CiA 810

Application note

CiA 434 implementation guideline

Version: 1.0.0

24 March 2010

HISTORY

Date	Changes
2010-03-24	*Publication of version 1.0* as application note

CONTENTS

1 Scope

This application note recommends and suggests how to implement and handle devices, which are compliant to CiA 434 profile family.

The application note discusses the different modes of operation for laboratory automation slaves (LAS), the transfer of the commands to the LAS as well as the handling of the command parameter RAM (CPRAM).

The purpose of this application note is to explain the major issues of specification CiA 434 standard part 1. This document is not normative but informative. In case of doubts always the specifications given in CiA 434 apply.

2 References

/CiA434/	CiA 434, CANopen profiles for laboratory automation systems

The references given in /CiA434/ apply for this document as well.

3 Abbreviations and definitions

The abbreviations and definitions given in /CiA434/ apply for this document as well.

4 Operating principles

4.1 General

This clause describes the general system architecture as well as the idea of command structures.

4.2 General system architecture

The intention of the application profile /CiA434-1/ is to define a common abstraction of a laboratory automation slave (LAS) device. This profile is suitable for a wide range of different laboratory automation devices. It is applicable for simple devices like diaphragm pumps as well as for very complex devices like multi axis positioning systems or pipetting systems. To provide this flexibility the finite state automaton (FSA) for LASs consists of mandatory as well as optional FSA states and sub-states. Any LAS supports the mandatory FSA states. Optional FSA states are only supported, in case the related functionality is supported respectively required by the LAS. State transitions within the FSA are based on device internal events (e.g. occurrence of device errors) or on the reception of the FSA command.

The functional behavior of the LAS is controlled via operation commands. The set of profiles for laboratory automation systems /CiA434/ offers two ways of providing operation commands to the LAS, the Direct execution mode and the Batch mode. In Direct execution mode, the laboratory automation master (LAM) feeds the LAS with single operation commands respectively command structures (A command structure consists of a single command influencing the LAS' process output, the related command parameters and a bit mask, which identifies the command parameters.). The LAS starts the execution of the command immediately after the entire command reception and generates the command result. In Batch mode, operation commands are not directly executed after reception but stored in so-called command buffers. The LAM is therefore enabled to pre-program complete work plans within the LAS. The execution of these work plans is triggered via the related FSA state transitions. Any LAS supports either Direct execution mode or Batch mode. Complex LAS devices may support both. Even in case a LAS supports Direct execution mode as well as Batch mode, this

LAS can only be operated in one of these modes at a certain time. Because the implementation of these two operating modes is optional the FSA differs for devices that support only one of these two operating modes. Figure 1 illustrates the minimum FSA implementation, separately for Batch mode or Direct execution mode.

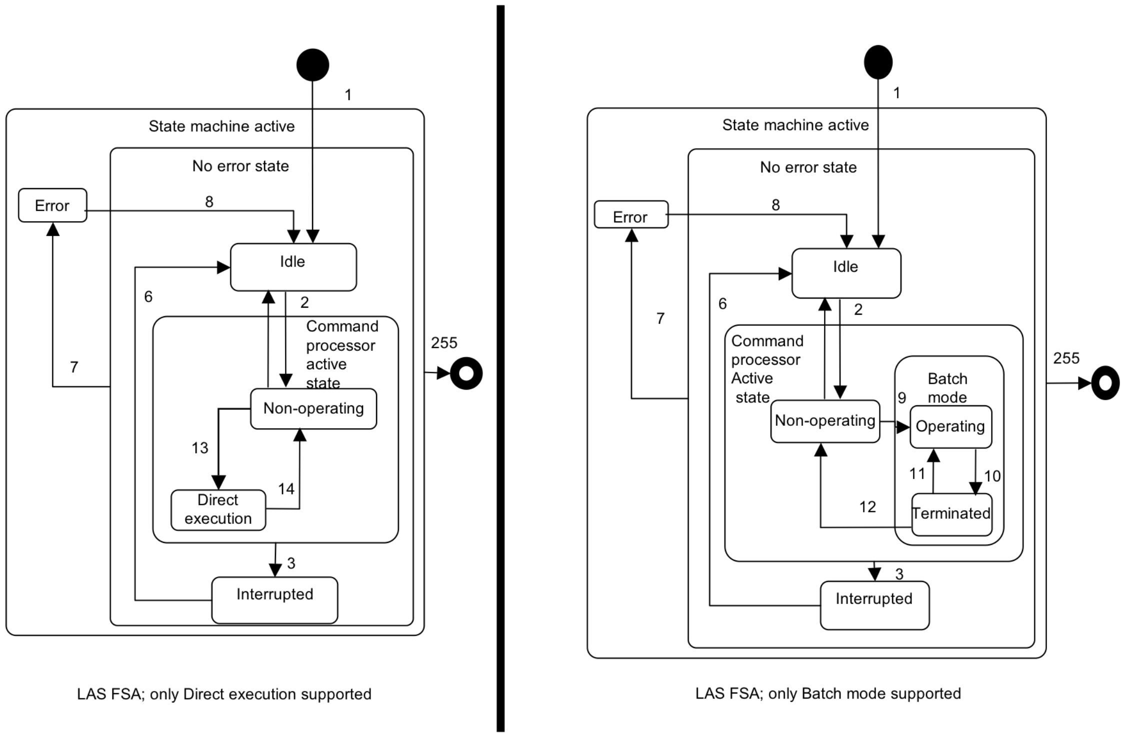


Figure 1 - LAS FSA minimum implementations

4.3 Command structures

Usually in CANopen device profiles, complex CANopen devices are controlled via a control word and a status word. This means the CANopen device supports one RPDO for the reception of the command and transmits its current status via one TPDO. The parameters, which may be required as inputs for certain device-internal functionality, initiated by the control word, are pre-configured within the CANopen device' (object dictionary).

