCMCIA 2.0 / JEIDA 4.1	4 (04H)	1 (01H)
PCMCIA 1.0 / JEIDA 4.0	4 (04H)	0 (00H)

NOTE: TPLLV1_MINOR was intentionally set to 01H for PCMCIA 2.1/JEIDA 4.2.

3.3 Configuration Tuples

The Configuration and Configuration-Entry tuples describe the configurable characteristics of 16-bit PC Card Memory Only and I/O cards as well as CardBus PC Cards. The card characteristics described by these tuples include:

- Class of interface (Memory Only, I/O or other);
- Use of Wait, **READY**, Write Protect, BVDs, and Interrupts;
- Card configurations requiring currents in excess of the nominal levels;
- Configurations of **VPP[2::1]** or **V**CC;
- I/O port and memory mapping requirements;
- CardBus PC Card Base Address Register size and characteristics;
- Interrupt levels;
- Unique identification of identical cards.

The Configuration tuple contains a number of fields within it which describe the interface(s) supported by the card, and, when present, the locations of the Card Configuration Registers and the Card Configuration Table.

Specific card configurations and their variations are defined in the Configuration-Entry tuples which make up the Card Configuration Table.

3.3.1 CISTPL_BAR: CardBus PC Card Base Address Register Tuple

The CardBus PC Card-bit Base Address Register tuple describes the size, characteristics and capabilities of the space mapped by a Base Address Register or an Expansion ROM Base Address Register.

Table 3-38 CISTPL_BAR: CardBus PC Card Base Address Register Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE CISTPL_BAR (07H)							
1	TPL_LINK Link to next tuple (at least 6)							
2	TPLBAR_ATTR Base Address Register Indicator and Attributes							
	Below 1 MB	Prefetchable / Cacheable		Address space	RFU (0)	Address Space Indicator		
3	Reserved, must be 0							
4··7	TPLBAR_SIZE Base Address Register Size							

***TPLBAR_ATTR* field**

Field	Description
Address Space Indicator	Same encoding as the *Address Space Indicator* field in the CISTPL_LONGLINK_CB tuple. 0 illegal value - zero indicates CardBus PC Card configuration space. 1 Base Address Register 1. 2 Base Address Register 2. 3 Base Address Register 3. 4 Base Address Register 4. 5 Base Address Register 5. 6 Base Address Register 6. 7 the Expansion ROM Base Address Register.
Address space	0 Base Address Register is of type memory. 1 Base Address Register is of type I/O, *Address Space Indicator allowed values 1*··5.
Prefetchable / Cacheable	When *Address Space* is set (1) this field is reserved and must be zero (0). When *Address Space* is reset (0): 0 neither prefetchable nor cacheable 1 prefetchable but not cacheable 2 both prefetchable and cacheable 3 Reserved value, do not use.
Below 1 MB	When *Address Space* is set (1) this field is reserved and must be zero (0). When Address Space is reset (0): 0 no restriction on mapping location. 1 must locate in the first megabyte of system address space on x86 architecture systems.

The *TPLBAR_SIZE* field indicates the size in bytes of the area mapped by the Base Address Register or Expansion ROM Base Address Register identified by the *Address Space Indicator* field of the *TPLBAR_ATTR* byte. All Base Address Register sizes are powers of two.

3.3.2 CISTPL_CFTABLE_ENTRY: 16-bit PC Card Configuration Table Entry Tuple

The initial portion of each entry of the 16-bit PC Card Configuration Table is given in a compact, standard form. Additional information may be added to the compact information by appending tuple-formatted information (tuple-code, offset to next tuple, and tuple contents) within the interface definition or configuration entry tuples containing the Configuration Table Entry. Subtuple codes 80H··BFH are reserved for vendor-specific configuration information.

The 16-bit PC Card Configuration Table organization for a Memory Only Card and an I/O Card are the same. However, a Memory Only Card should not include I/O specific fields or tuples. Fields related to **VCC**, **VPP[2::1]** and timing apply to all 16-bit PC Cards.

16-bit PC Card Configuration Table Entry tuples are used to specify each possible configuration of a card and to distinguish among the permitted configurations. The Configuration tuple must precede all Configuration Table Entry tuples. The Configuration Table Entry, whose Configuration Table Index matches the last index in the Configuration tuple, must appear after all other Configuration Table Entries.

When the default bit is set in the Configuration Table index byte, that Configuration Table Entry provides default values for entries which follow in the Card Configuration Table.

When the default bit is not set in an entry's Configuration Table index byte, the entry provides default values which apply only to the configuration indicated by the that entry's Configuration Table Index (*TPCE_INDX*) byte.

Values which are not present in a specific entry are taken from the card's most recently scanned default entry, as indicated in the specific fields below.

Table 3-39 CISTPL_CFTABLE_ENTRY: Configuration Table Entry Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_CFTABLE_ENTRY (1BH)						
1	TPL_LINK	Link to next tuple (n-1, [2 minimum])						
2	TPCE_INDX	Configuration Table Index Byte.						
··	TPCE_IF	Interface Description field. This field is present only when the *Intface* field of the Configuration Table Index Byte is set.						
··	TPCE_FS	Feature Selection byte. This field indicates which optional fields are present.						
··	TPCE_PD	Power Description Structure.						
··	TPCE_TD	Configuration Timing Information field.						
··	TPCE_IO	I/O Space description field.						
··	TPCE_IR	Interrupt Request Description structure.						
··	TPCE_MS	Memory Space Description structure.						
··	TPCE_MI	Miscellaneous Features field.						
··n	TPCE_ST	Additional information about the configuration in subtuple format.						

3.3.2.1 TPCE_INDX: The Configuration Table Index Byte

The Configuration Table index byte contains the value to be written to the Card Configuration Option Register in order to enable the configuration described in the tuple. In addition, this byte contains a bit to indicate that the configuration defined in this tuple should be taken as the default conditions for Configuration-Entry tuples that follow.

***TPCE_INDX*: Configuration Table Index Byte**

7	6	5	4	3	2	1	0
Intface	Default	Configuration-Entry-Number					

Field	Description
Configuration-Entry-Number	Contains the value to be written to the Card Configuration Register to enable the configuration described in the tuple.
Default	This bit indicates that this entry provides default values for succeeding entries until another entry is encountered with its Default bit set. 1 The default conditions for following table entries are exactly those conditions specified in this entry. 0 The default conditions are the conditions specified by the last entry encountered with its default bit set.
Intface	0 No interface configuration byte is present following this byte. When the default bit is set in this byte or no Configuration-Entry tuple has been scanned with its default bit set then the standard, Memory Only Interface with **READY**, Write Protect and Battery Voltage Detects are present at pins but not necessarily in a Pin Replacement Register. **WAIT#** is not generated for memory cycles. (TPCE_IF value of 70H) Otherwise, the timing is specified by the most recently scanned Configuration-Entry tuple with its default bit set. 1 An interface configuration byte follows this byte.

3.3.2.2 TPCE_IF: Interface Description Field

The *Interface Description* field indicates the particular card interface which the configuration uses. The interface type (Memory Only or Memory and I/O) is specified. In addition, the card's requirements for system support of the Battery Voltage Detect, Write Protect, **READY** and Wait functions may be indicated.

***TPCE_IF*: Interface Description field**

7	6	5	4	3	2	1	0
M Wait Required	READY Active	WP Active	BVDs Active	Interface Type			

Field	Description
Interface Type	0 Memory 1 I/O and Memory 2 Reserved for future standardization 3 Reserved for future standardization 4 Custom Interface 0 5 Custom Interface 1 6 Custom Interface 2 7 Custom Interface 3 8··15 Reserved for future standardization Interfaces 4 through 7 correspond to interfaces which are defined in CCSTPL_CIF subtuples in the Configuration Tuple. The custom interface number is the relative position of the CCSTPL_CIF subtuple used by this configuration in the set of CCSTPL_CIF subtuples within the Configuration Tuple (CISTPL_CONFIG tuple, TPCC_SBTPL field) for this card.
BVDs Active	**BVD1** and **BVD2** signals are active and should be recovered from the Pin Replacement Register if not available on Pins 62 and 63.
WP Active	Write Protect Status is active and should be recovered from the Pin Replacement Register if not available on Pin 33.
READY Active	**READY** Status Active and should be recovered from the Pin Replacement Register if not available on Pin 16.
M Wait Required	**WAIT#** Signal support required for Memory Cycles. *(This bit should not be set by I/O PC Cards that do not use Common Memory.)*

The status bits *BVDs Active*, *WP Active*, *READY Active*, and *M WAIT Required* are TRUE when set (1).

3.3.2.3 TPCE_FS: Feature Selection Byte

The feature selector byte indicates which compact descriptive fields are present within the table entry. Those fields which are present are indicated by a 1 in the corresponding bit of the selector byte. Within the entry, the fields are ordered in the order of the selector bits starting from bit 0 (LSB) to bit 7 (MSB).

***TPCE_FS*: Feature Selection Byte**

7	6	5	4	3	2	1	0
Misc	Mem Space		IRQ	IO Space	Timing	Power	

Field	Description	
Power	The power supply requirements and load characteristics for this configuration are indicated. There may be 0, 1, 2 or 3 fields following representing **Vcc**, **Vpp[2::1]**, or **Vpp1** and **Vpp2** (individually) in that order. The coding is as follows:	
	0	No power-description structures, use the default.
	1	**Vcc** power-description-structure only.
	2	**Vcc** and **Vpp[2::1]** (**Vpp1 = Vpp2**) power-description-structures.
	3	**Vcc**, **Vpp1** and **Vpp2** power-description-structures.
Timing	0	When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no timing is specified. **READY** may indicate busy indefinitely, **WAIT#** will be active from 0 to 12 microseconds. Otherwise, the timing is specified by the most recently scanned Configuration-Entry tuple with its default bit set.
	1	A timing-description structure is present following the power-description structure.
IO Space	0	When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no I/O space is used. Otherwise, the I/O space requirement is specified by the most recently scanned Configuration-Entry tuple with its default bit set.
	1	An I/O space description structure is present following the timing description structure.
IRQ	0	When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no Interrupt is used. Otherwise, the Interrupt request requirement is specified by the most recently scanned Configuration Entry tuple with its default bit set.
	1	An Interrupt request description structure is present following the I/O space description structure.
Mem Space	Memory address space mapping requirements for this configuration. There may be 0, 2, 4 or N bytes of information following the Interrupt Request Structure. The coding is as follows:	
	0	When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no configuration dependent, memory address space is used. Otherwise, the memory address space requirement is specified by the most recently scanned Configuration-Entry tuple with its default bit set.
	1	Single 2-byte length specified. Card's Base Address is zero (0) and any host address may be mapped. The length is specified in 256-byte units.
	2	Length (2 bytes) and Card Address (2 bytes) specified. Host address equals card address. The length and card address are each 2-bytes long and represent the number of 256- byte pages which are mapped, and the starting page, which is mapped, respectively. The length field appears before the card address field.
	3	A memory space descriptor byte followed by one or more window descriptors is present.
Misc	0	When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then the miscellaneous fields are interpreted to be all zero. Otherwise, the miscellaneous fields are specified by the most recently scanned Configuration-Entry tuple with its default bit set.
	1	A miscellaneous fields structure is present following the memory space description structure.

3.3.2.4 TPCE_PD: Power Description Structure

Each power description structure has the following format:

Parameter Selection Byte
First Parameter Definition
..
Last Parameter Definition

Each bit in the parameter selection byte indicates whether or not the corresponding parameter is described by parameter definitions which follow that byte. The parameter definitions are present for each logic-1 bit in the parameter selection byte. Each byte in a parameter definition, except the last byte, has a logic-1 in bit 7 of the byte. The parameter definitions are ordered corresponding to bits 0 through 7 of the parameter selection byte.

1) Parameter Selection Byte

***Parameter Selection Byte* in Power Description Structure**

7	6	5	4	3	2	1	0
RFU (0)	Pdwn I	Peak I	Avg I	Static I	Max V	Min V	Nom V

Field	Description
Nom V	Nominal operating supply voltage. In the absence of other information the nominal operating voltage has a tolerance of ±5%.
Min V	Minimum operating supply voltage.
Max V	Maximum operating supply voltage.
Static I	Continuous supply current required.
Avg I	Maximum current required averaged over 1 second.
Peak I	Maximum current required averaged over 10 milliseconds.
PDwn I	Power-down supply current required.
RFU	Reserved for future standardization.

2) Parameter Definitions

Parameter definitions consist of a first byte containing a mantissa and exponent as described below. Additional bytes appearing after the first byte serve to modify the initial value as described below. An arbitrary number of additional bytes may be defined with the last byte in the list having a 0 in bit 7. The next parameter definition, or the next field of the I/O tuple, is in the byte immediately following the last byte of the parameter definition.

***Parameter Definition* in Power Description Structure**

7	6	5	4	3	2	1	0
EXT	Mantissa				Exponent		
EXT	Extension						

Extension bytes may be continued indefinitely until the first byte which contains a 0 in bit 7, which is the final byte.

