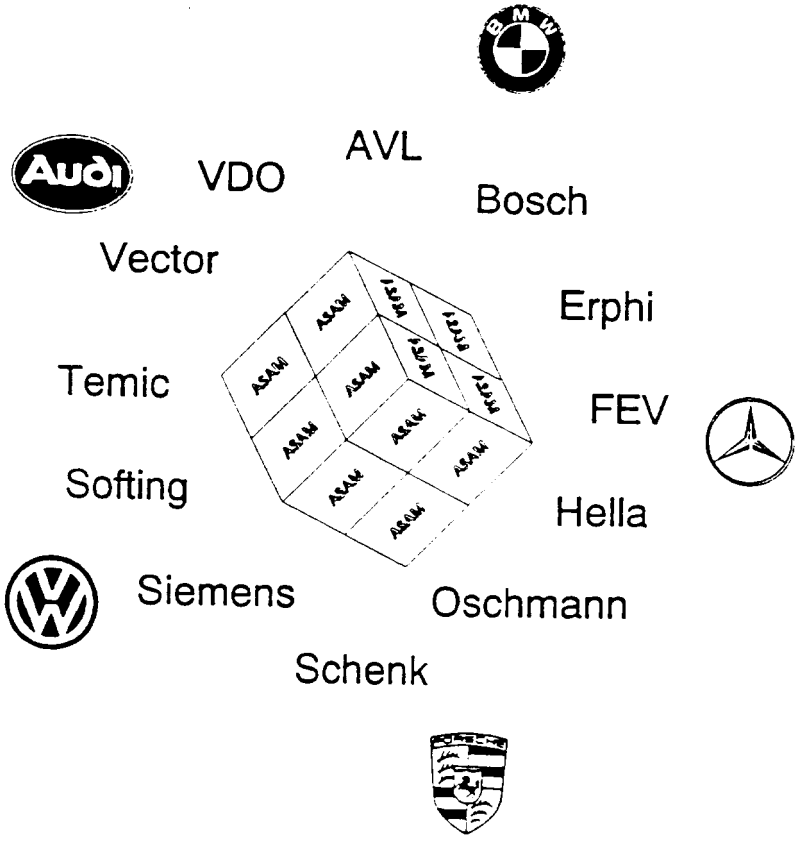


ASAP

Working Party on Standardisation of Application Systems

Interface Specification

Interface 1b

Version 1.0 dated 31.3.95

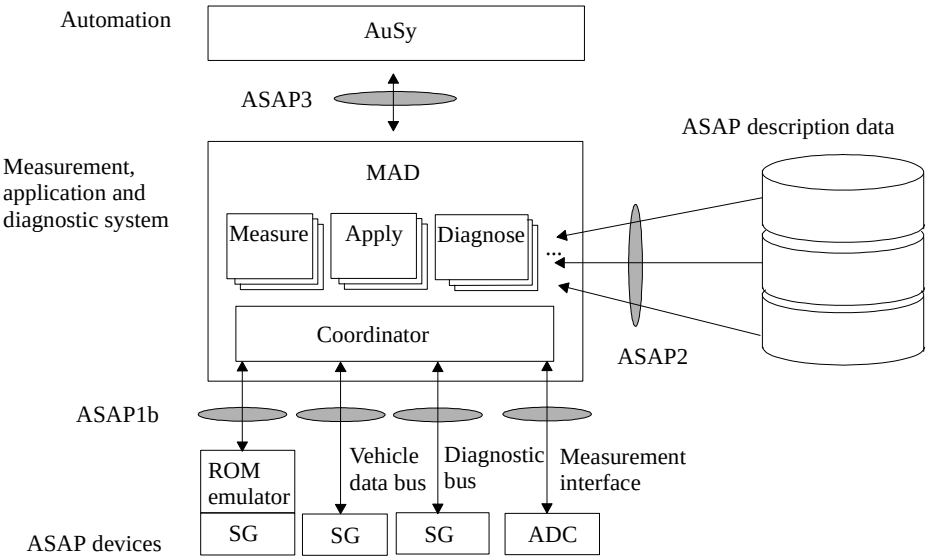
Contents

1. Summary

1.1 Scope of ASAP standardisation

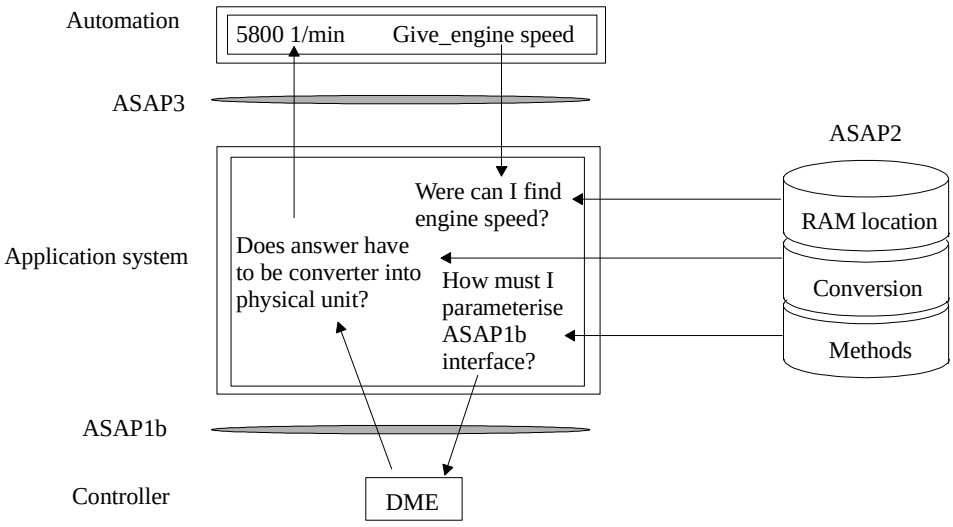
The scope of standardisation (Figur 1) covers three interfaces:

- ASAP3: between automation and application systems in the measurement, application and diagnostic system (MAD),
- ASAP2: the standardised description data, and
- ASAP1b: between the coordinator of the MAD and ASAP devices:



Figur 1 Scope of standardisation (schematic)

The allocation of tasks between subsystems and the interaction of all three ASAP interfaces is shown by an example (Figur 2).

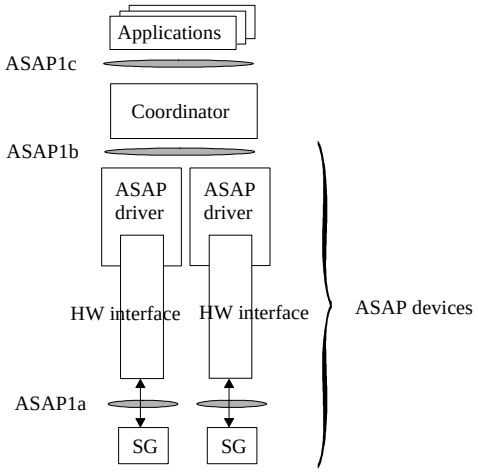


Figur 2 Example of interaction of ASAP interfaces

Automation requests the engine speed from the MAD application. Using interface 2, the application system communicates with the control device at hexadecimal level and provides conversion to a physical representation. The result is returned to Automation. As seen by Automation the various methods of accessing control devices as well as the different structures and functionalities in control devices are encapsulated by the application system and the results of requests are provided at a physical level.