Because laboratory automation devices may offer a higher level of abstraction and complexity than simple CANopen devices they may require much more complex commands with many pre-configured parameters for a single command. Though commands may typically have many parameters, most of them need to be specified only rarely, if not only once, at the beginning of the session. Later invocations of the command often need a reduced set of parameters, to specify the only values that change from one invocation to the other. It is thus necessary to provide a means of passing along with a command, an arbitrary number of parameters. Therefore the so-called command structures were introduced.

An entire command structure defines the maximum amount as well as type of the parameters, which are transmitted together with a command and need to be updated prior to command execution. In addition a command structure defines the order on how to transmit command and parameters.

The selection of the subset of parameters that are transmitted together with the command is performed by a bitmask, included in the structure as well.

This bitmask allows indicating, which of the command-related parameters are transmitted in fact within this command structure. Therefore busload can be reduced as only the really required parameters are transmitted. The others, which are not transmitted, remain as they are pre-configured within the LAS.

The command structure consisting of command, bitmask and parameters is illustrated in Figure 2. The definition of the entire command structures is done for each LAS device profile, part 2 and higher of /CiA434/. In a command, the parameters are numbered from 1 to n. In the structure, the parameters are also numbered from 1 to n, but some may be missing. The missing parameters are indicated in the bitmask as zeroes.

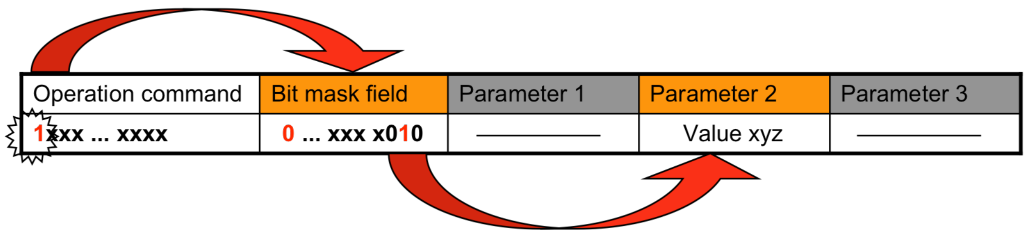
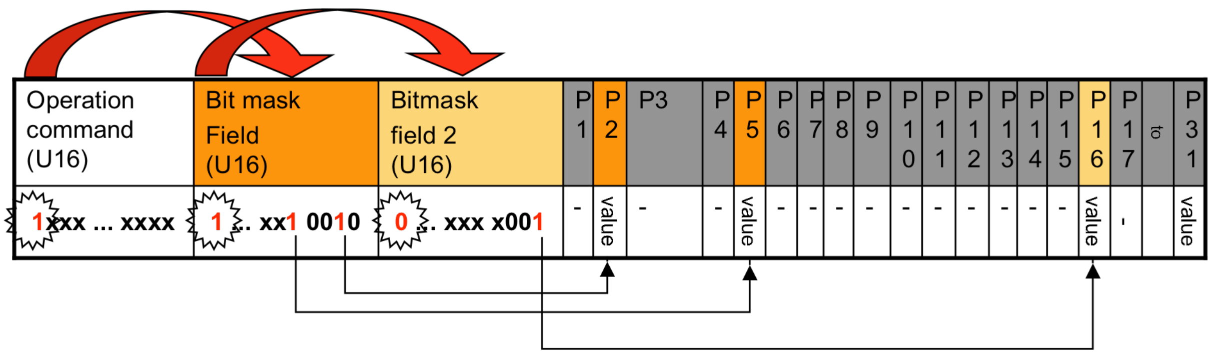


Figure 2 – General principle of a command structure

Therefore the bitmask field determines, which parameters are transmitted in fact within this command structures. In the example shown in Figure 2, only the value for parameter 2 would be transmitted. Parameter 1 and 3 would not be updated on reception of that command structure. They would remain as they are.



Caption:

✹	Bitmask indicator (bm bit)
Px	Parameter x; defined in CiA 434 part 2 or higher
value	Value transmitted in command structure; size determined by application object (related CANopen object dictionary entry)
-	Indicates that this parameter is not transmitted (size = 0)

Figure 3 - Command structure utilizing two bit masks and transmitting three parameters

The example illustrated in Figure 3 indicates a command structure, which consists of the command, two bitmasks and three parameters. In addition Figure 3 explains the coding of the bitmask in detail. Take care that these figures illustrate the command structure from point of view of the application layer! On data link layer the single bits (e.g. of the unsigned Integer 16 values) may appear in a different order (Intel representation)!