The Exponent of the Current and Voltage Values are given below:

Exponent	Current Scale	Voltage Scale
0	100 nA	10 µV
1	1 µA	100 µV
2	10 µA	1 mV
3	100 µA	10 mV
4	1 mA	100 mV
5	10 mA	1 V
6	100 mA	10 V
7	1 A	100 V

The mantissa of the value is given by the following table:

Mantissa	Value
0	1
1	1.2
2	1.3
3	1.5
4	2
5	2.5
6	3
7	3.5
8	4
9	4.5
AH	5
BH	5.5
CH	6
DH	7
EH	8
FH	9

Field	Description	
EXT	When *EXT* is set (1), an extension byte follows this byte. Extension bytes may be concatenated indefinitely. The final extension byte contains a 0 in the *EXT* field.	
Extension	The *Extension* field is defined as follows:	
	0··63H	Binary value for next two decimal digits to the right of the current value. (Note that this range is 0··99 decimal).
	64H··7CH	Reserved.
	7DH	No connection (i.e., high impedance) permitted during sleep or power-down only. (Must be last extension byte).
	7EH	Zero value required (Must be only extension byte). The voltage values given as nominal values are to be ignored.
	7FH	No connection (i.e., high impedance) is required (Must be only extension byte). The voltage values given as nominal values are to be ignored.

Here is an example power descriptor of a card with the following characteristics:

Operates over a **VCC** range of 2.5 to 7 volts. Operating current is 1 mA standby (static), 30 mA average operating current. When in power down mode, the card requires only 75 µA. It uses a **VPP[2::1]** of 12 volts ±5% at 50 mA peak which need not be connected during sleep or power down.

Power Description Example

Byte	Value	D7	D6	D5	D4	D3	D2	D1	D0	Notes
x + 0	02H	0	0	0	0	0	0	2 **Vcc** & **VPP[2::1]**		Feature Byte
+1	5EH	0 RFU	1 Power Down	0 No Peak I	1 Avg I	1 Static I	1 Max V	1 Min V	0 No Nominal V	Parameter Selection Byte for **Vcc**
+2	2DH	0 No EXT	52.5 X				5 1 Volt			**Vcc** Minimum Voltage
+3	6DH	0 No EXT	DH 7 X				5 1 Volt			**Vcc** Maximum Voltage
+4	04H	0 No EXT	0 1…0				4 1 mA			**Vcc** Static Current
+5	35H	0 No EXT	6 3.0 X				5 10 mA			**Vcc** Average Current
+6	EAH	1 EXT Next	DH 7.0 X				2 10 µA			**Vcc** Power Down Current
+7	32H	0 No EXT	32H +.50 (gives 7.50 x 10 µA)							**Vcc** Power Down Current Extension Byte
+8	21H	0 RFU	0 No Pwr Down I	1 Peak I Specified	0 No Avg I	0 No Static I	0 No Max V	0 No Min V	1 Nominal V	**VPP[2::1]** Parameter Selection Byte, Nominal V and Peak I given
+9	8EH	1 EXT Next	1 1.2 X				6 10 Volts			Nominal **VPP[2::1]** is 12.0 Volts ±5%
+AH	7DH	0 No EXT	7DH **VPP[2::1]** may be made no connect during power down and sleep modes							Extension byte indicates that No Connect is permitted on **VPP[2::1]** during Sleep and power Down
+BH	55H	0 No EXT	AH 5.0 X				5 10 mA			Peak **VPP[2::1]** current is 50 mA.

3.3.2.5 TPCE_TD: Configuration Timing Information

Timing information for Wait, **READY** and Initialization Delay are given in scaled, extended device speed code format. The first byte contains the scale factors and the other bytes contain the unscaled factors in the extended device speed code format.

(See also ***3.2.2.1.1 Device ID Fields***.)

TPCE_TD: **Configuration Timing Information field**

7	6	5	4	3	2	1	0
Reserved Scale 7			READY Scale			Wait Scale	

The timing descriptions, when present, occur in the same order as the fields are listed below:

Field	Description
WAIT Scale	This field is the power of 10 scale factor to be applied to the MAX WAIT time in extended device speed format which follows. The value 3 indicates that the **WAIT#** signal is not used and the MAX WAIT Speed is not present following the timing byte.
READY Scale	This field is the power of 10 scale factor to be applied to the MAX time in the Busy State for the **READY** signal. A value of 7 indicates that **READY** is not used and no maximum time is present following the timing scale factor byte.
Reserved Scale 7	This field is the power of 10 scale factor which is to be applied to a reserved time definition. A value of 7 indicates that no initial or extended reserved speed bytes follow the **READY** extended speed bytes.

3.3.2.6 TPCE_IO: I/O Space Addresses Required For This Configuration

The I/O Space description indicates that the card requires a portion of its I/O address space be accessible within the host's I/O address space. The I/O Space entry consists of, at minimum, the I/O Space Definition byte. If this byte has the Range bit set, a Range Descriptor byte follows the Definition byte, which is then followed by a series of Range Description fields. Each Range Description consists of either a Starting Address field, a Length field, or both. Multiple Range Descriptions can be defined if the card has multiple I/O windows.

***TPCE_IO*: I/O Space Description**

7	6	5	4	3	2	1	0
Range	Bus 16/8		IOAddrLines				

Field	Description		
Bus 16/8	Recommended values for *Bus 16/8* are:		
	Value	**Card-Related Description**	**Host Requirements**
	00B	Reserved.	
	01B	Card supports 8-bit I/O only. All registers are accessible only with byte accesses on **D0** through **D7**.	Host must be able to access I/O on card using byte accesses only on **D0** through **D7**. When 16-bit access is requested, two cycles must be generated with the even byte first.
	10B	All registers accessible by 16-bit hosts which generate byte accesses for byte registers and 16-bit accesses for word registers. Eight-bit accesses to 16-bit registers are not supported. Byte accesses to odd-byte registers may take place on either **D0** through **D7** or **D8** through **D15**.	Host must access 16-bit registers using 16-bit accesses, and access 8-bit registers using 8-bit accesses. **IOIS16#** is used to determine whether an access is to an 8- or 16-bit register.
	11B	All registers are accessible by both 8-bit or 16-bit accesses. If bit IOIS8 in the Card Configuration and Status Register is set, the host only supports 8-bit access and operation is as described for Bus 16/8 = 01. If IOIS8 is zero, all registers can be accessed using either 8-bit or 16-bit accesses.	This mode supports either 16-bit or 8-bit accesses, but 8-bit hosts can specify operation as in mode 01, above, by setting the IOIS8 bit in the Card Configuration and Status Register to 1. If IOIS8 is cleared to zero either 8-bit or 16-bit access is supported to all registers.

Field	Description	
Range	0	The I/O Space Definition byte is the only byte needed for this definition. The card will respond to all enabled I/O accesses and uses *IOAddrLines* of address to distinguish among its I/O ports. The host can select any 2^N boundary (N = IOAddrLines) for the card and generate enables for accesses at this host base address, with the amount of space also equal to 2^N.
	1	The I/O Space definition byte is followed by an *I/O Range Descriptor* byte, and one or more *I/O Address Range Description* fields.
IOAddrLines	Total number of address lines that are used by the card to determine when the card is selected.	
	0	When *IOAddrLines* is zero, a range must also be specified, and the card will respond to all addresses presented to the card. The host is entirely responsible for when the card is selected, and at what addresses the card is selected. The host must assign to the card a portion of the address space which is at least as large as the number of bytes indicated in the length field of the following range entry. The Base Address for the I/O space (assigned to the card by the host) must begin on a 2^N address boundary such that 2^N is greater than the number of bytes indicated in the length field.
	1··26	When IOAddrLines is non-zero, the card performs address decoding to determine when it is selected. In this case, the card and the host share the determination of when a card is actually selected. The card must indicate in IOAddrLines the highest address line (plus 1) which it decodes to determine when it has been selected. The card will determine which I/O addresses are actually selected within the I/O space that it decodes. The host and the card then share the task of determining when the card is selected. The host uses information provided by *IOAddrLines* and any range descriptions to determine if Card Enables should be generated during I/O cycles. The card then determines if it should respond to an address that has been enabled by the card enables. The card returns the Input-Acknowledge signal to the host whenever the card can actually respond to an I/O address on the bus.
	27··31	Reserved. (Beyond the 64 Megabyte I/O space.)
	Recommended values for *IOAddrLines* are:	
	0	Card responds to all I/O cycles when ***CE***'s are true, and specifies the desired address ranges to enable. It is suitable for all architectures.
	N	Where the card has $2^{(N-1)} <$ I/O ports $\leq 2^N$, and can be placed on any multiple of 2^N boundary. It is suitable for any architecture. For example, if a card has 32 eight-bit I/O ports and the card decodes 5 address lines (**A4**··**A0**), then N = 5, but the card can still be placed on a 32-byte I/O boundary; it does not need to be placed on a 64 byte I/O boundary.
	10	A 1 Kilobyte I/O address space. This is software compatible for decoding in the top 768 bytes of each 1 KByte of I/O space. This is the same as is done in the ISA and EISA PC's.
	16	Full decoding of 64 Kilobytes of I/O space. This is not compatible with EISA or ISA bus architectures.

***I/O Range Descriptor* Byte**

7	6	5	4	3	2	1	0
Size of Length		Size of Address		Number of I/O Address Ranges (-1)			

The *I/O Range Descriptor* byte is only provided if the *Range* bit in the I/O Space definition byte was set to one. This byte determines how many *I/O Address Range Description* fields will follow and how many bytes will be used to encode the *Start of I/O Address Block* and the *Length of I/O Address Block* portion of each of the *I/O Address Range Description* fields that follow. Either the *Start of I/O Address Block* or the *Length of I/O Address Block,* but not both portions of the descriptions can be eliminated by encoding the respective size field as zero. Every *I/O Address Range Description* that follows must have the same number of bytes as defined by the sum of the encoded sizes from the two size fields in the *I/O Range Descriptor* byte.

Field	Description
Number of I/O Address Ranges	This field specifies how many *I/O Address Range Description* fields will follow the *I/O Range Descriptor* byte. This value is encoded as one less than the number of descriptions. (A value of 0 indicates 1 *I/O Address Range Description* field, a value of 15 indicates 16 fields.)
Size of Address	This field specifies how many bytes will be used to encode the *Start of I/O Address block* in each of the *I/O Range Description* fields that follow. 0 address is not present in *I/O Address Range Description*s. 1 address is 1 byte long. 2 address is 2 bytes long. 3 address is 4 bytes long.
Size of Length	This field specifies how many bytes will be used to encode the *Length of I/O Address block* in each of the *I/O Range Description* fields that follow. 0 length is not present in *I/O Address Range Description*s. 1 length is 1 byte long. 2 length is 2 bytes long. 3 length is 4 bytes long.

The *Size of Address* and *Size of Length* fields may not both be zero (0).

***I/O Address Range Description* Field**

Byte	7	6	5	4	3	2	1	0
n	Start of I/O Address Block 0, 1, 2 or 4 bytes long as determined in the *I/O Range Descriptor*. Implied address bits in bytes that are not provided are zeros.							
n+(0··4)	Length of I/O Address Block (value is length -1) 0, 1, 2 or 4 bytes long as determined in *I/O Range Descriptor*.							

When the *Start of I/O Address Block* field is not present, this indicates that the starting address of the block is on an arbitrary 2^N boundary in the host space, where N is the number of address lines defined by *IOAddrLines*.

3.3.2.6.1 I/O Space Encoding Guidelines

Many permutations of the three main I/O Space encoding parameters (IOAddrLines, Start Address, and Length) are possible. These fields define the requirements of the card with respect to I/O addresses. In all cases, the host and the card share the same address (no translation is performed), the tuple merely defines when the card will be selected. The table below and associated text, is intended to give a precise definition for all the possible combinations of these parameters. There are two areas of explanation: when the host must generate Card Enables based on I/O address decoding, and how the host allocates I/O addresses.

I/O Space Definition Fields

IOAddrLines	Start Address	Length	Interpretation
*	*	0	Invalid combination. Should not be used.
0	NP	NP	Invalid combination. Should not be used.
0	NP	L	Host must at least generate ***CE*** for a host selected I/O address range of length L. Host sets the I/O range beginning on multiple of a power of 2 boundary. Host address selection range can be defined as follows (host chooses x and y): $[y * 2^x] \cdot\cdot [y * 2^x + L - 1]$, where $2^x >= L$ and x should be the smallest value that the host hardware will allow.
0	B	NP	Illegal combination.
0	B	L	Host must at least generate ***CE*** for the I/O address range specified by Start and Length. Host allocates the specified I/O address range, i.e., the address range is: $[B] \cdot\cdot [B + L - 1]$.
A	NP	NP	Host must at least generate ***CE*** for a host selected I/O address range of length 2^A. Host selects the I/O address range beginning on a 2^A boundary of length 2^A, i.e., the address range is (host chooses x): $[x * 2^A] \cdot\cdot [(x + 1) * 2^A - 1]$
A	NP	L	Host must at least generate ***CE*** for a host selected I/O address range of length L. Host selects the I/O address range beginning on a 2^A boundary of length L, i.e., the address range is (host chooses x): $[x * 2^A] \cdot\cdot [x * 2^A + L - 1]$
A	B	NP	Host must at least generate ***CE*** for a host I/O address range of length 2^A that includes B, i.e., the address range is (host determines x): $[x * 2^{(A)}] \cdot\cdot [(x + 1) * 2^A - 1]$ where: $x * 2^{(A)} <= B < (x + 1) * 2^A$
A	B	L	Host must at least generate ***CE*** for an address range of length L that includes base B, i.e., the address range is (host determines x): $[x * 2^A] \cdot\cdot [x * 2^A + L - 1]$ where: $x * 2^{(A)} <= B < x * 2^A + L$

*	any value that can be defined for the field
0	value is zero
NP	field is not present
A	a non zero value for IOAddrLines field
B	the base value defined for Start Address field
L	the value defined for Length field
x	a value chosen by the host to define host address selections
y	a value chosen by the host to define host address selections

When *IOAddrLines* <> 0, whether a base address is specified or not, the host can generate ***CE*** for any combination of the upper (26 - *IOAddrLines*) address lines. If a base address is specified and requires more address bits to represent than was specified by *IOAddrLines*, the full base address indicates a desired I/O address range allocation which should be included in the decode by the host. If a base address and *IOAddrLines* <> 0 are both specified, all recommended configurations will have a base address value that is a multiple of $2^{(\mathbf{IOAddrLines})}$.

