

Controller Communication Toolbox User's Guide

Version 1.0

Copyright © 2004 Emmeskay, Inc.
All rights reserved.

EMMESKAY, INC.
47119 Five Mile Road
Plymouth, Michigan 48170 USA
www.emmeskay.com

Tool Support Line: cctsupport@emmeskay.com

Table of Contents

1.	Introduction.....	4
2.	Configuration	5
3.	System Requirements	6
4.	Installation instructions	6
4.1	Installation from a CD.....	6
5.	Quick Start	8
6.	Command Details.....	10
6.1	General Commands.....	13
6.1.1	Init.....	13
6.1.2	Identify.....	13
6.1.3	Emergency.....	14
6.1.4	Exit.....	14
6.2	Configuration Commands.....	15
6.2.1	SetDescriptionFile	15
6.2.2	DefDescriptionFile	16
6.2.3	CopyBinFile.....	18
6.2.4	ChangeBinFile.....	19
6.3	Parameter Manipulation Commands.....	19
6.3.1	GetParam.....	19
6.3.2	SetParam	20
6.4	Map Manipulation Commands	20
6.4.1	SelectLookUpTable	21
6.4.2	PutLookUpTable	22
6.4.3	GetLookUpTable.....	23
6.4.4	SetLookUpTableValue	24
6.4.5	GetLookUpTableValue.....	25
6.4.6	IncrLookUpTableValue	25
6.5	Recorder Commands	27
6.5.1	DefRecParam	27
6.5.2	TrigCondition	27
6.5.3	ActRecorder	29
6.5.4	GetRecStatus	29
6.5.5	LoadRecorder	30
6.5.6	SaveRecorder	31
6.6	Measurement Data Acquisition Commands	32
6.6.1	ParamForAcq	32
6.6.2	SwitchOnlineOffline.....	32
6.6.3	GetOnlineValue.....	33
6.6.4	GetUserDefinedValue.....	34
6.7	Miscellaneous Commands.....	35
6.7.1	ResetDevice.....	35
6.7.2	SetCaseSensLabels	35
6.7.3	SetGraphicMode.....	36
6.7.4	flags.....	36
6.7.5	Version.....	37

7.	Programming Techniques	37
8.	FAQs and Troubleshooting.....	40
9.	Reference	41

1. Introduction

The Controller Communication Toolbox (ASAP3 toolbox) enables communication between MATLAB (the Automation System) and the Measurement and Calibration System via serial link using the ASAP3 protocol¹. This toolbox can be used

- To implement automated design verification tests written in MATLAB for the ECU in a batch mode, programmatically.
- To perform analysis (using MATLAB's powerful analysis tools) on the calibration results obtained from the Measurement and Calibration System.

A typical use case scenario for this toolbox would be a Hardware-in-the-Loop testing of an engine control unit. The commands to the engine control unit during the test can be transmitted to the calibration unit (connected the control unit) from MATLAB using this toolbox.

The essential components for the ASAP3 interface are the **Automation System (AUSY)** and a **Measurement and Calibration System (MC System)**. The ASAP3 toolbox is designed to remotely perform calibration tasks and control Measurement and Calibration System from MATLAB[®]. The following section describes a typical application and helps understand various components of the system.

The example under consideration is a Hardware-in-the-Loop (HIL) system for Electronic Control Unit (ECU) validation. The components of a typical HIL setup are (see Figure 1):

- Real Time Platform:** The Real Time platform simulates the vehicle for which the ECU is being validated or tested.
- ECU:** The ECU is the hardware under consideration, used in the HIL system. The ECU needs to go through the testing and validation process.
- Calibration computer:** Calibration computer (Calibration Laptop in Figure 1) interfaces with the ECU and provides means of monitoring the ECU and also do online calibration changes for the ECU. This device can be used in an interactive or stand-alone mode.
- Real Time Platform interface computer:** This device (Matlab and RT interface in Figure 1) provides interface with the Real Time platform. This helps to monitor the model (running on the Real Time platform) parameters. In the current scenario this computer will also be able to act as an AUSY and interfaces with the Calibration computer, which is the MC System.

¹ ASAP3 Protocol: Currently known as ASAM MCD 3MC protocol. This protocol is developed by ASAM (Association for Standardisation of Automation- and Measuring Systems) and defines the functions & procedures required for connecting Measurement & Calibration System with Automation/Optimization system. See www.asam.net for additional information.

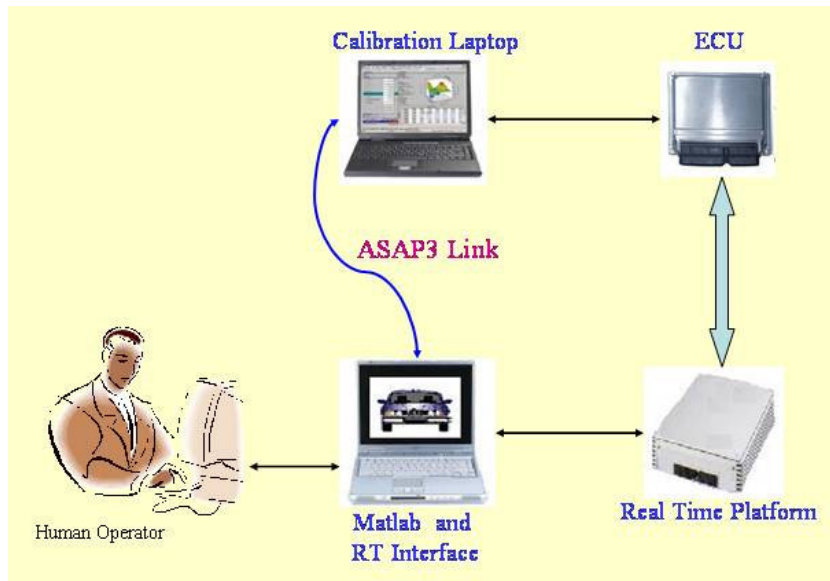


Figure 1: HIL Setup with ASAP3 Interface

Thus, in the case of ASAP3 toolbox, these components are identified as:

- a. Calibration computer: MC System
- b. Real Time Platform interface computer: Automation System

The ASAP3 protocol has been implemented for a serial port in this toolbox. Thus, standard RS232 ports are needed to establish connection between the two computers. After establishing this connection, the user would be able to issue commands over this serial link to the MC system using the ASAP3 toolbox. Following sections enable user to properly configure the system and be able to carry out the calibration using the ASAP3 toolbox.

2. Configuration

ASAP3 protocol uses standardized serial port protocol. To establish this protocol, typical configuration of the serial line can be set as:

- a. Baud Rate: 9600 (can be increased upto 115200)
 - b. Number of Bits: 8
 - c. Number of Stop Bits: 1
 - d. No parity bit
 - e. Does not support XON/XOFF
-

One of the most important facts to be remembered is that ASAP3 operates on a strictly Master-Slave principle. Thus, commands can only be issued by AUSY, which acts as the master. MC System operates as a slave device.

For the proper functioning of the ASAP3 protocol, minimum baud rate should be set to 9600. This can be increased up to 115200. To establish connection between the 2 serial ports, a [Null-Modem cable](#) is needed. Connection requirements for the [Null-Modem cable](#) are included in the [reference](#) section of this document.