An additional example clarifies more detailed the relationship between command (definitions), command parameters, defined command structures and command structures that are really transmitted via the CAN bus. In case a notional command 0x12 requires 6 input parameters for execution. These parameters are represented in the LAS' object dictionary at the indices e.g. 6050h/01h, 6053h/00h, 6055/04h, 6055/06h, 6057/02h, and 6057/05h. At the given the indices provide the the following values:

Parameter 1	6050h/01h:	12
Parameter 2	6053h/00h:	2356
Parameter 3	6055/04h:	4345
Parameter 4	6055/06h:	0000
Parameter 5	6057/02h:	7644
Parameter 6	6057/05h:	4574

A LAS master may evaluate the object dictionary entries and may come the result that only the objects 6050h/01h, 6055h/06h, 6057h/02h have to be updated prior to command execution. Therefore the LAS master wont send the entire command structure as illustrated in Figure 4 but a reduced one, where the already correctly configured parameters are omitted. The bit mask provides the information which parameters are transmitted.

Cmd	Bitmask field	Param. 1	Param. 2	Param. 3	Param. 4	Param. 5	Param. 6
0012h	0000000000011001	<new value> for object 6050h/01h	(keep previous value in 6053h/00h)	(keep previous value in 6055h/04h)	<new value> for object 6055h/06h	<new value> for object 6057h/02h	(keep previous value in 6057h/05h)

Figure 4 – Structure of the command and its parameters actually sent

The message actually transmitted via the bus is illustrated in Figure 5.

0012h	0019h	Parameter 1 value (e.g. 10)	Parameter 4 value (e.g. 4231)	Parameter 5 value (e.g. 0)

Figure 5 – Transmitted command structure

It is important to notice that the command parameters are handled in the same order as they are defined in the related command structure definition! For manufacturer-specific commands, this relationship has to be published in the manual of the device.

After successful command structure reception and updating of the object dictionary entries, the command parameters own the following values prior to command execution:

Parameter 1	6050h/01h:	10
Parameter 2	6053h/00h:	2356
Parameter 3	6055/04h:	4345
Parameter 4	6055/06h:	4231
Parameter 5	6057/02h:	0
Parameter 6	6057/05h:	4574

Mode-specific handling of commands and parameter update is provided in the related clauses, explaining batch mode as well as direct execution mode.

5 Modes of operation

5.1 General

The profile /CiA434/ differentiates between to modes of operations, the batch mode and the direct execution mode. The main difference between these modes is storage and handling of LAS application programs.

A LAS supporting batch mode may process autonomously a comprehensive LAS application program and will inform from time to time about the current status. Therefore the LAM is unburdened from some controlling issues and just monitors the LAS. In addition the busload is decreased. But the implementation of a batch mode may lead to high demands on the LAS' microcontroller with regard to performance and memory resources.

A device supporting direct mode only, is not capable to store a complete LAS application program. In direct mode only a single command can be processed. Therefore a LAS supporting direct mode only, may run on a simple microcontroller as well. But shifting the application program handling to the LAM leads to high communication effort resp. high busload.

5.2 Batch mode

In batch mode the laboratory automation unit processes a preconfigured command sequence, an application program. This application program was pre-programmed in the LAS by the LAM prior to the execution. The batch mode enables the autonomous processing of a complex batch command sequence by the LAS without interaction of the LAM.

In case a LAS supports this operation mode, at least one command buffer shall be implemented. In this command buffer the LAM can pre-program the LAS application program. Prior to the execution of certain commands, some command-related parameters need to be updated. In order to avoid that an interaction of the LAM is required, the LAS may support a so-called command parameter RAM (CPRAM) and a parameter locator object.

The CPRAM provides all sets of parameters, which are required for executing the LAS application program. One parameter set is started with a bit mask. This bit mask indicates how many as well as which parameters are provided in this parameter set. The different parameter sets may be provided in any order in this CPRAM. To determine, which set of parameters belongs to which command in the command buffer, the parameter locator is used. The parameter locator points to the CPRAM and identifies the bit mask of that parameter set, which belongs to the related operation command.

A complex LAS may support a second command buffer. Such devices enable the LAM to execute one application program while the other command buffer is modified or re-programmed. LASs supporting two command buffers support an additional CPRAM as well as parameter locator.

5.3 Direct execution mode

The direct execution mode enables the immediate execution of a received command. In this mode the application program is in the LAM not in the LAS. The LAM provides the application program command by command. Optionally the command may be encoded and transferred to the LAS as so-called command structure. Each command is processed immediately after entire command (structure) reception.

In comparison to batch mode the communication and command processing in direct execution mode may be more similar to other existing CANopen profiles.

6 Storage and handling of operation commands and application programs in batch mode

6.1 General structure

Storage and handling of LAS application programs requires in minimum the mandatory object 6003h "Command buffer1". In case all parameters, which are required for correct command execution, are preconfigured in the LAS, no further objects are required for LAS application program execution. The reception of the FSA command triggers the state transition from "FSA

Non-operating state" to "FSA Operating state" and initiates the execution of the LAS application program, which is provided in command buffer 1 (CB1).

In case some parameters need to be updated prior to the execution of an operation command, the values to be configured can already be stored within the LAS. For this purpose the LAS can optionally support the CPRAM for CB1 (Object 6700h to 677Eh). In case CPRAM is supported, for each command a pointer has to be provided which indicates, which parameter set provided in CPRAM is related to which command. This pointer is provided in the object 6005h "Parameter locator".