1.2 ASAP1

ASAP1 connects individual applications to the control devices and control-device-related metrology. It is split up into three sub-interface levels (figur 3):



Figur 3 ASAP1 interfaces and ASAP devices

The various applications communicate with the coordinator layer via ASAP1c. This takes care of the merging of streams of information from ASAP devices and distributes these to the applications. ASAP1b is the software interface to ASAP devices via which information flows from control devices and metrology. ASAP1a represents a protocol interface within the ASAP device.

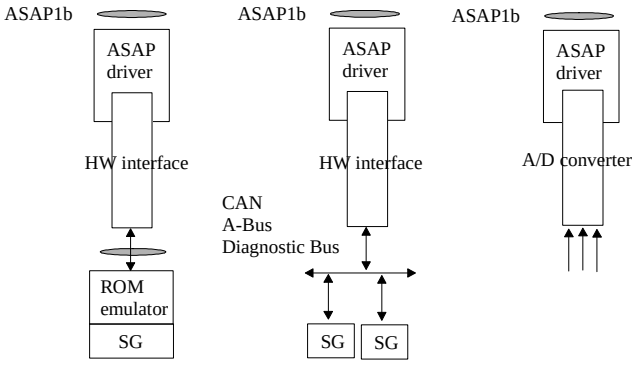
1.2.1 ASAP1a

ASAP1a contains the protocol definitions for the data transfer for application needs via the CAN, diagnostic and A-bus vehicle buses. This interface is currently not dealt with in the ASAP working party. It is a matter for bilateral negotiations between vendor and manufacturer.

1.2.2 ASAP1b

The entire subsystem below interface ASAP1b is designated a ASAP device and consists of the control device, an associated hardware interface and the ASAP driver (figur 3).

In turn, an ASAP device may, depending on the type of connection selected and the interface to the control device, have various distinctive features (figur 4). As an example, one might mention ROM emulators from different control-device manufacturers (DIM, MAC, INKA, DOSE, SAM2000, MCS, etc.), connection of
vehicle buses (CAN, A-Bus, diagnostic bus, etc.) and control-device-related metrology (A/D converter,. VME measuring system, etc.).



Figur 4 Examples of distinctive features of ASAP device

ASAP devices may differ markedly in terms of the intelligence implemented below ASAP1b. Acquired measurement data, depending on the design of hardware and software, may either be stored completely inside the ASAP device at the time of acquisition or be stored in the application system. Because stipulation in this respect cannot be regarded as meaningful, module types that differ in terms of their standalone measuring capability and the location of measurement-data storage, measurement-clock generation and trigger generation were defined for the acquisition of measurement data.

Building on this classification of ASAP devices into module types, eight ASAP services have been agreed as a further, essential characteristic of the ASAP1b interface. The services regulate the exchange of parameters and data via the interface:

Service	**Brief functional description**
INIT_READ	Initialise ASAP device for measuring system
INIT_ACCESS	Initialise ASAP device for adjusting system
SYNC	Synchronise several ASAP devices to each other by setting system clocks
READ	Data transfer from ASAP device to application system
ACCESS	Data transfer from application system to ASAP device
STOP	Terminate storage of measurement data started with SYNC in case of intelligent module types with own measurement data memory
FREE_HANDLE	Free references assigned at time of initialisation, terminate a measurement or adjustment cycle
GIVE_STATUS	Facility to poll information and states of ASAP device
RESOURCES	Request and allocate system resources for ASAP device

Table 1 ASAP services interface 1b

Semantics which regulate the passing of parameters and data have been defined for the individual services. Parameters and data are generally transferred between application and ASAP device through common memory structures organised as branched structures. The specification paid attention to ensuring that services were as independent as possible from the operating systems and programming languages used. Specialities and different communication methods within a ASAP device are also encapsulated

within that device. Any special parameters and binary objects needed to do this are obtained from interface 2. Once they have been transferred in blobs (binary large objects), it is then possible to interpret them in the ASAP device. A further result is uniform communication with the control device regardless on the connection type selected. Different capabilities of various control-device connections can be compensated within the ASAP driver. Functional communication, parameterisation, is regulated in a standard manner.

1.2.3 ASAP1c

The stipulations for this interface were deferred by the ASAP working party in favour of the 1b interface. It will be specified at a later date.

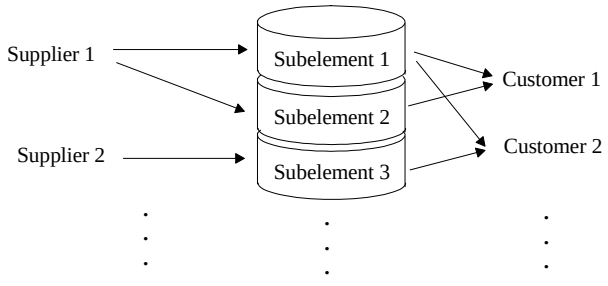
1.3 ASAP2

The ASAP description file (ASAP2) is used as a knowledge base for the application system. It contains, in its subelements, information arranged according to topic concerning the connected control device and its coupling to ASAP interface 1b; there are the following subelements (Table 2):

Subelement	Examples
Project-related data	Name of person in charge at component supplier, data reference number
Structures inside SG	Location and structure of warm-up characteristic map, contents of user information field
Conversion specifications	Scaling values for conversion from hexadecimal to physical for sources, structures and interfaces
Sources	Addresses, resolutions and update rates of measurable RAM locations
SW interfaces	Description of communication contents on vehicle buses, e.g. CAN identifier and associated data contents
Methods	Parameterisation of ASAP device for setting up communication with SG, e.g. special hex code for emulator

Table 2 Subelements of ASAP interface 2

The basic concept of ASAP2 can be understood as a format for the exchange of control-device descriptions. Its individual subelements may differ in terms of content and depend on their component supplier as well as on the relevant customer and his application system (Figur 5). Nevertheless, the methodology for storing information is standardised in order to allow uniform access by means of the individual components. This was achieved by using a key technique, 7-bit ASCII representation and nested elements for modular construction:



Figur 5 Customers and suppliers for ASAP2 subelements

The control-device manufacturer supplies the subelement of the standardised description file and is responsible for its functional equivalence in the control device. It is possible, for instance, to
replace subelement 7 (SW interfaces, CAN section) as well as subelement 1 (project-related data) in the event of functional expansion of the CAN interface. This gives an updated description that corresponds to the appropriate development status of the control device.

1.4 ASAP3

This interface is used to connect application systems to an automation system (AuSy) on a strict master/slave principle. Application systems are therefore seen by the AuSy as an intelligent device, e.g. an indicating device or a fuel weighing device.

The standard achieved represents the merging and extension of different existing implementations by the firms AVL, Bosch, Kleinknecht & Nirschl, Schenk, Siemens and VW/Gedas. A total of 28 commands were defined; these are transmitted in serial form and are divided up into the following topic areas (Table 3):

Contents	**Brief description**
Initialisation, identification, emergency	Establishment of communication, verification of version, cancellation of measures
Configuration	Selection and renaming of ASAP2 files and binary files
Characteristic map manipulation	Transfer, selection and modification of data elements (e.g. characteristic maps) in the control device
Parameter manipulation	Transfer and selection of sets of parameters in the application system
Acquisition of measured data	Initialisation of acquisition of measured values in MAD and transfer of results for automation
Recorder	Transfer, selection and starting of measured-value recording, transfer of results
Miscellaneous	Basic settings of application system transmitted for automation

Table 3 ASAP3 commands

Using ASAP3 commands, it is possible, for example, to obtain automatic alteration of characteristic maps with associated measured-value recording.