3. System Requirements

This section gives a quick, but detailed look at the requirement for the using the ASAP3 toolbox. These classified as hardware requirements and software requirements.

a. Hardware requirements: Depending upon the configuration in use, these requirements change. But, in a broad way, in theory, two free standard RS232 serial ports are required. In general there would be two computers as shown in a typical HIL setup. In this configuration, it is required that each of the computers have at least one free standard RS232 serial port. In case user plans to use just one computer to interface the MC system and the AUSY, then the computer should be equipped with two free standard RS232 serial ports. A null modem cable is required to connect these serial ports. The pin out for the null modem cable is provided in the reference section of this document.

b. Software Requirements: Toolbox is designed to work with MATLAB[®] version 6.1 and above.

Operating System Requirement: Windows NT/98/2000/XP

Operating System : Windows 2000, XP

MATLAB Version : MATLAB 6.1(R12.1), 6.5(R13), 6.5.1 (R 13.1)

Memory : 128 MB RAM Minimum

Minimum Serial Ports: 1 Serial Port Minimum (2 Serial Port for Single Computer Setup)

Other Hardware Accessories: Null Modem (Serial to Serial Communication) Cable

Other Software Requirement: ECU Calibration software, with ASAP3 v1.0+ support

4. Installation instructions

4.1 Installation from a CD

1. The installation process depends on the file format sent to the user
 - a. If the CD contains all the files in an uncompressed format, copy the folders and files to the destination directory on the computer.
 - b. If the CD contains the executable (*.exe) or the compressed (*.zip) file:

- i. Copy the executable, *CntrlrComm*.exe* (or the corresponding zipped file if you had received a *.zip file), to the destination directory where you want to install the files.
 - ii. Run the executable, *CntrlrComm *.exe*, by double clicking on it. (If you had received a zipped file, then double click it to open the zipped folders. This will require the user to have access to WinZip or a similar utility. You can extract the zipped files to a destination of your choice.)
 - iii. The program would prompt you with the destination directory where the files would be installed. If you want to change the destination directory, you can do so now.
 2. Instructions for Installing the HASP® drivers
 - a. Check your user privileges. You must have administrator privileges to install HASP® drivers.
 - b. Run the *hdd32.exe* file (which is located under \driver directory) by double-clicking it in Windows Explorer.
 - c. Follow the instructions displayed on the screen.
 - d. HASP® drivers can be uninstalled from ControlPanel\Add or Remove Programs.
 3. Connect the supplied hardware dongle.
 - a. If you have been provided with a parallel port dongle, please connect this dongle to the parallel port.
 4. If you are provided with a USB dongle, please connect it to one of the available USB ports. After the files are installed, start MATLAB® and include the folder *CntrlrComm* and all the directories beneath the folder *CntrlrComm* (in your destination directory for controller communication toolbox in the previous step) into your MATLAB® path. The procedure to do this:
 - a. Open Set [Path](#) Utility from the MATLAB® window menu toolbar (see Figure 2).
 - b. Use the “Add with Subfolders” button to add the Controller Communication Toolbox folders into path. When “Add with Subfolders” button is pressed it will bring up another window titled “Browse For Folder”.
 - c. In the “Browse For Folder” window, navigate to the folder where you have installed the Controller Communication Toolbox files. Pick *CntrlrComm* and click on “OK”. This will add the folder *CntrlrComm* and all the folders below it to the MATLAB® path. This can be confirmed by looking into the MATLAB® search path shown in “Set Path” window.
-

- d. On closing the Set Path window, click on 'Yes' to permanently include the new file paths in your MATLAB® path (otherwise you have to repeat this procedure at the beginning of every MATLAB session).

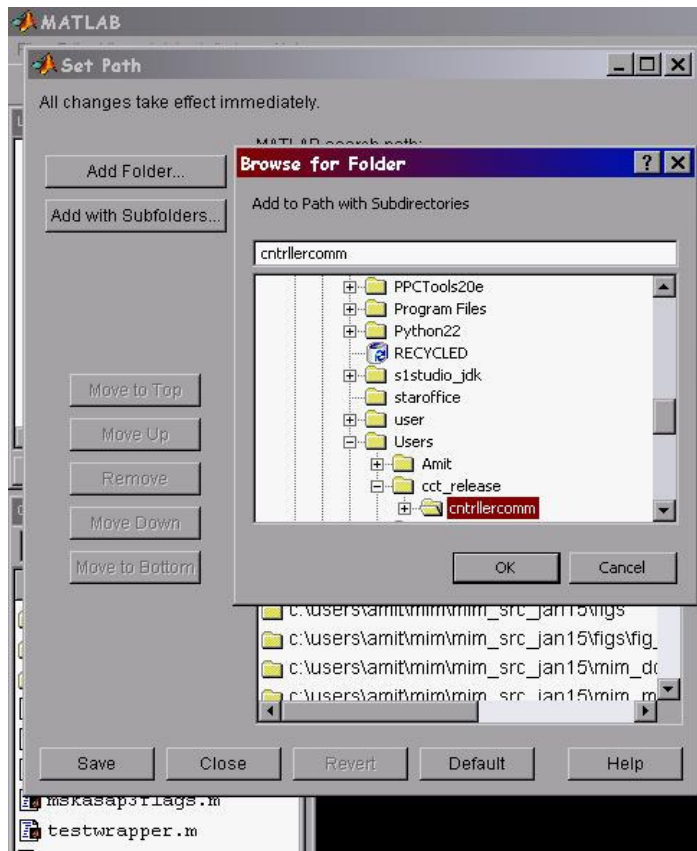


Figure 2: Path Browser

5. Quick Start

The section aims to give the user a quick start to use the toolbox.

1. Connect the null modem cable between the two serial ports. Depending upon the configuration user intends to use, the two serial ports can be on two different computers or on the same computer.
2. The assumption made is that the calibration software is running on the computer connected to the ECU and it supports ASAP3 v1.0+. Also, it is the responsibility of the user to make sure that the proper serial port on this computer is configured for the ASAP3 interface. Please refer to the help provided by the calibration software or contact the technical assistance of the company that developed the calibration software, if you have any questions on configuring the calibration software so as to get ASAP3 interface working.

-
3. Open MATLAB and include the required ASAP3 toolbox directories in path. At the command prompt, type:

```
>> s = serial('COM1');
```

An assumption made here is that COM1 port is used for the communication purpose. If the user is using any other port, the command argument needs to be changed accordingly. The command returns the serial object. This needs to be stored and passed to all the commands for ASAP3 toolbox. Open the serial port for communication:

```
>> fopen(s);
```

Initialize the ASAP3 protocol between the two computers:

```
>> mskasap3cmd(s, 'Init');
```

<p>Note: If the execution of any of the commands results in errors, the toolbox would report the error and the cause. All this information would be available to the user in the MATLAB command prompt.</p>
--

```
>> mskasap3cmd(s, 'Identify');
```

```
>> DataFlg = 1; % display the incoming and outgoing data streams
```

```
>> FileFlg = 1; % create a log file
```

```
>> mskasap3cmd(s, 'flags', DataFlg, FileFlg);
```

Complete the initialization process by issuing this command:

Depending upon calibration software, user needs to identify the Description file and the Binary file.

```
>> LUN = mskasap3cmd(s, 'SetDescriptionFile', '<DescFileName>', '<BinFile>')
```

User can now start changing the parameters. User will need to know the parameter labels and the look-up table labels. To illustrate use of the toolbox commands, the following assumptions are made:

Variable label: OUTSIDE_TEMP

Parameter label: REFERENCE_VOLTAGE

Look up Table label: TEMPSSENS_RTTABLE

```
>> TempVal = mskasap3cmd(s, 'GetParam', 'OUTSIDE_TEMP', LUN);
```

If the command executes without errors, then current value of OUTSIDE_TEMP would be returned in TempVal.

```
>> mskasap3cmd(s, 'SetParam', 'REFERENCE_VOLTAGE', 6.5, LUN);
```

The command enables user to change the parameter value. The values of only those parameters which are calibratable can be changed.

```
>> [MInd, X, Y] = mskasap3cmd(s, 'TEMPSENS_RTTABLE', LUN);
```

The command causes MC system to return the information for the specified look up table. Rest of the look up table can be experimented in a similar way.

```
>> mskasap3cmd(s, 'Exit');
```

In order to stop the using the toolbox, clean up is needed. The above command should allow the clean up and free the serial port for other applications.

