Document number 2005099

IEEE 1394 Application Layer for Industrial Automation

December, 09 2005

Sponsored by:

1394 Trade Association and 1394automation e.V.

Accepted for publication by

The 1394 Trade Association Board of Directors
accepted this specification January 23, 2006

Abstract

This specification covers the communication for industrial automation applications based on the standard IEEE 1394. In particular an application layer for this application area is specified. After the description of an overall model, application layer service elements containing services and attributes are described. Afterwards the application layer transfer syntax is specified covering application layer protocol data unit (APDU) definitions, APDU field descriptions and protocol behavior. The document is completed by the Control and Status Register (CSR) architecture.

Keywords

Industrial Automation, Industrial Communication, Application Layer, Drive Control, Remote I/O, Vision, PLC, IPC

1394 Trade Association Specification

1394 Trade Association Specifications are developed within Working Groups of the 1394 Trade Association, a non-profit industry association devoted to the promotion of and growth of the market for IEEE 1394-compliant products. Participants in Working Groups serve voluntarily and without compensation from the Trade Association. Most participants represent member organizations of the 1394 Trade Association. The specifications developed within the working groups represent a consensus of the expertise represented by the participants.

Use of a 1394 Trade Association Specification is wholly voluntary. The existence of a 1394 Trade Association Specification is not meant to imply that there are not other ways to produce, test, measure, purchase, market or provide other goods and services related to the scope of the 1394 Trade Association Specification. Furthermore, the viewpoint expressed at the time a specification is accepted and issued is subject to change brought about through developments in the state of the art and comments received from users of the specification. Users are cautioned to check to determine that they have the latest revision of any 1394 Trade Association Specification.

Comments for revision of 1394 Trade Association Specifications are welcome from any interested party, regardless of membership affiliation with the 1394 Trade Association. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally, questions may arise about the meaning of specifications in relationship to specific applications. When the need for interpretations is brought to the attention of the 1394 Trade Association, the Association will initiate action to prepare appropriate responses.

Comments on specifications and requests for interpretations should be addressed to:

Editor, 1394 Trade Association
1560 East Southlake Blvd., Suite 242
Southlake, TX 76092-6492
USA

1394 Trade Association Specifications are adopted by the 1394 Trade Association without regard to patents which may exist on articles, materials or processes or to other proprietary intellectual property which may exist within a specification. Adoption of a specification by the 1394 Trade Association does not assume any liability to any patent owner or any obligation whatsoever to those parties who rely on the specification documents. Readers of this document are advised to make an independent determination regarding the existence of intellectual property rights, which may be infringed by conformance to this specification.

Published by

1394 Trade Association
1560 East Southlake Blvd., Suite 242
Southlake, TX 76092-6492 USA

Printed in the United States of America

Contents

Tables

Figures

Annexes

Foreword (This foreword is not part of 1394 Trade Association Specification 2005099)

This specification defines an application layer for the communication in industrial automation applications based on the standard IEEE 1394. It exploits the isochronous feature of IEEE 1394 for highly synchronized communication in large distributed networks. Application layer service elements containing services and attributes are described, the application layer transfer syntax is specified, and the Control and Status Register (CSR) architecture is defined.

There are 5 annexes in this specification. Annex A is normative and part of this specification. Annexes B through E, inclusive, are informative and are not considered part of this specification.

This specification was accepted by the Board of Directors of the 1394 Trade Association. Board of Directors acceptance of this specification does not necessarily imply that all board members voted for acceptance. At the time it accepted this specification, the 1394 Trade Association. Board of Directors had the following members:

Eric Anderson, Chair
Max Bassler, Vice-Chair
Dave Thompson, Secretary

Organization Represented	*Name of Representative*
Agere Systems	Dave Thompson
Apple	Eric Anderson
Congruent Software Inc.	Peter Johansson
Firecomms	Declan O'Mahoney
Fraunhofer IPMS	Michael Scholles
Interactive Technologies	Max Bassler
Newnex	Sam Liu
Oxford Semiconductor	Jalil Oraee
Samsung	Jong-Wook Park

The Technical Working Group of the 1394automation association, which developed and reviewed this specification before submitting the document to the 1394 TA Industrial & Instrumentation Working Group and transferring all rights to the 1394 Trade Association had the following members:

Michael Scholles, Chair

Dirk Büsching
Jeroen Koeter
Ulrich Koch
Christoph Korb
Petra Nauber
Thomas Odermatt
Lutz Rauchhaupt
Elmar Senneka
Oliver Wetter

Revision history

Revision 1.0 (December 09, 2005)

Original version

Internal versions of 1394automation before submission of document to 1394 Trade Association are not listed.

IEEE 1394 Application Layer for Industrial Automation

1 Scope and purpose

1.1 Scope

This specification describes an application layer for industrial automation applications based on IEEE 1394 [R1]. It includes the definition of

- application layer services elements
- application layer transfer syntax
- Control and Status Register architecture.

Application layer service elements (ASE) are communication objects with determined purposes. Application layer service elements are described for

- process data variables (PDs), which cover usually time critical control, sensor and actuator data
- service data variables (SDs), which cover usually non-time critical parameter (and even complete software modules) for the automation devices
- CSR data, to provide direct access to CSR for higher layers and profiles of the automation branch,
- emergency data, to handle critical events of the automation control process
- network management data, to control the devices concerning their application related behavior

The application layer service element specification includes attribute and service definitions. Attributes are variables which can be accessed via a communication system. The services describe the interface between the application and the communication system.

The application transfer syntax describes in detail the application layer protocol units (APDU) which are transferred to read and write the attributes of different ASEs. Furthermore the behavior of several transfer tasks is specified.

Finally the Control and Status Registers (CSR) are defined which contain the attributes of the ASEs.

This document is based on a draft paper [E1] and was revised by the 1394 Automation Technical Working Group.

1.2 Purpose

The purpose of this specification is to achieve interoperability between industrial devices such as programmable logic controllers (PLC), Industrial PCs (IPC), sensors and actuators by defining an application layer for industrial automation applications. In particular the combination of devices with high performance requirements such as drives, simple remote I/O-devices and devices with a significant band width requirement such as industrial cameras is in the scope of this specification.

Distributed automation systems, and in particular drive control systems, require a highly deterministic isochronous high speed data transmission as it is specified in IEEE 1394 to implement distributed control loops. Furthermore there is a need for a network wide synchronisation of sensor input data and actuator output data to synchronise distributed processes such as the speed and acceleration behaviour of distributed drives. The high performance requirements in automation systems are also satisfied by the possibility to transmit data directly between devices without transferring them via a master device.

Furthermore the specification provides mechanisms to access automation device specific and automation application specific parameters without influencing the isochronous traffic. These mechanisms include program downloads and uploads as well.

Finally the network management with respect to the needs of devices of industrial automation is defined.

This specification is described in order to use available communication, device and application profiles of industrial communication systems on top (e.g. [E3], [E4]).

2 Normative references

2.1 Reference scope

The specifications and standards named in this section contain provisions, which, through reference in this text, constitute provisions of this 1394 Trade Association Specification. At the time of publication, the editions indicated were valid. All specifications and standards are subject to revision; parties to agreements based on this 1394 Trade Association Specification are encouraged to investigate the possibility of applying the most recent editions of the specifications and standards indicated below.

2.2 Approved references

The following approved specifications and standards may be obtained from the organizations that control them.

IEEE Std 1394-1995, Standard for a High Performance Serial Bus

IEEE Std 1394a-2000, Standard for a High Performance Serial Bus—Amendment 1

IEEE Std 1394b-2002, Standard for a High Performance Serial Bus—Amendment 2

Throughout this document, the term “IEEE 1394" shall be understood to refer to IEEE Std 1394-1995 as amended by IEEE Std 1394a-2000 and IEEE Std 1394b-2002.

2.3 Reference acquisition

The references cited may be obtained from the organizations that control them:

1394 Trade Association, 1111 South Main Street, Suite 100, Grapevine, TX 76051 USA; (817) 410-5750 / (817) 410-5752 (FAX); http://www.1394ta.org/

American National Standards Institute (ANSI), 11 West 42nd Street, New York, NY 10036, USA; (212) 642-4900 / (212) 398-0023 (FAX); http://www.ansi.org/

Institute of Electrical and Electronic Engineers (IEEE), 445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA; (732) 981-0060 / (732) 981-1721 (FAX); http://www.ieee.org/

In addition, many of the documents controlled by the above organizations may also be ordered through a third party:

Global Engineering Documents, 15 Inverness Way, Englewood, CO 80112-5776; (800) 624-3974 / (303) 792-2192; http://www.global.ihs.com/

3 Definitions and notation

3.1 Definitions

3.1.1 Conformance

Several keywords are used to differentiate levels of requirements and optionality, as follows:

3.1.1.1 expected: A keyword used to describe the behavior of the hardware or software in the design models assumed by this specification. Other hardware and software design models may also be implemented.

3.1.1.2 ignored: A keyword that describes bits, bytes, quadlets, octlets or fields whose values are not checked by the recipient.

3.1.1.3 may: A keyword that indicates flexibility of choice with no implied preference.

3.1.1.4 reserved: A keyword used to describe objects (bits, bytes, quadlets, octlets and fields) or the code values assigned to these objects in cases where either the object or the code value is set aside for future standardization. Usage and interpretation may be specified by future extensions to this or other specifications. A reserved object shall be zeroed or, upon development of a future specification, set to a value specified by such a specification. The recipient of a reserved object shall ignore its value. The recipient of an object defined by this specification as other than reserved shall inspect its value and reject reserved code values.

3.1.1.5 shall: A keyword that indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to assure interoperability with other products conforming to this specification.

3.1.1.6 should: A keyword that denotes flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “is recommended.”

3.1.2 Glossary

The following terms are used in this specification:

3.1.2.1 Application Cycle: minimum cycle of the application in which one or more process data variables have to be updated

3.1.2.2 Application Layer Service Element: communication objects including attributes and services with determined communication purposes

3.1.2.3 Application Layer Protocol Unit: payload of a transaction layer or link layer packet

3.1.2.4 Application Master: central control instance from the point of view of control applications

3.1.2.5 Application Slave: controlled node from the point of view of control applications

3.1.2.6 Byte: Eight bits of data, used as a synonym for octet

3.1.2.7 CSR Architecture: A convenient abbreviation of the following reference (see clause 2): ISO/IEC 13213 : 1994 [ANSI/IEEE Std 1212, 1994 Edition], Information Technology—Microprocessor systems— Control and Status Register (CSR) Architecture for Microcomputer Buses

3.1.2.8 Device: node in the network, which is controlled by a central node (the application master) from the point of view of control applications, used as a synonym for application slave

3.1.2.9 Node: network element, where an application master or application slave application is implemented

3.1.2.10 Process Data Variables: cover time critical control, sensor and actuator data

3.1.2.11 Service Data Variables: cover non-time critical parameters (and even complete software modules) for the automation devices

3.1.2.12 Quadlet: four bytes of data …

3.1.3 Abbreviations

The following are abbreviations that are used in this specification:

1394AP	IEEE 1394 Application Layer for Industrial Automation
APDU	Application Layer Protocol Unit
ASE	Application Layer Service Element
CSR	Control and Status Register
DDT	Device Data Telegram
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
MDT	Master Data Telegram
NMT	Network Management
PD	Process Data Variable
SD	Service Data Variable
PLC	Programmable Logic Controller

3.2 Notation

3.2.1 Numeric values

Decimal and hexadecimal are used within this specification. By editorial convention, decimal numbers are most frequently used to represent quantities or counts. Addresses are uniformly represented by hexadecimal numbers. Hexadecimal numbers are also used when the value represented has an underlying structure that is more apparent in a hexadecimal format than in a decimal format.

Decimal numbers are represented by Arabic numerals without subscripts or by their English names. Hexadecimal numbers are represented by digits from the character set 0 – 9 and A – F followed by the subscript 16. When the subscript is unnecessary to disambiguate the base of the number it may be omitted. For the sake of legibility hexadecimal numbers are separated into groups of four digits separated by spaces.

As an example, 42 and $2A_{16}$ both represent the same numeric value.

3.2.2 Bit, byte and quadlet ordering

This specification uses the facilities of Serial Bus, IEEE 1394, and therefore uses the ordering conventions of Serial Bus in the representation of data structures. In order to promote interoperability with memory buses that may have different ordering conventions, this specification defines the order and significance of bits within bytes, bytes within quadlets and quadlets within octlets in terms of their relative position and not their physically addressed position.

Within a byte, the most significant bit, *msb*, is that which is transmitted first and the least significant bit, *lsb*, is that which is transmitted last on Serial Bus, as illustrated below. The significance of the interior bits uniformly decreases in progression from *msb* to *lsb*.

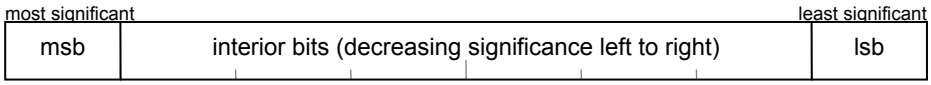

Figure 3.1 – Bit ordering within a byte

Within a quadlet, the most significant byte is that which is transmitted first and the least significant byte is that which is transmitted last on Serial Bus, as shown below.

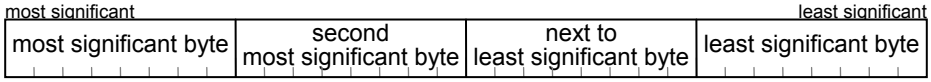

Figure 3.2 – Byte ordering within a quadlet

Within an octlet, which is frequently used to contain 64-bit Serial Bus addresses, the most significant quadlet is that which is transmitted first and the least significant quadlet is that which is transmitted last on Serial Bus, as the figure below indicates.

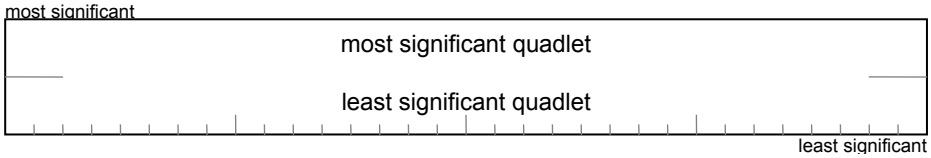

Figure 3.3 – Quadlet ordering within an octlet

When block transfers take place that are not quadlet aligned or not an integral number of quadlets, no assumptions can be made about the ordering (significance within a quadlet) of bytes at the unaligned beginning or fractional quadlet end of such a block transfer, unless an application has knowledge (outside of the scope of this specification) of the ordering conventions of the other bus.

3.2.3 Register specifications

This specification defines the format and function of control and status registers, CSRs. Some of these registers are read-only, some are both readable and writable and some generate special side effects subsequent to a write.

In order to define CSRs, their bit fields, their initial values and the effects of read, write or other transactions, the format illustrated by Figure 3.4 is used.

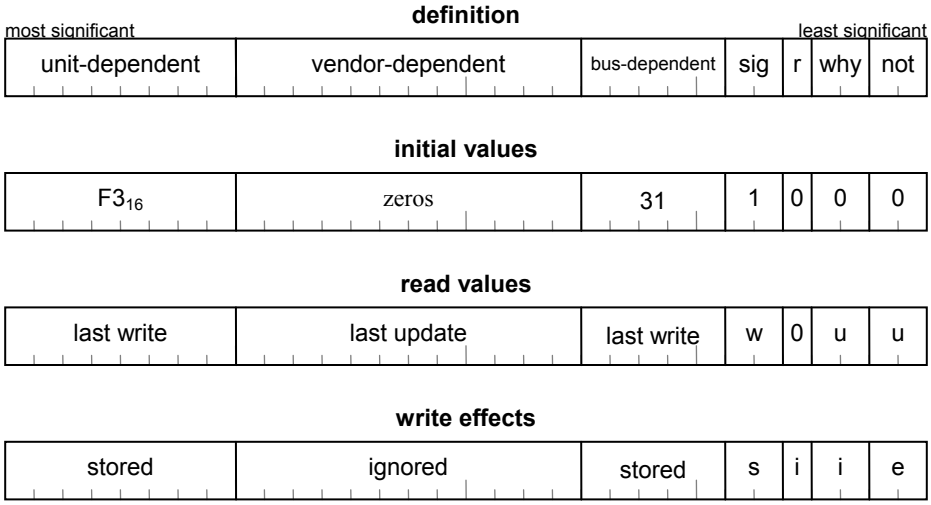

Figure 3.4 – CSR specification example

The register definition contains the names of register fields. The names are intended to be descriptive, but the fields are defined in the text; their function should not be inferred solely from their names. However, the following field names have defined meanings.

Name	Abbreviation	Definition
bus-dependent		The meaning of the field is defined by the bus standard, in this case IEEE 1394
reserved	r	The field is reserved for future standardization (see 3.1.1)
unit-dependent		The meaning of the field shall be defined by the organization responsible for the unit architecture
vendor-dependent		The meaning of the field shall be defined by the node's vendor

CSRs shall assume initial values upon the restoration of power (a power reset) or upon a write to the node's RESET_START register (a command reset). If the power reset values differ from the command reset values, they are separately and explicitly defined. Initial values for register fields may be described as numeric constants or with one of the terms defined for the register definition. Values for register fields subsequent to a reset may be described in the same terms or as defined below.

Name	Abbreviation	Definition
unchanged	x	The field retains whatever value it had just prior to the power reset, bus reset or command reset.

In addition to numeric values for constant fields, the read values returned in response to a quadlet read transaction may be specified by the terms below.

Name	Abbreviation	Definition
last write	w	The value of the field shall be either the initial value or, if a write or lock transaction addressed to the register has successfully completed, the value most recently stored in the field.[1]
last update	u	The value of the field shall be that most recently updated by the node hardware or software. An updated field value may be the result of a write effect to the same register address, a different register address or some other change of condition within the node.

The effects of data written to the register are specified by the terms below.

Name	Abbreviation	Definition
effect	e	The value of the data written to the field may have an effect on the node's state, but the effect might not be immediately visible by a read of the same register. The effect may be visible in another register or might not be visible at all.
ignored	i	The value of the data written to the field shall be ignored; it shall have no effect on the node's state.
stored	s	The value of the data written to the field shall be immediately visible by a read of the same register; it may also have other effects on the node's state.

Reserved fields within a register shall be explicitly described with respect to initial values, read values and write effects. Initial values and read values shall be zero while write effects shall be ignored. CSRs that are not implemented, either because they are optional or they fall within a reserved address space, shall abide by these same conventions if a successful completion response is returned for a read, write or lock request.

3.2.4 State machines

All state machines in this specification are defined in the style illustrated by Figure 3.5. The conditions and actions of the state machine transitions are formally defined by C code, as specified by ISO/IEC 9899:1990 [E2][E1].

[1] For clarity, read values for a field in a register that accepts lock transactions may be described as *last successful lock* rather than *last write*. However, the abbreviation in both cases remains *w*. Similar liberties may be taken with the use of *conditionally stored* in place of *stored* when the action occurs as the result of a lock transaction, but the corresponding one-letter abbreviation, *s*, is also unchanged.

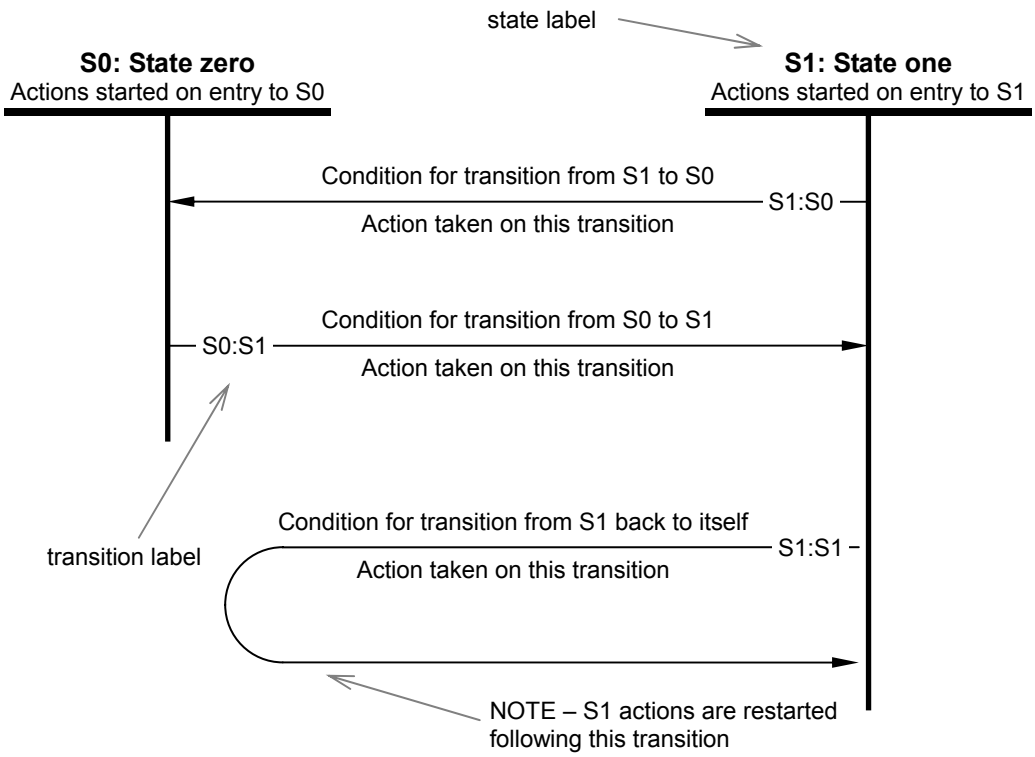

Figure 3.5 – State machine example

The state machines in this specification make three assumptions:

- Time elapses only within a discrete state;
- State transitions are conceptually instantaneous; the only actions taken during the transition are the setting of flags or variables and the sending of signals; and
- Each time a state is entered (or reentered from itself), the actions of that state are performed.

Multiple transitions may connect two states. In this case, the transitions are uniquely labeled by appending a character to the transition label, *e.g.*, S0:S1a and S0:S1b.

4 Model (informative)

4.1 System model

A communication system in accordance to the IEEE 1394 Application Layer for Industrial Automation (1394AP) consists of one application master and one to 62 application slaves. The application master is the central control instance in the distributed control system. The application master can be a programmable logic controller (PLC) or a personal computer (PC). The application slaves are controlled by the application master. Below in the document they are called devices. Example for devices are sensor devices, remote IO devices, drives, cameras etc.