2. ASAP1b detailed specification

2.1 General stipulations

2.1.1 Module types (MT) of ASAP device

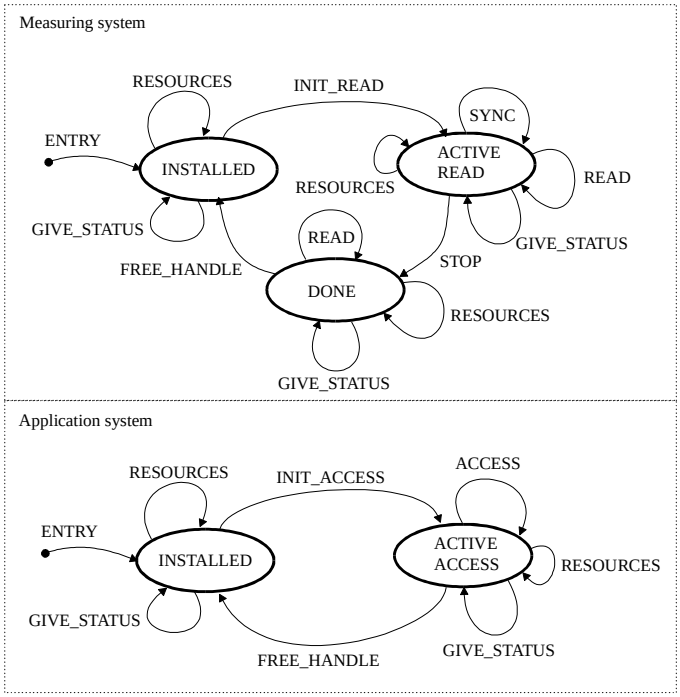
The module type is used to classify the various capabilities of the ASAP device. They are divided up as follows:

Module type 1		Can merely, in the "Active Read" state, supply the last valid measurement data without a time stamp
Module type 2	In addition to module type 1 can:	Generate the time stamp and has ring buffer store for exclusively one measurement job
Module type 3	In addition to module type 2 can:	Process and manage several measurement jobs sequentially
Module type 4	In addition to module type 2 can:	Calculate trigger conditions
Module type 5	In addition to module type 2 can:	Process and manage several measurement jobs sequentially and calculate their trigger conditions

Table 4 Module types

2.1.2 States of driver

The ASAP device can assume the following states:

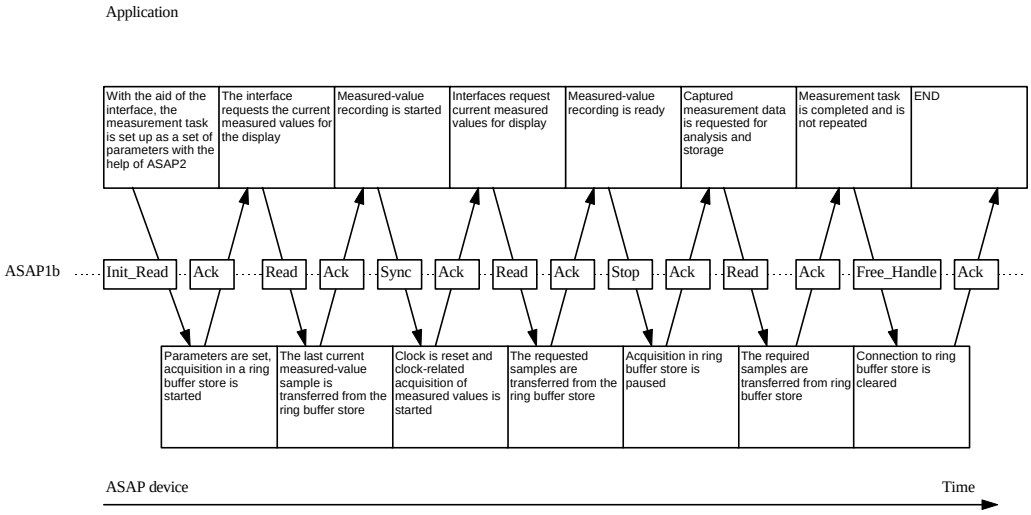


Figur 6 States and transitions between them

No state transitions are performed in the event of an error status signal from services with "NACK" (see also 2.1.5 Error handling). Cycles can also be executed several times in parallel if the ASAP device is designed accordingly. For example, the ASAP device could accomplish several tasks in parallel: process the acquisition of measured values for the higher-level measuring system and supply measured values for the on-line display of the adjusting system.

2.1.3 Referencing

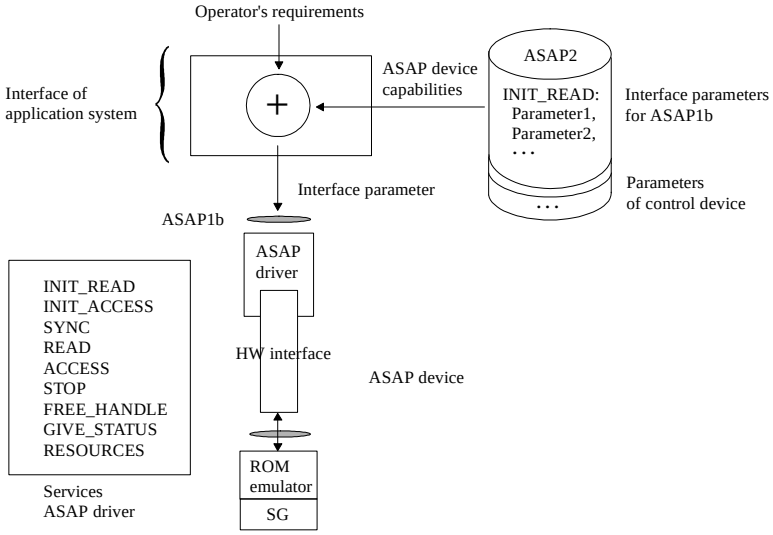
The selective responding of drivers is obtained through handles. There is precisely one handle for each individual cycle that is assigned by the higher-ranking system in the case of the INIT_READ and INIT_ACCESS services. The following services, e.g. READ, GIVE_STATUS, etc. reference each other through these unique handles.
Figur 7 shows a typical, abridged sequence using the example of a metrology task:



Figur 7 Typical sequence and allocation of tasks

2.1.4 Parameter flow (SS2)

The range of values of parameters of ASAP services are partly stipulated by ASAP, e.g. ACKNACK, or are found as elements in ASAP2, similarly to the course of action for control-device parameters, as typical for that ASAP device (Figur 8). They are then retrieved from the database as needed and passed to the interface of the application system:



Figur 8 Interaction of ASAP1b and ASAP2

The parameters contained in the database may describe the capabilities of the ASAP device (e.g. "knows module types 1 and 2") and so be used as a basis for the selection of parameters for the application system or they may contain specific parameterisation information for the ASAP device as blobs. The origin of the parameters is described in SS2 columns as per Table 5 Parameters from ASAP2 in the parameter description of the services|:

Contents	-	Opt.	Yes
Significance	Parameter not defined in ASAP2, originates from application system (operator prompt or from ASAP3)	Parameter can be contained in ASAP2	Parameter must be contained in ASAP, can be characteristic quantity of ASAP device or possible range of values of it