When *IOAddrLines* <> 0 and Length L is specified, the host need not reserve any excess I/O address space ($2^{(\mathbf{IOAddrLines})}$ - L bytes) for this PC Card. If L is not specified, the host must allocate all $2^{(\mathbf{IOAddrLines})}$ bytes of address space to this PC Card.

In all cases, the host may allocate a larger address space than that indicated for a particular configuration entry. The host may also allocate multiple (aliased) address ranges for a given PC

Card. If such aliased address ranges are allocated, a PC Card with *IOAddrLines* = 0 will respond to all aliased ranges since it does not perform upper address line decoding. A PC Card with *IOAddrLines* <> 0 will respond to other aliased ranges based on the number of decoded address lines specified with *IOAddrLines*.

If multiple address ranges are specified in a configuration entry, the *IOAddrLines* value must span the sum of all range lengths. For example, if an entry has a range with a length of 4 and a second range of length 8, *IOAddrLines* must be at least 4 (4 bits of addressing are required to access all 12 bytes).

3.3.2.7 TPCE_IR: Interrupt Request Description Structure

When the IRQ bit is set, it indicates that an interrupt description follows the I/O address bytes, if any. The interrupt request levels specified by the Configuration Table Entry Tuple describe the preferred routing for the card's **IREQ#** line. Routing of the **IREQ#** interrupt is performed by the host system which actually determines the system interrupt level used. A client which is the sole consumer of the card's **IREQ#** interrupt may elect to route the **IREQ#** line to a level not specified by the Configuration Table Entry Tuple. A generic card configuration utility which is not the ultimate consumer of the card's **IREQ#** interrupt should only route to the specified interrupt levels. There are either one or three interrupt description bytes as shown below:

***TPCE_IR*: Interrupt Request Description Structure**

Byte	7	6	5	4	3	2	1	0
+0	Share	Pulse	Level	Mask	IRQN Line 0··15			
					VEND	BERR	IOCK	NMI
+1 *	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
+2 *	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8

* These bytes present only when bit 4 of byte 0 is set (1). See *Mask* description below.

Field	Description
Share	The card contains interrupt sharing support logic in the Card Control and Status Register which may be used to allow interrupt levels to be shared among several cards.
Pulse	When this bit is one, the interrupt request pin in this mode may be pulsed low momentarily to request an interrupt.
Level	The Level Bit, when set, indicates that in this configuration the card's interrupt request pin can remain asserted at the active low level until the interrupt is serviced. When a configuration is capable of supporting level mode interrupts, the IRQLEV bit in the Configuration Register controls whether the level or pulsed mode interrupts are selected.
Mask	The Mask bit indicates whether the possible destinations of the interrupt request line are indicated by a single interrupt line value, or a mask of possible interrupt lines. When this bit is zero, the destination is given by the IRQN field, formed from right hand nibble of byte 0, which selects 1 of 16 possible interrupt request lines. In this case, bytes 1 and 2 are not present. When this bit is a one, the 4 least significant bits of byte 0 are masks for possible interrupt destinations. In addition, two additional bytes are present which contain mask bits for 16 possible interrupt request lines.
NMI IOCK BERR VEND	These bits are valid only when the Mask bit is 1. When set, each of these bits indicates that the corresponding special interrupt signal may be assigned by the host to the card's interrupt-request pin. Systems are not required to support these interrupts. NMI is non-maskable interrupt. IOCK is the I/O-check signal. BERR is Bus-Error signal. VEND is a Vendor-Specific signal.
IRQ0··15	These bytes are present only when the Mask bit is a 1. When set, each of these bits indicates that the corresponding interrupt signal may be assigned by the host to the card's interrupt request pin. IOCK is the I/O check signal. General purpose systems must support all those interrupts which are available on the I/O bus normally associated with the environment they are supporting.

3.3.2.8 TPCE_MS: Memory Space Description Structure

The Memory Space description indicates the card requires a configuration-entry related portion of its Common Memory space be mapped into the host's address space. This memory may be either direct mapped (no address translation is performed by the host), or may require that the host translate the host address to the corresponding card address in order to support the card. In either case, for the host to support the card under this configuration, the host must access the card when the corresponding addresses are accessed by software.

TPCE_MS*: Memory Space Descriptor Byte

7	6	5	4	3	2	1	0
Host Addr	Card Address Size		Length Size		Number of Windows		

* This byte and the associated Window Descriptors are present only if the *Mem Space* field in the Feature Selection byte is 3.

Field	Description
Number of Windows	The number of window descriptors following minus 1.
Length Size	The size of the length fields in bytes. The length is in 256-byte pages
Card Address Size	The size of the card address fields in bytes. The card address is in 256-byte pages.
Host Addr	0 No host address field is present and the host address is arbitrary. 1 A host address field, the same size as the card address field, is present.

Window Descriptor

Byte	7	6	5	4	3	2	1	0
+0··3	Length	The length of the window in units of 256 bytes. The least significant part appears in byte 0. 0··3 bytes depending upon *Length Size* field above.						
+0··3	Card Address	The address to be accessed on the card corresponding to the host address. The least significant part appears in byte 0. 0··3 bytes depending upon *Card Address Size* field above.						
+0··3	Host Address	The physical address in the host-address space where the block of memory must be placed. This field is present only when the *Host Addr* field is set (1). The least significant part appears in byte 0. 0··3 bytes depending upon *Card Address Size* field above.						

3.3.2.9 TPCE_MI: Miscellaneous Features Field

When the *Misc* bit is set in the Feature Selection Byte, the Miscellaneous Features Field follows prior Card Information Structure (CIS) fields. The Miscellaneous Features Field is one or more bytes. The most significant bit of each byte in the field, except the last byte, is set to one (1). The most significant bit of the last byte in the Miscellaneous Features Field is reset to zero (0).

If a configuration does not support any of the features described in the latter bytes of the Miscellaneous Features Field, these bytes may be omitted and only the initial bytes shall be present. In all cases, the most significant bit of the last byte present shall be reset to zero (0). For example, if the configuration does not support DMA transfers, but the *Pwr Down* feature is supported, only a single byte may be present with its most significant bit reset (0) and the *Pwr Down* bit set to one (1).

The two bytes following the DMA specification contain the PC Card's thermal rating. The first byte in bits 0 - 6 contain the XX part of a XX.YY thermal rating. The range of the PC Card's thermal rating is 00.00 to 99.99. Bit 7 is always set to 1. The second byte in bits 0 - 6 is the YY part of the XX.YY thermal rating. Bit 7 is used to extend the TPCE_MI field. See the ***Physical Specification*** for more information on the PC Card thermal rating.

***TPCE_MI*: Miscellaneous Features Field**

7	6	5	4	3	2	1	0
EXT	RFU (0)	Pwr Down	Read Only	Audio	Max Twin Cards		
EXT	RFU (0)	RFU (0)	DMA Width	DMA Request Signal		RFU (0)	RFU (0)
1	PC Card's Thermal Rating, XX of XX.YY value						
EXT	PC Card's Thermal Rating, YY of XX.YY value						

Field	Description
Max Twin Cards	This field is used to indicate the card requires that identical cards installed in the system be differentiated from each other in a sequential manner. For example, first twin is card 0, second is card 1, and so on. This allows the cards to share I/O ports and interrupts in a manner consistent with some peripherals commonly used in PC computers, such as fixed disk drives. The *Max Twin Cards* field specifies the maximum number of other identical cards which can be configured identically to this card. This permits more than one card to be installed in host which responds to the identical I/O addresses. The host allows the cards to distinguish among themselves by writing their "Copy" numbers (e.g. 0, for the first card, 1 for the second, etc.) into the copy field of the Socket and Copy Register in the Card Configuration Registers.
Audio	This bit indicates the card allows the **BVD2** signal to be used as Audio Waveform for the speaker. This operation is controlled by the Audio Enable Bit in the Card Control and Status Configuration Register.
Read Only	This bit indicates the card contains a data storage medium which is read-only for this configuration. There may be other configurations for which the storage medium is read/write.
Pwr Down	This bit indicates the card supports a power-down mode controlled by the power-down bit in the Control and Status Register.
RFU (0)	These bits are reserved for future definition and must be 0.
EXT	An extension follows this byte. A series of extension bytes is terminated when an extension byte is encountered which does not have the EXT bit set to one (1).
DMA Request Signal	A binary value identifying the pin on the interface used to signal a DMA request. 0 DMA not supported by this configuration 1 **DREQ#** uses **SPKR#** 2 **DREQ#** uses **IOIS16#** 3 **DREQ#** uses **INPACK#**
DMA Width	The DMA data transfer width. 0 the DMA data width is 8-bits. 1 the DMA data width is 16-bits.
Thermal Rating, Major	Bits 0-6 The XXh value of a PC Card's Thermal Rating. Valid range is 0 (00h) - 99 (63h). For consistency, bit 7, EXTension bit, is set (1).
Thermal Rating, Minor	Bits 0-6 The YYh value of a PC Card's Thermal Rating. Valid range is 0 (00h) - 99 (63h). Bit 7, EXTension flag, is set according to future specification fields.

3.3.2.10 TPCE_ST: Additional information in subtuple format

The *TPCE_ST* field extends to the end of the tuple. It contains tuple-formatted information relating to the configuration defined in this tuple.

3.3.2.10.1 STCE_EV: Environment Descriptor Subtuple

The environment-descriptor subtuple is an optional subtuple used to describe the environment in which the configuration is used. It describes the system and, optionally, the operating system for which the configuration is intended.

The subtuple contains one or more strings, the first of which contains 3 parts. The three parts are:

1. System Name
2. Operating System Name
3. Comment

The Operating System Name is preceded by a colon (:) and the comment is preceded by a semicolon (;). The remaining strings contain comment (and, optionally, the system name and OS name) in alternate languages. Each string is terminated by a 00_H byte. The list of strings is terminated by reaching the end of the subtuple. When the default bit is set in this configuration entry tuple, the environment descriptor becomes the default environment descriptor for succeeding entries as well.

***STCE_EV*: Environment Descriptor Subtuple**

Byte	7	6	5	4	3	2	1	0
0	ST_CODE	STCE_EV (C0H).						
1	ST_LINK	Link to next subtuple (at least m-1).						
2··m	STEV_STRS	A list of strings, the first being ISO 646 IRV coded, and the rest being coded as ISO alternate language strings, with the initial escape character suppressed. Each string is terminated by a 0 byte, and the last string, if it does not extend to the end of the subtuple, is followed by an FFH byte.						

3.3.2.10.2 STCE_PD: Physical Device Name Subtuple

The physical device name subtuple is an optional subtuple used to describe the physical device being implemented by the configuration.

The subtuple contains one or more strings, the first of which contains up to two parts. The first part is the physical device name; the second part, preceded by a semicolon (;) is comment.

The remaining strings contain comment (and, optionally, the physical device name) in alternate languages.

When the default bit is set in this configuration entry tuple, the environment descriptor becomes the default environment descriptor for succeeding entries as well.

***STCE_PD*: Physical Device Name Subtuple**

Byte	7	6	5	4	3	2	1	0
0	ST_CODE	STCE_PD (C1H).						
1	ST_LINK	Link to next subtuple (at least m-1).						
2··m	STPD_STRS	A list of strings, the first being ISO 646 IRV coded, and the rest being coded as ISO alternate language strings, with the initial escape character suppressed. Each string is terminated by a 0 byte, and the last string, if it does not extend to the end of the subtuple, is followed by an FFH byte.						

3.3.2.10.3 Additional I/O Feature Definitions within Entry

Additional information about the configuration may be embedded, at the end of, but within, a table entry in the Card Configuration Table. The information takes the form of configuration-entry-specific subtuples. These tuples begin at the byte following the last feature-description byte indicated by the feature-selector byte. These additional tuples must be in standard tuple form (tuple code, followed by offset to next tuple, followed by tuple info) and may not extend beyond the size for the entry, indicated by the offset to the next table entry. The tuple-codes definitions used are valid only within the interface definition tuple, or the configuration entry tuple.

3.3.3 CISTPL_CFTABLE_ENTRY_CB: CardBus PC Card Configuration Table Entry Tuple

Configuration Table Entry tuples are used to specify each possible configuration of a card and to distinguish among the permitted configurations. The Configuration tuple must be located before all Configuration Table Entry tuples. The Configuration Table Entry, whose Configuration Table Index matches the last index in the Configuration tuple, must appear after all other Configuration Table Entries.

When the default bit is set in an entry's Configuration-Table index byte, that Configuration Table Entry provides all default values for following entries in the Card Configuration Table.

While the default bit is not set in an entry's Configuration-Table index byte, the entry provides default values which apply only to the configuration indicated by the that entry's Configuration-Table Index (*TPCE_INDX*) byte.