Optionally a LAS may provide a second command buffer. In case the operation commands provided in command buffer 2 require an update of the related command parameters as well, it is not allowed to utilize the CPRAM for CB1. Such a device shall provide the required parameters in CPRAM for CB2 (Object 6780h to 67FEh). In addition the parameter locator for CB2 (Object 6006h) shall be supported.

6.2 Storage

The command buffer stores all operation commands of the preconfigured command structure. The CPRAM stores the bit masks and the parameters of these command structures and the parameter locator assigns the commands in the command buffer with the relevant parameters in the CPRAM. All three parts form one entire application program. A CPRAM is only required if the device needs to provide additional parameters in its command structure.

NOTE Typical examples for commands that require further parameters are motion- or dosing commands. For each motion- or dosing action, new target positions (x-, y-, z-Parameter) resp. target volumes are required. These new target vales may be provided in the CPRAM for batch mode.

If all operation commands use only preconfigured parameters, no CPRAM is required. The parameter locator is optional as well. The implementation of a parameter locator is mandatory, if a CPRAM is implemented.

The command buffer and the parameter locator are organized as an array of the data type unsigned Integer 16. A command buffer may store up to 254 commands. Each operation command, which is stored in the command buffer, owns a parameter locator in the very same sub-index of the parameter locator array. This is why the parameter locator owns the

same number of sub-indices like the related command buffer. The correlation of command buffer and parameter locator is illustrated in Figure 6.

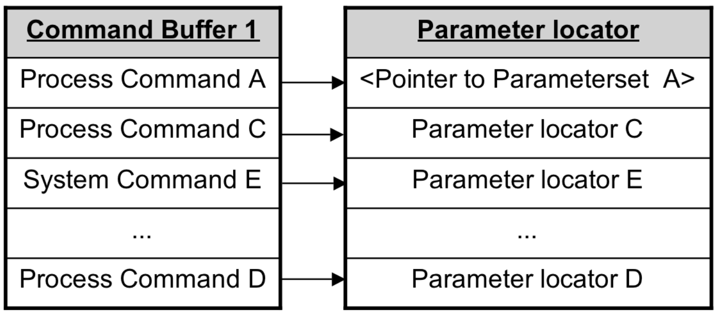
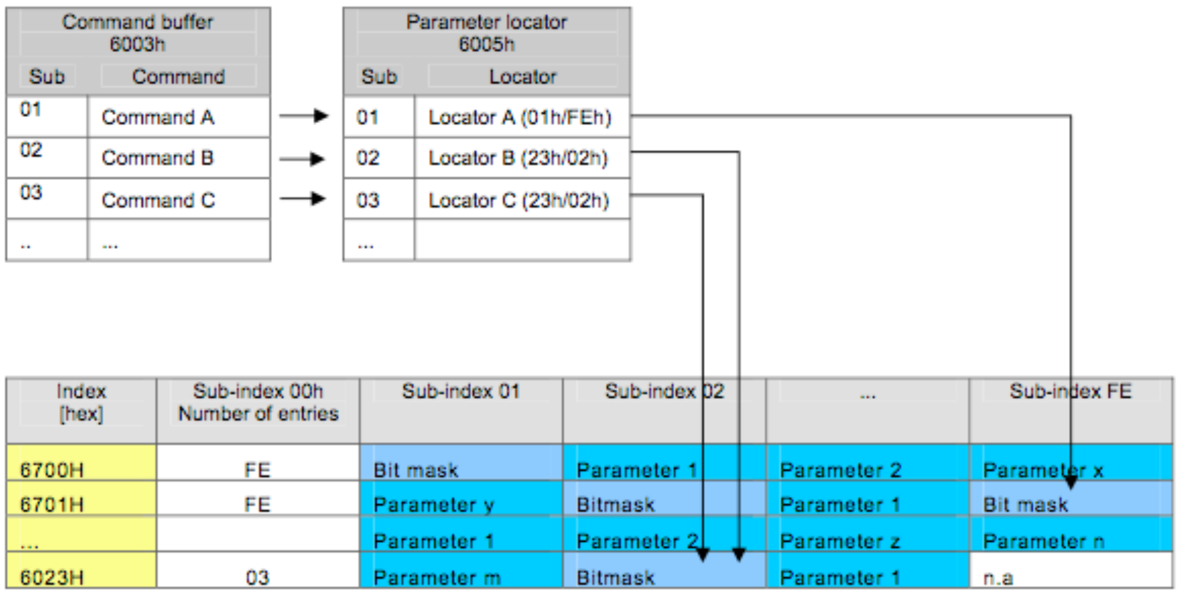


Figure 6 – Relationship between command buffer and parameter locator

The parameter locator connects the command in the command buffer with the parameters in the CPRAM. Different command buffer entries may use the same parameters if the parameter locator of these commands point to the same parameter record in CPRAM.



Command buffer 6003h	
Sub	Command
01	Command A
02	Command B
03	Command C
..	...

Parameter locator 6005h	
Sub	Locator
01	Locator A (01h/FEh)
02	Locator B (23h/02h)
03	Locator C (23h/02h)
...	