Table 5 Parameters from ASAP2

2.1.5 Error handling

The driver assumes the agreed sequence of individual services and does not allow any error handling for polling in an improper cycle. Communication concerning the error status is handled through the parameter ACKNACK of the individual services . This always refers to the last service and can assume three states:

Type	Description	State transitions	Example
Fatal error = NACK	Parameter set cannot be executed	No transition	Voltage supply of HW of ASAP device has failed and driver detects this
Warning	Parameter set can be executed but was modified by the driver to do this	Transition to new state	The desired sampling frequency cannot be set by the ASAP device; another, equivalent frequency is possible. This is set and entered in the parameter set and passed back.
Acknowledge = ACK	Parameter set can be executed exactly as it was passed to the ASAP device	Transition to new state	All parameters are suitable

Table 6 Error types

2.1.6 Transfer direction (Dir.)

The application and ASAP device communicate with each other in master/slave mode in principle. Parameters can be communicated between the application system or coordinator and the ASAP device or driver in three different ways:

Symbol	Description
↓	Application system specifies the parameter. If latter cannot be executed by ASAP device it answers with "NACK".
↓↑	The application system specifies the parameter. If the latter cannot be executed by the ASAP device, it corrects the parameter into an executable suggestion and answers with "warning".
↑	The parameter is returned by the ASAP device after execution of a service.

Table 7 Transfer directions

The application system therefore obtains its knowledge of the validity of the parameters passed by it from the status of the returned parameter ACKNACK. If "warning" is returned, it has to search for the parameter(s) that caused it on its own.

2.1.7 Data formats (type)

The parameters in the individual services of ASAP1b have the following data types (lower value addresses in the main memory of the MAD are on the left, higher value addresses are on the right):

Lower address in MAD Higher address in MAD

Byte

Byte

Range of values: 0 ... 255

Unsigned

Low byte	High byte

Range of values: 0 ... 65535

UnsignedLong

Low word		High Word	
Low byte	High byte	Low Byte	High Byte

Range of values:0 ... 4294967295

SignedLong

Low word		High Word		
Low byte	High byte	Low Byte	VZ	High 7 bits

Range of values: -2147483648 ... 2147483647

Pointer

Low word		High Word	
Low byte	High byte	Low Byte	High Byte

Maximum address range: 32 bits

PC string

Number of bytes	ASCII	...	ASCII	0-string

Range of values, number of bytes: 0 ... 62, 0 string = 00 (hex)

Blob

Length, low byte	Length, high byte	
Blob bytes		

The first two bytes specify the length of the remaining blob,
Range of values, length blob: 0 ... 65533 bytes (length specification without length bytes)

2.1.8 Measurement data format

During the read operation using the READ service, the measurement data is stored in the destination memory range in the following format: initially the timestamp (time of day) and then the useful bytes in the sequence and with the number of bytes which was determined at the time of parameterisation with INIT_READ. Taking an example with 3 measuring channels with 2 bytes, 1 byte and 4 bytes, the measurement data is placed in the target memory area as follows:

Memory map	Data format	Contents
Byte 1	Low word/low byte	Timestamp
Byte 2	Low word/high byte	
Byte 3	High word/low byte	
Byte 4	High word/high byte	
Byte 5	Low byte	Measuring channel 1
Byte 6	High byte	
Byte 7	Byte	Measuring channel 2
Byte 8	Low word/low byte	Measuring channel 3
Byte 9	Low word/high byte	
Byte 10	High word/low byte	
Byte 11	High word/high byte	

Figur 9 Examples for data storage in case of READ service

2.1.9 Codes for scaling units (CSE)

During parameterisation, specifications concerning the scaling units must be made at various points. The values of these specifications are uniformly laid down:

Scaling units CSE			
Value	Unit	Referred to	Comment
0	1 µsec	Time	
1	10 µsec	Time	
2	100 µsec	Time	
3	1 msec	Time	
4	10 msec	Time	
5	100 msec	Time	
6	1 sec	Time	
7	10 sec	Time	
8	min	Time	
9	h	Time	
10	d	Time	
100	Angular degrees	Angle	
101	Revolutions 360 degrees	Angle	
102	Cycle 720 degrees	Angle	e.g. in case of IC engines
103	Cylinder segment	Combustion	e.g. in case of IC engines
999	Always if there is new value		Calculation of a new upper range limit after receiving a new partial value, e.g. when calculating a complex trigger condition
1000	Non deterministic		Without fixed scaling

When listing the parameters of the individual services, possible limiting of the range of values is stated.

2.1.10 Conventions regarding indexes and handles

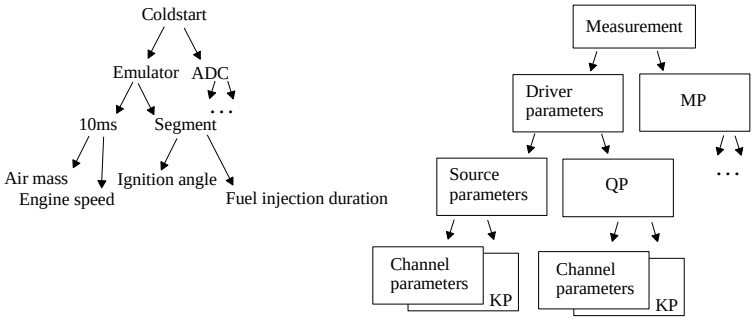
The convention is as follows when using and assigning indexes

Indexes	Start at 0 in principle and are used in ascending order, e.g. the indexes of source parameter blocks QP start at QP0 and, if there are three sources, would end at QP2.
Handles	Can be used arbitrarily within the range of values for handles but never have the value 0 in principle

2.2 INIT_READ

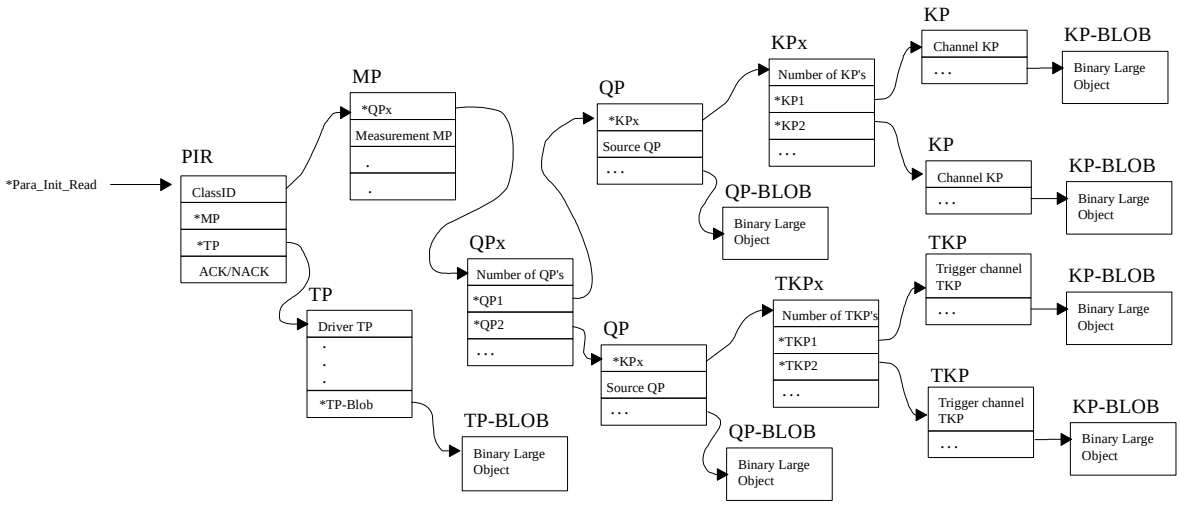
Task: to initialise a driver for the transfer of measured and processed variables to the MAD. The proper execution of this service changes the state of the driver from INSTALLED to ACTIVE READ. This service can be used by Metrology to display and record RAM locations as well as by the adjusting system, e.g. in order to display the current operating parameters.