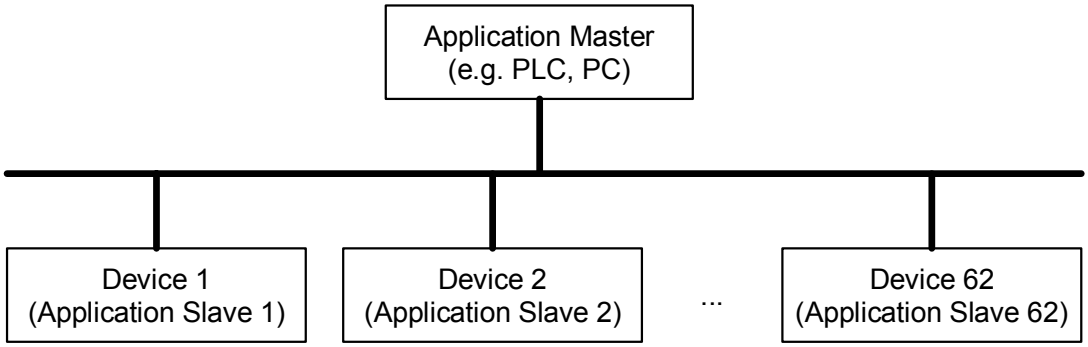

Figure 4.1 – 1394AP System Architecture

Based on the IEEE 1394 network structure, the architecture of a 1394AP system can be seen as a logical bus system (see Figure 4.1). All nodes (application master and devices) share a common communication medium. However, depending on the communication purpose, one of the communication relationships depicted in chapter 4.3 is used.

4.2 Communication Layer Model

The 1394AP communication layer model is based on the IEEE 1394 communication layer model. The IEEE 1394 Application Layer for Industrial Automation is the user of the services provided by the IEEE 1394 layer. Chapter 5 explains in detail which application layer service element uses which of the provided IEEE 1394 services. Specified in the same chapter is which services the 1394AP provides to communication profiles, device profiles and application profiles. These profiles specify object dictionaries and behavior for specific device families and application areas.

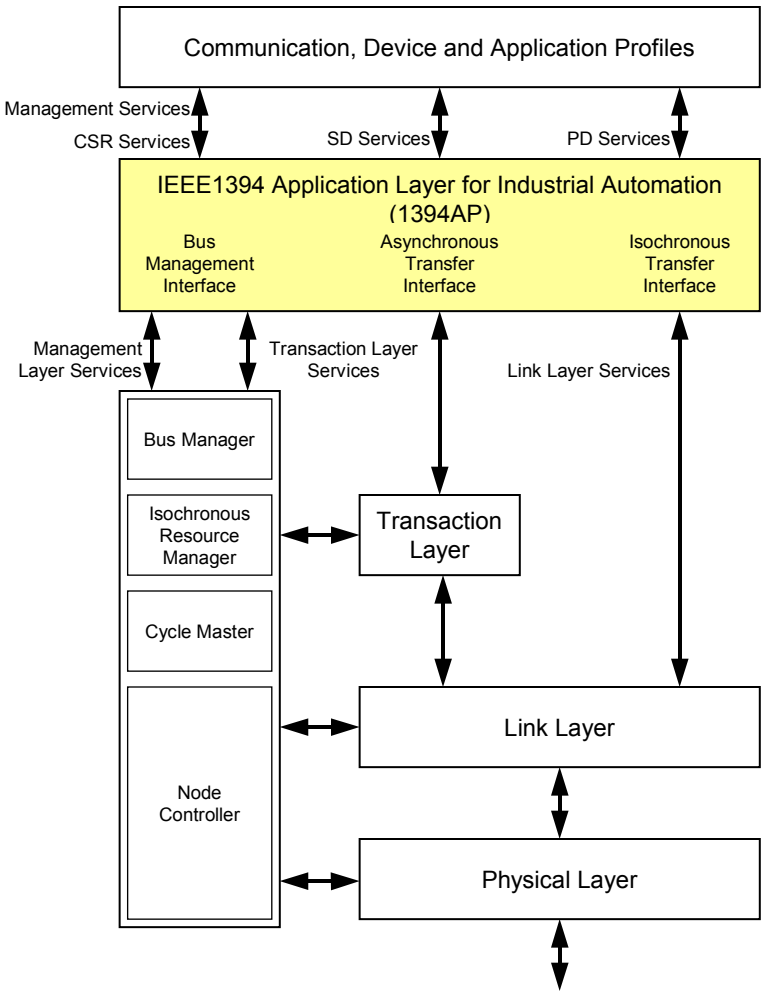

Figure 4.2 – 1394AP Communication Layer Model

4.3 Device Model

A device is structured as follows: (see Figure 4.3):

- Communication – This function unit provides the appropriate functionality to transport data items via the underlying communication.
- Control Status Register – The CSR is a collection of all the data items which have an influence on the behaviour of the application objects, the communication and the state machine used on this device.
- Application – The application comprises the functionality of the device with respect to the interaction with the process environment.

Thus the CSR serves as an interface between the communication and the application. The complete description of a device's application with respect to the data items in the Object Dictionary is named device profile.

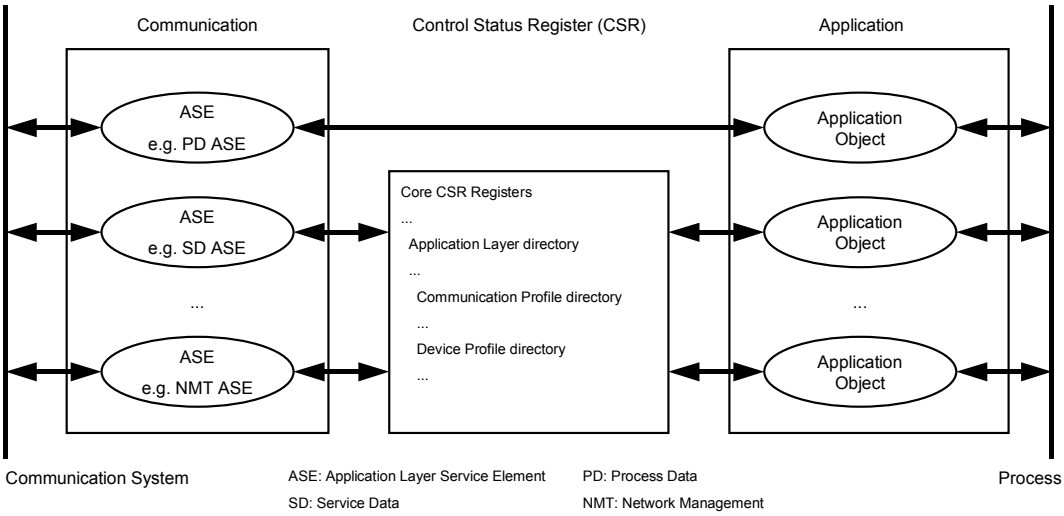

Figure 4.3 – 1394AP Device Model

4.4 Communication Relationships

4.4.1 Master-Slave-Relationship

There is always only one node in the network serving as a master for a specific functionality. All other nodes in the network are considered as slaves. The master issues a request and the addressed slave(s) respond(s) if the protocol requires this behavior (Figure 4.4 and Figure 4.5). The master-slave-relationship is used for network management transmissions.

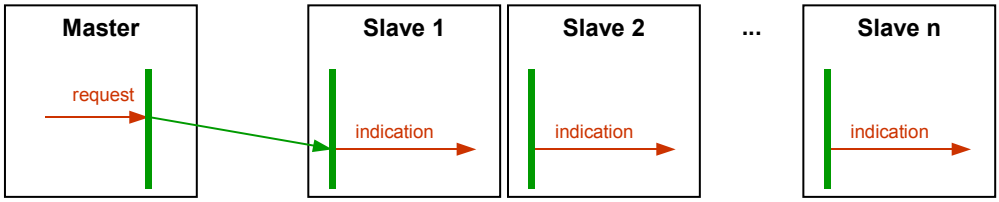

Figure 4.4 – Unconfirmed Master-Slave-Communication

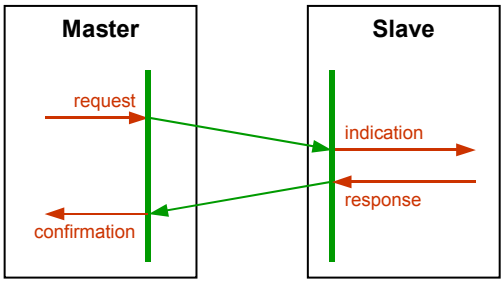

Figure 4.5 – Confirmed Master-Slave-Communication

4.4.2 Client-Server-Relationship

This is a relationship between a single client and a single server. A client issues a request thus triggering the server to perform a certain task. After finishing the task the server answers the request. The client-server-relationship is used for service data and CSR transmissions.

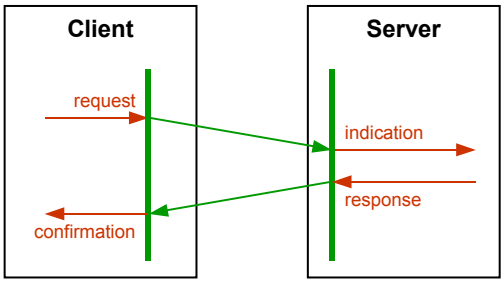

Figure 4.6 – Client-Server-Communication

4.4.3 Producer-Consumer-Relationship

The producer-consumer-relationship model involves a producer and zero or more consumer(s). The push model is characterised by an unconfirmed service requested by the producer. The producer-consumer-relationship is used for process data and emergency transmissions.

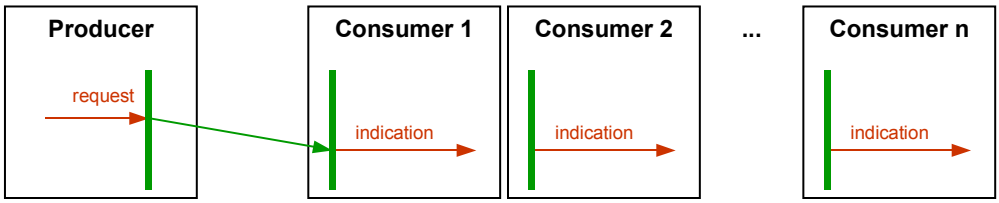

Figure 4.7 – Producer-Consumer-Communication

5 Application Layer Service Elements (ASE)

5.1 General remarks

The application layer service element description contains the attribute and service definitions. The attribute definitions include the attribute type, the allowed values and a default value where applicable. Tables are used for attribute definitions. The meaning of the columns is explained in Table 5.1.

Table 5.1 – Attribute Definition

Column	Description
Abbreviation	abbreviation of the attribute
Description	description of the attribute
M/O	definition of the attribute is mandatory (M) or optional (O)
CSR coding	reference to the description of the related entry in the Control and Status Register area

The service definitions include the service primitives and the arguments. Tables are used for service definitions (see Table 5.2). The first column contains the arguments for the service primitives. If applicable the arguments are combined e.g. in success arguments and failure arguments. The second column contains the definition concerning the availability of the request and indication service primitive. It is also defined if the arguments of the service primitive are optional or mandatory or if there is a selection between arguments. If available the third column contains the definitions for the response and confirmation service primitive.

Table 5.2 – Service Definition

Parameter	Request / Indication	Response / Confirm
Argument Argument 1 Argument 2 Argument 3	**Mandatory** mandatory mandatory optional	
Remote Result Success Argument 4 Argument 5 Failure Argument 6		**Mandatory** selection mandatory optional selection optional

5.2 Process Data ASE

5.2.1 Overview

The process data application layer service element describes the means, provided by the application layer, to transmit time critical control, sensor and actuator data.

The Process Data ASE includes the attributes listed in Table 5.3. The Process Data ASE attributes are described in detail in chapter 5.2.2. The coding is described in chapter **7**.

Table 5.3 – Process Data ASE Attributes

Abbreviation	Description	M/O	CSR coding
amc	Application Master Capability	M	7.5.4
amf	Application Master Functionality	O/M[2]	7.7.2.6
min_appl_cycle	Minimum Application Cycle of the node	M	7.7.2.10
basic_appl_cycle	Application Cycle of the master	O/M	7.7.2.11
dev_appl_cycle	Application Cycle of the device	O/M[3]	7.7.2.12
appl_cycle_list	Application Cycle list of the node	O	7.7.3.1
PD_loss_count_list	Process Data Loss Count list	O	7.7.3.2
astc	Asynchronous process data Transmission Capability	M	7.5.4
astf	Asynchronous process data Transmission Functionality	O/M[2]	7.7.2.6
cstc	Clock synchronised process data Transmission Capability	M	7.5.4
cstf	Clock synchronised process data Transmission Functionality	O/M[2]	7.7.2.6
evtc	Event triggered process data Transmission Capability	M	7.5.4
evtf	Event triggered process data Transmission Functionality	O/M[2]	7.7.2.6
min_inp_trigger_time	Minimum Input Trigger Time of the device	O	7.7.2.13
global_inp_trigger_time	Global Input Trigger Time of the device	O	7.7.2.14
dev_inp_time	Device Input Time of the device	O	7.7.2.15
dev_inp_time_res	Device Input Time Resolution of the device	O	7.7.2.16
min_outp_trigger_time	Minimum Output Trigger Time of the device	O	7.7.2.17
global_outp_trigger_time	Global Output Trigger Time of the device	O	7.7.2.18
number_of_R_PD	Number of Receive Process Data of the node	M	7.7.3.5
PD_source	Process Data Source	M	7.7.3.5
R_PD_header_offset	Receive Process Data Header Offset within process data APDU	M	7.7.3.5
R_PD_length	Receive Process Data Length	M	7.7.3.5
R_PD_offset	Receive Process Data Offset within process data APDU	O	7.7.3.5
R_process_ID	Receive Process Identification of the process data	O	7.7.3.5
number_of_T_PD	Number of Transmit Process Data of the node	M	7.7.3.6
PD_destination	Process Data Destination	M	7.7.3.6
T_PD_header_offset	Transmit Process Data Header Offset within process data APDU	M	7.7.3.6
T_PD_length	Transmit Process Data Length	M	7.7.3.6
T_PD_ offset	Transmit Process Data Offset within the process data APDU	O	7.7.3.6
T_process_ID	Transmit Process Identification of the process data	O	7.7.3.6

[2] Mandatory if capable.
[3] Mandatory if minimum application cycle is greater than zero.

As illustrated in Figure 5.1, the process data is transferred using the Process Data Update service. Depending on the requirements, the process data is passed to the link data service or to the asynchronous write service. The SYNC service is used if it is intended to transmit the process data clock synchronized.

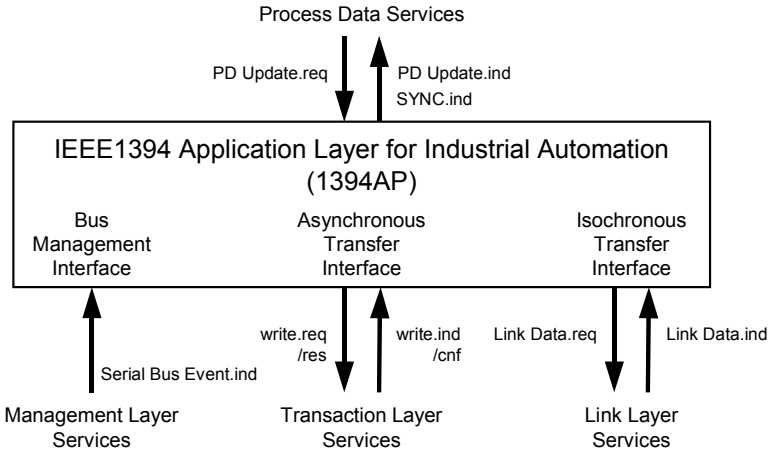

Figure 5.1 – Process Data Services

5.2.2 Attributes

5.2.2.1 Application Cycle Attributes

The application cycle attributes can be used to characterize and control the cyclic process data communication. They are stored in registers of the application layer CSR area. The meaning of the application cycle attributes is depicted in Figure 5.2 and explained in the following sections.

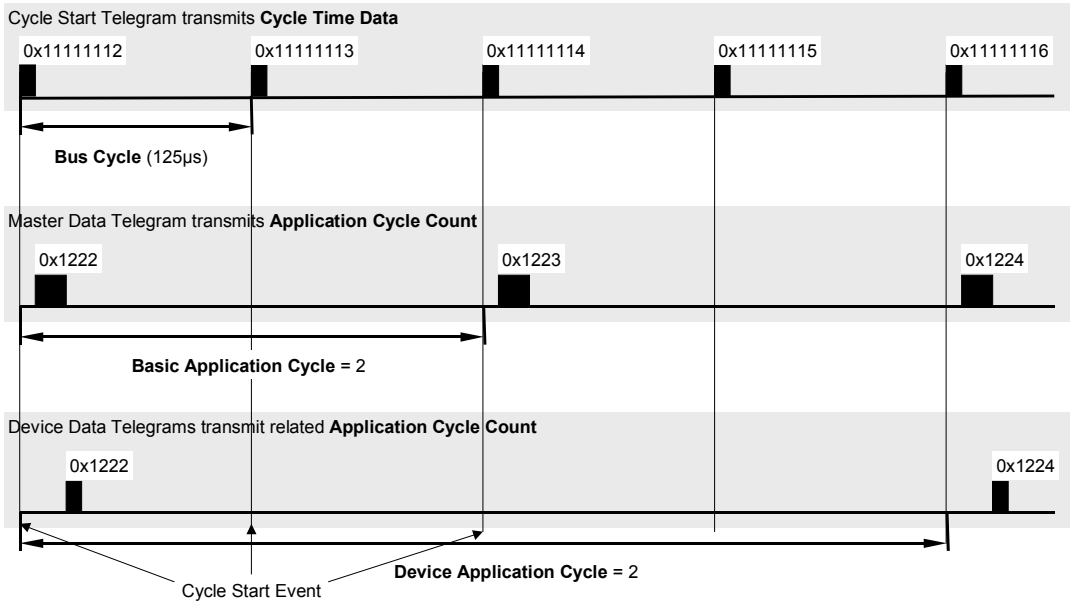

Figure 5.2 – Cycle Attributes

5.2.2.1.1 Application Master Capability (amc)

This attribute indicates whether a node is able to take over the part of an application master or not. The application master is the manager of the process data transmission. It determines the basic application cycle value and the trigger time values. Furthermore it provides the basic application cycle by transmitting the application cycle count value. The application master is usually a programmable logic controller (PLC). However, if there is no PLC in the network then another node may play the roll of application master.

Attribute Type: BOOLEAN

Allowed Values:

TRUE	node is application master capable
FALSE	node is not application master capable

5.2.2.1.2 Application Master Functionality (amf)

A remote node (e.g. a configuration device) may use this attribute to determine whether a node acts as an application master or not. This attribute is mandatory if a node is application master capable.

Attribute Type: BOOLEAN

Allowed Values:

TRUE	node acts as an application master
FALSE	node does not act as an application master

Default Value: FALSE

5.2.2.1.3 Minimum Application Cycle (min_appl_cycl)

This attribute contains the minimum number of bus cycles a device is able to support as application cycle. An application master may read this value out of the devices to be able to calculate the basic application cycle.

Attribute Type: UNSIGNED32

Allowed Values:

0	cyclic process data transmission not supported
> 0	minimum application cycle

5.2.2.1.4 Basic Application Cycle (basic_appl_cycle)

The basic application cycle indicates the duration between two consecutive process data transmissions of the master. This attribute contains the number of bus cycles between these two process data transmissions (see also Figure 5.2).

Attribute Type: UNSIGNED32

Allowed Values:

0	cyclic process data transmission not supported
> 0	basic application cycle

5.2.2.1.5 Device Application Cycle (dev_appl_cycle)

The device application cycle indicates the number of basic application cycles between two consecutive process data transmissions of the given device. This value may be set up by the application master using the Service Data or CSR Data services (see chapters 5.3 and 5.4).

Attribute Type: UNSIGNED32

Allowed Values:

0	cyclic process data transmission not supported
> 0	device application cycle

Default Value: Minimum Application Cycle Value

5.2.2.1.6 Application Cycle List (appl_cycle_list)

The application cycle list contains the application cycle values of process data producers, which are relevant for the given node. The list consists of 64 entries, one for each node. The first element (channel 0) contains the basic application cycle, which is the application cycle of the master.

Attribute Type: ARRAY of UNSIGNED32

Allowed Values: see 5.2.2.1.4 and 5.2.2.1.5

Default Value: 0 for each entry (cyclic process data transmission not supported for this channel)

5.2.2.2 Common Trigger Attributes

The common trigger attributes are used to characterize and control the trigger behavior of a device during process data transmission. They are stored in registers of the application layer CSR area. The meaning of the common trigger attributes is depicted in Figure 5.3 and Figure 5.4 and are explained in the following sections.

5.2.2.2.1 Asynchronous Process Data Transmission Capability (astc)

This attribute indicates whether a node is able to transmit process data using the asynchronous write service.

Attribute Type: BOOLEAN

Allowed Values: see Table 5.4

5.2.2.2.2 Asynchronous Process Data Transmission Functionality (astf)

A remote node may use this attribute to determine if this node transmits process data using the asynchronous write service. This attribute is mandatory if a node is able to transmit process data using the asynchronous write service.

Attribute Type: BOOLEAN

Allowed Values: see Table 5.4

Default Value: FALSE

5.2.2.2.3 Clock Synchronized Process Data Transmission Capability (cstc)

This attribute indicates whether a node is able to transmit process data synchronized or not. The process data may be transmitted using the link data service or to the asynchronous write service.

Attribute Type: BOOLEAN

Allowed Values: see Table 5.4

5.2.2.2.4 Clock Synchronized Process Data Transmission Functionality (cstf)

A remote node may use this attribute to determine whether this node transmits process data synchronized or not. This attribute is mandatory if a node is able to transmit process data clock synchronized.