Values which are not present in a specific entry are taken from the card's most recently scanned default entry, as indicated in the specific fields below.

Table 3-40 CISTPL_CFTABLE_ENTRY_CB: CardBus PC Card Configuration Table Entry Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_CFTABLE_ENTRY_CB (05H)						
1	TPL_LINK	Link to next tuple (n-1, 2 minimum)						
2	TPCE_INDX	Configuration Table Index Byte.						
3	TPCE_CBFS	Feature Selection byte, This field indicates which optional fields are present.						
··	TPCE_PD	Power Description Structure. (See ***3.3.2.4 TPCE_PD: Power Description Structure***.)						
··	TPCE_CBIO	I/O Base Address Registers Used.						
··	TPCE_IR	Interrupt Request Description structure.						
··	TPCE_CBMS	Memory Base Address Registers Used.						
··	TPCE_CBMI	CardBus PC Card Miscellaneous Features field.						
··n	TPCE_ST	Additional information about the configuration in subtuple format.						

3.3.3.1 TPCE_INDX: The Configuration Table Index Byte

The Configuration-Table index byte, *TPCE_INDX*, contains a bit to indicate that the configuration defined in this tuple should be taken as the default conditions for Configuration-Entry tuples that follow. The *TPCE_INDX field* is also used to identify the last CISTPL_CONFIG_ENTRY_CB in the CIS chain when the value matches the value in the *TPCC_LAST* field of the CISTPL_CONFIG_CB tuple.

***TPCE_INDX*: Configuration Index Byte**

7	6	5	4	3	2	1	0
RFU (0)	Default	Configuration-Entry-Number					

Field	Description
Configuration-Entry-Number	Contains the value to be written to the Card Configuration Register to enable the configuration described in the tuple.
Default	This bit indicates that this entry provides default values for succeeding entries until another entry is encountered with its Default bit set. 1 The default conditions for following table entries are exactly those conditions specified in this entry. 0 The default conditions are the conditions specified by the last entry encountered with its default bit set.

3.3.3.2 TPCE_CBFS: Feature Selection Byte

The feature-selector byte indicates which compact descriptive fields are present within the table entry. Those fields which are present are indicated by a 1 in the corresponding bit of the selector byte. Within the entry, the fields are ordered in the order of the selector bits starting from bit 0 (LSB) to bit 7 (MSB).

***TPCE_FS:* Feature Selection Byte**

7	6	5	4	3	2	1	0
Misc	RFU (0)	Mem Space	IRQ	IO Space	RFU (0)	Power	

Field	Description
Power	The power supply requirements and load characteristics for this configuration are indicated. There may be 0, 1, 2 or 3 fields following representing **Vcc**, **VPP[2::1]**, or **VPP1** and **VPP2** (individually) in that order. The coding is as follows: 0 No power-description structures, use the default. 1 **Vcc** power-description-structure only. 2 **Vcc** and **VPP[2::1]** (**VPP1** = **VPP2**) power-description-structures. 3 **Vcc**, **VPP1** and **VPP2** power-description-structures.
IO Space	0 When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no I/O space is used. Otherwise, the I/O space requirement is specified by the most recently scanned Configuration-Entry tuple with its default bit set. 1 An I/O space description structure is present.
IRQ	0 When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no Interrupt is used. Otherwise, the Interrupt request requirement is specified by the most recently scanned Configuration Entry tuple with its default bit set. 1 An Interrupt request description structure is present.
Mem Space	0 When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then no memory space(s) are used. Otherwise, the memory space requirement(s) are specified by the most recently scanned Configuration-Entry tuple with its default bit set. 1 A memory space description structure is present.
Misc	0 When the default bit is set in this tuple, or no Configuration-Entry tuple has been scanned with its default bit set, then the miscellaneous fields are interpreted to be all zero. Otherwise, the miscellaneous fields are specified by the most recently scanned Configuration-Entry tuple with its default bit set. 1 A miscellaneous fields structure is present.

3.3.3.3 TPCE_PD: Power Description Structure

The CardBus PC Card Power Description structure, *TPCE_PD* is identical to the 16-bit PC Card structure used in the CISTPL_CFTABLE_ENTRY tuple. (See ***3.3.2.4 TPCE_PD: Power Description Structure***.)

Note: CardBus PC Cards allow three **V**CC operational voltage settings: 3.3 volts and reserved values X.X volts and Y.Y volts. The 5 volt **V**CC setting is illegal for CardBus PC Cards.

3.3.3.4 TPCE_CBIO: I/O Base Address Registers Used

This field indicates the Base Address Registers which are used for I/O in this configuration. The potential values are limited to the range one through five by the requirement that all CardBus PC Card I/O space(s) be configurable as memory mapped I/O. The associated registers must be provided in order, with the I/O Base Address Register (or Registers) immediately preceding the memory Base Address Register, to avoid inadvertent double mapping, i.e., there may be a memory Base Address Register following each I/O Base Address Register to map the I/O space, or a contiguous set of I/O Base Address Registers may be followed by a single memory Base Address Register which maps all of the I/O spaces described by the set.

TPCE_CBIO: **Base Address Registers Used For I/O In This Configuration**

7	6	5	4	3	2	1	0
RFU	RFU	IO_BAR_Used (set = true)					RFU

The *IO_BAR_Used* field is a bit-mapped representation of the I/O Base Address Registers required by a configuration. For example, if bits one and two are set, Base Address Registers one and two map I/O space used in the configuration.

3.3.3.5 TPCE_IR: Interrupt Request Description Structure

The CardBus PC Card Interrupt Request Description structure, *TPCE_IR* is identical to the 16-bit PC Card structure used in the CISTPL_CFTABLE_ENTRY tuple. (See ***3.3.2.7 TPCE_IR: Interrupt Request Description Structure***.)

Note: All CardBus PC Cards generate level mode interrupts whenever interrupts are selected. *Level* shall be set (1) and *Pulse* shall be reset (0) for all CardBus PC Cards.

3.3.3.6 TPCE_CBMS: Memory Base Address Registers Used

This field indicates the Base Address Registers which are used for memory space in this configuration. The potential values are limited to the range one through seven by the encoding of configuration space accesses as zero (0). Note that the Expansion ROM Base Address Register is listed with all other memory space accesses when it is used in the configuration.

TPCE_CBMS: **Base Address Registers Used For Memory In This Configuration**

7	6	5	4	3	2	1	0
MEM_BAR_Used (set = true)							RFU

The *MEM_BAR_Used* field is a bit-mapped representation of the memory Base Address Registers required by a configuration. For example, if bits one and two are set, Base Address Registers one and two map memory space used in the configuration.

3.3.3.7 TPCE_CBMI: CardBus PC Card Miscellaneous Features Field

The CardBus PC Card Miscellaneous Features Field is one or more bytes. The most significant bit of each byte in the field, except the last byte, is set to one (1). The most significant bit of the last byte in the *TPCE_CBMI* field is reset to zero (0).

If a configuration does not support any of the features described in the last bytes of the Miscellaneous Features Field, these bytes may be omitted and only the initial bytes shall be present. In all cases, the most significant bit of the last byte present shall be reset to zero (0).

A number of the fields defined in the *TPCE_CBMI* field are used to give information on how the **Command** register fields should be set. With the exception of the *Fast Back-to-Back* field, each **Command** register field should have the same values as the corresponding feature indicator in this tuple field. For example, if the *Parity Error Response* field is set here, then the *Parity Error Response* field should be set in the **Command** register as well. (See also the ***Electrical Specification***.)

The two bytes following the RFU feature bits contain the PC Card's thermal rating. The first byte in bits 0 - 6 contain the XX part of a XX.YY thermal rating. The range of the PC Card's thermal rating is 00.00 to 99.99. Bit 7 is always set to 1. The second byte in bits 0 - 6 is the YY part of the XX.YY thermal rating. Bit 7 is used to extend the TPCE_MI field. See the ***Physical Specification*** for more information on the PC Card thermal rating.

TPCE_CBMI: **Miscellaneous Features Field**

7	6	5	4	3	2	1	0
EXT (must be 1)	Fast Back-to-Back	SERR# Enable	Wait Cycle Control	Parity Error Response	VGA Palette Snoop	Memory Write & Invalidate	Bus Master
EXT	RFU (0)	RFU (0)	RFU (0)	RFU (0)	Wakeup Event Support	Binary Audio Enable	PWM Audio Enable
1	PC Card's Thermal Rating, XX of XX.YY value						
EXT	PC Card's Thermal Rating, YY of XX.YY value						

Field	Description
RFU (0)	These bits are reserved for future (or historical) definition and must be 0.
EXT	If set (1), another byte follows this byte. A series of extension bytes is terminated when an extension byte is encountered which has the EXT bit reset to zero (0).
Bus Master	(See the ***Electrical Specification***.)
Memory Write & Invalidate	(See the ***Electrical Specification***.)
VGA Palette Snoop	(See the ***Electrical Specification***.)
Parity Error Response	(See the ***Electrical Specification***.)
Wait Cycle Control	(See the ***Electrical Specification***.)
SERR# Enable	(See the ***Electrical Specification***.)
Fast Back-to-Back	This bit field does not control the corresponding value in the Command register. Rather, this is the value which should be placed in the Status register indicating if this particular function, when acting as a target, is Fast Back-to-Back capable
PWM Audio Enable	This field is used to indicate that the card can supply audio data in a Pulse Width Modulation encoded format.
Binary Audio Enable	This field is used to indicate that the card can supply audio data in a binary wave format.
Wakeup Event Support	This read only field is used to indicate that the card supports (can provide) wakeup events. If the field is set (1), then the card supports wakeup events. If the field is reset (0), then the card does not support wakeup events.
Thermal Rating, Major	Bits 0-6 hold the XXH value of a PC Card's Thermal Rating. Valid range is 0 (00H) - 99 (63H). For consistency, bit 7, EXTension bit, is set (1).
Thermal Rating, Minor	Bits 0-6 hold the YYh value of a PC Card's Thermal Rating. Valid range is 0 (00H) - 99 (63H). Bit 7, EXTension flag, is set according to future specification fields.

3.3.3.8 TPCE_ST: Additional information in subtuple format

The *TPCE_ST* field extends to the end of the tuple. It contains tuple-formatted information relating to the configuration defined in this tuple. The CardBus PC Card definitions for this field are identical to the 16-bit PC Card structures used in the CISTPL_CFTABLE_ENTRY tuple. (See ***3.3.2.10 TPCE_ST: Additional information in subtuple format***.)

3.3.4 CISTPL_CONFIG: 16-bit PC Card Configuration Tuple

The Configuration tuple is used to describe the general characteristics of 16-bit PC Cards which can be configured for operation. This includes cards containing I/O devices or using custom interfaces. It may also describe PC Cards, including Memory Only cards, which exceed nominal power supply specifications, or which need descriptions of their power requirements or other information.

The default interface, in the absence of a Configuration tuple, is a Memory Only interface.

There may be no more than one Configuration tuple in the CIS.

There are two types of 16-bit PC Card defined host interfaces:

- Those supporting Memory Card Only interface and,
- Those supporting both the Memory and the I/O Card interfaces.

In addition, custom interfaces are also permitted which can be recognized and enabled in a standardized manner.

The card indicates which type(s) of custom interface(s) it can support in Custom Interface subtuples of the Configuration Tuple. The interpretation of the rest of the bytes of the configuration tuple is determined by the size-of-fields byte, *TPCC_SZ*, and consist of two Configuration Register Descriptors, *TPCC_RASZ* and *TPCC_RMSZ*, and a Reserved Size field, *TPCC_RFSZ*.

The Card Configuration Table allows the system to determine the characteristics of the card for each allowable configuration. The system may then configure the card into any allowable configuration, or leave the card unconfigured. It may use the configuration to gracefully reject the card if the card is found to be incompatible with the system hardware and software.

Table 3-41 CISTPL_CONFIG : 16-bit PC Card Configuration Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_CONFIG (1AH)						
1	TPL_LINK	Link to next tuple (n-1; minimum 5)						
2	TPCC_SZ	Size of Fields Byte						
	TPCC_RFSZ (Reserved Size, must be 0)		TPCC_RMSZ (Size of *TPCC_RMSK* field)				TPCC_RASZ (Size of *TPCC_RADR*)	
3	TPCC_LAST	Index Number of the last entry in the Card Configuration Table.						
	RFU (0)		Last Index					
4··	TPCC_RADR	Configuration Registers Base Address in Attribute Memory Space. 1, 2, 3 or 4 bytes depending upon the *TPCC_RASZ* field in *TPCC_SZ*.						
··	TPCC_RMSK	Configuration Registers Present Mask. 1 to 16 bytes as indicated by the *TPCC_RMSZ* field in *TPCC_SZ*.						
m	TPCC_RSVD	Reserved area 0··3 bytes. Must be 0 bytes until defined.						
m··n	TPCC_SBTPL	The rest of the tuple is reserved for subtuples containing standardized optional additional information related to the Card Configuration.						

3.3.4.1 TPCC_SZ: Size of Fields Byte

TPCC_SZ: **Size of Fields Byte**

7	6	5	4	3	2	1	0
TPCC_RFSZ (Reserved Size)		TPCC_RMSZ (Size of TPCC_RMSK field)				TPCC_RASZ (Size of TPCC_RADR)	

Field	Description
TPCC_RASZ	The number of bytes in the Configuration Registers Base Address in Attribute Memory Space field (*TPCC_RADR*) of this tuple is the value of this field plus 1.
TPCC_RMSZ	The number of bytes in the Configuration Register presence mask field (*TPCC_RMSK* field) of the tuple is this value plus 1.
TPCC_RFSZ	Size of area presently reserved for future use 0, 1, 2 or 3 bytes. Must be 0 at present.