Parameterisation: the transfer of individual parameters is handled through chained lists. Parameters related to measurements are located in field MP, those related to drivers are located in TP (in common with field TP in INIT_ACCESS). Channels having the same sampling criterion are linked through the source parameters QP and are themselves described in channel parameters KP. The transfer of trigger conditions takes place through field TKP that is treated as a measuring channel, i.e. a KP, and can accordingly be read out as a measuring channel. All parameters essentially originate either from the operator's requirement or from the described interface ASAP2.



Figur 10 Hierarchy of parameters in case of INIT_READ with example

The handle that is also transferred (measurement job handle in MP) is used from this time on as a reference to that measurement job and is used by the READ, FREE_HANDLE and GIVE_STATUS services. The measuring channels and trigger channels are referenced through indexing in fields KPx.



Figur 11 Chaining of parameter blocks in INIT_READ (schematic)

Comments: non-time critical command for preparing the READ service. Without functional overlap with the INIT_ACCESS command in order to ensure the independence of self-contained MAD Measurement and Apply applications.

Result: The ASAP device is parameterised, acquisition of measurement data takes place in the ring buffer store. READ can be used to access measurement data.

Para_INIT_READ PIR				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b01 (hex)
* MP	...↓	-	Pointer	Pointer to field MP (measurement parameter)
* TP	↓	-	Pointer	Pointer to field TP (driver parameter)
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 Warning:1 NACK: 2

Driver parameters TP (are identical to those in INIT_ACCESS!)				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b02 (hex)
* Expansion	↓	-	Pointer	Pointer for subsequent expansions
* TP-BLOB	↓	Opt.	Pointer	Used to transfer specific parameterisation and preparation values, the content of which are not known to the higher-ranking coordinator and which can only be passed by the latter. BLOBs originate generally from ASAP2 and are generated by the application.

Measurement parameters MP				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b03 (hex)
* Expansion	↓	-	Pointer	Pointer for subsequent expansions
* QPx	↓	-	Pointer	Pointer to field QPx
Measurement mode	↓	Yes	Unsigned	Sets the desired mode of measured-value recording: 0: standard 50: classification 100: standalone operation, i.e. "set out" according to parameterisation e.g. in case of MT5
Measurement job handle	↓	-	Unsigned	Handle of parameter set of a measurement job. Used as reference to: Measurement data in ASAP device in case of module types 2 to 5, Status of job in case of GIVE_STATUS service
Period duration code	↓↑	Yes	Unsigned	Setting of period duration, scaled in CSE; possible range of values: 0 ... 1000. Determines the reference unit with which all subsequent services are parameterized.

ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b06 (hex)
Number of channels	↓	-	Unsigned	Number of channels of source number of table elements (here n+1)
* KPO or TKPO	↓	-	Pointer	Pointer to field KP0 or TKP0
...	↓	-	Pointer	...
* KPn or TKPn	↓	-	Pointer	Pointer to field KPn or TKPn

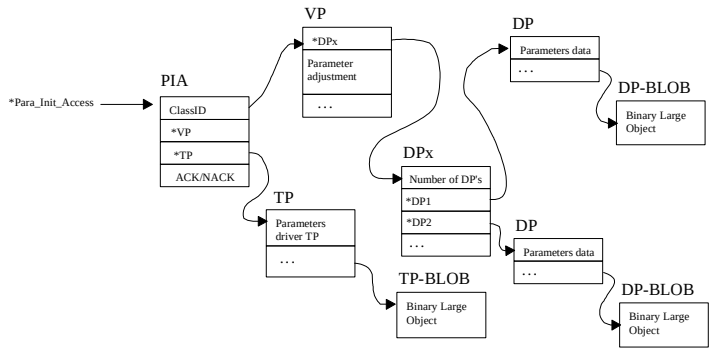
Channel parameters KP or trigger channel parameters TKP				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b07 (hex) for KP or 1b08 (hex) for TKP
* Expansion	↓	-	Pointer	Pointer for subsequent expansions
Data length	↓	Yes	Unsigned	To prepare interpretation of byte stream provided using READ service; number of bytes of a measured value: Range of values: 0 ... 4 bytes
Data type	↓	Yes	Unsigned	Definition of format conversion, needs ASAP device for internal calculation of trigger condition 0: Unsigned Intel 1: Signed Intel 2: Unsigned Motorola 3: Signed Motorola 4: Float (Borland C/4 bytes) IEEE754
*KP BLOB or *TKP BLOB	↓	Opt.	Pointer	Measuring channel parameters and trigger channel parameters are of identical structure and only differ with regard to blobs: KP blob: use to transfer individual channel parameters. Coordinator does not know their structure and can only differentiate them by indexing through ASAP1c or QPx. TKP blob: defined below

TKP blob				
Name	Dir.	SS2	Type	Description
Length, trigger parameters	↓	-	Unsigned	Specifies the total length of the parameters that then follow, i.e. 18 bytes
Operand A Type	↓	-	Unsigned	Identification of operand type: 1: Constant 2: Measuring channel
Operand A Source	↓	-	Unsigned	If operand A type = 2 (measuring channel): Index of source in QPx (starting at 0!)
Operand A Channel	↓	-	Unsigned	If operand A type = 2 (measuring channel): Index of channel in KPx (starting at 0!)
Operator	↓	Yes	Unsigned	Selection of trigger channel or constant logic operation: 0: >1: <2: =3: ≠ etc. Reference values to ASCII of operator in ASAP2
Operand B Type	↓	-	Unsigned	Identification of operand type: 1: Constant 2: Measuring channel
Operand B Source	↓	-	Unsigned	If operand B type = 2 (measuring channel): Index of source in QPx (starting at 0!)
Operand B Channel	↓	-	Unsigned	If operand B type = 2 (measuring channel): Index of channel in KPx (starting at 0!)
Constant	↓	Opt.	Signed Long	Transfer of constant if operand type = 1

2.3 INIT_ACCESS

Task: Initialisation of driver for adjusting operations, in oder to access memory locations and areas in the control device in the memory location read / modify / writeback cycle. The proper execution of this service changes the state of the driver from INSTALLED to ACTIVE ACCESS. This service is generally used by adjusting systems (A in MAD).

Parameterisation: In a similar way to INIT_READ, the transfer of individual parameters is handled through branched structures. The parameters from the adjustment job are located in field VP, those related to a driver are located in TP (in common with field TP in INIT_READ). Parameterisation of individual adjustment objects is located in the data parameters DP. Reference to adjustment objects is obtained through the handle of the adjusting job and the indexing of the data parameters in field DPx.



Figur 12 INIT_ACCESS parameter blocks

Result: The connection to the memory blocks to be altered is set up and the desired adjustment modes are set. The ACCESS service can be used to describe these structures.