Attribute Type: BOOLEAN

Allowed Values: see Table 5.4

Default Value: TRUE

5.2.2.2.5 Event Triggered Process Data Transmission Capability (evtc)

This attribute indicates whether a node is able to transmit process data event triggered or not. The process data may be transmitted using the link data service or to the asynchronous write service.

Attribute Type: BOOLEAN

Allowed Values: see Table 5.4

5.2.2.2.6 Event Triggered Process Data Transmission Functionality (evtf)

A remote node may use this attribute to determine whether this node transmits process data event triggered or not. This attribute is mandatory if a node is able to transmit process data event triggered.

Attribute Type: BOOLEAN

Allowed Values:

Table 5.4 – Allowed Combinations

asynchronous	clock synchronised	event triggered	Description
FALSE	FALSE	FALSE	passive node, e.g. for diagnostic purposes
FALSE	FALSE	TRUE	event triggered process data transmission using an isochronous channel
FALSE	TRUE	FALSE	clock synchronised process data transmission using an isochronous channel; minimum input trigger time and minimum output trigger time shall not be zero simultaneously
FALSE	TRUE	TRUE	reserved
TRUE	FALSE	FALSE	reserved
TRUE	FALSE	TRUE	event triggered process data transmission using an asynchronous connection
TRUE	TRUE	FALSE	clock synchronised process data transmission using an asynchronous connection; minimum input trigger time and minimum output trigger time shall not be zero simultaneously
TRUE	TRUE	TRUE	reserved

Default Value: FALSE

5.2.2.3 Input Trigger Attributes

The input trigger attributes are used to implement a system wide synchronized acquisition of process data. The meaning of the input trigger attributes is depicted in Figure 5.3.

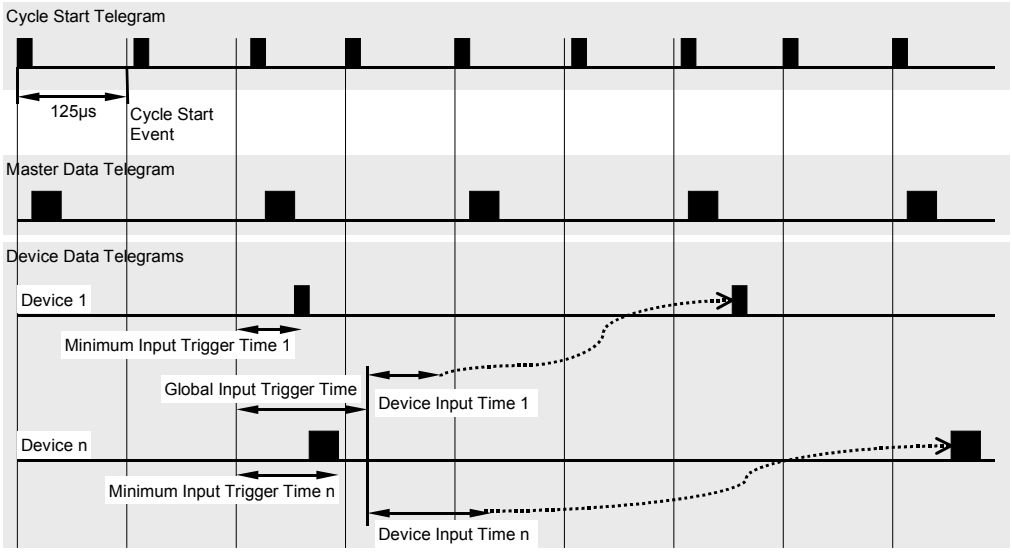

Figure 5.3 – Input Trigger Attributes

5.2.2.3.1 Minimum Input Trigger Time (min_inp_trigger_time)

The minimum input trigger time is the period of time a device needs to be prepared for sampling the input data. An application master may read this value to be able to calculate the global input trigger time.

Attribute Type: FTIME (see [E5])

Allowed Values:

0	clock synchronised input is not supported
> 0	minimum input trigger time

5.2.2.3.2 Global Input Trigger Time (global_inp_trigger_time)

The global input trigger time is a system wide time which represents the time period between the relevant cycle start event and the input sample instruction. The relevant cycle start event is associated to the application cycle of the device.

An application master may set this value in order to get a system wide global input trigger time.

Attribute Type: FTIME

Allowed Values: ≥ 0

Default Value: minimum input trigger time value

5.2.2.3.3 Device Input Time (dev_inp_time)

The device input time represents the time a device needs to prepare input data for transmission. This value can be used to determine how many application cycles go by until the input data is sent.

Attribute Type: FTIME

Allowed Values: ≥ 0

5.2.2.3.4 Device Input Time Resolution (dev_inp_time_res)

The device input time resolution represents the minimum difference between two input time values that can be distinguished by the device.

Attribute Type: FTIME

Allowed Values:

0	global input trigger time can not be written
> 0	device input time resolution

5.2.2.4 Output Trigger Attributes

The output trigger attributes are used to implement a system wide synchronised output of process data. The meaning of the output trigger attributes is depicted in Figure 5.4.

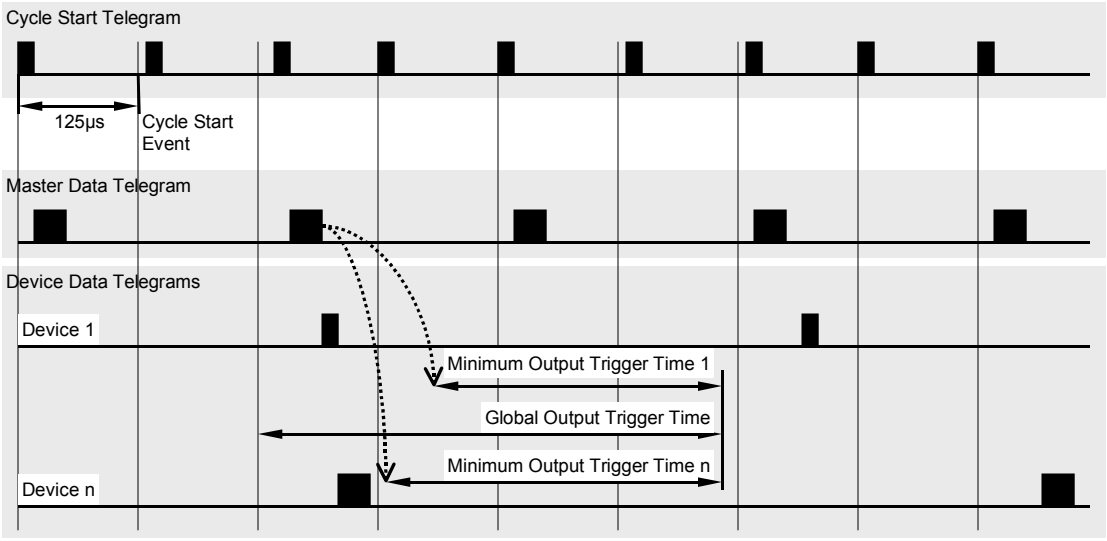

Figure 5.4 – Output Trigger Attributes

5.2.2.4.1 Minimum Output Trigger Time (min_outp_trigger_time)

The minimum output trigger time is the period of time a device needs to read the received data and to output it. An application master may read this value to be able to calculate the global output trigger time.

Attribute Type: FTIME

0	clock synchronised output is not supported
> 0	minimum output trigger time

5.2.2.4.2 Global Output Trigger Time (global_outp_trigger_time)

The global output trigger time is a system wide time which represents the time period between the relevant cycle start event and the data output instruction. The relevant cycle start event is associated to the application cycle of the device.

Attribute Type: FTIME

Allowed Values: ≥ 0

Default Value: minimum output trigger time value

5.2.2.4.3 Device Output Time Resolution

The device output time resolution represents the minimum difference between to output time values that can be distinguished by the device.

Attribute Type: FTIME

Allowed Values:

0	global output trigger time can not be written
> 0	device output time resolution

5.2.2.5 Receive Process Data Mapping Attributes

The receive process data mapping attributes describe how PDs are received by the node. A node can receive PDs via one or more isochronous channels or via one or more asynchronous connections. The attributes are stored in a Receive Process Data Mapping Parameter List (see chapter **7.7.3.5**). In chapter **6.1** the PD APDU structure is described which considers the contents of these attributes.

5.2.2.5.1 Number of Receive Process Data (number_of_R_PD)

This attribute indicates the number of process data for which the given node is the consumer.

Attribute Type: UNSIGNED12

Allowed Values: UNSIGNED12 range

5.2.2.5.2 Process Data Source (PD_source)

This attribute contains the source of process data. If the attribute asynchronous process data transmission functionality is TRUE this attribute contains the physical identifier concerning [R1] of the producer. Otherwise this attribute contains the isochronous channel. A node may receive PDs via one or more isochronous channels or via one or more asynchronous connections.

Attribute Type: UNSIGNED6

Allowed Values: UNSIGNED6 range

5.2.2.5.3 Receive Process Data Header Offset (R_PD_header_offset)

This attribute indicates the position of the process data header within the receive APDU. It contains the number of quadlets before the process data header, counted from the beginning of the APDU.

Attribute Type: UNSIGNED13

Allowed Values: > 0 and ≤ number of process data

A process data APDU contains one APDU header of one quadlet, and one process data header of one quadlet for each process data. The position of the first quadlet of the first process data header is behind the APDU header.

5.2.2.5.4 Receive Process Data Length (R_PD_length)

This attribute contains the number of octets of the associated receive process data.

Attribute Type: UNSIGNED13

Allowed Values:

0	The associated PDs are dynamic PDs. This means the process data header in the process data APDU has to be checked to discover whether process data is contained and what the current length is.
>0	The associated PDs are static PDs. This means the PDs are transmitted in each node specific application cycle with the given length. The PD update service shall provide the size argument.

5.2.2.5.5 Receive Process Data Offset (R_PD_offset)

This attribute indicates the position of the associated process data within the receive APDU. It contains the number of quadlets before the process data, counted from the beginning of the APDU.

Attribute Type: UNSIGNED13

Allowed Values: > number of process data within the receive APDU

A process data APDU contains one APDU header of one quadlet, and one process data header of one quadlet for each process data. The position of the first quadlet of the first process data is behind these headers. The offset is one quadlet less than the number of all headers.

5.2.2.5.6 Receive Process Identifier of Receive Process Data (R_process_ID)

The receive process identifier is intended for communication purposes of the user. The contents may be specified in profiles.

Attribute Type: UNSIGNED4

Allowed Values: specified in profiles

Default Value: specified in profiles

5.2.2.6 Transmit Process Data Mapping Attributes

The transmit process data mapping attributes describe how PDs are transmitted by the node. A node can transmit process data via one isochronous channel or via one or more asynchronous connections, whereas one and the same APDU is transmitted to different destinations. The attributes are stored in a Transmit Process Data Mapping Parameter List (see chapter **7.7.3.6**). In chapter **6.1** the PD APDU structure is described which considers the contents of these attributes.

5.2.2.6.1 Number of Transmit Process Data (number_of_T_PD)

This attribute indicates the number of process data which are produced by the given node.

Attribute Type: UNSIGNED12

Allowed Values: UNSIGNED12 range

5.2.2.6.2 Process Data Destination (PD_destination)

This attribute contains the destination of process data. If the asynchronous process data transmission functionality value is TRUE then this is the physical identifier of the consumer. Otherwise it is the isochronous channel. A node may transmit process data via one isochronous channels or via one or more asynchronous connections.

Attribute Type: UNSIGNED6

Allowed Values: UNSIGNED6 range

5.2.2.6.3 Transmit Process Data Header Offset (T_PD_header_offset)

This attribute indicates the position of the process data header within the transmit APDU. It contains the number of quadlets before the process data header, counted from the beginning of the APDU.

Attribute Type: UNSIGNED13

Allowed Values: > 0 and ≤ number of process data

A process data APDU contains one APDU header of one quadlet, and one process data header of one quadlet for each process data. The position of the first quadlet of the first process data header is behind the APDU header.

5.2.2.6.4 Transmit Process Data Length (T_PD_length)

This attribute contains the number of octets of the associated transmit process data.

Attribute Type: UNSIGNED13

Allowed Values:

0	The associated PDs are dynamic process data. This means the PDs are transmitted only if the PD update service was requested. The number of octets are taken from the service argument size.
>0	The associated PDs are static PDs. This means the PDs are transmitted in each node specific application cycle with the given length. The size provided by the PD update service may be ignored.

5.2.2.6.5 Transmit Process Data Offset (T_PD_offset)

This attribute indicates the position of the associated process data within the transmit APDU. It contains the number of quadlets before the process data, counted from the beginning of the APDU.

Attribute Type: UNSIGNED13

Allowed Values: > number of process data

A process data APDU contains one APDU header of one quadlet, and one process data header of one quadlet for each process data. The position of the first quadlet of the first process data is behind these headers. The offset is one quadlet less than the number of all headers.

5.2.2.6.6 Transmit Process Identifier of Transmit Process Data (T_process_ID)

The transmit process identifier is intended for communication purposes of the user. The contents may be specified in profiles.

Attribute Type: UNSIGNED4

Allowed Values: specified in profiles

Default Value: specified in profiles

5.2.3 Services

5.2.3.1 Process Data Update Service

The process data transmission follows the producer/consumer relationship as described in chapter 4. For the Update PD service the push model is valid. There are zero or more consumers of PD. A PD has exactly one producer.

Through this service the producer updates the mapped process data. The consumer gets an indication if new data is available. The PD update service is defined as shown in Table 5.5.

Table 5.5 – PD Update Service Primitives

Parameter	Request / Indication
Argument	**Mandatory**
PD Number	mandatory
Size	mandatory
Data	mandatory

Table 5.6 – PD Update Service Arguments

Argument	Description
PD Number	The process data number is a local reference to process data of the node.
Size	The size argument contains the number of octets of process data provided to and by the service.
Data	Process data

The PD number indicates the position in the transmit process data mapping parameter list respectively in the receive process data mapping parameter list contained in the CSR.

5.2.3.2 Synchronization Service

Through this service clock synchronized behavior can be implemented. It includes one service primitive only, the SYNC indication. The isochronous data transfer is maintained by the IEEE 1394 link layer. That is why no request is necessary by an application layer service.

The synchronization service indicates the expiring of the application cycle. It is based on a cycle start event which is associated to the application cycle of the device.

Table 5.7 – SYNC Service Primitive

Parameter	Indication
Argument Application Cycle Count	**Mandatory** mandatory

Table 5.8 – SYNC Service Argument

Argument	Description
Application Cycle Count	The current application cycle count refers to the number of basic application cycles since bus reset.

5.3 Service Data ASE

5.3.1 Overview

The service data application layer service element describes the means, provided by the application layer, to transmit non-time critical parameter (and even complete software modules) for automation devices. The services are used by the application layer user, which can be a communication profile or device profiles.

The Service Data ASE includes the attributes listed in Table 5.9. The Service Data ASE attributes are described in detail in chapter 5.3.2. The coding is described in chapter **7**.

Table 5.9 – Service Data ASE Attributes

Abbreviation	Description	M/O	CSR coding
num_of_SD_con	Number of Service Data Connections provided by the node	M	**7.7.3.7**
R_SD_destination	Receive Service Data Destination Offset	M	**7.7.3.7**
R_SD_length	Receive Service Data Length	M	**7.7.3.7**
T_SD_destination	Transmit Service Data Destination Offset	M	**7.7.3.7**
T_SD_length	Transmit Service Data Length	M	**7.7.3.7**

As illustrated in Figure 5.5 three services are available for service data transmission. Through the service data services a sequence of asynchronous write and read services of the IEEE 1394 transaction layer is induced.

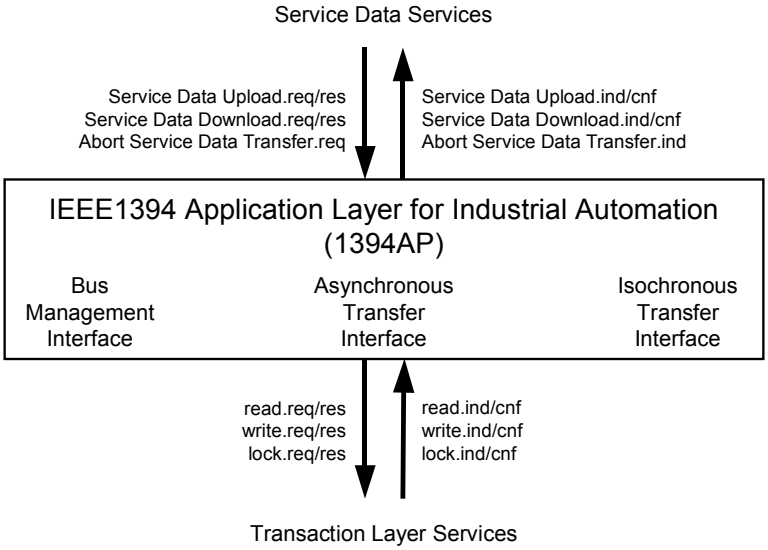

Figure 5.5 – Service Data Services

5.3.2 Attributes

The service data mapping attributes describe how SDs are received and transmitted by the node. A node can receive and transmit service data via one or more asynchronous connections. The attributes are stored in a Service Data Mapping Parameter List (see chapter 7.7.3.7). In chapter 6.3 the SD APDU structure is described which considers the contents of these attributes.

5.3.2.1 Number of Service Data Connections (number_of_SD_con)

This Attribute indicates the maximum number of service data connections, which are provided simultaneously by the service data server.

Attribute Type: UNSIGNED8

Allowed Values: UNSIGNED8 range

5.3.2.2 Receive Service Data Destination Offset (R_SD_destination)

This Attribute contains the destination offset for service data packets, directed to the SD server. An SD client may read this value to be able to execute service data uploads or downloads.

Attribute Type: UNSIGNED24

Allowed Values: UNSIGNED24 range

5.3.2.3 Receive Service Data Length (R_SD_length)

This Attribute contains the maximum number of octets of service data, that can be received by the SD server with one SD packet.

Attribute Type: UNSIGNED12

Allowed Values: UNSIGNED12 range

5.3.2.4 Transmit Service Data Destination Offset (T_SD_destination)

This Attribute contains the destination offset for service data packets, provided by the SD server. An SD client may read this value to be able to execute service data uploads or downloads.

Attribute Type: UNSIGNED24

Allowed Values: UNSIGNED24 range

5.3.2.5 Transmit Service Data Length (T_SD_length)

This Attribute contains the maximum number of octets of service data, that can be transmitted by the SD server with one SD packet.

Attribute Type: UNSIGNED12

Allowed Values: UNSIGNED12 range

5.3.3 Services

5.3.3.1 Service Data Upload Service

Through this service a client uploads service data from a server. The service is confirmed. The remote result parameter indicates the success or failure of the request. In the case of a failure, optionally the reason is confirmed. In the case of success, the data and its size are confirmed.

Table 5.10 – SD Upload Service Primitives

Parameter	Request / Indication	Response / Confirm
Argument	**Mandatory**	
SD number	mandatory	
Multiplexor	mandatory	
Size	mandatory	
Remote Result		**Mandatory**
Success		selection
Data		mandatory
Size		optional
Failure		selection
Reason		optional

Table 5.11 – SD Upload Service Arguments

Argument	Description
SD Number	The service data number is a local reference to a service data connection of the node. It considers the AP_device_ID and the receive and transmit SD destination offsets.
Multiplexor	This argument is used to address a certain object to be uploaded.
Size	The size argument contains the number of octets of service data, which are provided to the service or provided by the service.
Data	Service data
Reason	Reason for an unsuccessful upload.

5.3.3.2 Service Data Download Service

Through this service an SD client downloads service data to the SD server. The service is confirmed. The remote result parameter indicates the success or failure of the request. In the case of a success, optionally the size is confirmed. In the case of failure, optionally the reason is confirmed.

Table 5.12 – SD Download Service Primitives

Parameter	Request / Indication	Response / Confirm
Argument	**Mandatory**	
SD Number	mandatory	
Multiplexor	mandatory	
Size	mandatory	
Data	mandatory	
Remote Result		**Mandatory**
Success		selection
Size		optional
Failure		selection
Reason		optional

Table 5.13 – SD Download Service Arguments

Argument	Description
SD Number	The service data number is a local reference to a service data connection of the node. It considers the AP_device_ID and the receive and transmit SD destination offsets.
Multiplexor	This argument is used to address a certain object to be downloaded.
Size	The size argument contains the number of octets of service data, which are provided to the service or provided by the service.
Data	Service data
Reason	Reason for an unsuccessful download.

5.3.3.3 Service Data Abort Service

This service aborts the upload or download of service data. Optionally the reason is indicated. The service is unconfirmed. The service may be executed at any time by both the client and the server of the service data. If a confirmed service is outstanding, the indication of the abort is taken to be the confirmation of that service.

Table 5.14 – SD Abort Service Primitives

Parameter	Request / Indication
Argument	**Mandatory**
SD number	mandatory
Reason	mandatory

Table 5.15 – SD Abort Service Arguments

Argument	Description
SD Number	The service data number is a local reference to a service data connection of the node. It considers the AP_device_ID and the receive and transmit SD destination offsets.
Reason	Reason for an unsuccessful upload.

5.4 CSR Data ASE

5.4.1 Overview

The CSR data application layer service element describes the means, provided by the application layer, to transmit CSR data. The services are used by the application layer user, which can be a communication profile or device profiles. With these services a direct access to CSR registers is possible, using the IEEE 1394 destination address scheme. In particular IEEE 1394 specific CSR register or manufacturer specific CSR register can be accessed.

As illustrated in Figure 5.6 the CSR data services are directly mapped to the transaction layer services.