6. Command Details

NOTE: The following section discusses the ASAP3 command supported by the Controller Communication Toolbox. User should note that the MC system being used may not support all the commands listed here. To find out about the commands supported by the MC system in use, please check with the MC system developer. In case a particular command is not support by the MC system, the MC system may return error codes, but in some extreme condition, the command may cause MC system to close.

Following section covers each of the commands in detail. The discussion concentrates on explaining each command, the syntax and example. The discussion groups the commands according to the functionality. Later, in the next section a brief outline of the command sequence and other programming techniques are discussed.

The toolbox offers a standard and easy to use interface for all the command of the toolbox. A typical command has following syntax:

mskasap3cmd(SrObj, '<Command String>', arg1, arg2,...argN)

This syntax is discussed below.

mskasap3cmd acts as a gateway function to all the commands which are supported by the toolbox.

SrObj: Since ASAP3 implementation is done via a simple serial port, the command needs the Serial Object for execution.

<Command String>: This is the actual command that needs to be executed. Details on each available command are included in the following sections of the document.

arg1, arg2,..., argN: Depending upon the type of the ASAP3 command, the tool command might expect some input arguments. Some of the commands don't need any of these arguments.

The inputs SrlObj and the Command String are required for all the commands while the input arguments arg1, arg2 and argN are optional (depend on the type of ASAP3 command). Also, some of the commands have output arguments. All the commands are discussed in detail in the following sections. Please refer to the individual commands for the return arguments.

The following table covers a bird's eye view at the various commands available.

Section Number	Command	Command Function Name	Code (hex)	Input arguments Required	Output arguments available
General Commands					
6.1.1	Initialize	Init	02	No	No
6.1.2	Identify	Identify	14	Yes	No
6.1.3	Emergency	Emergency	01	Yes	No
6.1.4	Exit	Exit	32	No	No
Configuration Commands					
6.2.1	Select Description File and Binary File	SetDescriptionFile	03	Yes	Yes
6.2.2	Define Description File and Binary File	DefDescriptionFile	1E	Yes	Yes
6.2.3	Copy Binary File	CopyBinFile	04	Yes	No
6.2.4	Change Binary File Name	ChangeBinFile	05	Yes	No
Parameter Manipulation Commands					
6.3.1	Get Parameter	GetParam	0E	Yes	Yes
6.3.2	Set Parameter	SetParam	0F	Yes	No
Map Manipulation Commands					
6.4.1	Select Look-up Table	SelectLookUpTable	06	Yes	Yes
6.4.2	Put Look-up Table	PutLookUpTable	07	Yes	No
6.4.3	Get Look-up	GetLookUpTable	08	Yes	Yes

Deleted: 1

Deleted: 3

	Table				
6.4.4	Set Look-up Table	SetLookUpTableValue	0B	Yes	No
6.4.5	Get Look-up Table Value	GetLookUpTableValue	09	Yes	Yes
6.4.6	Increase Look-up Table	IncreaseLookUpTable	0A	Yes	No
<u>Recorder Parameter Commands</u>					
6.5.1	Define Recorder Parameters	DefRecParam	29	Yes	No
6.5.2	Define Trigger Condition	TrigCondition	2A	Yes	No
6.5.3	Activate Recorder	ActRecorder	2B	Yes	No
6.5.4	Get Recorder Status	GetRecStatus	2C	No	Yes
6.5.5	Load Recorder File	LoadRecorder	30	Yes	Yes
6.5.6	Save Recorder File	SaveRecorder	2F	Yes	No
<u>Measurement Data Acquisition Commands</u>					
6.6.1	Parameter for Value Acquisition	ParamForAcq	0C	Yes	No
6.6.2	Switching Offline/Online	SwitchOnlineOffline	0D	Yes	No
6.6.3	Get Online Value	GetOnlineValue	13	Yes	Yes
6.6.4	Get User Defined Value	GetUserDefinedValue	15	Yes	Yes
<u>Miscellaneous Commands</u>					
6.7.1	Reset Device	ResetDevice	11	Yes	No
6.7.2	Set Case Sensitive Labels	SetCaseSensLabels	3D	No	No
6.7.2	Set Graphic Mode	SetGraphicMode	10	Yes	No
6.7.3	Flag	flags	-	Yes	No
6.7.4	Version	Version	-	No	Yes

6.1 General Commands

6.1.1 Init

The INIT command initializes the communication link between the MC System and the AUSY system. This is the first command that needs to be sent. The MC System will do all the initializations needed.

Command syntax:

```
mskasap3cmd(s1, 'Init');
```

Arguments passed:

s1	Serial Port Object
Init	Command String

The response to this command results in information about the success of the command. If the command executes with some errors, then corresponding error codes would be sent back. Toolbox will output the error information.

6.1.2 Identify

The IDENTIFY command is issued after successful execution of INIT command. The command sends the version number of the ASAP3 protocol from the AUSY system and the name of the AUSY. The response to the command results in version number of the ASAP3 protocol on the MC System and the name of the MC System.

Command syntax:

```
[Ver, Sys] = mskasap3cmd(s1, 'Identify', 'Matlab');
```

Arguments passed:

s1	Serial Port Object
Identify	Command String
Matlab	Name of the AUSY

The response from the MC System results in Name (Sys) of the MC System and the Version number (Ver) of the ASAP3 protocol.

In case the response received has error code, then it is assumed that the protocol version V1.x is implemented on the MC System.

In case this command is not issued, then MC System assumes that protocol version V1.x is implemented on the AUSY.

6.1.3 Emergency

The Emergency (EMERGENCY) command may be sent to the MC system during active communication in an emergency situation. ECU-specific emergency reactions may be initiated on the MC system.

Command syntax:

```
mksasap3cmd(SrlObj, 'Emergency', Event);
```

Arguments passed:

s1	Serial Port Object
Emergency	Command String
Event	The value allocated to the event. Event is the ECU specific emergency reactions

The ECU specific events will be defined by MC system. A definition of such a situation, the value allocated to the 'Event' and how it must be dealt with, depends on the actual realisation of the MC system.

6.1.4 Exit

The EXIT command is the last command that needs to be sent to the MC System. After receipt of this command from the AUSY, MC System would free all the resources allocated for the protocol.

Command syntax:

```
mksasap3cmd(s1, 'Exit');
```

Arguments passed:

s1	Serial Port Object
Exit	Command String

The response to this command results in information about the success of the command. If the command executes with some errors, then corresponding error codes would be sent back. Toolbox will output the error information.

If user needs to carry out some ASAP3 communication again, INIT command needs to be sent again.

6.2 Configuration Commands

6.2.1 SetDescriptionFile

The Select Description File command can be sent only after successful execution of INIT command. The command requires two file names to be passed as inputs.