Para_INIT_ACCESS PIA				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b09 (hex)
* VP	↓	-	Pointer	Pointer to field VP (adjustment parameters)
* TP	↓	-	Pointer	Pointer to field TP (driver parameters)
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 NACK: 2

Adjustment parameters VP				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0a (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
* DPx	↓	-	Pointer	Pointer to field DPx
Adjustment job handle	↓	-	Unsigned	Handle of parameter set of an adjustment job. Used as reference to: • futher operations executed by means of ACCESS service • status of job in case of service GIVE_STATUS

Driver parameters TP (<u>are identical to those in INIT_READ!</u>)				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b02 (hex)
* Expansion	↓	-	Pointer	Pointer for subsequent expansions
* TP BLOB	↓	Opt.	Pointer	Used to transfer specific parameterisation and preparation values, content of which is not known to the higher-ranking coordinator and which can only be passed by the latter. BLOBs originate generally from ASAP2 and are generated by the application.

Table of data parameters DPx				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0b (hex)
Number of data items	↓	-	Unsigned	Number of data parameter fields DP number of table elements (here n-1)
* DP0	↓	-	Pointer	Pointer to field DP0
...	↓	-	Pointer	...
* DPn	↓	-	Pointer	Pointer to field DPn

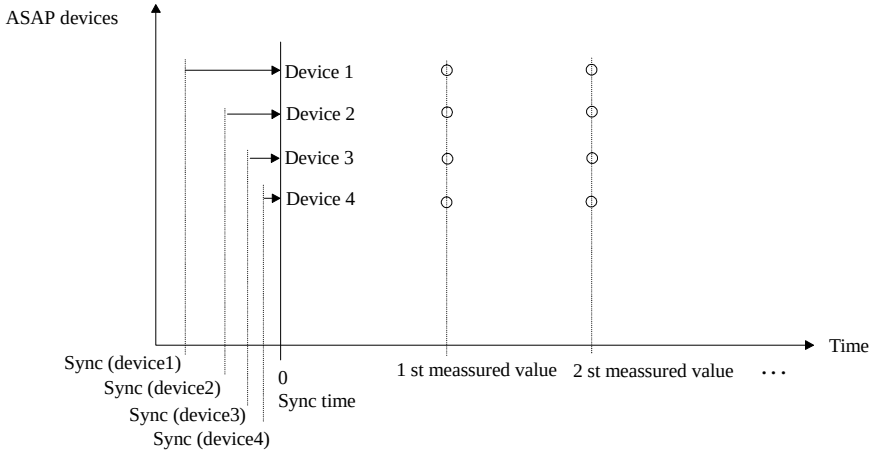
Data parameters DP				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0c (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Data type Note: parameter retained for compatibility reasons until next version but has no function and no effect	↓	-	Unsigned	Types of data transferred using ACCESS: 0: Unsigned Intel 1: Signed Intel 2: Unsigned Motorola 3: Signed Motorola 4: Float (Borland C/4 bytes) IEEE754
* DP BLOB	↓	Opt.	Pointer	e.g. pointer to address and length of data to be transferred and/or pointer to transferring data.

2.4 SYNC

Task: Synchronisation of one or more drivers for data acquisition.

Sequence: Data acquisition in the ring buffer store was already started using the service INIT_READ, SYNC is then used to set the clocks and correct the ring buffer store management of the ASAP devices so that time-related read operations can subsequently be executed using READ with module types > 1. The synchronisation process involving several connected ASAP devices is executed as a sync-in-future process. After initialisation with INIT_READ, the data from the ASAP devices is on hand in the drivers. Module types with a storage capability (2 to 5) write to their ring buffer store. The synchronisation time of individual ASAP devices that is contained in ASAP2 as a typical synchronisation time or can be determined by measurement by coordinator, is used when the service SYNC is invoked (sync-in-future). Starting from the ASAP device with the longest synchronisation time, the system clocks in the subsystems are set to the reference standard for synchronisation (sync time) in descending order of synchronisation time Running through all the ASAP devices achieves synchronisation of individual drivers from this time on. At the same time, this instant (Figur 13, sync time) is the reference point for time-related read operations using READ:

Important: on proper completion (ACK) of this service, the time reference for any previous measurement data remaining in the subsystem that has not been deleted (FREE HANDLE) is lost due to the newly defined system time.



Figur 13 Sync-in-future synchronisation

Para_Sync PS				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0d (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Period duration code	↓	Yes	Unsigned	Setting of unit for synchronisation, number of these values in "PS, number of periods": scaled in CSE, possible range of values: 0 ... 9 (only time-related units)
Number of periods	↓	-	Unsigned Long	Time to synchronisation, unit in "period duration code" parameter in PS
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 NACK: 2

2.5 READ

Task: Data transfer of measurement-data blocks from ASAP device to application.

This service is functionally identical for both the "Active Read" as well as the "Done" state. When doing so, various read strategies may come into play for the parameters Start time and Length under the control of the parameter Mode (see parameter Mode).

Para_Read PR				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0e (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Measurement-job handle	↓	-	Unsigned	Reference to measurement that was parameterised using INIT_READ. This makes it possible, for example, to distinguish "normal" read and on line read for the same ASAP device. Parameter originates from "INIT_READ, MP, measurement-job handle"
Mode	↓	Opt.	Unsigned	Specification of correction algorithm for cases where requested data is not available exactly as parameterised. "Warning" is then returned in ACKNACK 0: Initially reduce parameter Start time, then parameter Length 1: Initially reduce parameter Length, then parameter Start time 2: Read next block, next block with the parameters of the previously executed service READ with mode parameters 0 or 1
Start time	↓↑ (depen-ding on mode)	-	Unsigned Long	First sample requested, referred to time of day that was set to "0" using SYNC. Scaling is specified in parameter INIT_READ, QP, "C time of day, measurement data resolution". Note: if mode = 2, no valid start time must be passed
Length	↓↑ (depen-ding on mode)	-	Unsigned Long	Number of requested samples, referred to start time, scaling is specified in the parameter INIT_READ, QP, time of day. Using "Number of" instead of "time" makes it possible to calculate the size of the buffer neededin the main memory of MAD. In principle, this parameter exceeds 1 and can be corrected in the case of mode parameters equal to 1 and 2.
Source index	↓	-	Unsigned	Specification of source to which this service is to relate. It is taken from the field INIT_READ, QPx
Destination address	↓	-	Pointer	Specification of storage address beyond which driver must store measurement data
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 Warning:1 NACK: 2

2.6 ACCESS

Task: Data transfer of adjustment data between application and the ASAP device. In doing so, the read/modify/write cycle is generally used equivalently to the functions fetch and display control-device date/modify content/write back date to control device. The corresponding initialisation takes place using the "INIT ACCESS" service.

Para_Access PA				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b0f (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Adjustment job handle IMPORTANT: new parameter in Version 1.0	↓	-	Unsigned	Handle of parameter set of an adjustment job. Originates from initialisation by means of INIT ACCESS.
Direction	↓	-	Unsigned	Indication of transfer direction: 0: from coordinator to ASAP device 1: from ASAP device to coordinator
Data index	↓	-	Unsigned	Indication of target data area too/from which data is to be read/written. Parameter is the index in the structure "INIT_ACCESS, DPx"
Offset memory	↓	-	Unsigned Long	Specification of address to or from which data is to be written or read; is the offset between the destination data area stipulated in case of "INIT_ACCESS"
Length	↓	-	Unsigned Long	Number of bytes to be read and/or written, length within destination data area from INIT_ACCESS. Indication measured in bytes
* Data	↓	-	Pointer	Pointer to data to be transferred
* PA BLOB	↓	Opt.	Pointer	Pointer to any special parameters needed for this service; these are obtained from ASAP2
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 NACK: 2

2.7 STOP

Task: Pauses data acquisition in ring buffer stores in case of module types 2 to 5. Data in the ASAP device can then only be read offline using READ.