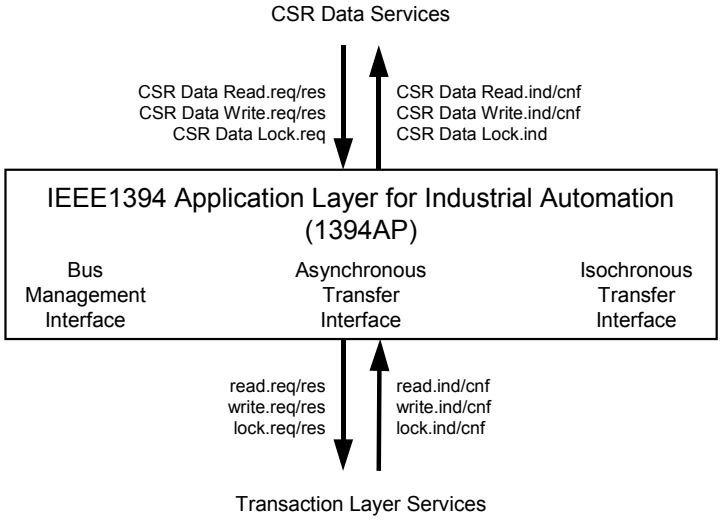

Figure 5.6 – CSR Data Services

5.4.2 Services

5.4.2.1 CSR Data Read Service

Through this service a client reads CSR entries. The definition of the service primitives and their arguments is in accordance with IEEE 1394.

5.4.2.2 CSR Data Write Service

Through this service a client writes CSR entries. The definition of the service primitives and their arguments is in accordance with IEEE 1394.

5.4.2.3 CSR Data Lock Service

Through this service a client performs an atomic operation to the CSR entries. The definition of the service primitives and their arguments is in accordance with IEEE 1394.

5.5 Emergency Data ASE

5.5.1 Overview

The emergency data ASE is optional. An emergency data service is triggered by an emergency producer on the device if a device internal error situation occurs. The emergency ASE is suitable for interrupt type error alerts. An

emergency service is called only once per error event. As long as no new errors occur on a device no further calls are necessary.

The emergency data message may be received by zero or more emergency consumers. The reaction on the emergency consumer(s) is not specified and does not fall in the scope of this document. Device specific additional information and the emergency condition also do not fall into the scope of this specification.

The Emergency Data ASE includes the attributes listed in Table 5.16. The Emergency Data ASE attributes are described in detail in chapter 5.5.2. As far as the attributes are stored in the CSR area the coding is described in chapter **7**.

Table 5.16 – Emergency ASE Attributes

Abbreviation	Description	M/O	CSR coding
error_code	Error Code	M	**7.7.2.4**
emergency_err_code	Emergency Error Code	M	**7.7.2.4**
profile_spec_err_code	Profile Specific Error Code	O	**7.7.2.4**
man_spec_err_code	Manufacturer Specific Error Code	O	**7.7.2.4**

5.5.2 Attributes

5.5.2.1 Error Code (error_code)

The error code is used to indicate internal errors of the node. The error code is a field in the emergency service packet and is stored in the error register and the error list of the application layer CSR area.

Allowed Values:

Generic error
Current
Voltage
Temperature
Communication error (overrun, error state)

5.5.2.2 Emergency Error Code (emergency_err_code)

The emergency error code is used to indicate internal errors of the node. The emergency error code is an argument of the emergency service and a field in the emergency service packet.

Allowed Values:

Error Reset or No Error
Generic Error
Current
Current, device input side
Current inside the device
Current, device output side
Voltage
Mains Voltage

Voltage inside the device
Output Voltage
Temperature
Ambient Temperature
Device Temperature
Device Hardware
Device Software
Internal Software
User Software
Data Set
Additional Modules
Monitoring
Communication
Life Guard Error
Protocol Error
External Error
Additional Functions
Device specific

5.5.2.3 Profile Specific Error Code (profile_spec_err_code)

This attribute may be defined by a communication profile or by a device profile. The profile specific error code is a field in the emergency service packet and is stored in the error register and the error list of the application layer CSR area.

5.5.2.4 Manufacturer Specific Error Code (man_spec_err_code)

This attribute may be defined by the manufacturer. The manufacturer specific error code is a field in the emergency service packet and is stored in the error register and the error list of the application layer CSR area.

5.5.3 Services

5.5.3.1 Emergency Notification Service

This emergency notification service is used by a node to notify the occurrence of an error to one or all nodes in the network.

Table 5.17 – Emergency Notification Service Primitives

Parameter	Request/Indication
Argument	**Mandatory**
AP_device_ID	selection
All	selection
error_code	mandatory
emergency_err_code	optional
profile_spec_err_code	optional
man_spec_err_code	optional

Table 5.18 – Emergency Notification Service Arguments

Argument	Description
AP_device_ID	This argument identifies the destination node in the request services primitive and the source in the indication service primitive.
All	This argument is used in the request service primitive to address all nodes in the network.
error_code	Error code indicating the emergency event.
emergency_err_code	Error code defined in the application layer specification.
profile_spec_err_code	Error code defined by a communication or device profile.
man_spec_err_code	Error code defined by the manufacturer of the node.

5.6 Network Management ASE

5.6.1 Overview

The network management application layer service element describes the means, provided by the application layer, for station and network management.

The network management ASE includes the attributes listed in Table 5.19. The Network Management ASE attributes are described in detail in chapter 5.6.2. If the attributes are stored in the CSR area, then the coding is described in chapter **7**.

Table 5.19 – Network Management ASE Attributes

Abbreviation	Description	M/O	CSR coding
AP_device_ID	Automation Profile Device Identifier	M	**7.7.2.1**
AL_functionality	Application Layer Functionality	M	**7.7.2.7**
NMT_state	Network Management State	M	**7.7.2.6**
boot_up_result	Boot up result	M	**7.7.2.6**
start_cycle_cnt	Start bus cycle count	O	-
Alloc_iso_channel	Allocated Isochronous Channel	O	**7.7.2.8**
alloc_bandwidth	Allocated Bandwidth	O	**7.7.2.9**
guard_time	Guard Time	O	**7.7.2.19**
life_time_factor	Life-time Factor	O	**7.7.2.19**

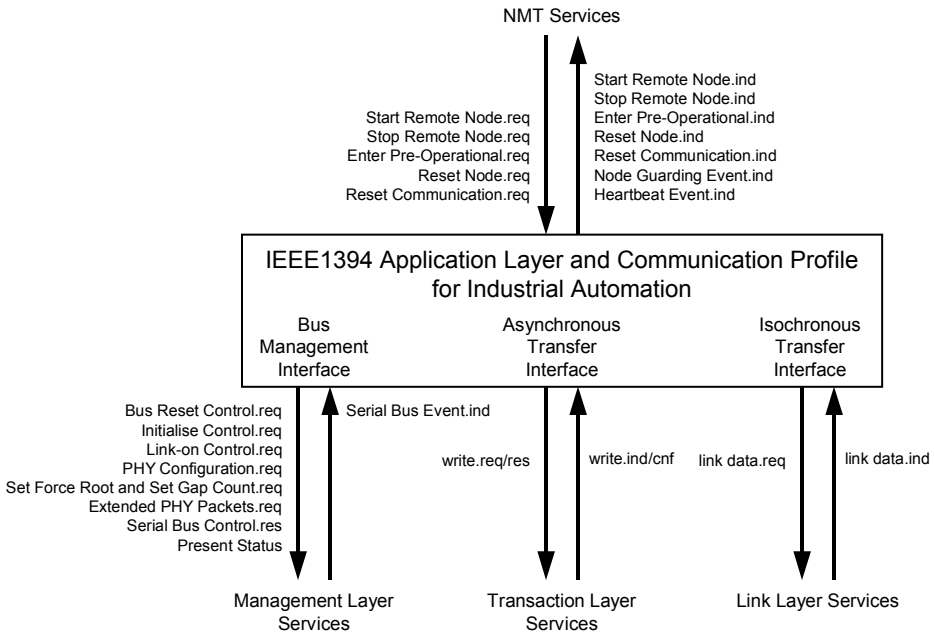

Figure 5.7 – NMT Services

5.6.2 Attributes

5.6.2.1 Automation Profile Device Identifier (AP_device_ID)

The AP Device ID identifies the place of a node within the automation application. The values of the AP Device IDs are specified in the planning phase of the automation application. The AP Device ID insures the unproblematic exchange of a node in the case of maintenance or fixing. A dynamic address table contains the relations between the AP Device ID, specified for the application, and the physical ID, which could be changed after each bus reset.

Attribute Type: UNSIGNED6

Allowed Values: UNSIGNED6 range

5.6.2.2 Application Layer Functionality (AL_func)

This attribute refers to the 1394AP capabilities. It is used to activate or deactivate the capabilities as well as to indicate the current status.

Allowed Values:

application master functionality
asynchronous process data transmission functionality
event triggered transmission functionality
clock synchronised transmission functionality

5.6.2.3 Network Management State (NMT_state)

This attribute is used to control the NMT state machine and to indicate the current state.

Allowed Values:

INITIALIZATION
STOPPED
OPERATIONAL
PRE-OPERATIONAL

5.6.2.4 Boot Up Result (bu_result)

This attribute indicates the result of the node after boot up.

Allowed Values:

successful
isochronous band width not available
no isochronous channel available
synchronisation impossible

5.6.2.5 Application Cycle Start Bus Cycle Count (ac_start_cycle_cnt)

This attribute contains the bus cycle count value at which the application cycle count value is zero (0). If a node identifies that the bus cycle count transmitted with the cycle start telegram is equal to the given attribute application cycle start bus cycle count value, it resets the application cycle counter to zero (0) and starts the application cycle counter.

Attribute Type: UNSIGNED32

Allowed Values: UNSIGNED32 range

5.6.2.6 Allocated Isochronous Channel (alloc_iso_channel)

This attribute holds the isochronous channel allocated by the node for process data transmission. Each node can allocate one channel only.

Allowed Values: ≤64

0	The isochronous channel 0 shall be used by the application master.
>0 and ≤63	Isochronous channel of the device.
64	No isochronous channel allocated for process data transmission.

5.6.2.7 Allocated Bandwidth (alloc_bandwidth)

This attribute informs about the bandwidth allocated by the node for process data transmission.

Allowed Values: as specified in IEEE 1394 [R1]

5.6.2.8 Guard Time (guard_time)

This attribute defines the period an NMT service provider on the NMT master shall transmit a guarding request.

Attribute Type: UNSIGNED16

Allowed Values:

0	guard time not supported
> 0	guard time in milliseconds

Default Value: 0

5.6.2.9 Life-time Factor (life_time_factor)

The value of this attribute multiplied by the guard time value result in the life time. If the life time elapsed without an NMT slave recognizing a guarding request, then the node initiates a life guarding event. If the life time elapsed without an NMT master recognizing a guarding response, then the node initiates a node guarding event.

Attribute Type: UNSIGNED8

Allowed Values:

0	life-time factor not supported
> 0	life-time factor

Default Value: 0

5.6.3 Services

5.6.3.1 Module Control Services

Through Module Control Services, the NMT master controls the state of the NMT slaves. The state attribute is one of the values {STOPPED, PRE-OPERATIONAL, OPERATIONAL, INITIALISING}. The Module Control Services can be performed with a certain node or with all nodes simultaneously. The NMT master controls its own NMT state machine via local services, which are implementation dependent. The Module Control Services except Start Remote Node can be initiated by the local application.

5.6.3.1.1 Start Remote Node

Through this service the NMT Master sets the state of the selected NMT Slaves to OPERATIONAL. The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be OPERATIONAL.

Table 5.20 – Start Remote Node Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All ac_start_cycle_count	**Mandatory** selection selection mandatory

Table 5.21 – Start Remote Node Service Arguments

Argument	Description
AP_device_ID	The automation profile device identifier is used to address a single device.
All	This argument is used to address all devices in the network.
ac_start_cycle_count	This argument contains the bus cycle count value at which the application count value is reset to zero (0).

5.6.3.1.2 Stop Remote Node

Through this service the NMT Master sets the state of the selected NMT Slaves to STOPPED. The service is unconfirmed and mandatory. After completion of the service, the state of the selected remote nodes will be STOPPED.

Table 5.22 – Stop Remote Node Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All	**Mandatory** selection selection

5.6.3.1.3 Enter Pre-Operational

Through this service the NMT Master sets the state of the selected NMT Slaves to PRE-OPERATIONAL. The service is unconfirmed and mandatory for all devices. After completion of the service, the state of the selected remote nodes will be PRE-OPERATIONAL.

Table 5.23 – Enter Pre-Operational Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All	**Mandatory** selection selection

5.6.3.1.4 Reset Node

Through this service the NMT Master sets the state of the selected NMT Slaves from any state to the RESET APPLICATION sub-state. The service is unconfirmed and mandatory for all devices. After completion of the service, the state of the selected remote nodes will be RESET APPLICATION.

Table 5.24 – Reset Node Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All	**Mandatory** selection selection

5.6.3.1.5 Reset Communication

Through this service the NMT Master sets the state of the selected NMT Slaves from any state to the RESET COMMUNICATION sub-state. After completion of the service, the state of the selected remote nodes will be RESET COMMUNICATION. The service is unconfirmed and mandatory for all devices.

Table 5.25 – Reset Communication Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All	**Mandatory** selection selection

5.6.3.1.6 Store Parameter

Through this service one or all node(s) may be forced to store the configuration parameter into a non-volatile memory or to restore the values back into the registers.

Table 5.26 – Store Parameter Service Primitives

Parameter	Request/Indication
Argument AP_device_ID All Store	**Mandatory** selection selection mandatory

Table 5.27 – Store Parameter Service Arguments

Argument	Description
AP_device_ID	The automation profile device identifier is used to address a single device.
All	This argument is used to address all devices in the network.
Store	This argument indicates whether the configuration registers shall be stored or restored.

5.6.3.2 Error Control Services

Through Error control services the nodes detect failures in a 1394AP Network.

Local errors in a node may e.g. lead to a reset or change of state. The definition of these local errors does not fall into the scope of this specification.

Error Control services are achieved principally through the periodically transmitting of messages by a node.

5.6.3.2.1 Node Guarding Event

Through this service, the NMT service provider on the NMT Master indicates that a remote error occurred or has been resolved for the remote node identified by AP_device_ID.

Table 5.28 – Node Guarding Event

Parameter	Indication
Argument AP_device_ID State Occurred Resolved	**Mandatory** mandatory mandatory selection selection

The service is provider initiated and optional.

5.6.3.2.2 Life Guarding Event

Through this service, the NMT service provider on an NMT Slave indicates that a remote error occurred or has been resolved.

Table 5.29 – Life Guarding Event

Parameter	Indication
Argument State Occurred Resolved	**Mandatory** mandatory selection selection

The service is provider initiated and optional.

6 Application Layer Transfer Syntax

6.1 Process Data APDU

6.1.1 Overview

The PDs are transmitted in form of the Process Data Application Layer Protocol Data Unit (PD APDU). It is embedded into the data block of an isochronous data packet or into an asynchronous write packet, depending on the transmission mode. The PDs within the APDU are updated by the Process Data Update service. The transmission is initiated according to the transmission mode and the triggering mode indicated in the 1394AP Capability Register and the 1394AP Status Register.

A process data APDU has to support static PD and dynamic PD.

Static PD are characterized by a fixed length. So the position of a static PD within an APDU is fixed. A node does not need to examine the PD header.

The length and the offset of dynamic PD can be changed during operational mode. Dynamic PD can even be absent in an APDU. Therefore a node has to examine the PD header which is associated to the relevant PD. Since the number and the sequence of PD headers in an APDU is fixed, a node knows where to find the relevant PD header.

All static PD shall be placed first in the APDU. Dynamic PD may follow.

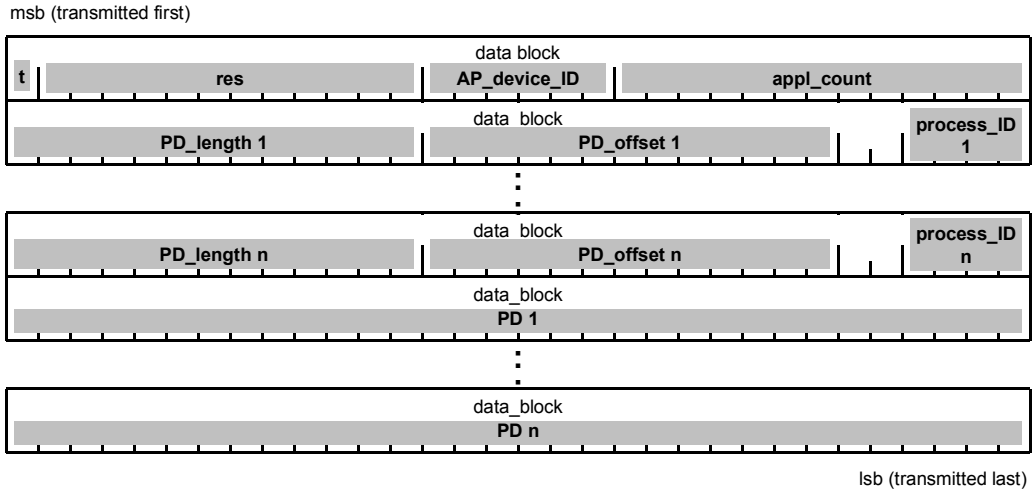

Figure 6.1 – Process Data APDU Structure

Table 6.1 – Process Data APDU Description

Abbreviation	Description
APDU Header	
t	Telegram Type
res	Reserved (always 0)
AP_device_ID	Automation Profile Device Identifier of the provider
appl_count	Application Cycle Count
Process Data Header x	
PD_length x	Process Data Length of PD x
PD_offset x	Process Data Offset of PD x
process_ID x	Process Identifier of PD x
Process Data x	
PD x	Process Data

6.1.2 APDU Fields

6.1.2.1 Telegram Type

This field is used to indicate the direction of the telegram.

Coding:

Code	Description
0	Device Data Telegram (DDT), transmitted from a device to the application master or to another device
1	Master Data Telegram (MDT), transmitted from the application master to one or more devices

6.1.2.2 Automation Profile Device Identifier

The AP device ID identifies the producer of the APDU.

The coding is specified in the planning phase of the automation application.

6.1.2.3 Application Cycle Count

The application cycle count indicates the application cycle, in which this telegram is sent. The application cycle count is a multiple of the bus cycle count (125µs interval). The application cycle counter is a continuous up-counter, which starts after bus reset with zero and counts the basic application cycles.

6.1.2.4 Process Data Length

This field contains the length of the process data.

Coding:

Code	Description
0	PD length unspecific It is not required to transmit the process data length for static PD. The receiver takes the value if necessary out of the CSR. The PD length field shall be zero in that case. A dynamic PD is not contained in the APDU if its process data length is zero.
>0	Number of octets of the process data.

6.1.2.5 Process Data Offset

This field contains the offset of the process data within the APDU.

Coding:

Code	Description
0	PD offset unspecific It is not required to transmit the process data offset for static PD. The receiver takes the value if necessary out of the CSR. The PD offset field shall be zero in this case. A dynamic PD is not contained in the APDU if its process data offset is zero.
> 0 and < number of PD headers within the APDU	These values must not be used.
> number of PD headers within the APDU	Number of quadlets before the process data, counted from the beginning of the APDU.

6.1.2.6 Process Identifier

The process ID is intended for communication purposes of the user.

The coding may be specified in profiles.

6.1.2.7 Process Data

The PDs are quadlet aligned, starting at a quadlet boundary. If necessary unused octets are padded. If an APDU contains static and dynamic PD, the static PD shall be placed first in the APDU.

6.2 Process Data Protocol

6.2.1 Overview

Process data may be transmitted using isochronous transactions or asynchronous transactions. However, a node supports one transmission type only at a time.

The transmission of process data may be clock synchronized or event triggered. However, a node supports one trigger mode only at a time.

An application master has to determine the process data attributes values of the relevant devices during the state RESET COMMUNICATON (see chapter 6.10.1.1). If necessary relevant attributes values have to be delivered to the devices as well.

6.2.2 Clock Synchronized Process Data Communication

The isochronous transaction is a fundamental capability defined in standard IEEE 1394. It is based on a 8kHz system clock.

The timing of clock synchronised isochronous transactions for process data transmissions is depicted in Figure 6.2.

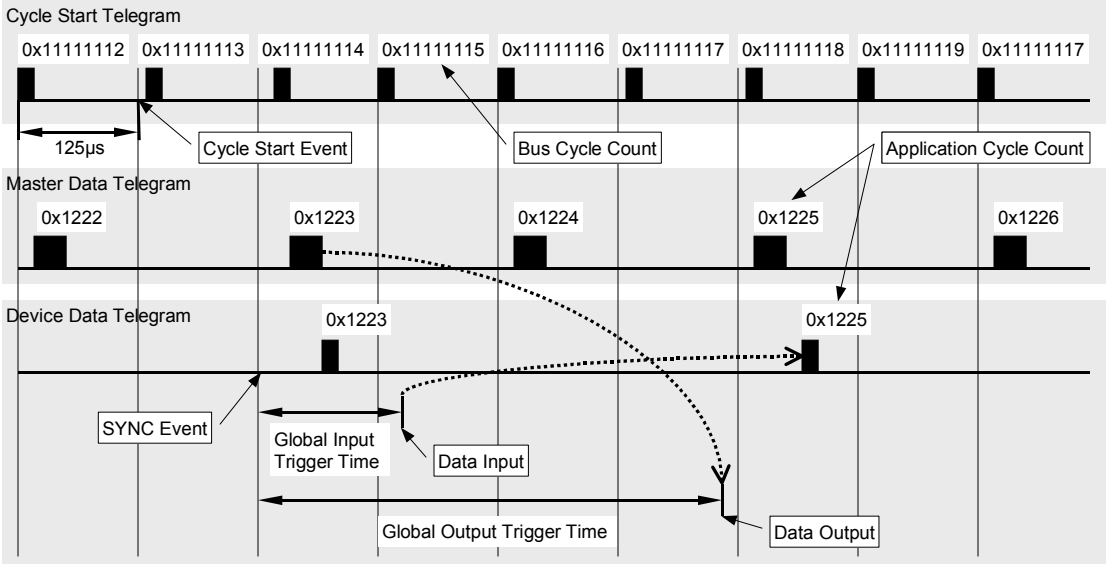

Figure 6.2 – Timing of a Clock Synchronised Isochronous Transaction

A cycle start telegram is transmitted approximately each 125µs and delivers the cycle time data (including the bus cycle count). This specification presupposes a cyclic clock event without jitter for isochronous transactions (cycle start event).

An application cycle is introduced, which is a multiple of the bus cycle e.g. for applications that do not require a cycle period of 125µs. The application cycle is indicated by the master data telegram, which follows the cycle start telegram and contains the application cycle count. The application master occupies channel 0 for the master data telegram.