Command syntax:

```
[LUN] = mskasap3cmd(s1, 'SetDescriptionFile', 'File1', 'File2', Dest);
```

Arguments passed:

s1	Serial Port Object
SetDescriptionFile	Command String
File1	Description File Name and Path string
File2	Binary File Name and Path string
Dest	Destination Number

The following section covers each of these input arguments in details.

Description File Name and Path string: This is the description file name (eg. ASAP2 file)

Binary File Name and Path string: This is the program code file name and its location string.

Destination: Destination specified either by the AUSY or the MC System.

The command helps in defining the description and the calibration files for the system. These two files store the information such as memory address of various parameters and look up table.

The last argument is the Destination number. Based on the value provided, action would be carried out.

Destination 0: Destination is automatically defined by the MC System

Destination 1, 2...: Final allocation to destinations 1, 2... by AUSY.

This command is important as it returns a Logical Unit Number (LUN). The Binary File and the Description file are together related to this LUN.

Example:

Description File Path and Name: 'C:\DescFile.dsc'

Binary File Path and Name: 'C:\BinFile.bin'

File1 = 'C:\DescFile.dsc'

File2 = 'C:\BinFile.bin'

Dest: 0 (Automatically defined by MC System.

[LUN] = mskasap3cmd(s1, 'SetDescriptionFile', File1, File2, Dest);

Note: The file extensions in the example are just for illustrative purposes. Based upon the calibration software the extension would change as specified by the vendor.

6.2.2 DefDescriptionFile

The Define Description File (*DefDescriptionFile*) command can be sent only after successful execution of INIT command. This command can be used as an alternate command to the Select Description File command. The command requires three file names to be passed as inputs.

Command syntax:

[LUN, F1, F2, F3] = mskasap3cmd(s1, 'DefDescriptionFile', 'File1', 'File2', 'File3', Dest, Mode);

Arguments passed:

s1	Serial Port Object
DefDescriptionFile	Command String
File1	Description File Name and Path string
File2	Program Code (+Calibration data) File Name and Path string
File3	Calibration File Name
Dest	Destination Number
Mode	Mode

The following section covers each of these input arguments in details.

Description File Name and Path string: This is the description file name (eg. ASAP2 file)

Program Code (+Calibration data) files name string: This is the location and name of the program code file. Calibration data can also be specified. This is used only if the Calibration file name (next argument for this command) is a blank string.

This argument can be a blank string, if the Mode is 0 or 2.

Calibration File Name: In case of Mode = 0, this argument can be set to a blank string. In any other case where mode is > 0, and a blank string for this argument, MC system will use the Calibration data provided from the previous argument.

Destination number: The possible options are:

0: Destination is automatically defined by MC System

- 1: Parallel EPROM Emulation
- 2: CAN-Bus
- 3: K-Line

Mode: The mode also has 4 possible options. If the mode is set to

0: There won't be any download to the destination memory. The system would only search for this type of already existing configuration. The result of the finding would be reported back. This data is discussed below.

1: This mode results in download of the program code and the calibration data to the destination memory.

2: This mode results in download of the calibration data to the destination memory.

3: In this mode the ECU program code would be downloaded to the destination only if the MC system recognizes that there is difference between program code of the destination memory and the desired configuration.

Response from the MC System:

LUN	Emulator LUN
F1	Description File Name
F2	Program Code (+Calibration data) File Name
F3	Calibration File Name

This information is returned back by the MC System, when the Mode is set to 0. The most important data returned is the Emulator LUN, similar to *SetDescriptionFile* command.

This command can be used as an alternative command to *SetDescriptionFile* command, but can only be sent after successful execution of *INIT* command.

Example:

Description File Path and Name: 'C:\DescFile.dsc'

Binary File Path and Name: 'C:\BinFile.bin'

File1 = 'C:\DescFile.dsc'

File3 = 'C:\CalFile.cal'

Dest: 0 (Automatically defined by MC System)

Mode: 0

[LUN] = mskasap3cmd(s1, 'DefDescriptionFile', File1, '', File3, Dest, Mode);

Note: The file extensions in the example are just for illustration purpose. Based on the calibration software the extension would change as specified by the vendor.

Note: By default, if *SetDescriptionFile* or *DefDescriptionFile* commands aren't used, the LUN is 0.

6.2.3 CopyBinFile

The CopyBinFile command (Copy Binary File) can be sent only after Emulator LUN is determined using *SetDescriptionFile* command and *DefineDescriptionFile* command

Command syntax:

```
mksasap3cmd(s1,'CopyBinFile',<Source>,<Target>,LUN);
```

Arguments passed:

s1	Serial Port Object
CopyBinFile	Command String
Source	BinaryFileSource
Target	BinaryFileTarget

This command is used to save the current calibration binary data file in different location particularly in the hard disk. This command transfers the current binary file from the specified source to the specified target of a particular EmulatorLUN. This command is used to save a copy of the current ECU setting in the ECU and the saved file can be reloaded to ECU using this same command.

Source: The source from which the binary data to be copied

Target: The target to which the binary data to be copied

The possible source and Target are

1. EPROM
2. File (Harddisk)
3. Virtual Emulator Board – Calibration Software internal database
4. Physical Emulation board – Emulator memory in ECU.

The combination of source and target is depends on the Calibration software. The saved date may overwrite the existing data in the specified target with or without Error messages or warnings or may not overwrite the data depends on the Calibration software implementation.

Example: `mksasap3cmd(s1,'CopyBinFile',3,2,LUN);`

6.2.3 ChangeBinFile

The ChangeBinFile command (Change Binary File) can be sent only after Emulator LUN is determined using *SetDescriptionFile* command and *DefineDescriptionFile* command

Command syntax:

```
mksasap3cmd(s1,'ChangeBinFile',<Binary Filename>_LUN);
```

Arguments passed:

s1	Serial Port Object
ChangeBinFile	Command String
Binary Filename	New Binary File Name
Target	BinaryFileTarget

This command is used to change the name of the binary file of the designated Emulator for the selected ECU. A subsequent *copy binary file* will store the binary file under this new name. This allows to save modified binary file contents without losing the original data.

Example:

```
mksasap3cmd(s1, 'ChangeBinFile', 'BinFile.Bin' LUN);
```

Note: The file extensions in the example are just for illustration purpose. Based upon the calibration software the extension would change as specified by the vendor.

6.3 Parameter Manipulation Commands

6.3.1 GetParam

The GetParam command can be sent only after successful execution of *INIT* command.

Command syntax:

```
[Val, MinVal, MaxVal, MinInc] = mksasap3cmd(s1, 'GetParam', <ParLabel>);
```

Arguments passed:

s1	Serial Port Object
GetParam	Command String
ParLabel	Parameter Name String

The response to this command results in information about the success of the command. If the command executes with some errors, then corresponding error codes would be sent back. Toolbox will output the error information.

The response from the MC system is:

Val	Value of the parameter
MinVal	Minimum value of the parameter
MaxVal	Maximum value of the parameter
MinInc	Minimum increment

Example:

```
[Val, MinVal, MaxVal, MinInc] = mskasap3cmd(s1, 'GetParam', 'Param1');
```

6.3.2 SetParam

The SetParam command can be sent only after successful execution of *INIT* command.