Para_Stop PST				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b10 (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Measurement-job handle	↓	-	Unsigned	Reference to measurement that was parameterised using INIT_READ and that is paused by this service in the case of module types 2 to 5. The parameter originates from INIT_READ, MP, measurement-job handle
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 Warning:1 NACK: 2

2.8 FREE_HANDLE

Task: Clearing of connection to measurement and adjustment jobs in ASAP device.
These jobs are referenced by means of the handles assigned in the appropriate INIT READ or INIT ACCESS.

Result: Data in the ring buffer store is discarded. The ASAP device frees the memory occupied by this measurement job for new requirements.

Para_Free_Handle PFH				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b11 (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Selection handle	↓	-	Unsigned	Reference to measurement job or adjustment job that was parameterised using INIT_READ or INIT_ACCESS. The parameter originates from fields MP or VP of these services.
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 NACK: 2

2.9 GIVE_STATUS

Task: Request to the ASAP device to return information stored in the subsystem. In order to be able to address different topic areas selectively, selection is made possible by the parameter "Info type". The information is placed in the main memory of MAD starting from the address specified by the parameter "destination address". The type and structure of the information are apparent from the "Info types" Table. A further referencing mechanism is obtained through the handle that belongs to the desired measurement or adjustment job (Selection handle). The parameters of the service GIVE_STATUS are made up as follows:

Para_Give_Status PGS				
Name	Dir.	SS2	Type	Description
ClassID	↓	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b12 (hex)
* Expansion	↓	-	Pointer	Pointer to field for subsequent expansion
Destination address	↓	-	Pointer	Specification of memory address beyond which information is to be stored by the driver
Info length	↓↑	-	Unsigned	Maximum length of information to be supplied; with this the ASAP device can decide whether the information originating from it in the case of this service can be stored completely. Otherwise nothing is transferred and "NACK" is returned, if there is sufficient storage space the desired information is transferred and the "Info length" parameter is corrected to the length actually needed, measured in bytes.
Info type	↓	Opt.	Unsigned	Used to select the desired information, the parameter signifies: 0: Information 1: Identification 2: Operating state 3: Error state 4: StatusInitRead 5: Parameter return ... 100 to 107 GS blobs 1 to 8
Selection handle	↓	-	Unsigned	Reference to measurement job that was parameterised with INIT_READ or adjustment job parameterised with INIT_ACCESS. The parameter originates respectively either from INIT_READ, MP, measurement-job handle or from INIT_ACCESS, VP, adjustment-job handle. It is needed for selection "StatusInitRead" and "Parameter return" (see above)
ACKNACK	↑	-	Unsigned	Feedback concerning error state of driver OK: 0 Warning:1 NACK: 2

The parameter "Info type" can be used to select specific information areas that return the following contents:

Info type parameter for selecting information	Content and format of return (entered in memory of MAD starting from "destination address")
0: Information	Details concerning driver and lower-ranking ASAP device: • Driver name PC string • Creation date PC string • Manufacturer PC string • Number of connected HW items Unsigned • Connected HW1 PC string • Connected HW2 PC string etc.
1:Identification	Details concerning driver software present: Version Unsigned Variant Unsigned Revision Unsigned (Note: these three values start with 1.0.0, in addition, these details are also found identically, but in ASCII, as the 1st comment line in the error file, e.g. -2, 1.0.0) ModulTypREAD Unsigned ModulTypACCESS Unsigned (to be defined)
2: Operating state	State which the driver is in: Number of handles Unsigned Handle, state Unsigned, Unsigned Handle, state Unsigned, Unsigned The handle is a measurement-job handle or an adjustment-job handle of which the state is defined as: 0: Active read 1: Done (measurement-job handle omly) 2: Active access 3: Installed
3: Error state	For comments, only the last warning or NACK returned by the driver: In doing so, only the cause or comment for just one error can ever be returned. Error number Unsigned Cause Unsigned Remedial measure 1 Unsigned Remedial measure 2 Unsigned Error text PC string
4: StatusInitRead Important: there are plans to modify the format of this info type from the next version of the specification onwards. See Annex to this Table.	Information concerning ring buffer store status of data acquisition started using INIT_READ; selection of data acquisition is obtained through the parameter "Selection handle" of the service Give_Status: Earliest time of day Unsigned Long Scaled in CSE from INIT_READ, QP, code time of day resolution Latest time of day Unsigned Long Scaled in CSE from INIT_READ, QP, code time of day resolution Overflow Unsigned Remaining measuring time Unsigned Long Scaled in CSE from INIT_READ, QP, code time of day resolution Sync Yes No Unsigned 0: Yes 1: No Trigger status Unsigned 0: Wait for trigger event 1: Trigger event occurred 2: Measurement has completed, ready Trigger time of day Unsigned Long Scaled in CSE from "INIT_READ, QP, time of day resolution code"
5: Parameter return	Return of parameter lists only for INIT_READ and INIT_ACCESS services, selection of lists is obtained by the parameter "PGS, measurement-job handle". The driver writes the result, i.e. the structures "PIR" or "PIA", back to the memory area specified in "PGS, destination address".
...	... spare for further info types
100 to 107: GS blobs 1 to 8	For 8 user-specific blobs. A blob is to be returned by the driver to the coordinator. The destination address to which it must be written is passed in the parameter "Target address". Used to transfer information that is not specifically defined from the driver to the coordinator.

4: StatusInitRead From next version of specification	Information concerning ring buffer store status of data acquisition started using INIT_READ; selection of data acquisition is obtained through the parameter "Selection handle" of the service Give_Status: Ÿ Overflow Unsigned Ÿ Remaining measuring time UnsignedLong Scaled in CSE from INIT_READ, QP, time of day resolution code Note: measuring time still to run for this measurement, i.e. this value = 0 in the "DONE" state. Ÿ Sync Yes No Unsigned 0: No 1: Yes Ÿ Trigger status Unsigned 0: Wait for trigger event 1: Trigger event occurred 2: Measurement has completed, ready Ÿ Trigger time of day UnsignedLong Scaled in CSE from "INIT_READ, QP, time of day resolution code" This is followed by indications of earliest and latest time of day for all sources: Source 1: Ÿ Earliest time of day UnsignedLong Scaled in CSE from INIT_READ, QP, time of day resolution code Ÿ Latest time of day UnsignedLong Scaled in CSE from INIT_READ, QP, time of day resolution code ... Source n: Ÿ Earliest time of day UnsignedLong Scaled in CSE from INIT_READ, QP, time of day resolution code Ÿ Latest time of day UnsignedLong Scaled in CSE from INIT_READ, QP, time of day resolution code

2.10 RESOURCES

Task: This service is used by the ASAP device as a request for system resources to the higher-ranking coordinator (resource types 0 to 10) or as a means of information concerning the configuration currently set (e.g. by means of jumpers, etc.) or occupancy (resource type 100).