The application cycle of the devices may differ from the basic application cycle (see Figure 5.2and Figure 6.2). To indicate the relation of a certain device data telegram to the application cycle, the application cycle count is contained in each device data telegram.

If the operational mode is activated (see **6.10.1.3**) the PD transmission is initiated by the synchronisation event. The synchronisation event is generated if the cycle start event occurs and the current application cycle is completed.

An application master prepares the output data and transmits the APDU using channel 0.

A device prepares the data acquisition if input data is supported. The data is sampled if the global input trigger time is expired. The input data is transmitted in the next possible isochronous channel slot associated with the device and its application cycle.

If output data is supported, this data, received with an MDT (channel 0), is written out if the global output trigger time is expired. The same procedure is valid if the output data is received with a DDT (device-to-device-communication) with the same application cycle count.

Process data may also be transmitted using asynchronous write packets. The synchronisation behaviour is in principle the same for isochronous and asynchronous transactions. Only the delay time and the jitter will probably be greater for asynchronous PD transactions.

6.2.3 Event Triggered Process Data Communication

The timing of an event triggered isochronous transaction is depicted in Figure 6.3.

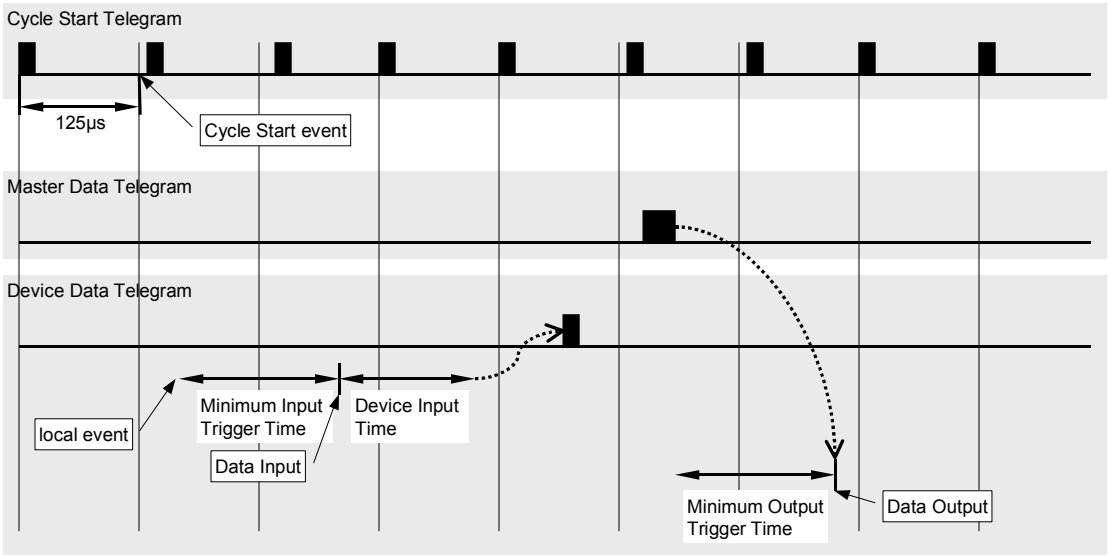

Figure 6.3 – Timing of an Event Triggered Isochronous Transaction

A local event triggers the data acquisition. The input data is prepared for transmission and is transmitted in the next possible isochronous channel associated with the device.

Receive output data is written to the periphery just after reception. In Figure 6.3 the reception of an MDT is depicted. The same procedure is valid if the output data is received in another channel (device-to-device-communication).

As for clock synchronized PD communication, asynchronous transactions can be used for event triggered PD communication. The restrictions concerning the broadcast is also for this transmission type valid.

6.3 Service Data APDU

6.3.1 Overview

The SDs are transmitted in form of the Service Data Application Layer Protocol Data Unit (SD APDU). It is embedded into the data block of an asynchronous write request packet or an asynchronous read response packet, respectively.

The structure of an SD APDU is depicted in Figure 6.4.

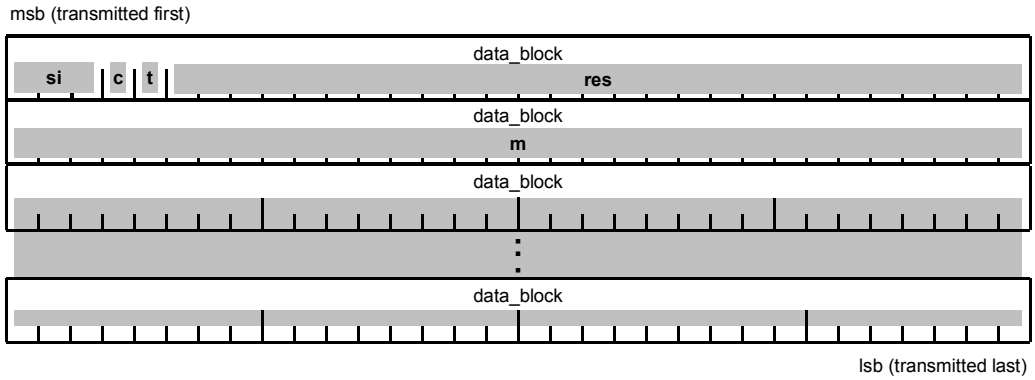

Figure 6.4 – Service Data APDU Structure

Table 6.2 – Service Data APDU Description

Abbreviation	Description
Si	Service Identifier
C	Complete Flag
T	Toggle Bit
Res	Reserved (always 0)
M	Multiplexor
D	Data

6.3.2 APDU Fields

6.3.2.1 Service Identifier

This field is used to indicate the service primitive within the upload or download procedure.

Coding:

Code	Description
0	Service data abort response The APDU is embedded in an asynchronous read response packet. The APDU contains the multiplexor and a reason code.
1	Service data download request The APDU is embedded in an asynchronous write request packet. The APDU contains the multiplexor and service data to be downloaded to the SD server.
2	Service data download response The APDU is embedded in an asynchronous read response packet. The APDU contains the multiplexor. In the case of failures the APDU contains a reason code provided by the SD server.
3	Service data upload request The APDU is embedded in an asynchronous write request packet. The APDU contains the multiplexor of the service data to be uploaded from the SD server.
4	Service data upload response The APDU is embedded in an asynchronous read response packet. The APDU contains the multiplexor and service data to be uploaded. In the case of failures the APDU contains a reason code instead of service data.

6.3.2.2 Complete Flag

This field is used to control a segmented upload or download sequence. It is relevant in service data download request packets and in service data upload response packets. In all other packets its value is 1.

Coding:

Code	Description
0	Continue transfer The requested data size is not yet transferred. Additional write packets, in the case of service data download, or additional read packets, in the case of service data upload, are necessary.
1	Transfer completed No write request follows for the current download sequence or no read response follows for the current upload sequence.

6.3.2.3 Toggle Bit

This field is used to monitor a segmented upload or download sequence. The value shall alternate for each subsequent segment during a segmented upload or download procedure. The first segment shall have the value 0. It is relevant in service data download request packets and in service data upload response packets. In all other packets its value is 0.

6.3.2.4 Multiplexor

This field contains an index, identifying the service data. The coding may be specified in profiles.

6.3.2.5 Data

The data field contains the service data or is empty. In the case of failure this field contains a reason code.

Coding:

Table 6.3 – Reason codes

Code	Description
0503 0000h	Toggle bit not alternated.
0504 0000h	SD protocol timed out.
0504 0001h	Command specifier not valid or unknown.
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	Object does not exist in the object dictionary.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to a hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	general error
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error).

The reason codes not listed here are reserved.

6.4 Service Data Protocol

6.4.1 Overview

Each service data server offers one or more buffer pairs to receive and transmit service data packets. Each pair of buffers belongs to a connection, which is offered to SD clients. If an SD client starts an upload or download process, the referred connection can not be used by other SD clients until the started transfer is complete. Moreover, also the relevant SD client must not use the same connection for another, parallel, upload or download process. To implement this behaviour it is necessary that the transaction layer rejects write requests to the SD receive buffer until a write response has been transmitted.

An SD client has to determine the service data attribute values of the relevant servers during the Reset Communication state (see chapter 6.10.1.1).

6.4.2 Service Data Upload Protocol

The SD upload consists of one IEEE 1394 Asynchronous Write Transaction and one or more IEEE 1394 Asynchronous Read Transactions (see Figure 6.5). The SD upload consists of the following steps:

1. The SD client initiates the SD upload service and hands over the SD number, the multiplexor and the required size of service data.

2. The SD upload request service primitive builds an SD APDU with service identifier (upload request) and multiplexor. The relevant parameters, such as destination address and data length are determined and an asynchronous write request is initiated.

3. The reception of the SD upload request is indicated to the application layer user by an SD upload indication. The SD server informs the SD client with the asynchronous write response that the SD upload request was received and will be processed. The SD server may also indicate with the asynchronous write response that the SD upload request was not accepted, since another download or upload process is active with the given SD number.

4. Afterwards the SD client uses the asynchronous read service addressed to the transmit SD destination offset of the SD server to get the requested SD.

5. Additional asynchronous read requests may follow if the size exceeds the transmit SD length of the server. The last asynchronous read response initiates an SD upload confirmation in the SD client. The SDs are given to the application layer user.

6. The transaction may be aborted at any time either by the SD client or by the SD server. The SD client transmits the abort APDU with an asynchronous write request. The SD server puts the abort APDU into the asynchronous read response.

7. The asynchronous read response may contain an upload or download packet, possibly directed to another client. This is indicated by the service identifier or the SD number.

8. The SD upload process is observed by the SD server using a timeout timer.

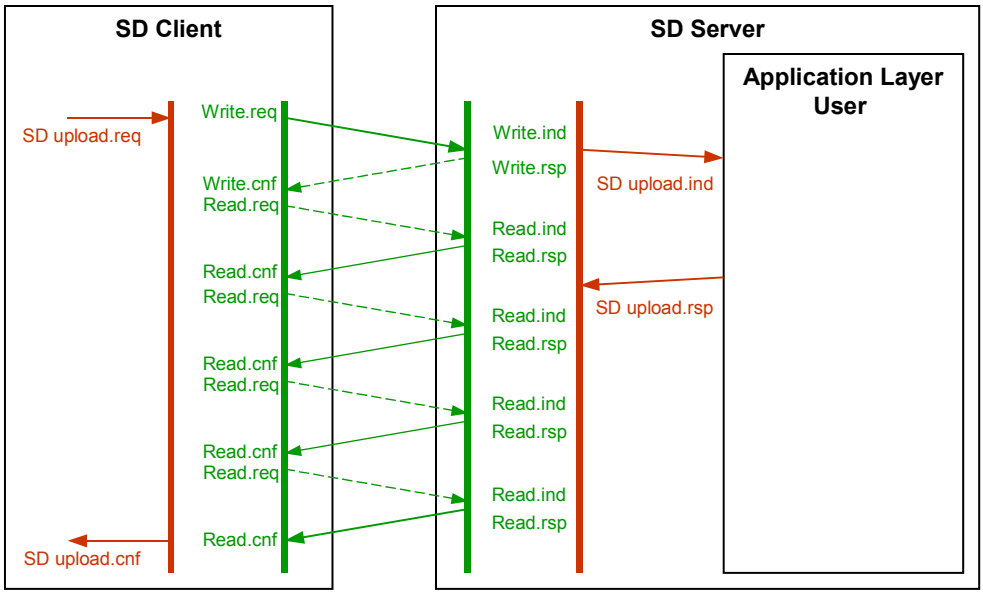

Figure 6.5 – Service Data Upload Transaction Chart

6.4.3 Service Data Download Protocol

The SD download consists of one or more IEEE 1394 Asynchronous Write Transactions and one IEEE 1394 Asynchronous Read Transaction (see Figure 6.6). The SD download consists of the following steps:

1. The SD client initiates the SD download service and hands over the SD number, the multiplexor, the size of service data and the service data itself.

2. The SD download request service primitive builds an SD APDU with service identifier (download request), complete flag, toggle bit, multiplexor, and service data. The relevant parameter, such as destination address and data length are determined, and an asynchronous write request is initiated.

3. The SD server indicates with the asynchronous write response that the SD download request was received and will be processed. The SD server may also indicate with asynchronous write response that the SD download request was not accepted, since another download or upload process is active with the given SD number.

4. Additional asynchronous write requests follow if the size exceeds the receive SD length of the server. The last asynchronous write request initiates an SD download indication. The service data are given to the application layer user. The SD download response is prepared.

5. The SD download is completed by an asynchronous read request service addressed to the transmit SD destination offset of the SD server. The asynchronous read response contains the result of the transaction. It is given with the SD download confirmation to the application layer user at the SD client.

6. The transaction may be aborted at any time either by the SD client or by the SD server. The SD client transmits the abort APDU with the asynchronous write request. The SD server responses negatively to an asynchronous write request and puts the abort APDU into the asynchronous read response. The SD client may inquire the reason by using an asynchronous read service.

7. The asynchronous read response may contain an upload or download packet, possibly directed to another client. This is indicated by the service identifier or the SD number.

8. The SD download process is watched by the SD server using a timeout timer.

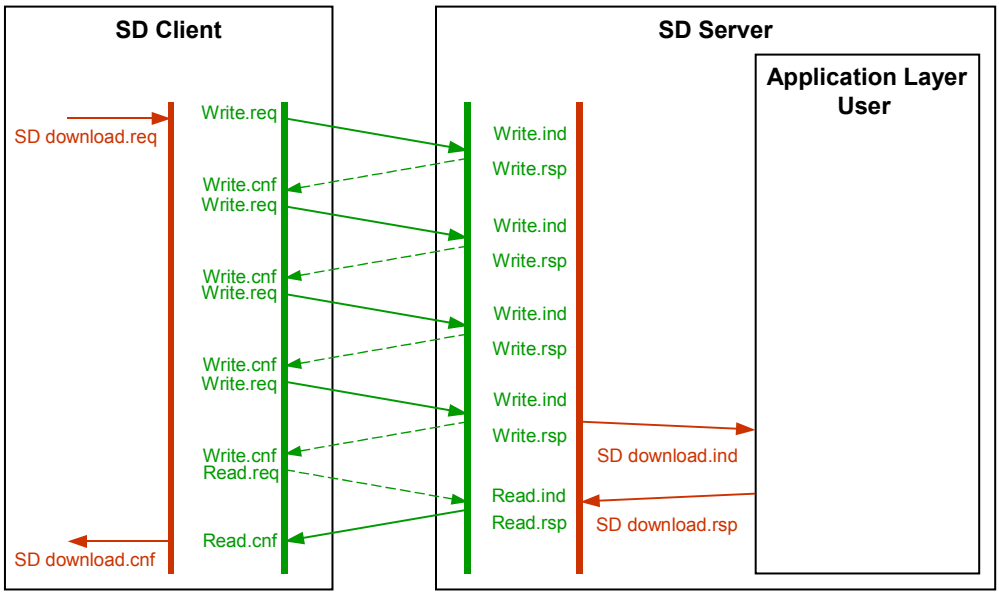

Figure 6.6 – Service Data Download Transaction Chart

6.5 CSR APDU

There is no CSR Application Layer Protocol Data Unit defined.

6.6 CSR Protocol

There is no CSR Application Layer Protocol defined.

6.7 Emergency APDUs

6.7.1 Emergency Indication APDU

An emergency indication is transmitted in the form of an Emergency Indication APDU. It is embedded in the data block of a CSR data lock packet. The structure of an EI APDU is depicted in Figure 6.7.

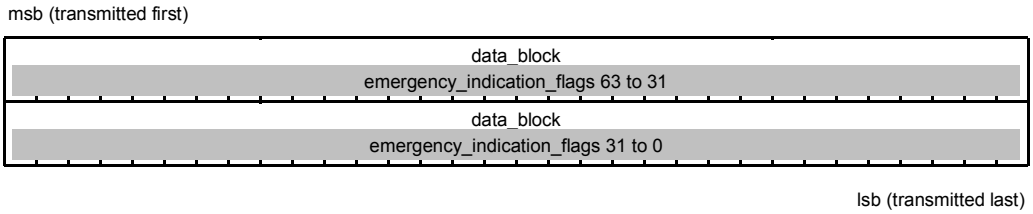

Figure 6.7 – Emergency Indication APDU Structure

6.7.2 Emergency Indication APDU Fields

The emergency indication APDU carries the emergency indication flag register. It contains 64 flags. The flag at the lowest bit position represents the node with the AP device ID zero (0), the flag with the highest bit position represents the node with the AP device ID sixty three (63).

6.7.3 Emergency Data APDU

Emergency data is transmitted in the form of an Emergency Data Application Layer Protocol Data Unit (ED APDU). It is embedded into the data block of an asynchronous read service. The structure is defined as shown in Figure 6.8.

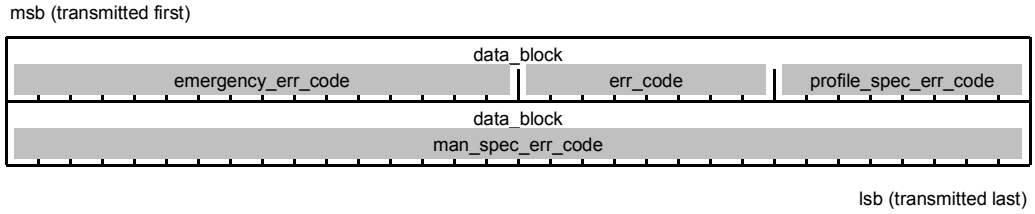

Figure 6.8 – Emergency Data APDU Structure

Table 6.4 – Emergency Data APDU Description

Abbreviation	Description
emergency_err_code	Emergency Error Code
error_code	Error Code
profile_spec_err_code	Communication or Device Profile Specific Error Code
man_spec_err_code	Manufacturer Specific Error Code

6.7.4 Emergency Data APDU Fields

The emergency data APDU carries the error register. That is why the coding of the APDU fields is as described in section **7.7.2.2**.

6.7.4.1 Emergency Error Code

This field describes in detail the reason of the emergency message.

6.7.4.2 Error Code

This field indicates the event which initiated the emergency messages.

6.7.4.3 Profile Specific Error Code

This field contains the profile specific error code stored in the error register.

6.7.4.4 Manufacturer Specific Error Code

This field contains the manufacturer specific error code stored in the error register.

6.8 Emergency Data Protocol

A device may be in one of two emergency states (Figure 6.9). Dependent on the transitions, an emergency notification and data will be transmitted. Links between the emergency state machine and the NMT state machine may be defined in the device profiles.

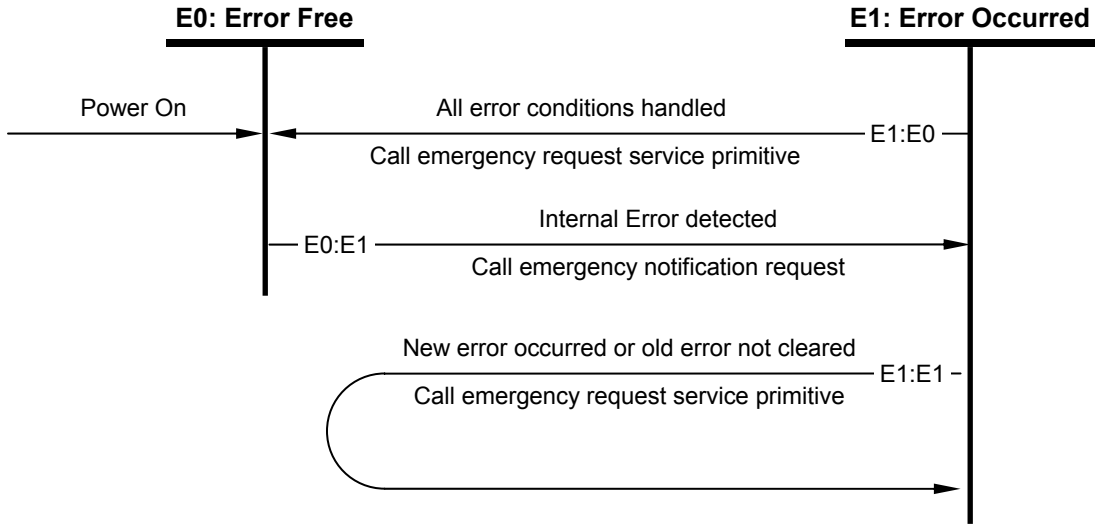

Figure 6.9 – Emergency State Machine

State E0: Error Free. After initialization, the device enters the state ERROR-FREE if no error is detected. No error message is sent.

Transition E0:E1. The device detects an internal error. The device enters the state ERROR-OCCURRED and calls the emergency notification request. This leads to one or more asynchronous data lock request(s) changing the associated emergency flag in the emergency indication register of the emergency consumer(s). The emergency consumer(s) execute(s) an asynchronous read of the error register and reset(s) the emergency flag. The emergency notification indication service primitive informs the application of the emergency consumer.

State E1: Error Occurred. Whenever an internal error has been detected, the state machine enters state Error Occurred

Transition E1:E1. One, but not all error reasons are gone. The emergency producer updates the error register, calls the emergency request service primitive to set the emergency flag within the emergency consumer(s). The emergency consumer reads the error register, resets the emergency flag and informs the application. The transition also happens, if a new error occurs on the device. The emergency producer updates the error register, calls the emergency request service primitive to set the emergency flag within the emergency consumer(s). The emergency consumer reads the error register, resets the emergency flag and informs the application.

Transition E1:E0. All errors are repaired. The emergency producer updates the error register, calls the emergency request service primitive to set the emergency flag within the emergency consumer(s) and enters the state ERROR-FREE. The emergency consumer reads the error register, resets the emergency flag and informs the application.

6.9 Network Management APDUs

6.9.1 Overview

The network management data is transmitted with different Application Layer Protocol Data Units (APDU).

6.9.2 NMT APDUs

6.9.2.1 Module Control Data APDU

The Module Control Data APDU (MCD APDU) is used to transmit network management commands from an NMT master to one or more NMT slaves. It is embedded into the data block of an asynchronous write request packet directed to the Message_Request register specified in the core register area of the CSR.

The structure of an MCD APDU is depicted in Figure 6.10.