Index [hex]	Sub-index 00h Number of entries	Sub-index 01	Sub-index 02	...	Sub-index FE
6700H	FE	Bit mask	Parameter 1	Parameter 2	Parameter x
6701H	FE	Parameter y	Bitmask	Parameter 1	Bit mask
...		Parameter 1	Parameter 2	Parameter z	Parameter n
6023H	03	Parameter m	Bitmask	Parameter 1	n.a

Figure 7 – CPRAM usage

As illustrated in Figure 7, the parameter locators are used as pointer to the CPRAM, in order to indicate the command parameters. These parameters need to be updated prior to command execution. By means of Command B and C an example is illustrated, that different commands may use the same parameter set, because parameter locator B and parameter locator C point to the same starting address (bitmask) in CPRAM.

6.3 Command processing in batch mode

The LAS' command processor is responsible for command processing. The procedure described in this clause is a rough guide how command processing in batch mode should be implemented.

1) Read the actual command from the currently processed command buffer (CB1: object 6003h; CB2: object 6004h). Sub-index for starting is indicated in related FSA command.

2) Check if the “bm bit” in the command word is set. The “bm bit” indicates whether a bitmask is provided or not resp. whether command-related parameters need to be updated prior to command execution. These parameters are available as raw data in the CPRAM. In case there is no “bm bit” provided (value equal to 0), continue the command processing procedure at step 11.

3) Evaluate where in the CPRAM the required command parameters are provided. Therefore read that entry of the parameter locator, which is provided in the very same sub-index than the related command in the command buffer (see Figure 6).

4) Calculate the CPRAM index and sub-index from the parameter locator values.

5) Read the bitmask from CPRAM location. The location of the bitmask is known by the parameter locator (see step 4).

6) Evaluate whether the bitmask is valid

7) Evaluate bit 15 of the bitmask. In case bit 15 is equal to 1, the following 16 bits provide a further bitmask (This additional bit mask may indicate up to further 15 command-related parameters). Read this additional bitmask in the following CPRAM

location and return to step 6. In case bit 15 of the bitmask is equal to 0 continue with step 8

8) Cycle through the parameter valid flags field of all provided bitmasks. In case a parameter flag is valid (value equal to 1) continue at step 9, otherwise continue at step10.

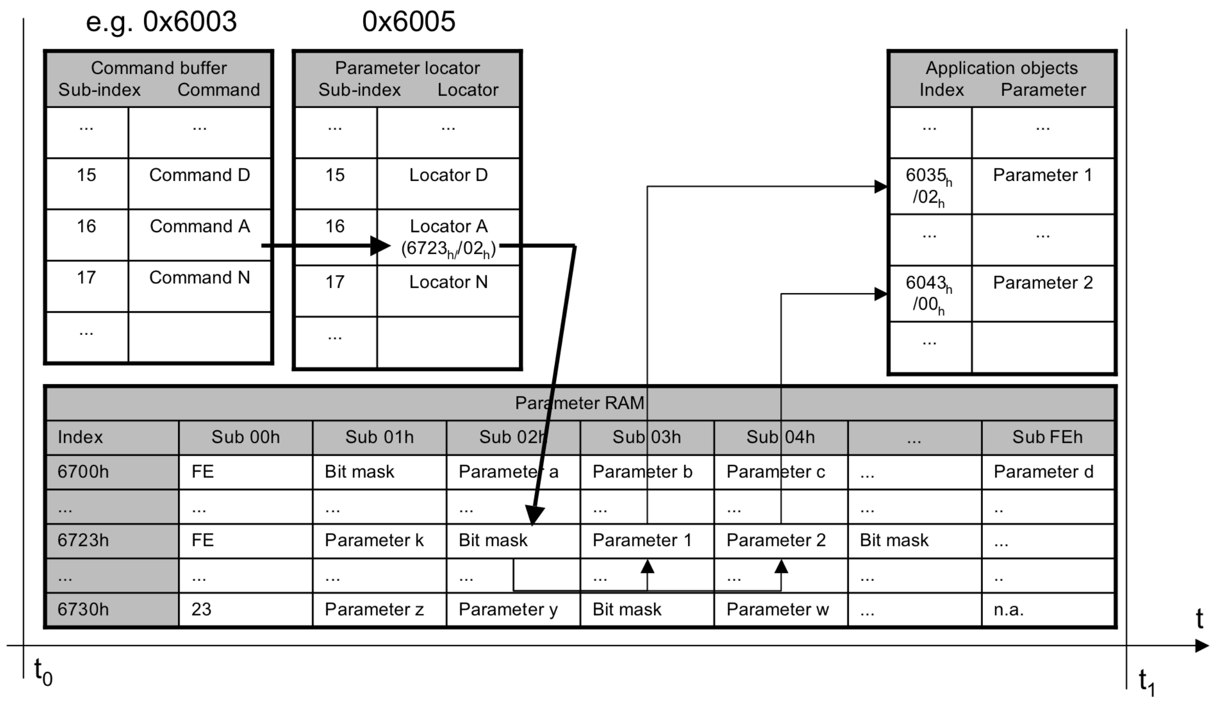
9) Copy the parameter value from the parameter location in CPRAM to the application object according to the command structure for the related command (see Figure 8).

10) Check whether there is a remaining parameter valid flag in one of the provided bitmasks, which was not evaluated yet. In case there are further remaining flags to check continue with step 8, otherwise continue with step 11.

11) All command-related parameters are updated resp. own the correct pre-configured value. Start command execution.