Para_Resources PRS				
Name	Dir.	SS2	Type	Description
ClassID	↓↑	-	Unsigned	Type number, one per parameter block, used only to identify relevant block Value: 1b13 (hex)
* Expansion	↓↑	-	Pointer	Pointer to field for subsequent expansion
Resource type	↑	-	Unsigned	Selection of resource needed by ASAP device: 0: Request ISR 1: Enable ISR 2: Inhibit IRQ 3: Enable IRQ 4: Exclusive CPU 5: Enable CPU 6: Request memory 7: Enable memory 8: Reset timer 9: Enable timer 10: Current systemtime 100: Request configuration A more detailed description of the parameters of these resource types is given in the Table below.
Destination address	↓↑	-	Pointer	Indication of memory address beyond which information is stored
ACKNACK	↓↑	-	Unsigned	Feedback concerning error state of driver or coordinator (different to the other services) OK: 0 Warning:1 NACK: 2

Table of resource type parameters:

Resource type	Dir.	SS2	Type	Description
0: Request ISR				Request insertion of interrupt service routine (ISR)
IRQ number	↑	-	Unsigned	Indication of desired IRQ: Range of values 0 ... 15
Routine address	↑	-	Pointer	Pointer to ISR to be inserted Comment: ISR contains EOI and is terminated with IRET
1: Enable ISR				Request to remove an interrupt service routine (ISR)
IRQ number	↑	-	Unsigned	Indication of desired IRQ; Range of values 0 ... 15
2: Disable IRQ				Disable precisely one interrupt
IRQ number	↑	-	Unsigned	Indication of desired interrupt as number, not bit mask; Range of values 0 ... 15
3: Enable IRQ				Enable precisely one interrupt
IRQ number	↑	-	Unsigned	Indication of desired IRQ as number, not bit mask; Range of values 0 ... 15
4: CPU exclusive				ASAP device requests CPU for itself alone in order not to be interrupted by other routines
5: Enable CPU				ASAP device releases protection against interruption (initiated with resource type 4)
6: Request memory				Request by ASAP device for allocation of memory by the coordinator
Quantity	↑	-	Unsigned Long	Indication of quantity of memory desired in bytes
Address Memory	↓	-	Pointer	Pointer to allocated memory
7: Enable memory				Request to enable memory requested by ASAP device with resource type 6
Address Memory	↑	-	Pointer	Pointer to memory to be enabled
8: Reset timer				Reset of timer in the coordinator; timer has "egg timer" characteristic and must therefore be reset again for cyclic operation
Time	↓	-	Unsigned Long	Indication of desired running time of timer, the parameter will be given the fixed value=1 when the next version is redrafted.
Pointer Routine	↓	-	Pointer	Pointer to routine to be jumped to on expiry of timer
Timer handle	↓	-	Unsigned	Return of reference to timer by the coordinator
9: Enable timer				Request to enable the timer requested by ASAP device with resource type 8
Timer handle	↑	-	Unsigned	Reference to timer to be enabled
10:Current system time				Request by ASAP device for transfer of central system time of coordinator
System time period duration code	↑	-	Unsigned	Requirement of ASAP device for system-time resolution; scaled in CSE, Range of values 0 ... 9

<table>
<tr><td>System time</td><td>↓</td><td>-</td><td>Unsigned Long</td><td colspan="3">System time of coordinator scaled as desired in previous "system time period duration code"</td></tr>
<tr><td>100: Request configuration</td><td></td><td></td><td></td><td colspan="3">Request by ASAP device for transfer of information concerning HW interfaces, DMA channels, UMB memory, I/O areas and IRQ assignments set. ASAP device can request information concerning one configuration object (see current, expandable list below). Data interchange takes place as a PC string.
For example, ASAP device sends the PC string "COM" and receives "2" as a response. In this way the driver knows that its subsystem is to be addressed via interface COM2. If no serial interface were available, the response of the coordinator would be "NACK".</td></tr>
<tr><td rowspan="15">Configuration</td><td rowspan="15">↓↑</td><td rowspan="15">-</td><td rowspan="15">PC string</td><td colspan="3">List of currently defined configuration objects:</td></tr>
<tr><td>Configuration object</td><td>Dir.</td><td>Content and range of values</td></tr>
<tr><td rowspan="2">Serial interface</td><td>↑</td><td>"COM"</td></tr>
<tr><td>↓</td><td>"1" ... "4"</td></tr>
<tr><td rowspan="2">Parallel interface</td><td>↑</td><td>"LPT"</td></tr>
<tr><td>↓</td><td>"1" ... "2"</td></tr>
<tr><td rowspan="2">DMA channel</td><td>↑</td><td>"DMA"</td></tr>
<tr><td>↓</td><td>"1" ... "7"</td></tr>
<tr><td rowspan="2">UMB window</td><td>↑</td><td>"UMB"</td></tr>
<tr><td>↓</td><td>"1234-ABCD"
(cf. exclude option EMM386)</td></tr>
<tr><td rowspan="2">I/O addresses</td><td>↑</td><td>"I/O"</td></tr>
<tr><td>↓</td><td>"123-ABC"
(cf. exclude option EMM386)</td></tr>
<tr><td rowspan="2">IRQ assignment</td><td>↑</td><td>"IRQ"</td></tr>
<tr><td>↓</td><td>"0" ... "15"</td></tr>
</table>

3. Parameters from ASAP2

The following information from ASAP2 is needed in order to parameterise ASAP1b services:

Name	SS2	Service	Structure
TP blob	opt.	INIT_READ	TP
Measurement mode	Yes	INIT_READ	MP
Period duration code	Yes	INIT_READ	MP
Measurement mode	Yes	INIT_READ	MP
Source number	Yes	INIT_READ	QP
Source period duration code	Yes	INIT_READ	QP
Time of day, measurement data resolution code	Yes	INIT_READ	QP
Sampling interval	Opt.	INIT_READ	QP
Time estimate	Yes	INIT_READ	QP
OP blob	Opt.	INIT_READ	QP
Data length	Yes	INIT_READ	KP
Data type	Yes	INIT_READ	KP
KP blob	Opt.	INIT_READ	KP
Operator	Yes	INIT_READ	TKP
Data length	Yes	INIT_READ	TKP
Data type	Yes	INIT_READ	TKP
Constant	Opt.	INIT_READ	TKP
TP blob	Opt.	INIT_ACCESS	TP
DP blob	Opt.	INIT_ACCESS	DP
Period duration code	Yes	SYNC	PS
Mode	Opt.	READ	PR
PA blob	Opt.	ACCESS	PA
Info type	Opt.	GIVE_STATUS	PGS
Configuration	Yes	RESOURCES	PRS
Module type	Yes	-	-
Maximum number of channels per source	Yes	-	-
Maximum number of channels	Yes	-	-

4. Abbreviations used

ACK	Acknowledge
ADC	Analogue/Digital Converter
ASAP	Working Party on Standardisation of Applications
AuSy	Automation System
BLOB	Binary Large Object
CAN	Controller Area Network
DME	Digital Motor Electronics
DP	Data Parameter
HW	Hardware
IF	Interface
KP	Channel Parameter
MAD	Measurement, Application and Diagnostic system
MP	Measurement Parameter
MS	Measuring System
MT	Module Type
NACK	Not Acknowledge
QP	Source Parameter
SG	Control Device
SW	Software
TP	Driver Parameter
VP	Adjustment Parameter
VS	Adjusting System