Command syntax:

```
mskasap3cmd(s1, 'SetParam', <ParamLabel>, ParamVal);
```

Arguments passed:

S1	Serial Port Object
SetParam	Command String
ParamLabel	Parameter Name String
ParamVal	New value of the parameter

The response to this command results in information about the success of the command. If the command executes with some errors, then corresponding error codes would be sent back. Toolbox will output the error information.

Val would contain the value of the parameter.

Example:

```
mskasap3cmd(s1, 'SetParam', 'Param1', 5.48);
```

6.4 Map Manipulation Commands

For this section, these look-up tables would be used for explanation purpose.

2D look-up table: LkUpTbl2D

X Value	Y Value
3	0
5	7
15	10

3D look-up table: LkUpTbl3D

X \ Y	0	5	10
3	3	6	9
5	4	8	12
15	5	10	15

6.4.1 SelectLookUpTable

The command can be executed only after successful execution of the *SetDescriptionFile* or *DefDescriptionFile* commands. This command requires LUN obtained from these commands.

Command syntax:

```
[No, XIdx, YIdx, MapAddr] = mskasap3cmd(s1, 'SelectLookUpTable',...
                                     '<MapLabel>', LUN);
```

Arguments passed:

s1	Serial Port Object
SelectLookUpTable	Command String
MapLabel	Parameter Name String

The two arguments required to process this command are the MapLabel and the LUN.

MapLabel is the string for name of the Map

LUN is the Emulator LUN returned by *SetDescriptionFile* or *DefDescriptionFile* command

Upon successful execution of the command, the MC system responds with following information

MapNo	Map Number (for reference)
YIdx	Dimension in Y direction
XIdx	Dimension in X direction
MapAddr	Map Address

The MapNo is the reference number that would be used to refer to a particular Look-up table. This number is generated by combining the Emulator LUN and the look-up table name.

XIdx: Dimension in X direction

YIdx: Dimension in Y direction

MapAddr: Map Address

These two indices are used to identify if the referred look-up table is a 2D or 3D. In case of a 2D look-up table, XIdx would be greater than 1, where as YIdx would be equal to 1.

In case of a 3D look-up table, both XIdx and YIdx would be greater than 1.

Example:

For 2D Table:

```
[No, XIdx, YIdx, MA] = mskasap3cmd(s1, 'SelectLookUpTable',  
                                'LkUpTbl2D', LUN);
```

For 3D Table:

```
[No, XIdx, YIdx, MA] = mskasap3cmd(s1, 'SelectLookUpTable',  
                                'LkUpTbl3D', LUN);
```

6.4.2 PutLookUpTable

The command can be executed only after successful execution of the *SelectLookUpTable* command. This command requires map data obtained from this command.

Command syntax:

For 2D Table:

```
mskasap3cmd(s1, 'PutLookUpTable', MapNo, XIdx, YIdx, [XValues],  
            [YValues]);
```

For 3D Table:

```
mskasap3cmd(s1, 'PutLookUpTable', MapNo, XIdx, YIdx, [XValues],  
            [YValues], [ZValues]);
```

Arguments passed:

s1	Serial Port Object
PutLookUpTable	Command String
MapNo	Map number
Xidx	Dimension in X direction
Yidx	Dimension in Y direction
[XValues]	Values along the X direction
MinZ	Minimum Z value
MaxZ	Maximum Z value
MinInc	Minimum increment along Z direction

[YValues]	Values along the Y direction
[ZValues]	Values along Z direction (only for 3D Look-up table)

This command enables a new look up table to be loaded on the controller. In order to carry out this command, the MapNo, XIdx, YIdx, MinZ, MaxZ and MinInc data are required. This is the data that is returned by successful execution of the *SelectLookUpTable* command. Apart from these three arguments, the actual data is also sent to the MC system. The [XValues] and [YValues] are the arrays of data needed for both 2D and 3D look up tables, whereas [ZValues] are needed only for 3D look-up tables.

Example:

For 2D Table:

```
mksasap3cmd(s1, 'PutLookUpTable', MapNo, XIdx, YIdx, [3, 5, 15], MinZ,
             MaxZ, MinInc, [1, 7, 10])
```

For 3D Table:

```
mksasap3cmd(s1, 'PutLookUpTable', MapNo, YIdx, XIdx, [3, 5, 15], MinZ,
             MaxZ, MinInc [1,5,10], [3, 4, 5, 6, 8, 10, 9, 12, 15])
```

6.4.3 GetLookUpTable

The command can be executed only after successful execution of the *SelectLookUpTable* command. This command requires map data obtained from this command.

Command syntax:

```
[ML,XD,YD,ZD,MinZ,MaxZ,MinInc] = mksasap3cmd(s1, 'GetLookUpTable',
                                             MapNo);
```

Arguments passed:

s1	Serial Port Object
GetLookUpTable	Command String
MapNo	Map number

The command returns the look up table referred to by MapNo. The arguments passed to the command are the MapNo. This is the data that is returned by successful execution of the *SelectLookUpTable* command.

In case of a 2D look up table, only XVal (data along X direction) and YVal (data along Y direction) are returned, whereas for a 3D look-up table ZVal(data along Z direction) are also returned.

ML represents the Map length. Also, the ZMin is the minimum value along the Z direction. ZMax is the maximum value along the Z direction. MinInc is the minimum increment along the Z direction.

Example:

MapNo: 1 (MapNo is obtained by a previous execution of *SelectLookUpTable* command)

```
[ML, XVal, YVal, ZVal] = mskasap3cmd(s1, 'GetLookUpTable', MapNo);
```

6.4.4 SetLookUpTableValue

The command can be executed only after successful execution of the *SelectLookUpTable* command. This command requires map data obtained from this command.

Command syntax:

```
mskasap3cmd(s1, 'SetLookUpTableValue', MapNo, YInd, XInd);
```

Arguments passed:

s1	Serial Port Object
PutLookUpTable	Command String
MapNo	Map number
XInd	X Index of the look up table element
YInd	Y Index of the look up table element
Val	Value of the element

The command enables change of value for a single element pointed at by the XInd and Yind of the look-up table referred to by MapNo. The argument passed to the command is the MapNo, XInd and YInd and the Val. Argument XInd would be neglected if the look-up table referred to by MapNo is a [2D](#) look-up table.

Example:

For [2D](#) and [3D](#) look-up tables:

```
mskasap3cmd(s1, 'SetLookUpTableValue', MapNo, 2, 1, 6);
```

In case of a [2D](#) look-up table as above, the last argument, 1 is neglected.

6.4.5 GetLookUpTableValue

The command can be executed only after successful execution of the *SelectLookUpTable* command. This command requires map data obtained from this command.

Command syntax:

Val = mskasap3cmd(s1, 'GetLookUpTableValue', MapNo, YInd, XInd);

Arguments passed:

s1	Serial Port Object
GetLookUpTableValue	Command String
MapNo	Map number
XInd	X Index of the look up table element
YInd	Y Index of the look up table element

The command returns the value of the element pointed at by the XInd and YInd of the look-up table referred to by MapNo. The argument passed to the command is the MapNo, XInd and YInd. Argument XInd would be neglected if the look-up table referred to by MapNo is a 2D look-up table.

Example:

For [2D](#) and [3D](#) look-up tables:

Val = mskasap3cmd(s1, 'GetLookUpTableValue', MapNo, 2, 1);

In case of a [2D](#) look-up table as above, the last argument, 1 is neglected. The value available in Val would be 7.

But in case of a [3D](#) look-up table, as above the value available in Val would be 12.