12) As soon as processing of this single command has finished, update command result filed, current- and next operation index in status word on transition from FSA state operating to FSA state terminated.

The update of the command parameters respectively the transfer of the raw parameters from CPRAM to the application objects, prior to the execution of the command, is illustrated in Figure 8.



Command buffer	
Sub-index	Command
...	...
15	Command D
16	Command A
17	Command N
...	

Parameter locator	
Sub-index	Locator
...	...
15	Locator D
16	Locator A (6723_h/02_h)
17	Locator N
...	

Application objects	
Index	Parameter
...	...
6035_h /02_h	Parameter 1
...	...
6043_h /00_h	Parameter 2
...	

Parameter RAM							
Index	Sub 00h	Sub 01h	Sub 02h	Sub 03h	Sub 04h	...	Sub FEh
6700h	FE	Bit mask	Parameter a	Parameter b	Parameter c	...	Parameter d
...	...	...	...	...	...	...	..
6723h	FE	Parameter k	Bit mask	Parameter 1	Parameter 2	Bit mask	...
...	...	...	...	...	...	...	..
6730h	23	Parameter z	Parameter y	Bit mask	Parameter w	...	n.a.

t0: Enter Operating; sub-index 16_h of command buffer provides next command
t1: Command parameters updated; Command A execution is started

Figure 8 – Example of command parameter update from CPRAM to application object prior to command execution

6.4 Command parameter RAM (CPRAM) handling

6.4.1 General

This sub-clause describes the handling and dangers of CPRAM handling. The CPRAM is required for LAS batch mode operation. It avoids a permanent command parameter update (e.g. via SDO) prior to command execution and enables an autonomous LAS operation.

For these purposes a memory area within the LAS is specified, which offers the LAM the possibility to download all required command parameters in raw format. Therefore the LAS is enabled to update the related command parameters autonomously prior to command execution, by copying the related information from this memory area (CPRAM) to the related object dictionary entries.

As a LAM may have read as well as write access to this memory area, this area is called "command parameter random access memory" (CPRAM). Nevertheless a LAS manufacturer may restrict the access type of the CPRAM to "read only".

6.4.2 General CPRAM structure

The command parameter area is in the LAS' device object dictionary index range from 6700h to 67FEh and it is divided in two parts:

1) 6700h to 677Eh: Exclusive CPRAM for command buffer 1 (mandatory if batch mode supported and several different parameter sets are required for the same command)

2) 6780h to 67FEh: Exclusive CPRAM for command buffer 2 (mandatory if batch mode as well as command buffer 2 are supported)

Take care that the single CPRAM is divided in fields with the size of 32 bit. The size of one CPRAM field does not change, even if the parameter to be stored in this field is smaller than 32 bit. The smaller parameter will be stored in one CPRAM field and will be right aligned!

6.4.3 Mechanisms for filling and editing CPRAM

6.4.3.1 General

There exist several mechanisms for filling the CPRAM. The different types of filling are explained in this clause. In addition the major pros and cons of each mechanism are listed.

In case the single sub-indices of CPRAM own the access type "read write", the LAS has to check permanently the consistency of the CPRAM.

6.4.3.2 Direct SDO access

The easiest way of filling and editing the CPRAM is by direct SDO access to the desired CPRAM field (sub-index). On one hand the direct access requires only minimum processing capability on the LAS. On the other hand configuring the CPRAM by single SDO access may block the bus for a long time and is much slower than transmitting all CPRAM information via one single SDO transfer. In addition a local consistency check of the CPRAM may become difficult.

6.4.3.3 Generic software download

In case the LAS provides the program download capability, which is specified in /CiA302/, the LAM may download the entire LAS application program to any sub-index of object $1F50_h$. Via SDO write access to object 1F51h the LAS application program can be initiated.

“Initiation of the program” means for a LAS application program that the LAS updates command buffer, CPRAM and parameter locator. The LAS application program will be ready for processing. Processing will start as soon as the related FSA command is received.

A software download speeds up the transfer of the data enormously as only one single SDO transfer is required. The usage of SDO block transfer could speed up the transfer further. On the other hand, higher resources and computing capabilities are required at the LAS. In case LAS supports both command buffers and the software download capability is used according /CiA302/, it is not specified how to distinguish between the two command buffers and CPRAMs. This is open to the device manufacturer. In addition the object 1F50h is an object, which was introduced for any kind of programs. This means a LAS application program may be managed in the same object where complete device firmwares may be stored.

6.4.3.4 LAS application program management

The object 6008h provides the LAM capabilities for CPRAM as well as command buffer update similar to the program download objects introduced by /CiA302/. But the functionality offered by LAS application program management is tailored much more to the LAS requirements.

Up to 254 different Las application programs can be downloaded to the object 6008h. Via object 6009h, LAS application program update, one of the downloaded programs can be transferred to either command buffer 1 or 2. This transfer is just initiated by the LAM via an expedited SDO transfer. The LAS is then updating the related command buffer as well as CPRAM autonomously. Via read access to object 6009h the LAM gets the information, which program is currently loaded and what is the status of the program. An illustration of this mechanism is already provided in /CiA434/.

As for generic software download, this type of handling CPRAM speeds up the transfer of the data enormously as only one single SDO transfer is required. The usage of SDO block transfer could speed up the transfer further. On the other hand, higher resources and computing capabilities are required at the LAS.