3.3.4.2 TPCC_LAST: Card Configuration Table Last Entry Index

The Card-Configuration-Table size is a one byte field which contains the Configuration Index Number of the last configuration described in the Card Configuration Table. Once the host encounters this configuration, when scanning for valid configurations, it will have processed all valid configurations.

TPCC_LAST: **Card Configuration and Last Entry Index**

7	6	5	4	3	2	1	0
RFU 0		Last Index					

Field	Description
Last Index	The Index Number of the final entry in the Card Configuration Table (the last entry encountered when scanning the CIS).
RFU 0	This area is reserved for future standardization and must be 0.

3.3.4.3 TPCC_RADR: Configuration Registers Base Address in Attribute Space

TPCC_RADR: **Configuration Registers Base Address in Attribute Space**

7	6	5	4	3	2	1	0
Base Address Bits 7··0 (always present)							
Base Address Bits 15··8 (present if TPCC_RASZ is ≥ 1)							
Base Address Bits 23··16 (present if TPCC_RASZ is ≥ 2)							
Base Address Bits 25··24 (present if TPCC_RASZ is = 3)							

The Base Address of the Configuration Registers, in an even byte of Attribute Memory (address of Configuration Register 0), is given in this field. The system uses this information together with the information in the Configuration Register's presence mask field to determine which registers are present on the card and where they are located in the card's Attribute Memory space.

The Base Address is a field which may be from 1 to 4 bytes long depending upon the number of bits needed to represent it. High order address bits, not present in the field, are always interpreted to be zeros. The size of this field in bytes is one greater than the value of the *TPCC_RASZ* size field in the *TPCC_SZ* byte.

It is recommended, but not required, that the Configuration Register's Base Address be placed near the beginning of the Attribute Memory space.

3.3.4.4 TPCC_RMSK: Configuration Register Presence Mask Field for Interface Tuple

TPCC_RMSK: **Configuration Register Presence Mask Field for Interface Tuple**

7	6	5	4	3	2	1	0
Configuration Register Presence Mask for Registers 7 to 0 (this byte is always present)							
Configuration Register Presence Mask for Registers 15 to 8 this byte is present only if TPCC_RMSZ ≥ 1)							
Etc.							

The presence mask for the Configuration Registers is given in this field. Each bit represents the presence (1) or absence (0) of the corresponding Configuration Register. The least significant bit represents the lowest Configuration Register represented by the byte, and the represented registers increase with higher bit significance. Those registers which would be indicated by bytes extending beyond the end of the field are not implemented on the card.

The system uses this information together with the information in the Configuration Registers Base Address field to determine which registers are present on the card, and where they are located in the card's Attribute Memory space.

The address of each configuration register is given by the formula:

Base Address + (Register Number * 2)

regardless of whether or not the register is present.

3.3.4.5 TPCC_SBTPL: Configuration tuple Sub-tuples

The *TPCC_SBTPL* field extends to the end of the tuple. It contains tuple formatted information relating to the card's configuration. These tuples are subtuples of the CISTPL_CONFIG tuple and cannot extend beyond the end of the configuration tuple. The chain of subtuples is ended by reaching an FFH subtuple, or reaching the end of the CISTPL_CONFIG tuple. Any subtuple is invalid if it contains a link field which links beyond the first byte of the tuple following the CISTPL_CONFIG tuple. The tuple codes of the subtuples have meaning only within the CISTPL_CONFIG tuple. Subtuple codes from 80H through BFH are vendor specific whereas the rest are reserved for standard definition. The only subtuples presently defined are the list terminating subtuple, FFH (a 1-byte-long subtuple), and the CCSTPL_CIF, custom interface subtuple.

3.3.4.5.1 CCSTPL_CIF: Custom Interface Subtuples

The custom interface subtuple provides a code which uniquely identifies a card supported interface. These interfaces are referenced in the configuration entry tuples (CISTPL_CFTABLE_ENTRY, TPCE_IF field) by their relative positions in the CISTPL_CONFIG tuple. A 1 to 4 byte, unique, Interface Number may be followed by series of strings which describe the interface.

A maximum of four custom interface subtuples may be present in the configuration tuple (CISTPL_CONFIG).

There are two classes of custom interfaces. The first class of custom interfaces are those jointly defined by PCMCIA and JEIDA. These interfaces have a published electrical interface which is documented in ***Appendix C*** in the ***Electrical Specification***. The second class of custom interfaces are those which have an assigned custom vendor interface code but no published interface definition within the ***PC Card Standard***.

The first-byte codes — 41H, 81H and C1H in the field STCI_IFN— define two (XX41H), three (XXXX81H) and four (XXXXXXC1H) byte numbers for custom interfaces which are jointly defined by PCMCIA and JEIDA. The custom interface codes defined by PCMCIA/JEIDA are provided in the following table:

STCI_IFN Code	STCI_IFN_1 Code	STCI_IFN_2 Code	STCI_IFN_3 Code	CIF Name
41H	00H	N/A	N/A	Reserved
41H	01H	N/A	N/A	Zoomed Video
41H	02H	N/A	N/A	Digital Video Broadcasting
41H	03-FFH	N/A	N/A	Reserved for future use.
81H	00-FFH	00-FFH	N/A	Reserved for future use.
C1H	00-FFH	00-FFH	00-FFH	Reserved for future use.

Vendors that wish to define proprietary custom interfaces may do so by requesting a custom vendor interface code assignment. To simplify initial assignments, manufacturers who possess a JEDEC programmable semiconductor manufacturer's ID may use first-byte codes 42h, 82h and C2h followed by a second byte, which is their JEDEC ID, to generate interface ID numbers. For example, with yy indicating values selected by the manufacturer and jj indicating their JEDIC ID, the interface ID numbers jj42h, yyjj82h, and yyyyjjC2h are appropriate. Currently, no custom vendor interface codes have been assigned.

The subtuple is defined as follows:

***CCSTPL_CIF*: Custom Interface Subtuples**

Byte	7	6	5	4	3	2	1	0
0	ST_CODE	CCSTPL_CIF (C0H)						
1	ST_LINK	Link to next tuple (n; minimum 1)						
2	STCI_IFN	Interface ID Number LSB - 1 to 4 bytes least significant byte first This field provides a unique 8- to 32- bit number which identifies the interface. Within the ID number are two bits (bits 6 and 7 of byte 0) which give the size of this field. The first two or three bytes of the ID number is assigned by PCMCIA, JEIDA or its designer. When a vendor has been assigned a two or three byte initial portion the remaining one or two bytes of the field are assigned by that vendor to uniquely identify the interface.						
	STCI_IFN_SIZE		STCI_IFN_BASE					
3	STCI_IFN_1	2nd Interface ID Number byte, exists if STCI_IFN_SIZE is 1, 2 or 3.						
4	STCI_IFN_2	3rd Interface ID Number byte, exists if STCI_IFN_SIZE is 2 or 3.						
5	STCI_IFN_3	4th Interface ID Number byte, exists if STCI_IFN_SIZE is 3.						
m	STCI_STR	Interface Description Strings A list of strings, the first being ISO 646 IRV coded and the rest being coded as ISO alternate language strings with the initial escape character suppressed. Each string is terminated by a 0 byte and the last string, if it does not extend to the end of the subtuple, is followed by an FFH byte.						

3.3.5 CISTPL_CONFIG_CB: CardBus PC Card Configuration Tuple

CISTPL_CONFIG_CB and *CISTPL_CONFIG* are extremely similar, although CISTPL_CONFIG_CB has only one variable size field: *TPCC_SBTPL*, the configuration sub-tuple, compared to three variable fields in CISTPL_CONFIG. *CISTPL_CONFIG_CB* gives the location of the CardBus PC Card status registers for the associated function in place of the configuration registers for a 16-bit PC Card identified by CISTPL_CONFIG.

CISTPL_CONFIG_CB indicates via Configuration sub-tuple(s) which type(s) of custom interface(s), if any, are supported by the CardBus PC Card.

The Card Configuration Table allows the system to determine the characteristics of the card for each allowable configuration. The system may then configure the card into any allowable configuration, or leave the card unconfigured. It may use the configuration to gracefully reject the card if the card is found to be incompatible with the system hardware and software.

Table 3-42 CISTPL_CONFIG_CB: CardBus PC Card Configuration Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_CONFIG_CB (04H)						
1	TPL_LINK	Link to next tuple (n-1; minimum 6)						
2	TPCC_SZ	Size of fields (must be 03H).						
3	TPCC_LAST	Index Number of the last entry in the Card Configuration Table.						
4··7	TPCC_ADDR	Pointer to CardBus PC Card status registers.						
8	TPCC_RSVD	Reserved area 0··3 bytes. Must be 0 bytes.						
8··n	TPCC_SBTPL	The rest of the tuple is reserved for subtuples containing standardized optional additional information related to the Card Configuration.						

3.3.5.1 TPCC_SZ: Size of Fields Byte

***TPCC_SZ:* Size of Fields Byte**

7	6	5	4	3	2	1	0
TPCC_RFSZ (Reserved Size, must be 0).		Reserved, must be 0.				Must be 3 (All TPCC_ADDR status registers are 4 bytes).	

3.3.5.2 TPCC_LAST: Card Configuration Table Last Entry Index

The Card-Configuration-Table size is a one-byte field which contains the Configuration Index Number of the last configuration described in the Card Configuration Table. Once the host encounters this configuration, when scanning for valid configurations, it will have processed all valid configurations.

***TPCC_LAST:* Card Configuration and Last Entry Index**

7	6	5	4	3	2	1	0
RFU 0		Last Index					

Field	Description
Last Index	The Index Number of the final entry in the Card Configuration Table (the last entry encountered when scanning the CIS).
RFU 0	This area is reserved for future standardization and must be 0.

3.3.5.3 TPCC_ADDR: CardBus PC Card Status Register Pointer

The encoding of this address is the same as for the *TPLL_ADDR* field in the CISTPL_LONGLINK_CB tuple and the CIS Pointer, with the additional restriction that CardBus PC Card status registers can only be placed in one of the memory spaces. The applicable values of the Address Space Indicator and the Address Space Offset fields are given below.

When the status registers are not implemented, the Address Space Indicator must be set to zero(0).

***TPCC_ADDR*: CardBus PC Card Status Register Pointer Layout**

DWORD	31	…	4	3	2	1	0
	Address Space Offset				Address Space Indicator		
Memory Space	offset within Base Address Register memory space				replace with 0s		

Field	Description		
Address Space Indicator	**Value**	**Meaning**	
	0	The status registers are not implemented.	
	1··6	The status registers are in the memory address space governed by one of the six Base Address Registers. For example, if the value is 2, then the status registers are in the memory address space governed by Base Address Register 2.	
	7	Reserved.	
Address Space Offset	**Address Space Indicator**	**Space Type**	**Address Space Offset Values**
	0	Reserved.	If Address Space Indicator is set to 0, this field must be set to 0.
	x; 1 ≤ x ≤ 6	memory space	0 ≤ value ≤ FFFFFFF8H. This is the offset into the memory address space governed by Base Address Register x. Adding this value to the value in the Base Address Register gives the location of the start of the status registers.
	7	Reserved.	Undefined.

3.3.5.4 TPCC_SBTPL: Additional Information Stored in Tuple Format

The *TPCC_SBTPL* field in the CISTPL_CONFIG_CB tuple does not have any sub-tuples defined.

3.3.6 CISTPL_PWR_MGMNT: Function State Save/Restore Definition

The function state save/restore definition tuple identifies the method of PC Card function state preservation supported by the function. When the PC Card function does not store state information on the card the tuple describes the size and location of a buffer through which the PC Card function state is retrieved when state is saved and returned to the card when state is later restored.

The CIS for a PC Card function shall contain at most one CISTPL_PWR_MGMNT tuple. Multiple Function PC Cards shall never place a CISTPL_PWR_MGMNT tuple in the PC Card's global CIS.

Instead, Multiple Function PC Cards shall place a CISTPL_PWR_MGMNT tuple in the function-specific CIS of each function that indicates the Power Management Support Register is present in the function's Configuration Register array.

Table 3-43 CISTPL_PWR_MGMNT: Function State Save/Restore Definition

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_PWR_MGMNT (08H)						
1	TPL_LINK	Link to next tuple (at least 6 when PWR_METHOD = 80H, at least 14 in all other cases)						
2	PWR_METHOD	type of support provided by the PC Card for saving/restoring card state 00H = PC Card state is saved/restored through buffer in Attribute space 01H = PC Card state is saved/restored through buffer in Common space 02H··7FH = Reserved for future use. 80H = PC Card state is saved in non-volatile storage on the PC Card 81H··FFH = Reserved for future use.						
3	PWR_TIME	Save/Restore time						
4··7	PWR_REG_ADDR	Offset of Power Management Support Register in Attribute Space						
8··11	PWR_OFFSET	Offset of save/restore buffer						
12··15	PWR_BUFFSIZE	Size in bytes of save/restore buffer						

The *PWR_METHOD* field identifies the type of support provided by the PC Card function for saving and restoring the state of the function. Two methods of state save/restore support are defined. When the *PWR_METHOD* field is 80H, the PC Card function uses storage on the card when saving and restoring the state of the function. When the *PWR_METHOD* field is 00H or 01H, the PC Card transfers the state of the function through a buffer and relies on storage in the host when saving and restoring the state of the function.