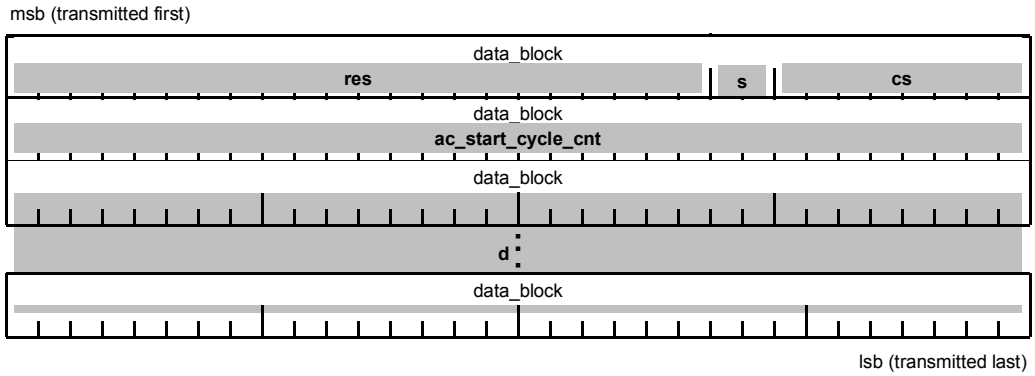

Figure 6.10 – Module Control Data APDU Structure

Table 6.5 – Module Control Data APDU Description

Abbreviation	Description
res	Reserved (always 0)
s	Store Command
cs	Command Specifier
ac_start_cycle_cnt	Application Cycle Start Cycle Count
d	Data

6.9.2.2 Module Control Data APDU Fields

6.9.2.2.1 Store Command

This field indicates whether the node shall store the configuration registers into a non-volatile memory or restore the values into the registers.

Coding:

Code	Description
0	Do not store or restore
1	Store configuration registers
2	Restore configuration registers
3	Restore default register values

The codes not listed here are reserved.

6.9.2.2.2 Command Specifier

This field contains module control command specifier. It is used to control the NMT state machine in NMT slaves.

Coding:

Code	Description
0	Do not change the state
1	Start Remote Node
2	Stop Remote Node
128	Enter Pre-Operational
129	Reset Node
130	Reset Communication

The codes not listed here are reserved.

6.9.2.2.3 Application Cycle Start Cycle Count

If the command specifier is 1 (Start Remote Node) this field contains the bus cycle count value at which the application cycle starts with application cycle count from value zero (0). Otherwise this field shall be cleared to zero.

6.9.2.2.4 Data

The data field has a fixed length of 14 quadlets. These bytes shall be cleared to zero.

6.9.2.3 Module Status Data APDU

The module status data APDU (MSD APDU) is transmitted from an NMT slave to the NMT master. It is embedded into the data block of an asynchronous read or write request packet. It is transferred after boot-up and for error control purposes (node guarding).

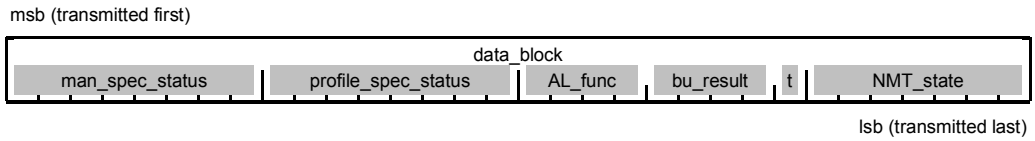

Figure 6.11 – Module Status Data APDU Structure

Table 6.6 – Module Status Data APDU Description

Abbreviation	Description
man_spec_status	Manufacturer Specific Status code
profile_spec_status	Communication or Device Profile Specific Status code
AL_func	Application Layer Functionality code
bu_result	Boot Up Result code
t	Toggle bit
NMT_state	NMT State code

The coding of the APDU fields is in accordance with the 1394AP Status register.

6.9.2.4 Module Status Data APDU Fields

The Module Status Data APDU carries the 1394AP status register. That is why the coding of the APDU fields is as described in section 7.7.2.6.

6.9.2.4.1 Manufacturer Specific Status Code

This field contains a manufacturer specific status stored in the 1394AP status register.

6.9.2.4.2 Communication or Device Profile Specific Status Code

This field contains communication or device profile specific status stored in the 1394AP status register.

6.9.2.4.3 Application Layer Functionality Code

This field contains the current application functionality stored in the 1394AP status register.

6.9.2.4.4 Boot Up Result Code

This field contains the result after entering the state PRE-OPERATIONAL. It is stored in the 1394AP status register.

6.9.2.4.5 Toggle Bit

The value of this field is toggle after each transmission. It is stored in the 1394AP status register.

6.9.2.4.6 NMT State Code

This field contains the current state of the NMT state machine. It is stored in the 1394AP status register.

6.9.3 Boot Up Indication APDU

A boot up indication is transmitted in the form of a Boot Up Indication APDU. It is embedded in the data block of a CSR data lock packet. The structure of a BU APDU is depicted in Figure 6.12.

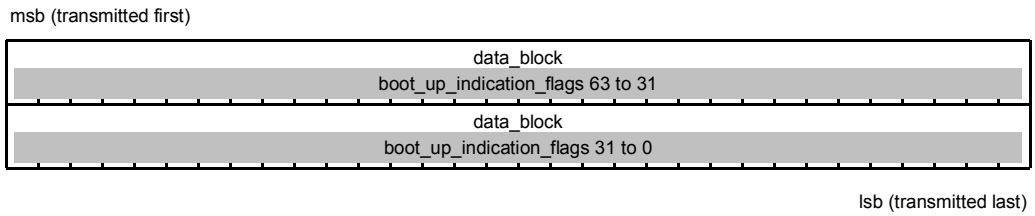

Figure 6.12´- Boot Up Indication APDU Structure

6.9.4 Boot Up Indication APDU Fields

The boot up indication APDU carries the boot up indication flag register. It contains 64 flags. The flag at the lowest bit position represents the node with the AP device ID zero (0), the flag with the highest bit position represents the node with the AP device ID sixty three (63).

6.10 Network Management Protocol

6.10.1 Module State Machine

In Figure 6.13 the state diagram of a device is shown. Devices enter the PRE-OPERATIONAL state directly after finishing the device initialization. During this state, device parameterization via SD services (e.g. using a configuration tool) is possible. The devices can then be switched into the state OPERATIONAL.

The NMT state machine determines the behavior of the communication function unit. The coupling of the application state machine to the NMT state machine is device dependent and falls into the scope of device profiles.

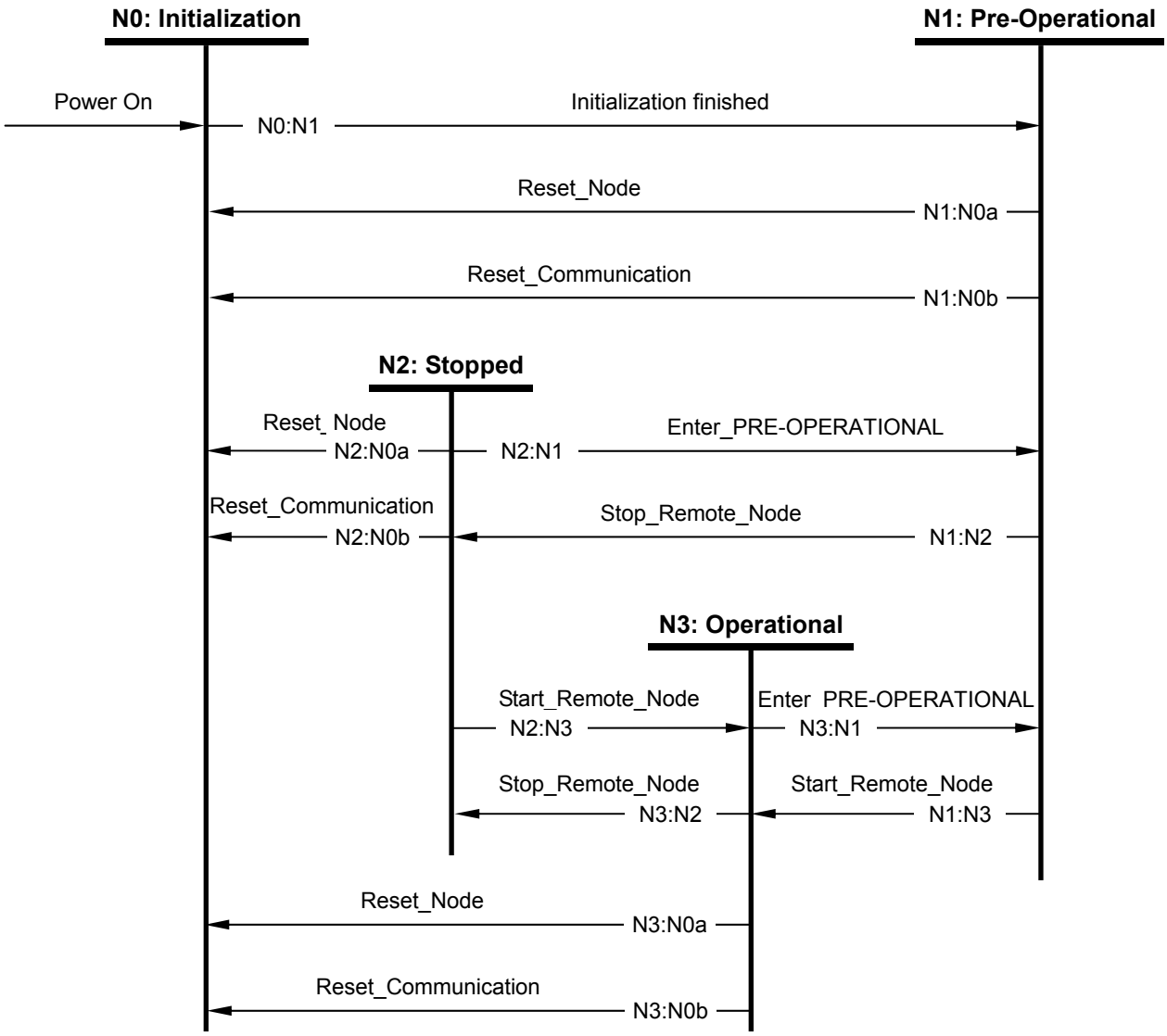

Figure 6.13 – State Diagram of a Device

6.10.1.1 Initialization

The INITIALISATION state is divided into three sub-states (see Figure 6.14) in order to enable a complete or partial reset of a device.

1. **INITIALISING**: This is the first sub-state the device enters after power-on or hardware reset. The definition of this procedure does not fall into the scope of this specification. After finishing the basic node initialisation the device enters autonomously into the state RESET-APPLICATION.

1. **RESET-APPLICATION**: In this state the parameters of the manufacturer specific profile area and of the standardised device profile area are set to their power-on values. After setting the power-on values the state RESET-COMMUNICATION is entered autonomously.

2. **RESET-COMMUNICATION**: This sub-state covers the IEEE 1394 bus configuration procedure including channel allocation and bus bandwidth allocation. The parameters are set to their power-on values. In the result

of the configuration, the configuration result is set. The application master is then determined using the following procedure:

a. The Bus Manager checks if its AP_MASTER_ID register has value 0x3F.

b. If yes, BUS MANAGER collects from all nodes the APMC bit and the AP_DEVICE_ID.

c. The APMC node with the highest node-ID (i.e. the node closest to rrot) becomes AP_MASTER.

d. The AP_DEVICE_ID is stored in the AP_MASTER_ID register.

e. For every BUS_MANAGER capable node the "abdicate" bit in STATE_CLEAR is set and a bus reset is issued.

The application master collects the minimum application cycle values of the devices using asynchronous read transactions. Afterwards the basic application cycle and the device application cycles are calculated. The application cycles are delivered to the devices using asynchronous write transactions. If a device accepts the received application cycle it updates the boot up indication register within the application master by an asynchronous lock request. After this the state INITIALISATION is finished and the device enters the state PRE-OPERATIONAL. If an error occurred the node enters the state PRE-OPERATIONAL without changing the boot up flag. The state RESET-COMMUNICATION is entered if the associated NMT service is called or when a bus reset is occurred.

Power-on values are the last stored parameters. If storing is not supported or has not been executed or if the reset was preceded by a restore_default command, the power-on values are the default values according to the communication and device profile specifications.

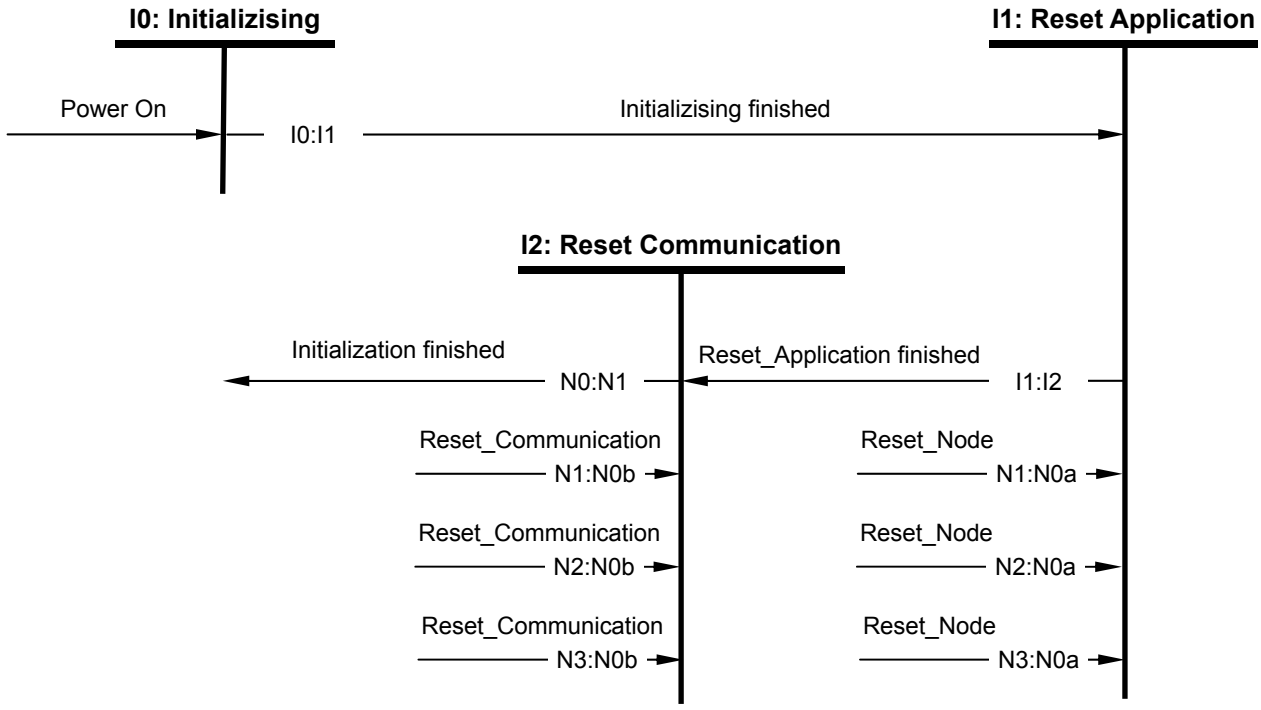

Figure 6.14 – State Diagram Fragment for Initialization

6.10.1.2 Pre-Operational

In the state PRE-OPERATIONAL, communication using CSR data services and service data services is possible. The application master may read at first the boot up result. Then the AP device ID list is delivered to the devices. The configuration of the CSR may be performed by a configuration application.

The device may be switched into the operational state by sending the Start Remote Node request. With this command the application master delivers the bus cycle count at which the application cycle starts with zero (0). The device enters the state OPERATIONAL when the given bus cycle count value was recognized.

6.10.1.3 Operational

In the state OPERATIONAL all communication services are active. The PD transfer starts with the first master data telegram. CSR access is possible. Implementation aspects or the application state machine however may require to limit the access to certain parts whilst in operation, e.g. a data area may contain the application program which cannot be changed during execution.

6.10.1.4 Stopped

By switching a device into the state STOPPED it is forced to stop the application layer functionality (except node guarding, if active). Furthermore, this state can be used to achieve certain application behaviour. The definition of this behaviour falls into the scope of device profiles.

6.10.1.5 States and Communication Services Relation

Table 6.7 shows the relation between communication states and communication services. Listed communication services may only be executed if the devices involved in the communication are in the appropriate communication states.

Table 6.7 – States and Communication Objects

	INITIALISING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
PD Services			X	
SD Services		X	X	
CSR Services	X	X	X	
Emergency Services		X	X	
Boot-Up Service	X			
Network Management Services		X	X	X

6.10.2 Error Control

6.10.2.1 Node Guarding

The guarding is achieved through transmitting guarding requests (Node guarding protocol) by the NMT Master. This is done by a CSR data read request of the 1304AP Status register. If a NMT Slave has not responded within a

defined span of time (node life time) or if the NMT Slave's communication status has been changed, the NMT master informs its NMT master application about that event.

If life guarding (NMT slave guarded NMT master) is supported, the slave uses the guard time and life-time factor to determine the node life time. If the NMT slave is not guarded within its life time, the NMT slave informs its local application about that event. If guard time and life time factor are 0 (default values), the NMT slave does not guard the NMT master.

Guarding starts for the slave when the first guarding request is received. This may be during the boot-up phase or later.

7 CSR Architecture

7.1 Overview

This chapter describes the Control and Status Register (CSR) architecture of the IEEE 1394 Application Layer for Industrial Automation (1394AP). This architecture is based on the CSR architecture specified in IEEE 1394. This specification does not restrict the CSR architecture specified in IEEE 1394 nor is any contents defined.

The overall structure of the 1394AP CSR is depicted in Figure 7.1. In the following chapters the entries of the Configuration ROM and of the register area are explained, which are specific for 1394AP.

CSR

Core CSR Registers

...

Serial Bus Dependent Registers (IEEE 1394)

...

Configuration ROM

Bus Info Block

Node Vendor ID value

...

Root Directory

Module Vendor ID value

Node Capabilities value

Node Unique ID offset

Unit Directory offset

Unit Spec ID value

Unit SW Version value

Unit Location of the Application Layer directory offset

Unit Location of the Communication Profile directory offset

Unit Location of the Device Profile directory offset

Unit Location of the Manufacturer Specific directory offset

Figure 7.1 – 1394AP CSR Structure

7.2 Bus Info Block

A device according to this specification shall implement the Bus Info Block as defined in IEEE 1394a.

7.3 Root Directory

A device according to this specification shall implement the Node_Unique_ID.

7.4 Unit Directory

7.4.1 Overview

The Unit_Directory_Offset of the IEEE 1394 CSR architecture points to the start address of the 1394AP Unit Directory. The structure of the 1394AP Unit Directory is depicted in Figure 7.2. The grey fields are mandatory.

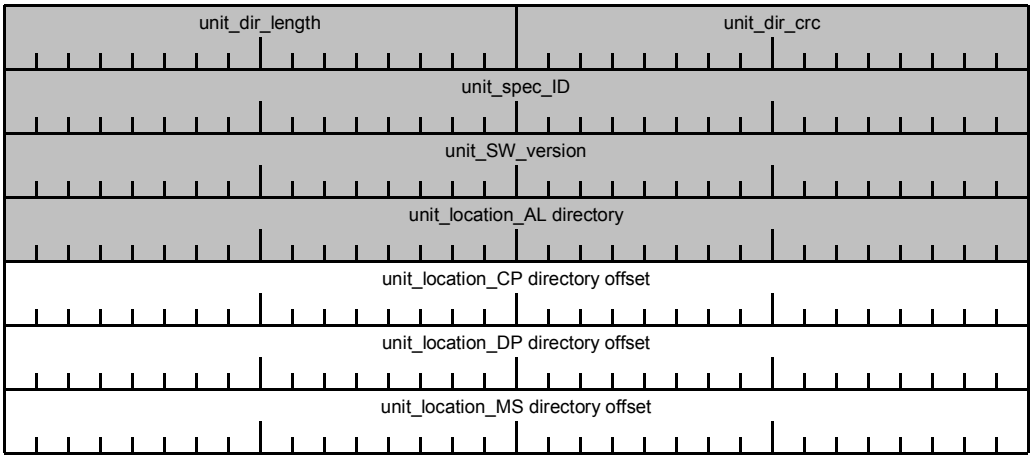

Figure 7.2 – Structure of 1394AP Unit Directory

The unit directory entries are explained in Table 7.1. The following sections contain the detailed specification.

Table 7.1 – 1394AP Unit Directory Entries

Abbreviation	Description	M/O
unit_dir_length	Unit Directory Length value	M
unit_dir_crc	Unit Directory CRC value	M
unit_spec_ID	Unit Specification Identifier value for IEEE 1394 Application Layer for Industrial Automation (1394AP)	M
unit_SW_version	Unit Software Version value	M
unit_location_AL	Unit Location of the application layer directory	M
unit_location_CP	Unit Location of the Communication Profile directory	O
unit_location_DP	Unit Location of the Device Profile directory	O
unit_location_MS	Unit Location of the Manufacturer Specific directory	O

7.4.2 Unit Directory Length

The data type of the unit directory length is UNSIGNED16. It contains the number of quadlets of the unit directory.

7.4.3 Unit Directory CRC

The data type of the unit directory CRC is UNSIGNED16. It contains the CRC value calculated over the unit directory. The CRC polynomial is defined in CSR specification.

7.4.4 Unit Specification ID

The data type of the unit specification identifier is UNSIGNED32 with the following structure:

Figure 7.3 – Structure of Unit Specification ID

Table 7.2 – Unit Specification ID Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 0h	M
k_value	key value according to ISO/IEC 13213 = 12h	M
unit_spec_ID_value	Identifier of IEEE 1394 Application Layer specification for Industrial Automation (1394AP) = 00A02Dh	M

7.4.5 Unit Software Version

The data type of the unit software version is UNSIGNED32 with the following structure:

Figure 7.4 – Structure of Unit Software Version

Table 7.3 – Unit Software Version Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 0h	M
k_value	key value according to ISO/IEC 13213 = 13h	M
unit_SW_version_value	Version of the IEEE 1394 Application Layer specification for Industrial Automation (1394AP) = 00.03:00h	M

7.4.6 Unit Location of the Application Layer Directory

The data type of the unit location of the application layer directory is UNSIGNED32 with the following structure:

Figure 7.5 – Structure of the Unit Location of the Application Layer Directory

Table 7.4 – Unit Location of the Application Layer Directory Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 3	M
k_value	key value according to ISO/IEC 13213 = 15h	M
unit_location_AL_offset	Offset to the directory of the IEEE 1394 Application Layer for Industrial Automation (1394AP)	M

The application layer directory is described in chapter 7.5.