In contrast to the generic software download the LAS application program management provides the LAM with standardized application program control capabilities. Any program may be transferred to any supported command buffer and CPRAM. Object 6009h provides the program transfer status.

6.4.4 Ways of filling CPRAM

There are no rules provided in /CiA434/ on what is the best way of filling the CPRAM. The only provided rule is that the parameters for one command follow directly in the higher sub-indices after the related bitmask.

In case there are only few memory resources on the LAS’ microcontroller and every supported field (sub-index) of the CPRAM points directly to a real memory address, it could be wise to fill the sub-indices of CPRAM in a consecutive way. This would mean that you start with e.g. object 6700h/01h for the bitmask of the first parameter set, store the parameters in sub-indices 02h and higher. Subsequently bitmasks and parameter sets for further commands follow. As soon as all sub-indices of object 6700h are filled (at most 254), the next parameter will be stored to object 6701h/01h. An example of such a way of filling CPRAM is illustrated in Figure 9.

Index [hex]	Sub-index 00h Number of entries	Sub-index 01h	Sub-index 02h	...	Sub-index FEh
6700	FE	Bit mask	Parameter 1	Parameter 2	Parameter x
6701	FE	Parameter y	Bit mask	Parameter 1	Bit mask
...		Parameter 1	Parameter 2	Parameter z	Parameter n
6723	03	Parameter m	Bit mask	Parameter 1	n.a.

Figure 9 – Consecutively filled CPRAM

Due to several device-specific reasons, the grouping of parameters in any other way is possible. There would be e.g. the possibility to group every parameter set in one index in the CPRAM or to leave gaps in the CPRAM.

Parameter RAM							
Index	Sub 00h	Sub 01h	Sub 02h	Sub 03h	Sub 04h	...	Sub FEh
6700h	03	Bit mask	Parameter a	Parameter b	n.a.	n.a.	n.a.
n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
6723h	02	Bit mask	Parameter c	n.a.	n.a.	n.a.	n.a.
n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.
6730h	23	Parameter z	Parameter y	Bit mask	Parameter w	...	n.a.

Figure 10 – Grouping of parameter sets in CPRAM

Nevertheless it is important to take care, that all parameters of one parameter set are provided in a consecutive way.

7 Handling of operation commands in direct execution mode

7.1 General

This clause provides the general principles for handling operation commands in direct mode. The LAM transfers command by command to the LAS. Directly after the entire reception of one command the command is executed. In case some command-related parameters need to be updated prior to command execution, the LAM may hand over that command encoded as command structure. A command structure covers in addition to the command a bitmask and the parameter to be updated.

Storage of entire application programs and a monitored, autonomous program processing is not in the scope of this mode. However pre-configuring manufacturer-specific, comprehensive macros within the LAS, and triggering execution of these macros via direct command execution is possible.

7.2 Handling of operation commands

The LAS' command processor is responsible for command processing. The procedure described in this clause is a rough guide on how command processing in direct execution mode should be implemented.

1) The entire command structure is transferred to the "command structure reception" object 6011h. Depending on the used communication object and profile, the command structure has to be transferred to the correct reception port resp. sub-index of object 6011h. Please take care, that at a certain point of time, only one single reception port

is open. All the others are closed and write PDO/SDO write access to these ports will be rejected by the LAS. The port selection is done by sub-index 01h of object 6011h.

2) Immediately after entire reception of the command structure, the LAS evaluates whether there is a bitmask supported. Check if the “bm bit” in the command word is set. The “bm bit” indicates whether a bitmask is provided or not resp. whether command-related parameters need to be updated prior to command execution. These parameters are available as raw data in the further data bytes of the command structure, following the command and the bitmask fields. In case there is no “bm bit” provided (value equal to 0), continue the command processing procedure at step 9.

3) Read the bitmask, which is provided in those 16 bits of the command structure, which follow directly to the command field.

4) Evaluate whether the bitmask is valid.

5) Evaluate bit 15 of the bitmask. In case bit 15 is equal to 1, those 16 bits of the command structure, which follow directly the first bitmask provide a further bitmask (This additional bit mask may indicate up to further 15 command-related parameters as well as further bitmask). Read this additional bitmask and continue with step 4. In case bit 15 of the bitmask is equal to 0 continue with step 6.

6) Cycle through the parameter valid flags of all provided bitmasks. In case a parameter flag is valid (value equal to 1) continue at step 7, otherwise continue at step 9.

7) Copy the parameter value from the parameter field of the command structure to the related application object in the object dictionary.

8) Check whether there is a remaining parameter valid flag in one of the provided bitmasks, which was not evaluated yet. In case there are further remaining flags to check continue with step 6, otherwise continue with step 9.

9) Check whether the command is valid which is provided in the command structure and copy the command in the object 6010h.

10) Start command execution.

11) As soon as processing of this single command is finished, update command result filed and remain in FSA direct execution.

A simplified illustration of the update of command parameters, prior to command execution is illustrated in Figure 11.