The *PWR_TIME* field is used by the PC Card function to specify the period of time required by the function to save and/or restore the function's state. The granularity of this count is four milliseconds (4 ms). A PC Card requiring eighty milliseconds (80 ms) to save and/or restore its state encodes a value of twenty (14H) in this field. The maximum value that can be encoded in the *PWR_TIME* field is FFH, corresponding to one thousand twenty milliseconds (1020 ms).

The *PWR_REG_ADDR* field indicates the location of the PC Card function's Power Management Support Register in Attribute Space. This address corresponds with register ten (10) at offset twenty (20) in the PC Card function's array of Configuration Registers. (See the ***Electrical Specification***.)

The *PWR_OFFSET* field is the address of the PC Card function's save/restore buffer in a card address space. When the *PWR_METHOD* field is 01H the save/restore buffer for the function is in Common Memory space. When the *PWR_METHOD* field is 00H the save/restore buffer for the function is in Attribute Memory space. Note that in Attribute Memory space only even bytes in the address range are valid and therefore the offset of the buffer must be a multiple of two (2).

The *PWR_BUFFSIZE* field indicates the number of bytes to be saved during a save operation and then later restored during a restore operation. When the *PWR_METHOD* field is 00H, indicating a save/restore buffer in Attribute Memory space, only even bytes in the address range are counted. For example, if *PWR_METHOD* = 00H, *PWR_OFFSET* = 0100H, and *PWR_BUFFSIZE* = 20H , the significant data bytes are the thirty-two (20H) bytes at even addresses in the range 0100H through 013EH in Attribute space.

4. DATA RECORDING FORMATS (LAYER 2)

This level defines the data-recording format for the card. If none of the layer-2 headers are present, software should assume that the card is organized as an unchecked sequence of bytes.

Card data-recording formats fall into two categories:

- Disk-like — the card consists of a number of data blocks, where each block consists of a fixed number of bytes. These blocks correspond to the sectors of rotating disk drives. Conceptually, an entire block must be updated if any byte in the block is to be changed.
- Memory-like — the card is treated as a sequence of directly-addressable bytes of data. Formats are further categorized according to how error checking is performed.

This Standard recognizes four basic possibilities:

- Unchecked — no data checking is performed at the data-format layer.
- Checked with in-line codes — data checking is performed by the data-format layer using check codes. The check code for a given block is recorded immediately after the block.
- Checked with out-of-line codes — data checking is performed by the format layer using check codes. The check code for a given block is recorded in a special table that resides separately from the data blocks.
- Checked over entire partition — data checking is performed only over the complete partition.

This Standard recognizes two kinds of check codes: arithmetic checksum and CRC. Arithmetic checksums are typically one-byte or two-bytes long; CRCs are always two-bytes long.

When cards with 16-bit or 32-bit data paths are used to record byte data, it is necessary to specify how the bytes of the data card correspond to sequential bytes of data. In this Standard, all disk-like organizations require that bytes be assigned to words, with the lowest-byte-address mapping to the least-significant byte of the word, and subsequent-byte-address mapping to increasingly significant bytes.

Memory-like formats also require that the byte mapping be specified. For maximum flexibility, both little-endian and big-endian byte orders are supported.

4.1 Card Information Tuples

The tuples listed in the following subsections provide generic information about how the card is to be used.

4.1.1 CISTPL_BATTERY: The Battery-Replacement Date Tuple

This optional tuple shall be present only in cards with battery-backed storage. It indicates the date at which the battery was replaced, and the date at which the battery is expected to need replacement.

Only one CISTPL_BATTERY tuple is allowed per card.

Its format is given below:

Table 4–1 CISTPL_BATTERY: Battery Replacement Date Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_BATTERY (45H).						
1	TPL_LINK	Link to next tuple (at least 4).						
2··3	TPLBATT_RDAY	date battery last replaced.						
4··5	TPLBATT_XDAY	date battery due for replacement.						

Bytes 2-3, *TPLBATT_RDAY*, indicate the date on which the battery was last replaced. This field has the same interpretation as the field TPLDATE_DAY in the CISTPL_DATE tuple.

Bytes 4-5, *TPLBATT_XDAY*, (the "expiration day") indicate the date on which the battery should be replaced. This field has the same format as the field *TPLDATE_DAY* in the CISTPL_DATE tuple.

If either TPLBATT_RDAY or TPLBATT_XDAY are zero, it indicates that the corresponding date was not known when the tuple was written.

4.1.2 CISTPL_DATE: The Card Initialization Date Tuple

This optional tuple indicates the date and time at which the card was formatted. Only one CISTPL_DATE tuple is allowed per card. The layout of the tuple is shown below.

Table 4–2 CISTPL_DATE: Card Initialization Date Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_DATE (44H).						
1	TPL_LINK	Link to next tuple (at least 4).						
2··3	TPLDATE_TIME	time the card was initialized.						
4··5	TPLDATE_DAY	date the card was initialized.						

Bytes 2-3, *TPLDATE_TIME*, indicate the time at which the card was initialized. It is a 16-bit packed structure with the following format:

***TPLDATE_TIME* field**

Byte	7	6	5	4	3	2	1	0
0	$\mathrm{MIN}_{\mathrm{LO}}$			SSS				
1	HHH					$\mathrm{MMM}_{\mathrm{HI}}$		

- The field HHH contains the hour at which the card was initialized. It is a binary number between 0 and 23.
- The field MMM contains the minute at which the card was initialized. It is a binary number between 0 and 59. MMM is a six bit field constructed by combining $\mathrm{MMM}_{\mathrm{HI}}$ and $\mathrm{MMM}_{\mathrm{LO}}$.
- The field SSS represents the two-second interval at which the card was initialized. It is a binary number between 0 and 29. To convert SSS to seconds, it should be multiplied by two.

Bytes 4-5, *TPLDATE_DAY,* indicate the date the card was initialized. It is a 16-bit packed structure with the following format:

***TPLDATE_DAY* field**

Byte	7	6	5	4	3	2	1	0
0	$\mathrm{MON}_{\mathrm{LO}}$			DAY				
1	YYYY							$\mathrm{MON}_{\mathrm{HI}}$

- The field *YYYY* represents the year. It is a binary number between 0 and 127, with 0 representing the year 1980.
- The field *MON* represents the month. It is a binary number between 1 and 12, with 1 representing January. *MON* is a four bit field constructed by combining MON_{HI} and MON_{LO}.
- The field *DAY* represents the day. It is a binary number between 1 and 31.

If the date and time components of the date are both zero, this should be taken as an indication that the date and time were unknown when the card was first initialized.

4.1.3 CISTPL_VERS_2: The Level-2 Version and Information Tuple

The Level-2 information tuple serves to introduce information pertaining to the logical organization of the card's data. The layout of the Level-2 tuple is shown below. This tuple should appear only one time in a CIS.

Table 4–3 CISTPL_VERS_2: Level-2 Information Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_VERS_2 (40H).						
1	TPL_LINK	Link to next tuple (at least m-1).						
2	TPLLV2_VERS	Structure version (00H).						
3	TPLLV2_COMPLY	Level of compliance claimed.						
4··5	TPLLV2_DINDEX	Byte address of first data byte in card (LSB first).						
6	TPLLV2_RSV6	Reserved; must be zero.						
7	TPLLV2_RSV7	Reserved; must be zero.						
8	TPLLV2_VSPEC8	Vendor-specific byte.						
9	TPLLV2_VSPEC9	Vendor-specific byte.						
10	TPLLV2_NHDR	Number of copies of CIS present on the device.						
11··k	TPLLV2_OEM	Vendor of software that formatted card (ASCII, variable length, terminated with NULL [00H]).						
k+1··m	TPLLV2_INFO	Informational message about the card (ASCII, variable length, terminated with NULL [00H]).						

TPLLV2_VERS represents the standardization version of the tuple. This byte should always be zero.

TPLLV2_COMPLY indicates the claimed degree of compliance with this Standard. At present, this should always be zero.

TPLLV2_DINDEX specifies the address of the card's first data byte. Setting this to non-zero reserves bytes at the beginning of Common Memory.

Note: The first data byte on the card must always be somewhere in the first 64 KBytes of the card. This field should be consistent with information provided in the format tuple (if that tuple is present).

TPLLV2_NHDR specifies the number of copies of the CIS that are present on the card. For compatibility with this Standard, this value should be 1.

TPLLV2_OEM specifies the vendor of the machine, or format program, that formatted the card. This is a text string terminated by a NULL byte (00H). The value of *TPLLV2_OEM*, combined with the value of *TPLLV2_INFO*, determines how to interpret vendor-specific fields in the Level-2 tuples. For alternate languages, CISTPL_ALTSTR tuples may follow this tuple, specifying the string value to be substituted when using alternate languages.

TPLLV2_INFO contains a text message terminated by a NULL byte (00H). This message should be displayed to users, by a computer, whenever the host needs to describe the type of card that is in the drive. For alternate languages, CISTPL_ALTSTR tuples may follow this tuple, specifying the string value to be substituted when using alternate languages.

Note: If the host system's format routine determines that the card is already formatted, it may display a message like: Caution! This card contains data for <info>, from <vendor>. The contents of the information field should be chosen appropriately. For example, a VCR setup card for a VCR by AlphaBeta Electronics might have <info> as "Model 9770 VCR" and the <vendor> field would be "AlphaBeta".

The characters used in *TPLLV2_INFO* and *TPLLV2_OEM* shall be chosen from the printing 7-bit ISO 646 IRV set (codes 20H through 7EH, inclusive).

TPLLV2_RSV6 and *TPLLV2_RSV7* are reserved for use in future versions of this Standard. They shall be set to zero.

TPLLV2_VSPEC8 and *TPLLV2_VSPEC9* are vendor specific. If not used, they shall be set to zero.

4.2 Data Recording Format Tuples

All information about a card's data-recording format is given in special tuples in the Card Information Structure. Each card that conforms to this Standard's Layer 2 shall contain at least one format tuple defining how the data is recorded on the card.

If the format is disk-like, the format tuple may be followed by a geometry tuple. This tuple indicates the cylinder, track and sector layout for operating environments that need to treat all mass-storage devices in that way.

If the format is memory-like, the format tuple may be followed by a byte-order tuple. The byte-order tuple specifies two independent (but related) parameters:

- How multi-byte numbers are recorded on the media, and
- How byte addresses are assigned within each word (for cards with 16-bit or wider data-paths).

4.2.1 CISTPL_BYTEORDER: The Byte-Order Tuple

This tuple shall only appear in a partition tuple set for a memory-like partition. It specifies two parameters: the order for multi-byte data, and the order in which bytes map into words for 16-bit cards.

Table 4–4 CISTPL_BYTEORDER: Byte Order Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_BYTEORDER (43H).						
1	TPL_LINK	Link to next tuple; should be at least 2.						
2	TPLBYTE_ORDER	Byte order code: see Byte Order Codes table						
3	TPLBYTE_MAP	Byte mapping code: see Byte Mapping Codes table						

Byte 2, *TPLBYTE_ORDER*, specifies the byte order for multi-byte numeric data. Symbolic codes for this field begin with the text “TPLBYTEORD_”, and are listed below.

TPLBYTE_ORDER Byte Order Codes.

Code	Name	Description
0	TPLBYTEORD_LOW	Specifies that multi-byte numeric data is recorded in little-endian order.
1	TPLBYTEORD_HIGH	Specifies that multi-byte numeric data is recorded in big-endian order.
02H··7FH		Reserved for future standardization.
80H··FFH	TPLBYTEORD_VS	Vendor-specific.

Byte 3, *TPLBYTE_MAP,* specifies the byte mapping for 16-bit or wider cards. Symbolic codes for this field begin with the text “TPLBYTEMAP_”, and are listed below.

TPLBYTE_MAP Byte Mapping Codes.

Code	Name	Description
0	TPLBYTEMAP_LOW	Specifies that byte 0 of a word is the least-significant byte (multi-byte cards).
1	TPLBYTEMAP_HIGH	Specifies that byte 0 of a word is the most-significant byte (multi-byte cards).
02H··7FH		Reserved for future standardization.
80H··FFH	TPLBYTEMAP_VS	Vendor-specific.

If a byte-order tuple is not present, the data shall be recorded using little-endian byte order, *TPLBYTEORD_LOW*, and shall be mapped with byte 0 of each word corresponding to the least-significant byte, *TPLBYTEMAP_LOW*.

For applications involving DOS file systems, little-endian byte order and low-to-high byte mapping are mandatory.

4.2.2 CISTPL_FORMAT, CISTPL_FORMAT_A: The Format Tuples

The format tuples define the data recording format for a region (usually all) of a card. Layouts are shown below.

Table 4–5 CISTPL_FORMAT and CISTPL_FORMAT_A: Format Tuples

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_FORMAT (41H) or CISTPL_FORMAT_A (47H).						
1	TPL_LINK	Link to next tuple (n-1: at least 12, typically 20)						
2	TPLFMT_TYPE	Format type code (TPLFMTTYPE_xxx)						
3	TPLFMT_EDC	Error-detection method and length of error-detection code						
	RFU	Error-detection-code type.			EDC Length			
4··7	TPLFMT_OFFSET	Byte address of the first data byte in this partition.						
8··11	TPLFMT_NBYTES	Number of data bytes in this partition.						
12··n		Additional information, interpreted based on value of TPLFMT_TYPE.						