7.4.7 Unit Location of the Communication Profile Directory

The data type of the unit location of the communication profile directory is UNSIGNED32 with the following structure:

Figure 7.6 – Structure of the Unit Location of the Communication Profile Directory

Table 7.5 – Unit Location of the Communication Profile Directory Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 3h	M
k_value	key value according to ISO/IEC 13213 = 15h	M
unit_location_CP_offset	Offset to a communication profile directory	M

The definition of a communication profile directory is not part of this specification.

7.4.8 Unit Location of the Device Profile Directory

The data type of the unit location of the device profile directory is UNSIGNED32 with the following structure:

Figure 7.7 – Structure of the Unit Location of the Device Profile Directory

Table 7.6 – Unit Location of the Device Profile Directory Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 3h	M
k_value	key value according to ISO/IEC 13213 = 15h	M
unit_location_DP_offset	Offset to a device profile directory	M

The definition of a device profile directory is not part of this specification.

7.4.9 Unit Location of the Manufacturer Specific Directory

The data type of the unit location of the manufacturer specific directory is UNSIGNED32 with the following structure:

Figure 7.8 – Structure of the Unit Location of the Manufacturer Specific Directory

Table 7.7 – Unit Location of the Manufacturer Specific Directory Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 3h	M
k_value	key value according to ISO/IEC 13213 = 15h	M
unit_location_MS_offset	Offset to a manufacturer specific directory	M

The definition of a manufacturer specific directory is not part of this specification.

7.5 Application Layer Directory

7.5.1 Overview

The directory of the IEEE 1394 Application Layer for Industrial Automation (1394AP) contains the values and references which are specific for the application layer. The structure of the directory is depicted in Figure 7.9.

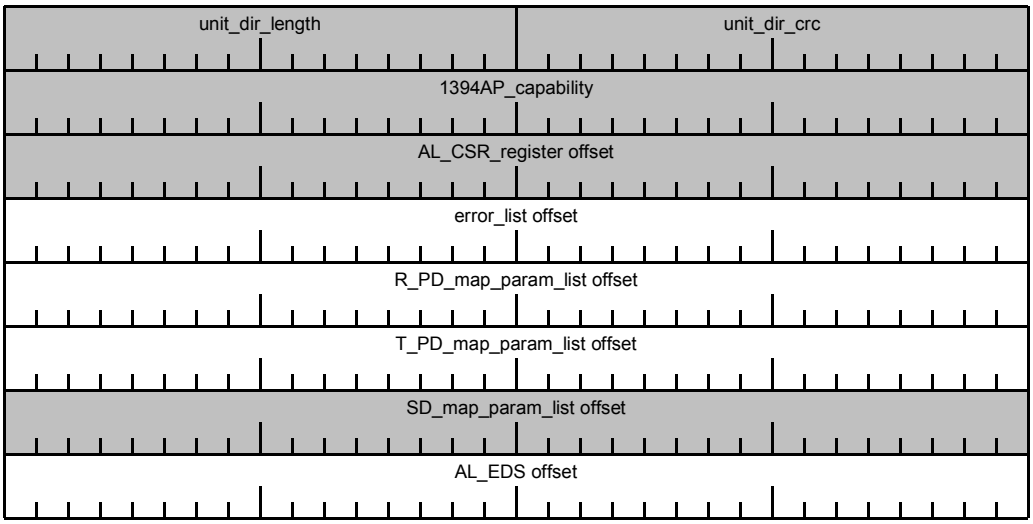

Figure 7.9 – Structure of the Application Layer Directory

The unit directory entries are explained in Table 7.8.The following sections contain the detailed specification.

Table 7.8 – Application Layer Directory Entries

Abbreviation	Description	M/O
unit_dir_length	Unit Directory Length value	M
unit_dir_crc	Unit Directory CRC value	M
1394AP_capability	1394AP capability value of a device which implements the IEEE 1394 Application Layer for Industrial Automation (1394AP)	M
AL_CSR	Application Layer CSR offset	M
error_list	Error List offset	O
R_PD_map_param	Receive Process Data Mapping Parameter List offset	O
T_PD_map_param	Transmit Process Data Mapping Parameter List offset	O
SD_map_param	Service Data Mapping Parameter List offset	M
AL_EDS	Application Layer Electronic Data Sheet List offset	O

7.5.2 Unit Directory Length

The data type of the unit directory length is UNSIGNED16. It contains the number of quadlets of the unit directory.

7.5.3 Unit Directory CRC

The data type of the unit directory CRC is UNSIGNED16. It contains the CRC value calculated over the unit directory.

7.5.4 1394AP Capability Register

The data type of the 1394AP capability register is UNSIGNED32 with the following structure:

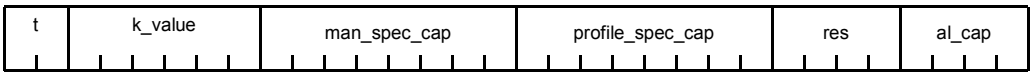

Figure 7.10 – Structure of 1394AP Capability Register

Table 7.9 – 1394AP Capability Register Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 0h	M
k_value	key value according to ISO/IEC 13213 = 14h	M
man_spec_cap	manufacturer specific capability flags	O
profile_spec_cap	communication or device profile specific capability flags	O
res	reserved (always 0)	M
al_cap	application layer capability code (see Table 7.10)	M

Table 7.10 – Structure of the Application Layer Capability Code

Bit	Abbreviation	Description
0	cstc	clock synchronised process data transmission capable
1	evtc	event triggered process data transmission capable
2	astc	asynchronous process data transmission capable
3	amc	application master capable

The node supports a capability if the associated bit is set to 1.

7.5.5 Application Layer CSR Register Offset

The data type of the application layer CSR register offset is UNSIGNED32 with the following structure:

Figure 7.11 – -Application Layer CSR Offset

Table 7.11 – Application Layer CSR Offset Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 1	M
k_value	key value according to ISO/IEC 13213 = 30h	M
AL_CSR_offset	Offset to the common register area of the IEEE 1394 Application Layer for Industrial Automation (1394AP)	M

The application layer register area is described in chapter 7.7.2.

7.5.6 Error List Offset

The data type of the error list offset is UNSIGNED32 with the following structure:

Figure 7.12 – Structure of the Error List Offset

Table 7.12 – Error List Offset Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 1h	M
k_value	key value according to ISO/IEC 13213 = 31h	M
error_register_offset	Offset to the error list in the register area	M

The error register list is described in chapter 7.7.3.4.

7.5.7 Receive Process Data Mapping Parameter List Offset

The data type of the receive process data mapping parameter list offset is UNSIGNED32 with the following structure:

Figure 7.13 – Structure of the Receive Process Data Mapping Parameter List Offset

Table 7.13 – Receive Process Data Mapping Parameter List Offset Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC13213 = 1h	M
k_value	key value according to ISO/IEC13213 = 32h	M
R_PDT_parameter_offset	Offset to the receive process data mapping parameter list	M

The process data mapping parameter list is described in chapter 7.7.3.5.

7.5.8 Transmit Process Data Mapping Parameter List Offset

The data type of the transmit process data mapping parameter list offset is UNSIGNED32 with the following structure:

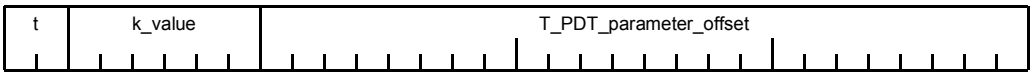

Figure 7.14 – Structure of the Transmit Process Data Mapping Parameter List Offset

Table 7.14 – Transmit Process Data Mapping Parameter List Offset Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC13213 = 1h	M
k_value	key value according to ISO/IEC13213 = 33h	M
T_PDT_parameter_offset	Offset to the transmit process data mapping parameter list	M

The process data mapping parameter list is described in chapter 7.7.3.6.

7.5.9 Service Data Mapping Parameter List Offset

The data type of the service data mapping parameter list offset is UNSIGNED32 with the following structure:

Figure 7.15 – Structure of the Service Data Mapping Parameter List Offset

Table 7.15 – Service Data Mapping Parameter List Offset Entries

Abbreviation	Description	M/O
T	key type according to ISO/IEC 13213 = 1h	M
k_value	key value according to ISO/IEC 13213 = 34h	M
SD_param_offset	Offset to the service data mapping parameter	M

The service data mapping parameter list is described in chapter 7.7.3.7.

7.5.10 Application Layer Electronic Data Sheet Offset

The data type of the application layer electronic data sheet offset is UNSIGNED32 with the following structure:

Figure 7.16 – Structure of the Application Layer Electronic Data Sheet Offset

Table 7.16 – Application Layer Electronic Data Sheet Offset Entries

Abbreviation	Description	M/O
t	key type according to ISO/IEC 13213 = 1h	M
k_value	key value according to ISO/IEC 13213 = 35h	M
AL_EDS_offset	Offset to the application layer electronic data sheet	M

The application layer electronic data sheet is still to be defined.

7.6 Core CSR Registers

A device according to this specification shall implement the Message_Request and Message_Response Registers. They are used for network management messages.

7.7 Application Layer CSR

The control and status register of the IEEE 1394 Application Layer for Industrial Automation (1394AP) are stored in a special area of the ASR. The register contents is defined using tables. The meaning of the columns is explained in Table 7.17.

Table 7.17 – Register Definition

Column	Description
Abbreviation	abbreviation of a control or status variable stored in a register or in part of a register
Description	description of the control or status variable
Acc	definition of the access rights of the variable via the network, the access rights can be read only (ro), read write (rw) or write only (wo)
M/O	definition if the variable is optional (O) or mandatory (M)

7.7.1 Overview

The CSR area of the IEEE 1394 Application Layer for Industrial Automation (1394AP) contains the registers which are specific for the application layer. Table 7.18 lists the entries of the application layer CSR register area.

Table 7.18 – Application Layer CSR Entries

Abbreviation	Description	Acc.	M/O
AP_device_ID	Automation Profile Device Identifier Register	ro	M
AP_master_ID	AP_device_ID of Application Master (only valid on Bus Manager)	ro	M
emergency_indication	Emergency Indication Flag Register	wo	O
error_register	Error Register	ro	M
boot_up_indication	Boot-up Indication Flag Register	wo	M
1394AP_status	1394AP Status Register	ro	M
1394AP_control	1394AP Control Register	wo	M
alloc_iso_channel	Allocated Isochronous Channel Register	ro	M
alloc_bandwidth	Allocated Bandwidth Register	ro	O
min_appl_cycle	Minimum Application Cycle Register	ro	O
basic_appl_cycle	Basic Application Cycle Register	rw	O
dev_appl_cycle	Device Application Cycle Register	rw	O
min_inp_trigger_time	Minimum Input Trigger Time of the device Register	ro	O
global_inp_trigger_time	Global Input Trigger Time of the device Register	rw	O
dev_inp_time	Device Input Time of the device Register	ro	O
dev_inp_time_res	Device Input Time Resolution of the device Register	ro	O
min_outp_trigger_time	Minimum Output Trigger Time of the device Register	ro	O
global_outp_trigger_time	Global Output Trigger Time of the device Register	rw	O
error_control_time	Error Control Time Register	rw	M
appl_cycle_list	Application Cycle List	rw	O
PD_loss_count_list	Process Data Loss Count List	ro	O
AP_device_ID_list	Automation Profile Device Identifier List	ro	M
error_list	Error list	ro	O
R_PD_map_param_list	Receive Process Data Mapping Parameter List	rw	O
T_PD_map_param_list	Transmit Process Data Mapping Parameter List	rw	O
SD_map_param_list	Service Data Mapping Parameter List	rw	M

All registers of the application layer CSR area can be accessed by using the CSR data services.

7.7.2 Application Layer CSR Registers

The structure of the application layer CSR registers is depicted in Figure 7.17. Reserved fields shall always contain zero (0) and shall be written with zero (0).

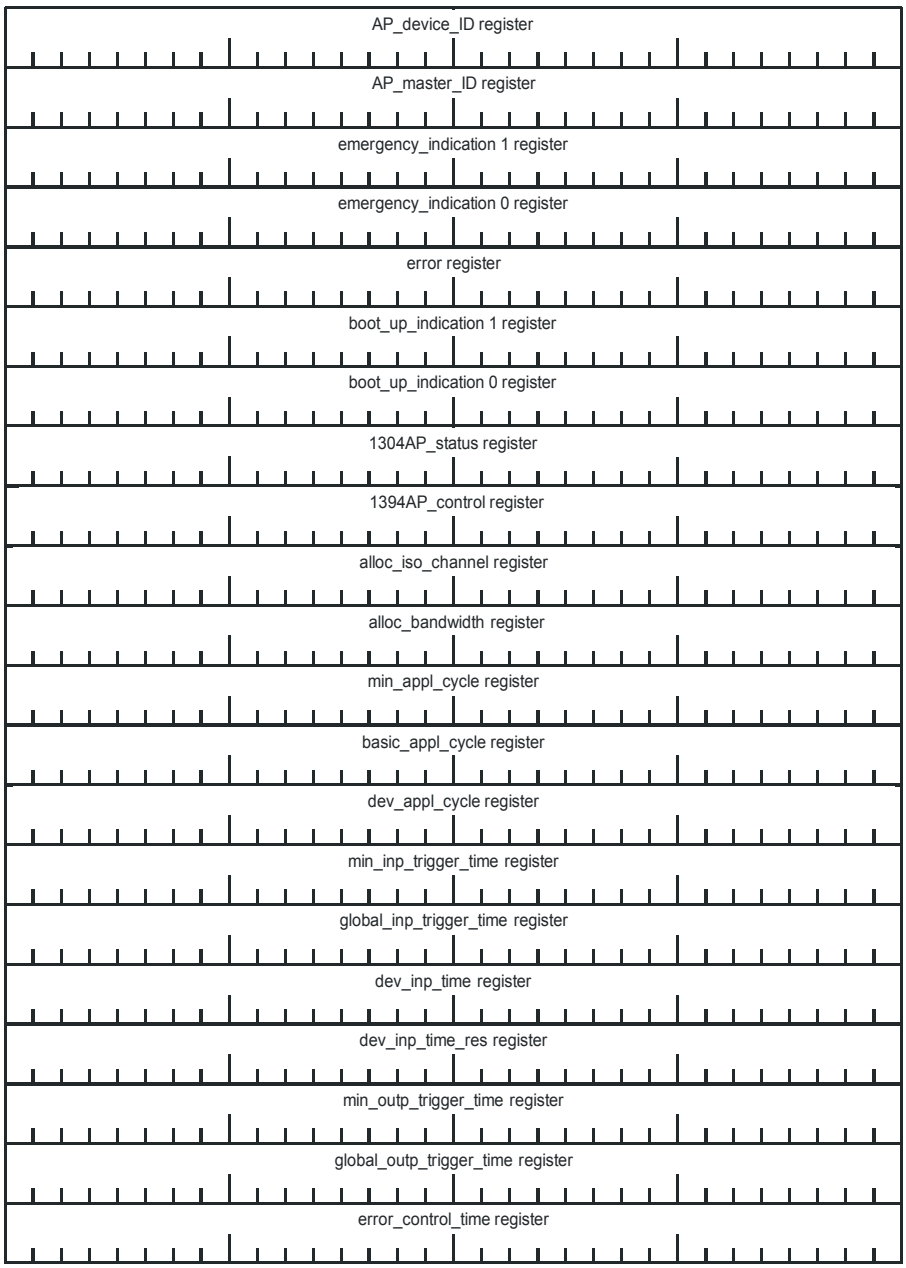

Figure 7.17 – Structure of the Application Layer CSR (Registers)

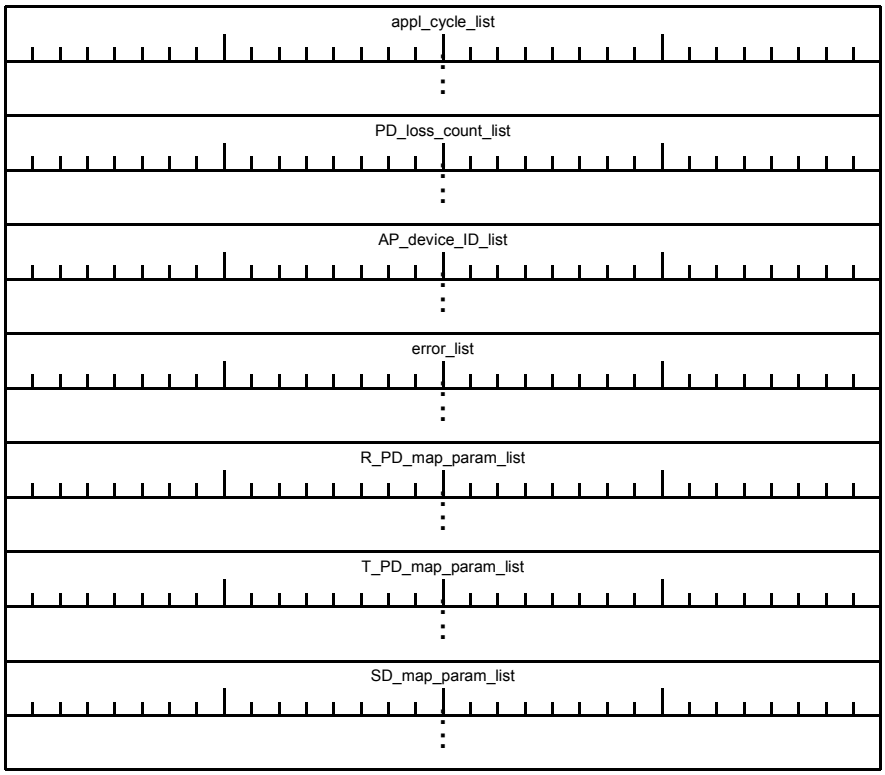

Figure 7.18 – Structure of the Application Layer CSR (Register Lists)

7.7.2.1 Automation Profile Device ID Register

The AP device ID register contains AP device ID. It identifies the place of a node within the automation application. The values of the AP device ID is specified in the planning phase of the automation application. The AP device ID is contained in the PD APDU.

The way the AP device ID is stored in the device is not a subject of this specification.

Figure 7.19 – Structure of the Automation Profile Device ID Register

Table 7.19 – Automation Profile Device ID Register Entries

Abbreviation	Description	Acc.	M/O
res	Reserved (always 0)	ro	M
AP_device_ID	Automation Profile Device Identifier	ro	M

7.7.2.2 Automation Profile Master ID Register

The AP master ID register contains AP device ID of the application master . The value of the AP master ID is determined during the RESET_COMMUNICATION state of the Network Management state machine. This register is only valid on the Bus Manager. Values of the Automation Profile Master ID Register of nodes which are not Bus Manager shall be ignored. Initial value of Automation Profile Master ID Register is 0x3F.

Figure 7.20 – Structure of the Automation Profile Master ID Register

Table 7.20 – Automation Profile Master ID Register Entries

Abbreviation	Description	Acc.	M/O
res	Reserved (always 0)	ro	M
AP_device_ID	Automation Profile Master Device Identifier	ro	M

7.7.2.3 Emergency Indication Flag Register

The emergency indication flag register contains one flag for each node which indicates the occurrence of an error within this node. The flag is set by the erroneous node and reset by the owner of the register.

The emergency indication register contains 64 flags. The flag at the lowest bit position represents the node with the AP device ID zero (0), the flag with the highest bit position represents the node with the AP device ID sixty three (63).

7.7.2.4 Error Register

The error register contains manufacturer specific and profile specific codes as well as error codes defined by this specification. The error register is contained in the emergency APDU.

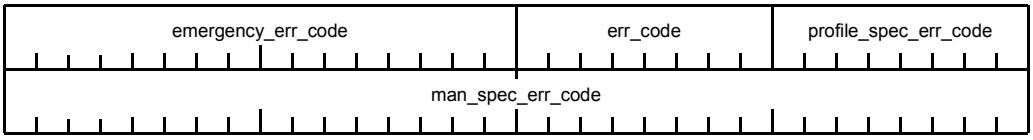

Figure 7.21 – Structure of the Error Register

Table 7.21 – Error Register Entries

Abbreviation	Description	Acc.	M/O
emergency_err_code	Emergency error code	ro	O
error_code	Error code (see Table 7.23)	ro	M
profile_spec_err_code	Communication or Device Profile specific error code	ro	O
man_spec_err_code	Manufacturer specific error code	ro	O

Table 7.22 – Emergency Error Code

Code	Description	M/O
00xxh	Error Reset or No Error	M
10xxh	Generic Error	M
20xxh	Current	O
21xxh	Current, device input side	O
22xxh	Current inside the device	O
23xxh	Current, device output side	O
30xxh	Voltage	O
31xxh	Mains Voltage	O
32xxh	Voltage inside the device	O
33xxh	Output Voltage	O
40xxh	Temperature	O
41xxh	Ambient Temperature	O
42xxh	Device Temperature	O
50xxh	Device Hardware	O
60xxh	Device Software	O
61xxh	Internal Software	O
62xxh	User Software	O
63xxh	Data Set	O
70xxh	Additional Modules	O
80xxh	Monitoring	O
81xxh	Communication	O
8130h	Life Guard Error	O
82xxh	Protocol Error	O
90xxh	External Error	O
F0xxh	Additional Functions	O
FFxxh	Device specific	O

Table 7.23 – Error Code

Bit	Description	Acc.	M/O
0	generic error	ro	M
1	current	ro	O
2	voltage	ro	O
3	temperature	ro	O
4	communication error (overrun, error state)	ro	O

If a bit is set to 1 the specified error has occurred. The only mandatory error that has to be signalled is the generic error. The generic error is signalled at any error situation.

7.7.2.5 Boot-up Indication Flag Register

The boot-up indication flag register contains one flag for each node which indicates the transition to the state PRE-OPERATIONAL. The flag is set by the node which transferred to the state PRE-OPERATIONAL and is reset by the owner of the register.