6.4.6 IncrLookUpTableValue

The command can be executed only after successful execution of the *SelectLookUpTable* command. This command requires map data obtained from this command.

Command syntax:

mskasap3cmd(s1, 'IncrLookUpTableValue', MapNo, yIdStart, xIdStart,...
 yDelta, xDelta, Offset);

Arguments passed:

s1	Serial Port Object
IncrLookUpTableValue	Command String
MapNo	Map number

yIdStart	Starting Y Index of the look up table element
xIdStart	Starting X Index of the look up table element
yDelta	Number of elements in Y direction to be offset
xDelta	Number of elements in X direction to be offset
Offset	Actual offset value

The command returns the look up table referred to by MapNo. The argument passed to the command is the MapNo. This is the data that is returned by successful execution of the *SelectLookUpTable* command.

The command results in adding offset to the look-up table values within a specified range.

Amongst the other information to be passed to this command are the 2 indices, yIdStart and xIdStart and the starting indices along the Y and X directions respectively. yDelta and xDelta are the number of elements along the Y direction and X direction to which the offset is to be added. And the last argument, Offset is the actual offset value. This value would be added to all the elements in the range specified by yIdStart, xIdStart, yDelta and xDelta.

Example:

MapNo: 1

For 2D Table:

mksasap3cmd(s1, 'IncrLookUpTableValue', MapNo, 2, 1, 2, 1, 5);

The command results in adding an offset at 2nd and 3rd element of the [2D](#) look up table. So, the look up table would look like:

X Value	Y Value
3	0
5	12
15	15

For 3D Table:

mksasap3cmd(s1, 'IncrLookUpTableValue', MapNo, 2, 2, 1, 3, 7);

The command would modify the above [3D](#) look up table in following way:

X \ Y	0	5	10
3	3	13	16
5	4	15	19

15	5	17	22
----	---	----	----

6.5 Recorder Commands

6.5.1 DefRecParam

The command enables user to define recorder parameter for a data acquisition purpose on the MC system. The command can be executed only after successful execution of the *SetDescriptionFile* or *DefDescriptionFile* command. This command requires the Emulator LUN obtained from this command.

Command syntax:

```
mksasap3cmd(s1, 'DefRecParam', RecParamList);
```

Arguments passed:

s1	Serial Port Object
DefRecParam	Command String
RecParamList	Parameter Name String
LUN	Emulator LUN

The command takes in the recorder parameter list and the emulator LUN. By definition, a recorder parameter list is just a collection of the labels of those parameters which need to be recorded during a simulation at a specific scan time for the recorder. The toolbox expects the recorder parameter list as a structure that contains the parameter list and the scan time for the recorder.

The scan time is a very important parameter that needs to be passed. The value passed as input is the scan time in milliseconds.

Example:

```
RecParamList.Name = ('Param1', 'Param2');
RecParamList.scanTime = 100;
mksasap3cmd(s1, 'DefRecParam', RecParamList);
```

Upon successful execution of the command, a recorder would be defined. It will enable user to log data for the two specified parameters. Data will be logged at the 100ms.

6.5.2 TrigCondition

The command enables user to define recorder trigger conditions for the data acquisition purpose on the MC system. The command can be executed only after successful execution of the *DefRecorderParam* command.

Command syntax:

```
mskasap3cmd(s1, 'TrigCondition', 'StartTrig', 'StopTrig', MaxSampleNo,...  
            StartDelay, StopDelay );
```

Arguments passed:

s1	Serial Port Object
TrigCondition	Command String
StartTrig	Start trigger transition parameter label
StopTrig	Stop trigger transition parameter label
MaxSampleNo	Maximum number of samples
StartDelay	Start delay time value
StopDelay	Stop delay time value

The command takes various parameters as inputs that are required to completely define activation of the data acquisition. All the input arguments are discussed in following section.

StartTrig: Defines the transition parameter to start the data acquisition, trigger on condition.

StopTrig: Defines the transition parameter to stop the data acquisition, trigger off condition.

MaxSampleNo: Defines maximum number of samples to be acquired before the data acquisition is stopped.

StartDelay: Defines the time delay for initiating the data acquisition, after the trigger on condition is satisfied.

StopDelay: Defines the time delay for stopping the data acquisition, after the trigger off condition is satisfied.

When StartTrig and StopTrig are empty string, the stop condition is achieved by the maximum number of samples provided. The recorder has to be commanded to start acquiring data. The recorder can be commanded to stop data acquisition or would stop data acquisition after maximum number of samples are reached. The commands for starting and stopping the recorder are discussed later in this section.

Example:

```
mskasap3cmd(s1, 'TrigCondition', 'Sig1', 'Sig2', 100000, 0, 0);
```

The above command would cause the recorder to start data acquisition on Sig1 transition and stop the data acquisition on Sig2 transition.

```
mskasap3cmd(s1, 'TrigCondition', ',', ',', 100000, 0, 0);
```

The above command would cause the recorder to be controlled by start and stop commands. The recorder would stop even if the stop command is not issued in case the maximum number of samples is reached.

6.5.3 ActRecorder

The command enables user to control the recorder start and stop conditions. The command can be executed only after successful execution of the *DefRecorder* command.

Command syntax:

```
mskasap3cmd(s1, 'ActRecorder', Cmd);
```

Arguments passed:

s1	Serial Port Object
ActRecorder	Command String
Cmd	Recorder command (On/Off/Act)

The recorder command is sent as the third argument of this command. This can take two values:

On: Recorder On

Off: Recorder Off

Act: Activate recorder

The command is used to control the recorder start and stop action programmatically.

Example:

To start the recorder:

```
mskasap3cmd(s1, 'ActRecorder', 'On');
```

To stop the recorder:

```
mskasap3cmd(s1, 'ActRecorder', 'Off');
```

6.5.4 GetRecStatus

The command enables user to access the recorder status, after the *ActRecorder* command is successfully executed.

Command syntax:

```
[Sts, CSample, StCond, StInfo] = mskasap3cmd(s1, 'GetRecStatus');
```

Arguments passed:

s1	Serial Port Object
GetRecStatus	Command String

Apart from the two standard arguments, the command does not take in any arguments. The command responds with following data from the MC system.

Sts	Recorder status
CSample	Current sample number
StCond	Stop condition
StInfo	Stop information string

Each of the above returned data is discussed here.

Sts: The recorder status code is returned by the MC system. The possible values are:

- 1: Recorder On
- 2: Recorder Off
- 3: Recorder Ready

CSample: The MC system also returns the current sample number that is being sampled.

StCond: The stop condition code is returned by MC system. The possible values are:

- 1: Manual
- 2: Trigger Stop
- 3: End of Recording
- 256: Stop due to error

StInfo: The MC system also returns more information about the stop condition in form of a string.

Example:

```
[Sts, CSample, StCond, StInfo] = mskasap3cmd(s1, 'GetRecStatus');
```

6.5.5 LoadRecorder

The command can be sent only after successful execution of *'Init'* command. The command enables user to load the recorder file by the MC system.

Command syntax:

```
[NoOfVals, Vals] = mskasap3cmd(s1, 'LoadRecorder', '<FName>');
```

Arguments passed:

s1	Serial Port Object
LoadRecorder	Command String
<FName>	Recorder file path and name

The command takes the name for the recorder file as the single input argument. The file name should be specified with its full path and extension. The command results in MC system loading the recorder file locally. Upon successful execution of the command, the MC system returns some information about the loaded recorder file. The information received is:

- NoOfVals: Number of values
- Vals: The values for the recorded data

Example:

FName: 'C:\RecFile.daq'

```
[No, Vals] = mskasap3cmd(s1, 'LoadRecorder', FName);
```

6.5.6 SaveRecorder

The command enables user to save the result of the recorder file. The file is saved locally to the MC system.