Note: Bytes 4··7, *TPLFMT_OFFSET*, contain the four byte address of the first data byte in this partition. In CISTPL_FORMAT, defining either CardBus PC Card memory or 16-bit PC Card Common Memory, this value is the physical offset in the partition. In CISTPL_FORMAT_A, defining 16-bit PC Card Attribute Memory only, the value in this field must be multiplied by 2 to obtain the physical address of the first data byte. This is done for consistency with the usage of the offset field in the Checksum tuple.

Each format tuple implicitly begins a partition tuple set. Subsequent geometry, byte order, software interleave and data organization tuples are implicitly associated with the preceding format tuple.

Byte one of the tuple specifies the link to the next tuple, and, therefore, (implicitly) the length of this tuple. Two ranges of values are permitted. Normally, the value will be at least 20 (014H), however, if the format tuple is specifying a memory-like format, the value may be as little as 12 (0CH), as bytes 13 through 21 must be zero for memory-like formats. If the partition does not use error-detecting codes, then the *TPLFMT_EDCLOC* field may be omitted.

Byte two of the tuple, *TPLFMT_TYPE*, specifies the kind of format used for this partition. The permitted values for this field are given below.

TPLFMT_TYPE Format Type Codes.

Code	Name	Description
0	TPLFMTTYPE_DISK	This partition uses a disk-like format.
1	TPLFMTTYPE_MEM	This partition uses a memory-like format.
02H··7FH		(Reserved for future standardization.)
80H··FFH	TPLFMTTYPE_VS	This partition uses a vendor-specific format.

- Byte 3, *TPLFMT_EDC*, specifies the error-detection method, and the length of the error-detection code. Byte 3 is generally only meaningful for disk-like formats. Bit 7 is reserved; it must be zero. Bits 3-6 specify the error-detection code. The legal values are given in Table 5-48. Bits 0-2, *TPLFMT_EDCLEN*, specify the length in bytes of the error-detection code; this is a number between 0 and 7. The legal values for the length field are determined by the error-detection method in use.
- Memory-like regions may use the PCC method of error detection.

TPLFMT_EDC Error Detection Type Codes

Code	Name	Description
0	TPLFMTEDC_NONE	No error-detection code is used. If the length field is non-zero, space will be reserved for the check code, but no checking will be performed.
1	TPLFMTEDC_CKSUM	An arithmetic checksum is used to check the data. The length field must be 1 if this code is selected. See text for details in calculating the checksum.
2	TPLFMTEDC_CRC	A cyclical redundancy check is used to check the data. The length field must be 2 if this code is selected. The CRC value is always recorded low-order byte first.
3	TPLFMTEDC_PCC	An arithmetic checksum is used to check the data. However, a single checksum is provided for the entire data partition. This technique is intended for use with static data on ROM or OTPROM cards. The PCC code itself is recorded in byte 18 of the tuple (field TPLFMT_EDCLOC, byte 0). The length field must be 1 if this option is selected.
4H··7H		(Reserved for future standardization.)
8H··FH	TPLFMTEDC_VS	A vendor-specific method of error checking is used.

The code in *TPLFMT_EDC* only specifies the method to be used to verify data integrity. The value in *TPLFMT_EDCLOC* must be consulted to determine whether the code is to be interleaved with the data, or stored in a separate table.

Bytes 4-7, *TPLFMT_OFFSET*, specify the absolute byte address of the first data byte governed by this tuple. The value is stored as a 32-bit quantity with LSB first.

Bytes 8-11, *TPLFMT_NBYTES*, specify the number of bytes in the partition, including (if present) the error-detection codes. The value is stored as a 32-bit quantity with LSB first.

4.2.2.1 The Format Tuple for Disk-like Regions

When the *TPLFMT_TYPE* field of the format tuple has the value TPLFMTTYPE_DISK, bytes 12 through 21 of the tuple are interpreted as shown in below.

Table 4–6 Format Tuple for Disk-like Regions

Byte	7	6	5	4	3	2	1	0
12··13	TPLFMT_BKSZ Block size. For unblocked formats, this value should be 0. This field corresponds to the number of data bytes per sector. The value in this field must be a power of 2.							
14··17	TPLFMT_NBLOCKS Number of data blocks in this partition.							
18··21	TPLFMT_EDCLOC Location of the error-detection code. If zero, the error-detection code is interleaved with the data blocks. If non -zero, the error-detection code is stored in a linear table starting at the specified address on the card. If using PCC, the first byte of this field contains the check code and bytes 19··21, if present, must be zero.							
22	TPLFMT_BAR CardBus PC Card Base Address Register Indicator (CardBus PC Card only).							

Bytes 12-13, *TPLFMT_BKSZ*, specify the number of data bytes in each block in the partition. This value does not include error-check bytes. The value in this field must be a power of 2 between 128 and 2048. This Standard recommends that 512 be used wherever possible. The value is stored as a 16-bit quantity with LSB first.

Bytes 14-17, *TPLFMT_NBLOCKS*, specify the number of data blocks in the partition. This value is stored as a 32-bit quantity with LSB first. The quantity:

TPLFMT_NBLOCKS *(*TPLFMT_BKSZ* + *TPLFMT_EDCLEN*)

shall be less than or equal to *TPLFMT_NBYTES*.

Bytes 18-21, *TPLFMT_EDCLOC*, specify where the error-detection codes are stored. This value is stored as a 32-bit quantity with LSB first. If the value stored in this location is zero, or if this field is not present, then codes (if present) are interleaved with the data blocks, with the code for a given data block to follow immediately after that block. If the value stored in this location is non-zero, it shall be the address of the first byte of the error-detection code table. This table shall be an array of values, with *TPLFMT_NBLOCKS* entries, containing the error-detection codes for each data block. Each entry in the table shall have a byte length as indicated by *TPLFMT_EDCLEN*. The value stored in *TPLFMT_EDCLOC* shall be at least the value in *TPLFMT_OFFSET*, and shall be no greater than the result of *TPLFMT_OFFSET* + *TPLFMT_NBYTES* - (*TPLFMT_EDCLEN* · *TPLFMT_NBLOCKS*). In other words, the table must be entirely contained in the range of bytes between *TPLFMT_OFFSET* and *TPLFMT_OFFSET* + *TPLFMT_NBYTES* - 1. Since the first data byte of block 0 resides at *TPLFMT_OFFSET*, this Standard requires that the EDC table appear after all the data blocks in the partition. This Standard does not require that the table occur immediately after the last block, nor does it preclude the use of spare space for vendor-specific purposes.

If PCC error checking is selected, then the *TPLFMT_EDCLOC* field is used to add the actual CRC value, rather than pointing to the cell that holds the PCC.

The bit *TPLFMT_EDC_RFU* is reserved for future use and shall always be zero.

The following table summarizes some possible error-detection strategies.

Error Detection Format Summary

EDC Format	EDC Length	EDC Location	Description
TPLFMTEDC_NONE	0	0	No error checking is performed and no room is reserved for error-detection tables. The data blocks are recorded sequentially.
TPLFMTEDC_NONE	2	0	No error checking is performed, but room is reserved for a two-byte error-detection code after each data block.
TPLFMTEDC_NONE	1	non-zero	No error checking is performed, but room is reserved for an out-of-line table of error-detection codes, with one byte per data block. The data blocks themselves are recorded contiguously.
TPLFMTEDC_CKSUM	1	non-zero	Data is checked using a one-byte arithmetic checksum of the data. The checksum is stored in an out-of-line table. The data blocks themselves are recorded contiguously.
TPLFMTEDC_CRC	2	0	Data is checked using SDLC CRC codes. The check-code for a data block is stored immediately following the data block.
TPLFMTEDC_PCC	1	special	Entire partition is checked using a one byte arithmetic checksum. The checksum is stored in the TPLFMT_EDCLOC field of the tuple itself.

Byte 22, *TPLFMT_BAR*, is present only on CardBus PC Cards and indicates the Base Address Register which is being defined. The *TPLFMT_BAR* field is identical for disk-like and memory-like partitions. (See ***4.2.2.2 The Format Tuple for Memory-like Regions***.)

4.2.2.2 The Format Tuple for Memory-like Regions

When the *TPLFMT_TYPE* field of the format tuple has the value TPLFMTTYPE_MEM, bytes 12 through 21 of the tuple are interpreted as shown below.

Table 4–7 Format Tuple for Memory-like Regions

Byte	7	6	5	4	3	2	1	0
12	TPLFMT_FLAGS Various flags							
	(Reserved)						AUTO	ADDR
13	(Reserved; must be zero)							
14··17	TPLFMT_ADDRESS Physical address at which this memory partition should be mapped, if so indicated by **TPLFMT_FLAG**. Four bytes stored with LSB first.							
18··21	TPLFMT_EDCLOC Error-detection code location, with same meaning as for disk-like regions. Used for PCC checking only. Byte 18 holds the check value, and bytes 19··21 must be zero or omitted.							
22	TPLFMT_BAR CardBus PC Card Base Address Register Indicator (CardBus PC Card only).							

As with the format tuple for disk-like regions, zero bytes may be omitted at the end of the format tuple for memory-like regions. To do this, the next tuple must begin immediately after the last non-zero byte in the format tuple.

Byte 12, *TPLFMT_FLAGS*, contains several control bits.

- Bit 0, *TPLFMTFLAGS_ADDR*, if set, indicates that bytes 14-17, *TPLFMT_ADDRESS*, represent a physical address to be associated with the first byte of this region. If clear, bytes 14-17 do not represent a physical address and must be zero.
- Bit 1, *TPLFMTFLAGS_AUTO*, tells the system whether to automatically map the region into memory when the card is inserted (or at system start-up). If set, the system should attempt to map the region into memory. If *TPLFMTFLAGS_ADDR* is set, the system should attempt to map the code at the specified address. A system shall ignore these fields if it cannot perform the specified mapping. It may also, at the designer's option, ignore these fields even if it could perform the mapping.

Byte 13 is reserved and, if present, must be set to zero.

Bytes 14 through 17, *TPLFMT_ADDRESS*, represent the physical address at which the partition should be mapped into the host's address space. For x86 architecture machines, this is a linear address, not a segment/offset address. If the flag *TPLFMTFLAGS_ADDR* is not set, then this field is reserved and must be zero.

Note: A system can comply fully with this Standard and not honor these fields. The automatic mapping feature is not intended for general-purpose use, or for building interchangeable BIOS extensions for general-purpose systems. This Standard's automatic mapping feature is included for use in low-cost embedded systems, and is not intended as a general, eXecute-In-Place, specification. Many important issues are deliberately not addressed here, such as what to do when the card is removed, or how to resolve conflicts when cards in different sockets both need to be mapped to a specific physical address. It is anticipated that general purpose DOS-based systems will ignore these fields.

Bytes 18-21, *TPLFMT_EDCLOC*, have the same meaning for memory-like regions that they have for disk-like regions. A memory-like region has only two options for error checking: none or PCC. Therefore, if this field is used, byte 18 contains the check code, and bytes 19-21 are reserved and must be zero.

Byte 22, *TPLFMT_BAR*, is present only on CardBus PC Cards and indicates the Base Address Register which is being defined.

***TPLFMT_BAR* field**

7	6	5	4	3	2	1	0
Reserved, must be 0.					Address Space Indicator		

Field	Description
Address Space Indicator	Same encoding as the *Address Space Indicator* field in the CISTPL_LONGLINK_CB tuple. 0 illegal value - zero indicates CardBus PC Card configuration space. 1 Base Address Register 1. 2 Base Address Register 2. 3 Base Address Register 3. 4 Base Address Register 4. 5 Base Address Register 5. 6 Base Address Register 6. 7 the Expansion ROM Base Address Register.

4.2.2.3 Arithmetic Checksums As Error-Detection Codes

Arithmetic checksums shall be computed by summing together all the data bytes of the block using eight-bit, twos-complement addition and ignoring any overflow that occurs. The resulting sum shall be stored in an external table (for block-by-block checksum,) or in the format tuple itself (for PCC checking). This method has the disadvantage that the checksum of a block of zero data is also zero; however, it is consistent with current practice.

4.2.2.4 CRC Error-Detection Codes

CRC codes shall be computed using the SDLC algorithm[1].

The SDLC CRC has a convenient property. When the check code is appended to the data stream, and the algorithm is run on the result, the remainder will always be $x^{12} + x^{11} + x^{10} + x^{8} + x^{3} + x^{2} + x^{1} + x^{0}$ (assuming that neither the data nor the CRC have been corrupted).

Despite its complicated formal definition, the SDLC CRC is quite easy to compute both in hardware and in software. Commercially available chips (such as the 9401) can compute the CRC directly from a serial-data stream. There are several well-known methods for computing the CRC, one-byte-at-a-time, using a lookup table. Even so, computing a CRC in software is somewhat slower than computing a simple checksum.

4.2.2.5 Byte Mapping for Disk-Like Media

Within a disk-like partition, cards with 16-bit data paths shall be byte mapped with the lowest-byte address of each word corresponding to the least-significant byte of that word, and with increasing-byte addresses corresponding to increasingly significant bytes.