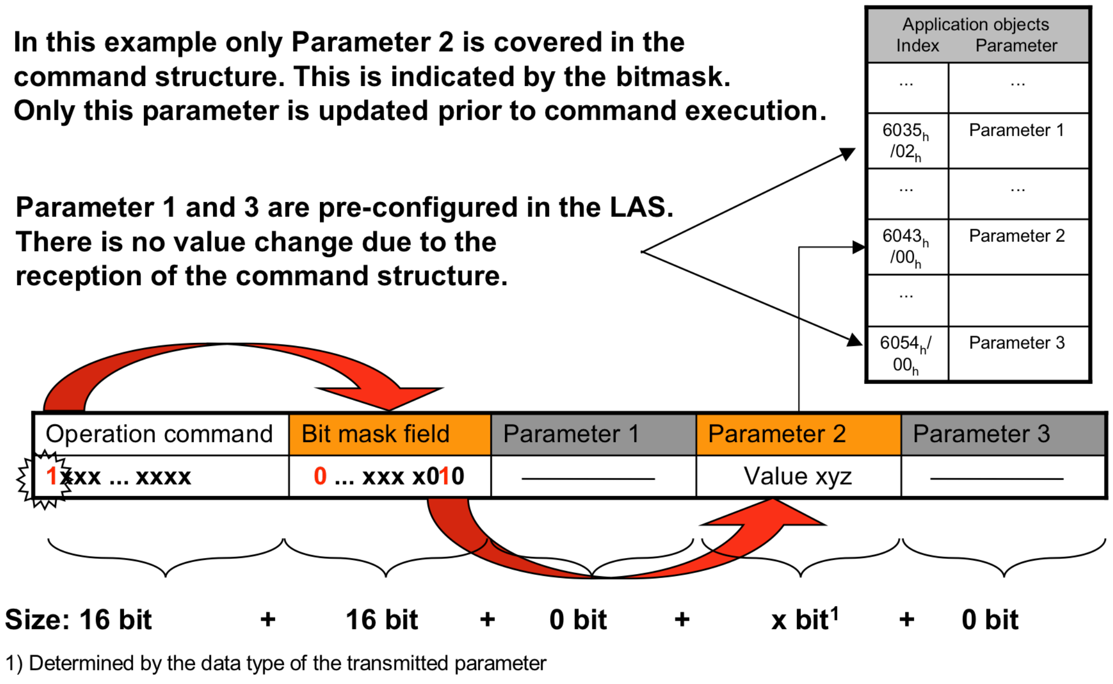

In this example only Parameter 2 is covered in the command structure. This is indicated by the bitmask. Only this parameter is updated prior to command execution.
Parameter 1 and 3 are pre-configured in the LAS. There is no value change due to the reception of the command structure.
Application objects
Index
Parameter
6035h/02h
Parameter 1
6043h/00h
Parameter 2
6054h/00h
Parameter 3
Operation command
1xxx ... xxxx
Bit mask field
0 ... xxx x010
Parameter 1
Parameter 2
Value xyz
Parameter 3
Size: 16 bit + 16 bit + 0 bit + x bit1 + 0 bit
1) Determined by the data type of the transmitted parameter

Figure 11 – Update of command parameters in direct execution mode

The general handling, which is illustrated in Figure 12, does not differ between providing a single command or an entire command structure. In case of single command operation, the command can directly be transferred to object 6010h and the command will be executed immediately.

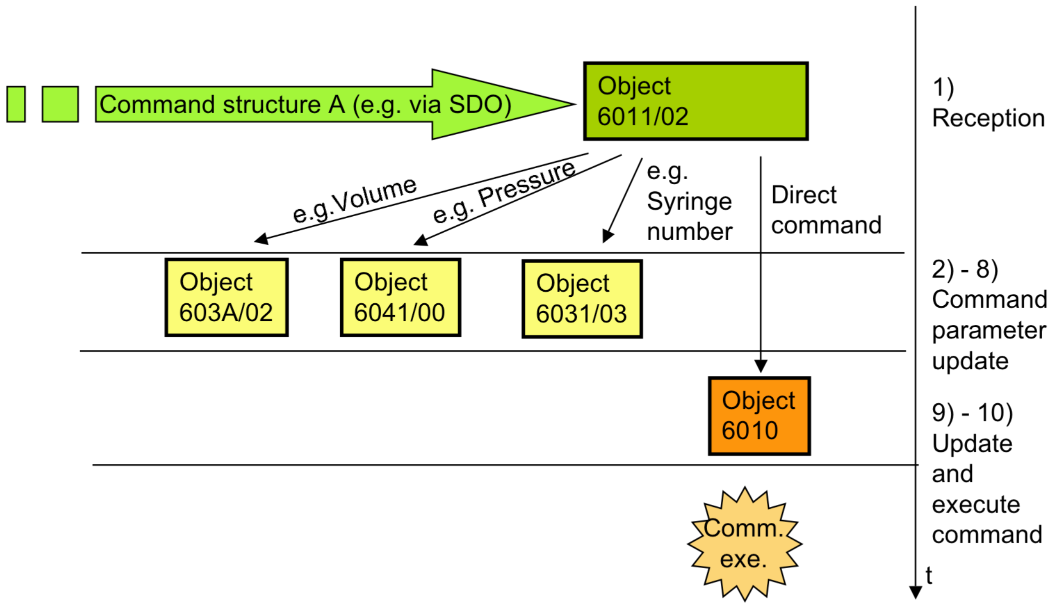


Figure 12 - Command (structure) handling in direct execution mode