Command syntax:

```
mskasap3cmd(s1, 'SaveRecorder', '<FName>');
```

Arguments passed:

s1	Serial Port Object
SaveRecorder	Command String
<FName>	Recorder file path and name

The command takes the name for the recorder file as the single input argument. The file name should be specified with its full path and extension.

Example:

FName: 'C:\RecFile.daq'

```
mskasap3cmd(s1, 'SaveRecorder', FName);
```

Note: The file extensions in the example are just for illustration purpose. Based upon the calibration software the extension would change as specified by the vendor.
--

6.6 Measurement Data Acquisition Commands

6.6.1 ParamForAcq

The command can be executed only after successful execution of the *SetDescriptionFile* or *DefDescriptionFile* commands. This command requires LUN obtained from these commands.

The command enables user to define a list of parameters for the data acquisition. The command not only defines the parameter list, but also the scanning rate and the sequence.

Command syntax:

```
mksasap3cmd(s1, 'ParamForAcq', ParamList, LUN);
```

Arguments passed:

s1	Serial Port Object
ParamForAcq	Command String
ParamList	Parameter list and scan time information
LUN	Emulator LUN

The arguments to be passed to the command are discussed in this section.

ParamList: This defines the parameter list and also the scan time. The scan time can take any value from 500 ms to 100000ms.

Along with this data, the Emulator LUN has to be passed as an argument.

If this command is repeatedly sent, the acquisition lists are concatenated in their access sequence. But, the scan time parameter is taken from the last time the command was issued.

Example:

```
ParamList.Name = { 'Param1', 'Param2', ..., 'ParamN' }
```

```
ParamList.ScanTime = 500 (in ms)
```

```
mksasap3cmd(s1, 'ParamForAcq', ParamList, LUN);
```

Note: Please contact Emmeskay Inc. if you have problems issuing this command.

6.6.2 SwitchOnlineOffline

The command can be executed only after successful execution of the *ParamForAcq* commands.

The command is used to control the transfer of modifications made to the Map and the parameters to the ECU.

Command syntax:

```
mskasap3cmd(s1, 'SwitchOnlineOffline', Mode);
```

Arguments passed:

s1	Serial Port Object
SwitchOnlineOffline	Command String
Mode	Mode for the parameter acquisition

The argument Mode needs to be passed to the command. Mode can take two values:

 'on': Online mode

 'off': Offline mode

When the command is sent for the first time with mode 'on', all the current and future modifications to Map and parameters are transferred to the controller. Thus, the AUSY system can request the current online value of the parameters. By switching the mode to 'off', this transparency is lost. So, any changes made to the maps and parameters would be stored locally on the MC system.

Example:

To set the mode online:

```
mskasap3cmd(s1, 'SwitchOnlineOffline', 'on');
```

To set the mode offline:

```
mskasap3cmd(s1, 'SwitchOnlineOffline', 'off');
```

Note: Please contact Emmeskay Inc. if you have problems issuing this command.

6.6.3 GetOnlineValue

The command can be sent to the MC system only after successful execution of *SwitchOnlineOffline* command with 'online' mode.

Upon receipt of this command by the MC system, the MC system would respond back with the actual values of the parameters as defined in the *ParamForAcq* command.

Command syntax:

```
[NoofVal, Vals] = mskasap3cmd(s1, 'GetOnlineValue');
```

Arguments passed:

s1	Serial Port Object
GetOnlineValue	Command String

The command responds back with the parameter actual values.

The information received is:

NoOfVals: Number of values

Vals: The values for the recorded data

Example:

```
[NoofVal, Vals] = mskasap3cmd(s1, 'GetOnlineValue');
```

Note: Please contact Emmeskay Inc. if you have problems issuing this command.

6.6.4 GetUserDefinedValue

The command is used to get the values of those quantities which the user has selected. The command can be sent to the MC system only after successful execution of *Init and Identify* command.

This command is similar to the *GetOnlineValue* command. But this command is used to get the values of group of variables in offline mode. Upon receipt of this command, the MC system would output the current values of the user defined variables.

Command syntax:

```
[NoofVal,Vals] = mskasap3cmd(s1, 'GetUserDefinedValue');
```

Arguments passed:

s1	Serial Port Object
GetUserDefinedValue	Command String

The command responds back with the parameter values.

The information received is:

NoOfVals: Number of values

Vals: The values for the recorded data

Example:

```
[NoofVal, Vals] = mskasap3cmd (s1, 'GetUserDefinedValue');
```

Note: Please contact Emmeskay Inc. if you have problems issuing this command.

6.7 Miscellaneous Commands

6.7.1 ResetDevice

The command enables the user to activate the reset line of the control device, via the MC system for the specified controller identified by the LUN. The command can be issued only after successful execution of *SetDescriptionFile* or *DefineDescriptionFile* command.

Command syntax:

```
mskasap3cmd(s1, 'ResetDevice', LUN);
```

Arguments passed:

s1	Serial Port Object
SetGraphicMode	Command String
Cmd	Command String

Cmd: This is the command string that needs to be passed to the MC system. It can have two values:

- on: Graphic mode on
- off: Graphic mode off

Example:

```
mskasap3cmd(s1, 'SetGraphicMode', 'on');
```

Note: Please contact Emmeskay Inc. if you have problems issuing this command.

6.7.2 SetCaseSensLabels

The command can be used to let the MC system know that all the labels for the remaining session should be treated as case sensitive. The command can be sent after *Identify* command has been sent.

Command syntax:

```
mskasap3cmd(s1, 'SetCaseSensLabels');
```

Arguments passed:

s1	Serial Port Object
SetCaseSensLabels	Command String

Care has to be taken that user takes into account the case of the label once this command is sent. If the case does not match, then the MC system will respond with error message. This would let user know that the specified parameter was not found.

6.7.3 SetGraphicMode

The command can be used to disable some of the allowed MC system settings, so that MC system resources can be made available for more important system tasks. The settings that can be disabled depends on the Calibration software.

Command syntax:

```
mskasap3cmd(s1, 'SetGraphicMode',Mode);
```

Arguments passed:

s1	Serial Port Object
SetGraphicMode	Command String
Mode	Mode can be 0 or 1

Mode: Mode represents the activation / deactivation.

0 : Visualisation is deactivated

1 : Visualisation is activated

Some of the critical tasks like ECU communication needs more memory in the MC system. By disabling some memory required tasks in the MC system, resources can be allocated for these important tasks.

Example:

```
mskasap3cmd(s1, 'SetGraphicMode', 0);
```

6.7.4 flags

This command is not defined for the ASAP3 specification. The command is used by the toolbox internally. The command allows user to set the display and log options. The command can be sent only after *Init* command has been successfully sent. The command causes the toolbox to update the data stored in a mat file in the current working directory. The file name is mskasap3flagdata.mat. If the file does not exist, then it is created when the *Init* command is issued. The flag statuses are stored in this file. By default both the flags are set to 1. Whenever user sends this command, the flags are updated within the mat file.