[1] Also known as CRC-ITU-T(CCITT) or HDLC CRC. In this algorithm, the data to be checked is considered as a serial-bit stream, with the low-order bit of the first byte taken as the first bit of the stream. This bit stream is conceptually taken as the coefficients of a polynomial in x^n, where n is the number of bits in the stream, and where the first bit is the coefficient of the term in x^{n-1}. This polynomial is multiplied by x^{16} and then divided (modulo 2) by the polynomial $x^{16} + x^{12} + x^{5} + 1$, leaving a remainder of order 15 or less. If the register initial-condition is set to all ones, the CRC code for a block of all zeros will be non-zero. The one's complement of this remainder is the error-check code. It is recorded with the complemented coefficient of x^{15} as its least-significant bit, and with the complemented coefficient of x^{0} as its most-significant bit.

4.2.3 CISTPL_GEOMETRY: The Geometry Tuple

This tuple shall only appear in partition descriptors for disk-like partitions. It provides instructions to those operating systems that require that all mass-storage devices be divided into cylinders, tracks and sectors.

Table 4–8 CISTPL_GEOMETRY: Geometry Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_GEOMETRY (42H).						
1	TPL_LINK	Link to next tuple. (at least 4)						
2	TPLGEO_SPT	Sectors per track.						
3	TPLGEO_TPC	Tracks per cylinder.						
4··5	TPLGEO_NCYL	Number of cylinders, total.						

Byte 2, *TPLGEO_SPT*, specifies the number of sectors per simulated track on the memory card. This is a number between 1 and 255. A value of zero is not permitted.

Byte 3, *TPLGEO_TPC*, specifies the number of tracks per simulated cylinder on the device. This is a number between 1 and 255. A value of zero is not permitted.

Bytes 4-5, *TPLGEO_NCYL*, specify the number of simulated cylinders on the device. This is a number between 1 and 65535, stored as a 16-bit integer with LSB first (this value is one greater than the same quantity as represented by the PC BIOS). This Standard records the number of simulated cylinders; the PC BIOS records the maximum cylinder number. Since cylinder numbers on the PC start at zero , the maximum cylinder number on the PC is one less than the number of cylinders.

The product

*TPLGEO_NCYL * TPLGEO_TPC * TPLGEO_SPT*

shall be less than or equal to the number of blocks recorded in field *TPLFMT_NBLOCKS* of the format tuple.

4.2.4 CISTPL_SWIL: Software Interleave Tuple

This tuple allows the software interleaving of data within a partition on a card. The presence of this tuple depicts that the partition that it is associated with is 2n (n = *TPL_SWIL_INTRLV*) interleaved. The presence of this tuple implies the non-sequential physical address sequence used to read and write data. The data is reconstructed after reading the data from the card using an n-way interleaved sequence. Note, this tuple is associated with the previous format tuple.

Table 4–9 CISTPL_SWIL: Software Interleave Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE		CISTPL_SWIL (23H).					
1	TPL_LINK		Link to next tuple (at least 2).					
2	TPL_SWIL_INTRLV		Value = n, where data is recorded via software interleaving by a factor of 2n. The partition total size divided by this interleaving factor defines the higher-order address "offset" by which addresses are incremented until the interleave "loop" is completed. Between loops, low-order addresses are incremented to the next physical word whereupon offset increments are used through the next loop (and so on). The value of n = 0 is reserved and shall not be used.					

Assumptions:

The software interleave tuple must be associated with a Format Tuple (partition tuple set) that describes a homogeneous partition beginning and ending on a physical device boundary and hence referring to one and only one Device Info entry and associated Device Geometry Info entry within the Device Information Tuple and the Device Geometry Tuple respectively.

The *DGTPL_HWIL* field within the Device Geometry Info entry must equal 1, i.e. non-hardware interleaved.

The following provide the association of the Software Interleave Tuple and the implied "software interleave partition" with the Device Information Tuple, Device Geometry Tuple and Format Tuple.

"software interleaved partition" base address = *TPLFMT_OFFSET*

"software interleaved partition" length = *TPLFMT_NBYTES*

The size or length of the partition must be a multiple of the stride (below).

"software interleaved partition" interleave width = *DGTPL_WBS* * *DGTPL_BUS*

"software interleaved partition" stride = *DGTPL_BUS* * (size in bytes of a physical device)

The device size is determined from the Device Info entry of the Device Information Tuple.

4.3 Standard Data Recording Formats

This Standard allows great flexibility in adjusting the card format to meet specific requirements. For simplicity, this Standard further specifies recommended formats for low-level data recording—formats which are expected to be commonly used. Higher level data formats will be specified in addition to this level.

- Generic — the bytes are recorded in 512-byte blocks with no error checking. The card's first data byte appears at byte address 512 (200H).
- Single-byte checksum format — the bytes are recorded in 512-byte blocks, with a separate region reserved for error-checking codes, and with a sector buffer.
- Two-byte Embedded CRC format — the bytes are recorded in 512-byte blocks, with each block followed by a 16-bit CRC.
- Raw- byte format — the bytes are recorded sequentially in an unblocked form.

In order to maintain a reasonable degree of interchangeability, this Standard recommends that all Layer-2 conforming implementations be able to read and write generic-format cards.

When an implementation is presented with a card, whose format is not supported by that implementation, the implementation shall refuse to write on the card, except to reinitialize it. If the basic format is not supported by the implementation at all, the implementation shall return an error to applications whenever they attempt to access the card. The implementation may allow read-only access to a card whose basic format is supported, but whose error-detecting code is not.

For example, an implementation that only supports the creation of generic-format cards could allow single-byte-checksum cards to be read by ignoring the checksum bytes. With a little more sophistication, the implementation could also allow embedded-CRC format cards to be read.

4.4 Mixed Data Formats

Layer 2 allows a card to have multiple partitions, with each partition having its own data-recording format. **In this case, there shall be one format tuple in the Card Information Structure for each partition on the card.** The additional tuples that refer only to a given partition shall appear immediately following the format tuple that defines the partition.

An implementation is not required to support multiple partitions on a single card. When presented with a card with multiple partitions, an implementation may:

- Only allow access to the first partition (if supported),
- Scan the CIS for the first partition of a supported type, and only allow access to that partition.
- Deny access to the card.

  Note: In most applications only one or two regions will be needed. (Two regions would be used by a ROM card that contained both executable ROM images and a DOS file system).

5. DATA ORGANIZATION (LAYER 3)

This layer defines the data organization of a particular partition on a memory card. At this level, the possibilities become complex. Some examples are:

- A partition can contain a DOS-file system (or a file system for some other operating system). This can be used with any disk-like, Level-2 format.
- A partition can contain a Flash file system (used with memory-like formats).
- A partition can use a vendor-specific organization.
- A partition can use an application-specific organization.

Layer 3 of this Standard provides an unambiguous means of specifying the organization of the partition. (See also the ***Media Storage Formats Specification***.)

5.1 Data Organization Tuples

All information about the organization of a given partition is given in special tuples in the Card Information Structure. Each card that conforms to Layer 3 of this Standard shall contain at least one data-organization tuple for each partition defined on the card.

At present, the data-organization tuple is the only defined Layer 3 tuple.

5.1.1 CISTPL_ORG: The Organization Tuple

The Organization Tuple appears in the series of partition-specific tuples that follows each format tuple. It has the format shown below.

Table 5–1 CISTPL_ORG: Data Organization Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_ORG (46H)						
1	TPL_LINK	Link to next tuple (at least n-1).						
2	TPLORG_TYPE	Data organization code						
3··n	TPLORG_DESC	Text description of this organization, terminated by NULL (00H).						

Byte 2, *TPLORG_TYPE*, specifies the type of data organization in use. The possible values of this byte are given below:

***TPLORG_TYPE* Codes**

Code	Name	Description
0	TPLORGTYPE_FS	This partition contains a file system. The description field specifies the file system type and version.
1	TPLORGTYPE_APP	This partition contains application-specific information. The description field specifies the application name and version.
2	TPLORGTYPE_ROMCODE	This partition contains executable code images. The description field specifies the name and version of the organization scheme.
03H··7FH		(Reserved for future standardization.)
80H··FFH	TPLORGTYPE_VS	This partition uses a vendor-specific organization. The contents of the description field are vendor-specific.

Bytes 3 through the end of the tuple, *TPLORG_DESC,* contain a NULL-terminated ASCII-text description of the organization. For file-system organizations, this field should specify the file-system type. This field shall contain only characters in the printing ASCII set, 020H through 07EH. (For international use, one or more **CISTPL_ALTSTR** tuples can follow this tuple.)

The intent of this field is two-fold:

- For operating systems with sufficient flexibility, it allows the appropriate file-system driver to be selected based on the value of this field.
- If a card cannot be read due to software incompatibilities, a utility program can display to the user the contents of this field along with other card information. This would inform the user what kind of information is actually on the card.

A summary of the defined *TPLORG_DESC* codes follows.

TPLORG_DESC Codes

TPLORG_TYPE	**Card Services Index**	**TPLORG_DESC**	**Description**
RESERVED	0000H	none	This value is used by Card Services to indicate that there is no CISTPL_ORG.
TPLORGTYPE_FS	0001H	"DOS"	DOS-file systems
TPLORGTYPE_FS	0002H	"Flash"	Microsoft Flash File System - Version 1
TPLORGTYPE_FS	0003H	"FFS20"	Microsoft Flash File System - see the Media Storage Formats Specification.
TPLORGTYPE_FS	0004H	"FTL100"	Flash Translation Layer - see the Media Storage Formats Specification.
TPLORGTYPE_FS	0005H	"LFS100"	Linear File Store - see the Media Storage Formats Specification.
TPLORGTYPE_FS	0006H	"VS"	Vendor-Specific partition
RESERVED	7FFFH		Card Services returns this value when indicating an “Unknown partition” to a client.

Note: All *TPLORG_DESC* fields must be null terminated.

The Card Services **GetFirst/NextPartition** services return values listed in the *Card Services Index* field. (See the ***Card Services Specification***.)

6. SYSTEM-SPECIFIC STANDARDS (LAYER 4)

Layer 4 of this standard specifies items that are only relevant in certain operating environments. The following DOS-specific items are defined in other volumes of this Standard:

- An interchange format for cards formatted with the DOS FAT-based file system. (See the ***Media Storage Formats Specification***.)
- A standard for directly-executable programs. (See the ***XIP Specification***.)

6.1 System-Specific Standard Tuples

6.1.1 CISTPL_SPCL: Special Purpose Tuple

The Special Purpose Tuple is identified by a thirty-two (32) bit field assigned by PCMCIA or JEIDA. An eight (8) bit field allows a series of Special Purpose Tuples to be used when the data exceeds the size that can be stored in a single tuple; the maximum data area of a series of Special Purpose Tuples is unlimited. Another eight (8) bit field gives the number of bytes in the data field in this tuple.

When the Special Purpose Tuple is used to address IPL from a PC Card, the tuple provides only enough information to allow a bootstrap loader to fetch significant data bytes and place them in host system memory for execution. Once in memory, it is the responsibility of the loader to determine whether they are appropriate for the environment. The Special Purpose Tuple contains the following fields:

Table 6–1 CISTPL_SPCL: Special Purpose Tuple

Byte	7	6	5	4	3	2	1	0
0	TPL_CODE	CISTPL_SPCL (90H)						
1	TPL_LINK	Link to next tuple (n-2 minimum)						
2··5	TPLSPCL_ID	This 32-bit field contains a PCMCIA or JEIDA assigned value that identifies this series of one or more Special Purpose Tuples. TPLSPCL_ID values are assigned by contacting either PCMCIA or JEIDA.						
6	TPLSPCL_SEQ	The last tuple in the series has END (bit 7 in this field) set to one (1); any other tuples in the series have *END* reset to zero (0). *SEQUENCE* (bits 0··6 in this field) is assigned ascending values starting with zero (0). When *SEQUENCE* reaches 127 (1111111B) and *END* is not set to one (1) the next tuple *SEQUENCE* value is set to zero (0).						
	END	SEQUENCE						
7	TPLSPCL_BYTES	The number of bytes in TPLSCPL_DATA, the data component of this tuple.						
8··n	TPLSPCL_DATA	The data component of this tuple (8 ≤ n ≤ 257)						

If more data is required than will fit in a single tuple, additional Special Purpose Tuples appear in the CIS. A grouping of Special Purpose Tuples with identical *TPLSPCL_ID* values is called a series of Special Purpose Tuples. A *TPLSPCL_ID* value can appear in only one series of Special Purpose Tuples in a CIS; it may never be repeated in another series of Special Purpose Tuples in the same CIS.

The *TPLSPCL_SEQ* field is divided into a single bit flag, *END* (bit 7), and a seven bit field, *SEQUENCE* (bits 0··6). *SEQUENCE* is assigned values ranging from zero (0) to one hundred twenty-seven (127) in ascending order beginning with zero (0) and resetting to zero (0) after reaching one hundred twenty-seven (127). The last tuple in the series is indicated by setting the *END* bit to one (1). A series consisting of a single tuple has a *TPLSPCL_SEQ* field with a value of one hundred twenty-eight (10000000B).

The *TPLSPCL_BYTES* field contains the number of data bytes in the *TPLSPCL_DATA* field.

A series of Special Purpose Tuples must be placed in the CIS in the order that it is to be processed in the host environment.

7. COMPATIBILITY ISSUES

7.1 Buffer Pages

Some vendors use a buffer page to improve the reliability of memory cards in the face of power failures. This standard does not directly provide a means for specifying the location of the buffer page. Space can easily be reserved for a buffer page by proper adjustment of the values in the format tuple. If needed, a vendor-specific tuple can be added to specify the location of the buffer page within the partition.

7.2 Media Storage Formats

See the ***Media Storage Formats Specification***.