The boot-up flag register contains 64 flags. The flag at the lowest bit position represents the node with the AP device ID zero (0), the flag with the highest bit position represents the node with the AP device ID sixty three (63).

7.7.2.6 1394AP Status Register

The 1394AP status register contains manufacturer specific and profile specific status codes as well as status codes defined by this specification. The 1394AP status register is contained in the boot up, the node guarding and the heart beat APDU.

Figure 7.22 – Structure of the 1394AP Status Register

Table 7.24 – 1394AP Status Register Entries

Abbreviation	Description	Acc.	M/O
man_spec_status	Manufacturer Specific Status code	ro	O
profile_spec_status	Communication or Device Profile Specific Status code	ro	O
al_func	Application Layer Functionality code (see Table 7.25)	ro	M[4]
bu_result	Boot Up Result code (see Table 7.26)	ro	M
t	Toggle bit	ro	M
NMT_state	NMT State code (see Table 7.27)	ro	M

Table 7.25 – Structure of the Application Layer Functionality Code

Bit	Abbreviation	Description
0	cstf	clock synchronised process data transmission functionality
1	evtf	event triggered process data transmission functionality
2	astf	asynchronous process data transmission functionality
3	amf	application master functionality

[4] Mandatory if capable.

Table 7.26 – Configuration Result Code

Code	Status
0	successful
1	isochronous band width not available
2	no isochronous channel available
3	synchronisation impossible

Table 7.27 – NMT State Code

Code	State
0	INITIALIZATION
4	STOPPED
5	OPERATIONAL
127	PRE-OPERATIONAL

The codes not listed here are reserved.

7.7.2.7 1394AP Control Register

The 1394AP control register contains manufacturer specific and profile specific control codes as well as control codes defined by this specification. The 1394AP control register can only be written in the station state PRE-OPERATIONAL.

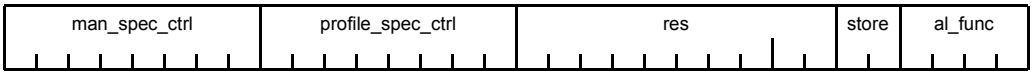

Figure 7.23 – Structure of the 1394AP Control Register

Table 7.28 – 1394AP Control Register Entries

Abbreviation	Description	Acc.	M/O
man_spec_ctrl	Manufacturer Specific Status code	wo	O
profile_spec_ctrl	Communication or Device Profile Specific Status code	wo	O
res	Reserved (always 0)	wo	M
store	Store code (see Table 7.29)	wo	O
al_func	Application Layer Functionality code (see Table 7.25)	wo	M[5]

[5] Mandatory if capable.

Table 7.29 – Structure of the Store Code

Bit	Description
0	store parameter
1	restore parameter

If bit 0 is set to 1 the node is requested to store the configuration parameter into a non-volatile memory area. If bit 1 is set to 1 the node is requested to restore the configuration parameter out of the non-volatile memory area.

If an application layer functionality bit is set to 1 the node is requested to activate the associated capability.

7.7.2.8 Allocated Isochronous Channel Register

The allocated isochronous channel register contains the number of the channel allocated by the node.

Figure 7.24 – Structure of the Allocated Isochronous Channel Register

Table 7.30 – Allocated Isochronous Channel Register Entries

Abbreviation	Description	Acc.	M/O
Res	Reserved (always 0)	ro	M
alloc_iso_channel	Allocated Isochronous Channel code (see Table 7.31)	ro	M

Table 7.31 – Allocated Isochronous Channel Code

Code	Description
0	The isochronous channel 0 shall be used by the application master.
>0 and ≤63	Isochronous channel of the device.
64	No isochronous channel allocated for process data transmission.

7.7.2.9 Allocated Bandwidth Register

The allocated bandwidth register contains the bandwidth allocated by the node. The coding of these registers corresponds to the coding of the Bandwidth_Available register in the serial bus dependent part of the core CSR register.

Figure 7.25 – Structure of the Allocated Bandwidth Register

Table 7.32 – Allocated Bandwidth Register Entries

Abbreviation	Description	Acc.	M/O
Res	Reserved (always 0)	ro	M
alloc_bandwidth	Allocated Bandwidth code	ro	M

7.7.2.10 Minimum Application Cycle Register

The minimum application cycle register contains the minimum number of bus cycles a device is able to support as an application cycle.

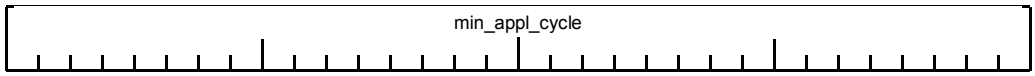

Figure 7.26 – Structure of the Minimum Application Cycle Register

7.7.2.11 Basic Application Cycle Register

The basic application cycle register contains the number of bus cycles between two consecutive process data transmissions of the master.

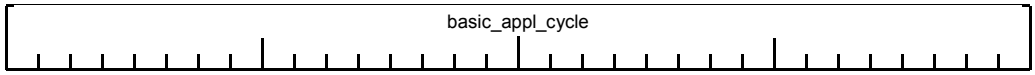

Figure 7.27 – Structure of the Basic Application Cycle Register

7.7.2.12 Device Application Cycle Register

The device application cycle register contains the number of basic application cycles between two consecutive process data transmissions of the given device.

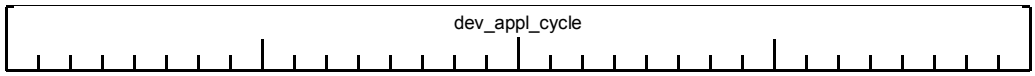

Figure 7.28 – Structure of the Device Application Cycle Register

7.7.2.13 Minimum Input Trigger Time Register

The minimum input trigger time register contains the period of time a device needs to be prepared for sampling the input data. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

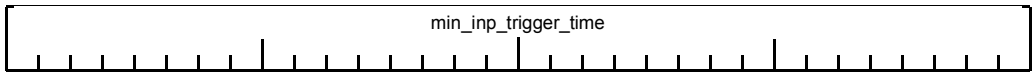

Figure 7.29 – Structure of the Minimum Input Trigger Time Register

7.7.2.14 Global Input Trigger Time Register

The global input trigger time registered contains a system wide time which represents the time period between the relevant cycle start event and the input sample instruction. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

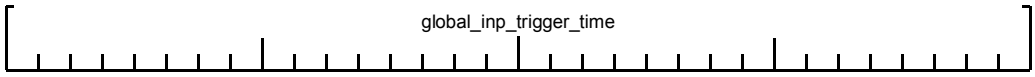

Figure 7.30 – Structure of the Global Input Trigger Time Register

7.7.2.15 Device Input Time Register

The device input time register contains the time a device needs to prepare input data for transmission. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

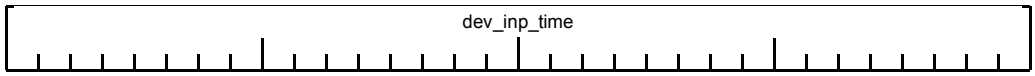

Figure 7.31 – Structure of the Device Input Time Register

7.7.2.16 Device Input Time Resolution Register

The device input time resolution register contains the minimum difference between two input time values that can be distinguished by the device. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

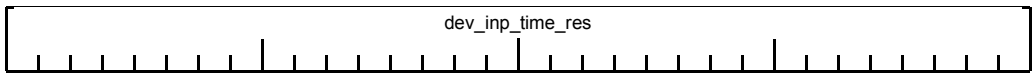

Figure 7.32 – Structure of the Device Input Time Resolution Register

7.7.2.17 Minimum Output Trigger Time Register

The minimum output trigger time register contains the period of time a device needs to read the received data and to output it. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

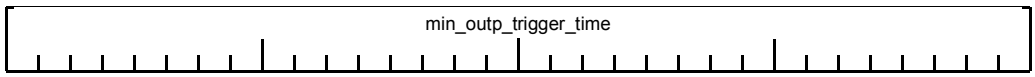

Figure 7.33 – Structure of the Minimum Output Trigger Time Register

7.7.2.18 Global Output Trigger Time Register

The global output trigger time register contains a system wide time which represents the time period between the relevant cycle start event and the data output instruction. This IEC 61131-3 type extension is a two's complement binary number with a length of four octets. The unit of time for this type is 1 µs.

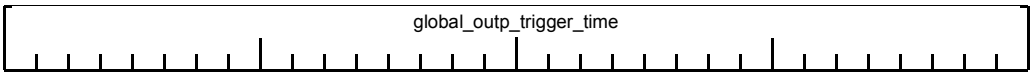

Figure 7.34 – Structure of the Global Output Trigger Time Register

7.7.2.19 Error Control Time Register

The error control time register contains the values of the guard time and the life-time factor.

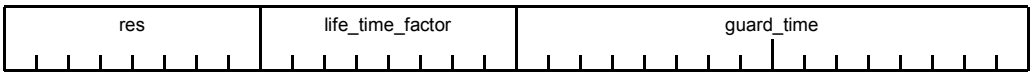

Figure 7.35 – Structure of the Error Control Time Register

Table 7.33 – Error Control Time Register Entries

Abbreviation	Description	Acc.	M/O
res	Reserved (always 0)	ro	M
life_time_factor	Life Time Factor	rw	M
guard_time	Guard Time	rw	M

7.7.3 Application Layer Register Lists

In the following chapters the structure of the application layer lists are defined.

7.7.3.1 Application Cycle List

The application cycle list is an ARRAY of UNSIGNED32 with the following structure:

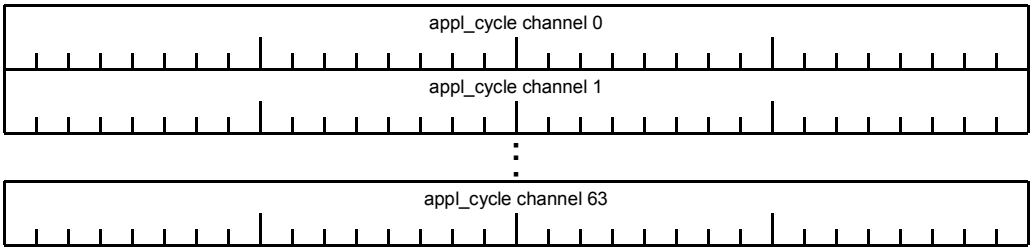

Figure 7.36 – Structure of the Application Cycle List

7.7.3.2 Process Data Loss Count List

The process data loss count list is an ARRAY of UNSIGNED32 with the following structure:

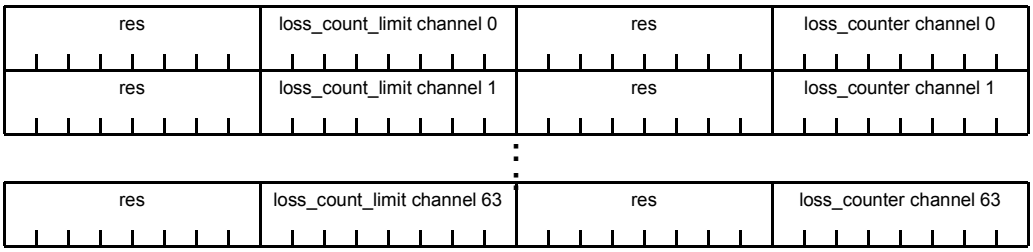

Figure 7.37 – Structure of the Process Data Loss Count List

7.7.3.3 Automation Profile Device Identifier List

The automation profile device identifier list is a STRUCT with the following structure:

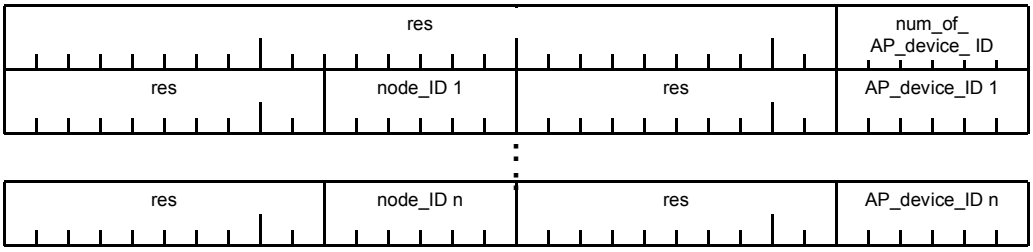

Figure 7.38 – Structure of the Automation Profile Identifier List

7.7.3.4 Error List

The error list is a STRUCT with the following structure. The current error register contents is contained in register number 1. The previous error register contents is contained in register t-1 and so on.

The handling of the error list is not a subject of this specification.

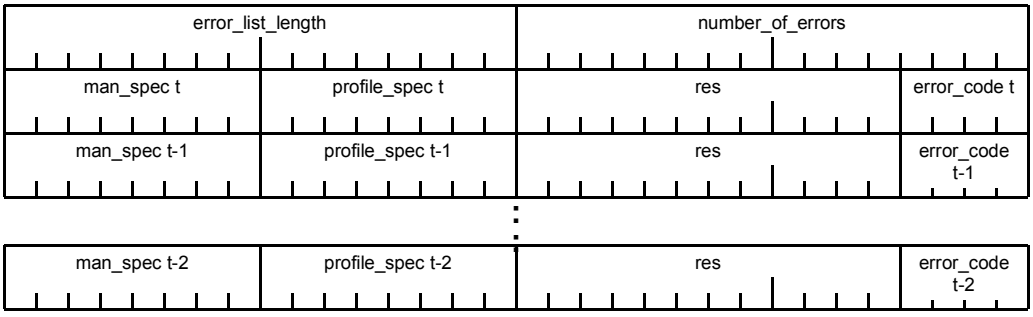

Figure 7.39 – Structure of the Error List

7.7.3.5 Receive Process Data Mapping Parameter List

The receive process data mapping parameter list is a STRUCT with the following structure:

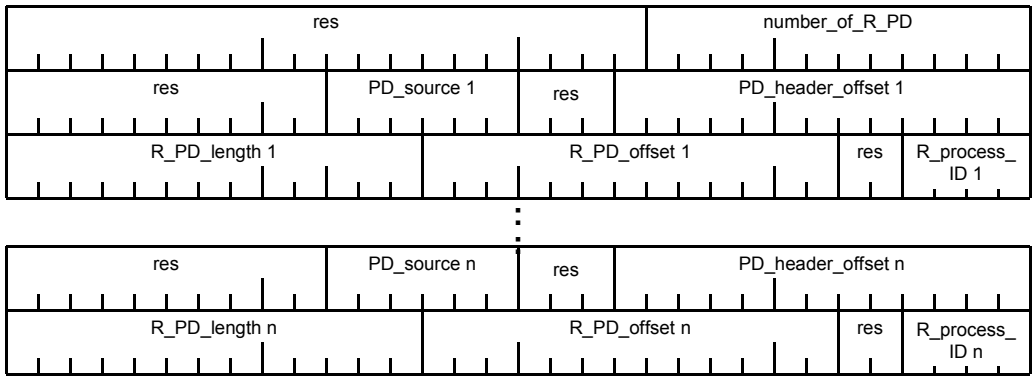

Figure 7.40 – Structure of the Receive Process Data Mapping Parameter List

7.7.3.6 Transmit Process Data Mapping Parameter List

The transmit process data mapping parameter list is a STRUCT with the following structure:

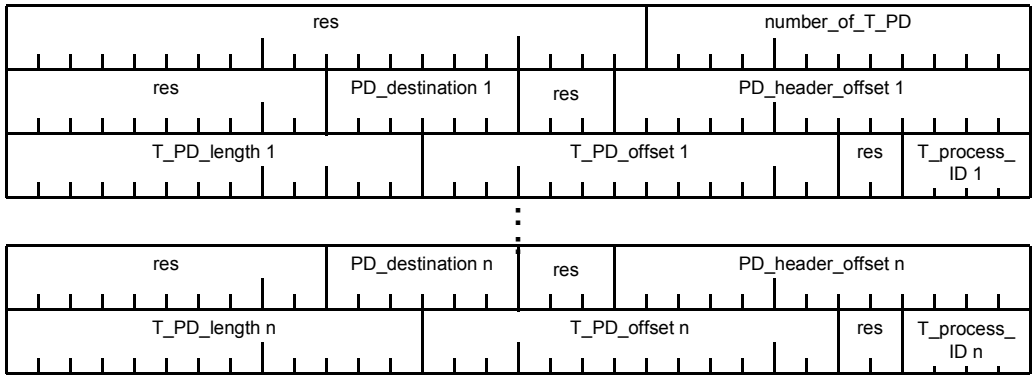

Figure 7.41 – Structure of the Transmit Process Data Mapping Parameter List

7.7.3.7 Service Data Mapping Parameter List

The service data mapping parameter list is a STRUCT with the following structure:

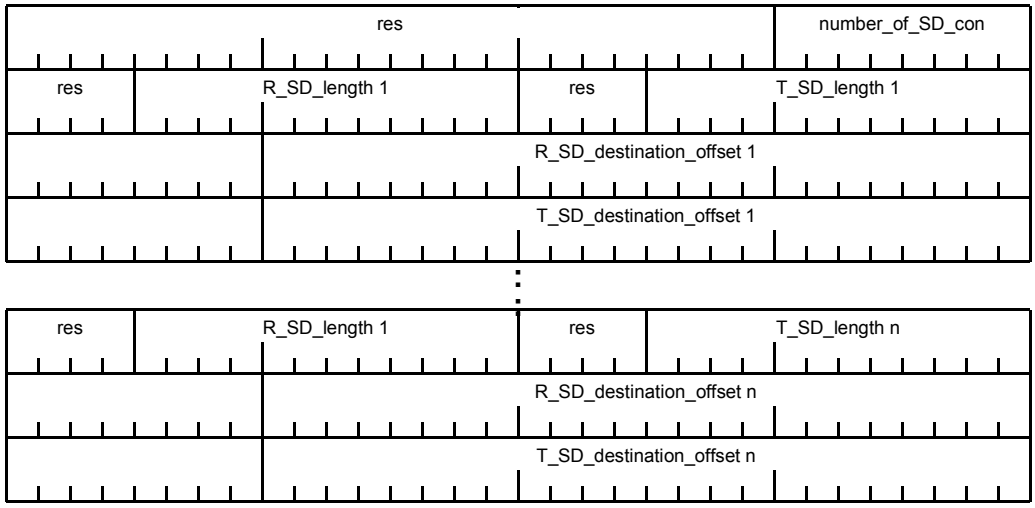

Figure 7.42 – Structure of the Service Data Mapping Parameter List

Annex A
(normative)

Compliance Annex

A device that supports this specification shall be IEEE1394a-2000 compliant.

A device that supports this specification and that claims to be Application Master Capable shall be IEEE1394 Bus Manager capable and shall indicate so by setting the BMC bit in the ConfigROM accordingly.

A device that supports this specification shall support all Process Data ASE attributes described in chapter 5.2.2 and marked as ‘Mandatory’.

A device that supports this specification shall support the Process Data Services described in chapter 5.2.3.

A device that supports this specification shall support the Process Data Protocol described in chapter 6.2.

A device that supports this specification shall support all Service Data Attributes described in chapter 5.3.2.

A device that supports this specification shall support all Service Data Services described in chapter 5.3.3.

A device that supports this specification shall support the Service Data Protocol as described in chapter 6.4.

A device that supports this specification shall support all Emergency Data Attributes described in chapter 5.5.2 and marked as ‘Mandatory’.

A device that supports this specification shall support all Emergency Data Services described in chapter 5.5.3.

A device that supports this specification shall support the Emergency Data Protocol described in chapter 6.8.

A device that supports this specification shall support the Network Management described in chapter 5.6, except for Node Guarding and Life Guarding, which are both optional.

A device that supports this specification shall support the Network Management Protocol described in chapter 6.10.

A device that supports this specification shall have a Configuration ROM Unit Directory according to chapter 7.4 providing all elements marked as ‘Mandatory’.

Annex B
(informative)

Features and Restrictions

The 1394AP envisages an isochronous and an asynchronous transmission of process data. The isochronous transmission is the intended method for process data.

In addition, nodes with restricted resources may use the asynchronous write service to provide process data. However, it is strongly recommended not to use broadcast transmission since non-1394AP devices could be influenced otherwise. Using the asynchronous write service directed sequentially to the consumers will worsen the time behaviour of the overall system significantly.

Annex C
(informative)

Implementation recommendations

This specification presupposes a cyclic start event without jitter for isochronous transactions. The cycle start event could be generated by the reception of the cycle start telegram, by a synchronised clock or by a phase locked loop.

Since there may be a time difference between the cycle start event and the reception of the master data telegram which contains the current application cycle count it could be necessary to "estimate" the current application count.

Process data may be transmitted using asynchronous write packets. Usually PD packets are broadcasted. However, this may cause problems, since memory areas may be overwritten in non-1394AP-devices, if any are in the network. The system integrator is responsible for avoiding this problem.

However, if this is not possible or if one of the processing units is overloaded due to the large number of broadcast messages, the destination may be addressed directly in the asynchronous write packet. This means one and the same process data is transmitted to different nodes. In this case the device has to provide the according number of entries in the transmit process data mapping parameter list.

The CSR data service may be used for "expedited" SD transmission. In this case communication and device profiles have to specify the mapping between destination offset and profile object index.

Annex D
(informative)

Coexistence Statement

The IEEE 1394 Application Layer for Industrial Automation has been specified with the intention to use conform device in parallel with devices which are not implemented in accordance to this specification. This means an application master should be able to communicate with 1394AP and non-1394AP devices at the same time and use the data to control the complete process.

In particular attention was paid not to use broadcast transmissions to certain destination offsets.

However it is up to the system planer to pre-estimate the bandwidth needs of the entire system.

Annex E
(informative)

Bibliography

[E1] Industrial Automation Protocol (IAP), Kommunikationsprotokoll für die industrielle Automation auf Basis der Norm IEEE 1394, Version 0.9.5, 04 June 2002

[E2] ISO/IEC 9899:1990, Programming Languages—C

[E3] CAN in Automation, CANopen Application Layer and Communication Profile, CiA Draft Standard DS-301, Version 4.02, 13 February 2002

[E4] CAN in Automation, CANopen Device Profile for Generic I/O Modules, CiA Draft Standard DS-401, Version 2.1, 17 May 2002

[E5] IEC 61131-3:1993, Programmable controllers — Part 3: Programming languages