Command syntax:

```
mskasap3cmd(s1, 'flags', DataFlg, FileFlg);
```

Arguments passed:

s1	Serial Port Object
flags	Command String
DataFlg	Flag for data stream display
FileFlg	Flag for log file generation

The flags can take a value of 0 or 1. If the DataFlg is turned on, both the outgoing data stream and the incoming data stream are displayed in the workspace. In case the FileFlg is turned on, then all the command responses are logged to a file mskasap3log.log in the present working directory.

Example:

```
mskasap3cmd(s1, 'flags', 1, 1);
```

6.7.5 Version

This command is not defined for the ASAP3 specification. The command is used by the toolbox internally. The command allows user to display the version number of the Controller Communication Toolbox. The command can be sent only after Init command has been successfully sent.

Command syntax:

```
[Ver] = mskasap3cmd(s1, 'Version');
```

Arguments passed:

s1	Serial Port Object
Version	Command String

Example:

```
[Ver] = mskasap3cmd(s1, 'Version');
```

7. Programming Techniques

This section describes sequence of command execution. The objective is to discuss typical steps of the validation/verification program using the ASAP3 toolbox.

Primarily, the ASAP3 toolbox is useful for automating the design validation/verification tests. In a typical process for carrying out a design validation test for a controller in a HIL environment following steps are involved:

1. Set initial conditions for controller
2. Set initial conditions for the vehicle model
3. Define controller side data acquisition parameters

4. Define HIL side data acquisition parameters
5. Start the controller side recorder
6. Start the HIL side data acquisition
7. Execute the test sequence
8. Stop the controller side data acquisition
9. Stop the HIL side data acquisition
10. Save the controller data
11. Save the HIL data
12. Stop test

In this type of typical test execution the human operator is "in-the-loop". In this section, this typical flow is assumed. Also, it should be noted that no attempt is made to show the HIL side interaction. The section strictly concentrates on the ASAP3 link program sequence.

Step 1:

Initialize: This is the first test which needs to be carried out in order to proceed. This is done as follows:

```
S = serial('COM1');  
fopen(S);
```

It is assumed that COM1 port is available for the ASAP3 toolbox. This port is connected to the MC system serial port.

The command is a standard MATLAB[®] function for serial port connection. This returns a serial port object in variable S and the second command opens the port for the ASAP3 communication. This is followed by:

```
mskasap3cmd(S, 'Init');  
mskasap3cmd(S, 'Identify');  
mskasap3cmd(S, 'flags', 1, 1);
```

Upon successful execution of Init command, Identify command should be sent. These two commands complete the initialization of the ASAP3 connection between the AUSY and the MC system. This should be followed by selecting or defining the description file:

```
LUN = mskasap3cmd(S, 'SetDescriptionFile', 'C:\DescriptionFile', 'C:\BinFile')
```

Successful execution of this command completes the initialization phase.

Step2:

Define Recorder Parameters: Next step is to initialize the recorder.

```
RecParamList.Name = {'P1', 'P2', 'V4', 'Var1'};  
RecParamList.ScanTime = 1;  
mskasap3cmd(S, 'DefRecParam', 0, RecParamList, LUN);
```

This initializes the recorder and defines the parameters for the recorder. The command is followed by

```
mskasap3cmd(S, 'TrigCondition', '', '', 100000, 0, 0);
```

This command defines the recorder mode. In current example, the recorder is controlled by commands from the user. To activate the recorder, send the command

```
mskasap3cmd(S, 'ActRecorder', 'act');
```

Step3:

Starting the data acquisition: Next, start the data acquisition on the MC system.

```
mskasap3cmd(S, 'ActRecorder', 'on');
```

This causes the MC system to start the data acquisition.

Step4:

Test execution: This is the most important part of the program for a design validation test. The user might have to carry out some parameter operations or look up table operations as part of the test. Thus, following commands would be useful in this step.

Working with look up tables:

```
[MapId, Xid, Yid] = mskasap3cmd(S, 'SelectLookUpTable', 'TableName', LUN)  
mskasap3cmd(S, 'SetLookUpTableValue', Map, 4, 1, 1, 1, 32.27)  
x = mskasap3cmd(S, 'GetLookUpTableValue', Map, 3, 2)
```

Working with parameters:

```
[TempVal] = mskasap3cmd(S, 'GetParam', 'ParamName', LUN)  
mskasap3cmd(S, 'SetParam', 'ModParam', 4.36, LUN)
```

In this step the complete test sequence execution is carried out.

Step5:

Ending the test: Once the test is complete, the data acquisition needs to be stopped and the data should be stored.

```
mskasap3cmd(S, 'ActRecorder', 'off')
```

mskasap3cmd(S, 'SaveRecorder', 'DataSaveFileName')

Step6:

Clean up: This is an important step that ensures that the communication is terminated and the program makes a graceful exit.

mskasap3cmd(S, 'Exit')

The command closes the ASAP3 connection between AUSY and MC system and releases the COM1 port on the AUSY for other programs.

8. FAQs and Troubleshooting

This section would be updated from time to time as users send us support questions.

Troubleshooting:

Issue	Potential cause	Suggested action
Communication failure	Serial port not free	Check if the serial port is available for MATLAB®
	Baud rate mismatch	Make sure that baud rates on the both the serial port settings match. If they are different, change either to match the other.
	Mismatched serial ports	Make sure that on both the computers correct serial ports are used.
	Connection issue	Make sure that the null modem cable is correct. Refer to the null modem pin out.
	Setting on calibration software	Check if the calibration software needs some setting (such as making it online) so as to enable the communication. Make sure that the serial port is active.
	Calibration software compatibility	Make sure that calibration software supports ASAP3 protocol.
	Calibration software not running	Please start the calibration software and configure as required for the ASAP3 protocol.
Licensing error	Driver not installed	If you were provided with a dongle, make sure that the HASP key drivers are installed.

	Dongle not connected	Make sure that the dongle is connected properly to the computer. If you were provided with USB dongle make sure that the USB port is properly installed.
	Path not set	Make sure that the paths are properly set for the MATLAB [®] being used. Please refer to installation instructions section of this document, to see how to set the path.
Command execution failure	Parameter label error	Make sure if accessing the parameters or look up table commands are causing errors intermittently, then check the labels for accuracy of spellings
	Case sensitivity turned on	If the case sensitivity turned on, make sure that commands are issued with proper cases for the labels.
	Unsupported commands	Though the toolbox support a lot of the ASAP3 commands, the calibration software may not support some of them. Contact your software vendor to find out if the command causing problem supported or not.
Output not visible	Option flag turned off	If the option flag for display of the output data is turned off, then the toolbox does not output command responses to the command prompt. Make sure this flag is not turned off. If it is turned off, then turn it on again by issuing the command with output flag turned on again.

9. Reference

The ASAP3 Controller Communication toolbox is based on the serial protocol. In order to work with the toolbox, the two serial ports (1 on MC system and 1 on AUSY) should be connected to each other via a Null Modem Cable.

Following table shows the null modem connection pin out.

Null Modem Cable Pin-out:

Pin	Serial port 1 (AUSY)		Serial Port 2 (MC System)		Pin
	25 Pin	9 Pin	9 Pin	25 Pin	
GND	1	-	-	1	GND
TD	2	3	2	3	RD
RD	3	2	3	2	TD
RTS	4	7	8	5	CTS
CTS	5	8	7	4	RDS
GND	7	5	5	7	GND
DSR	6	6	4	20	DTR
DTR	20	4	6	6	DSR

MATLAB[®], Simulink[®] and Stateflow[®] are Registered Trademarks of The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760-1500. Visit www.mathworks.com for details.