



The attached SMPTE Engineering Document has been declared “Stable” by the controlling Technology Committee.

The SMPTE Operations Manual for Standards states:

A document should be stabilized if it is believed to be substantially correct, does not contain harmful or misleading recommendations, may still be relevant to equipment or practices in use, is stable, but does not represent current technology, and need not be subject to future reviews.

A Stable document shall still be made available and offered for sale by the Society, but it shall be prefaced by a cover page explaining its current status.

At any time, a Technology Committee may revise, amend, or otherwise initiate a new Project on a Stable document.

A Stable document is “In Force”, and not deprecated or withdrawn.

*** * * * ***

Note:

SMPTE “Stable” documents were previously described as “Archived” and the attached document may be marked as “Archived”. The status of a SMPTE document described as “Archived” is exactly as described above for a “Stable” document.

Stable documents may not adhere to the latest style and format of SMPTE documents, or to current usage of normative language. Suitable care should be taken in interpretation.

SMPTE STANDARD

for Television — Status Monitoring and Diagnostics Protocol



Page 1 of 23 pages

1 General

This standard defines the status monitoring and diagnostics protocol (SMDP) used with a general-purpose communication link that connects to equipment used in the production, post-production, and/or transmission of visual and audio information. The communication link is separate and distinct from the supervisory interface for digital control (ANSI/SMPTE 207M). SMDP may be used when querying equipment for status and diagnostics information.

The primary intent of this standard is to establish the protocol between a supervisory controller and associated video/audio equipment. In addition, it seeks to establish a common set of commands that should be used in SMDP-based systems. A provision allows for manufacturer-specific commands in addition to the common set.

2 Normative reference

The following standard contains provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the edition indicated was valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standard indicated below.

SMPTE RP 165-1994, Error Detection Checkwords and Status Flags for Use in Bit-Serial Digital Interfaces for Television

3 Definitions

3.1 encapsulated SMDP: The lowest layer of the SMDP may be encapsulated as data into other protocols. Using this method, SMDP packets may be transported in open, standard protocols such as TCP/IP.

3.2 message set: From the standpoint of a sender, an issued command/data message sent to a destination and the associated response/data message from the destination. All message transactions between a supervisor and a virtual machine require a message set.

3.3 supervisor: Each system needs at least one supervisory controller. This controller connects to any and all video/audio equipment that is to be monitored.

3.4 virtual circuit: A transparent, unidirectional, logical communications connection between a supervisor and a virtual machine. The communication path may pass through other levels of protocol before the circuit is established.

3.5 virtual machine (VM): A logical device consisting of a single device or combination of devices that respond in essence as a generic type of equipment; e.g., VTR, video switcher, router, telecine, etc. The VM is the device under scrutiny when connected to a link that uses the SMDP.

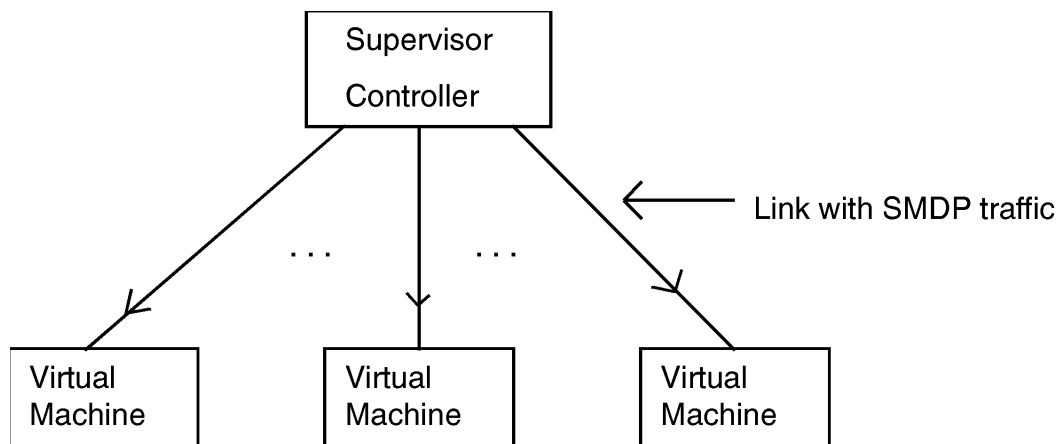


Figure 1 – Typical system

4 System elements

Figure 1 shows a typical system with links between a supervisory controller and several VM devices. Each device is connected to the supervisor using dedicated communication links (TIA-232). Small systems may use directly connected communication links as shown in figure 1. Medium and larger systems may use encapsulated SMDP to provide addressability, guaranteed delivery, and access to public networks. Encapsulated SMDP leverages the advantage of open, public network standards. Although SMDP may be used as shown in figure 1, its main application is as described in annex A. Annex A of this standard gives an example of encapsulated SMDP and how it may be used in a networked environment. As will be shown, SMDP traffic uses only a few ASCII control codes and, as such, may be carried by MODEMS without problems. The SMDP supports multidrop addressing. A discussion of multidrop modes is presented in annex B.

In general, the supervisor queries the VM using a command from the standard set or one provided by the manufacturer. The VM executes the command and always returns a response when finished. The VM may initiate a call to the supervisor by sending a special command for this purpose. So, the VM may be polled by the supervisor for information or it may send an interrupt to the supervisor indicating that it has information of interest. Most VMs also have a separate digital control interface input to remotely control the VM. It is the responsibility of the manufacturer to assign priorities to the two interfaces so that both are served but in a time and manner that is best suited for any particular VM.

SMDP traffic is processed by software routines (the SMDP processor) in each VM and the supervisor. The SMDP processor may be run as a task under the auspices of the VM's or supervisor's operating system. The processor is not multitasking and as such can only process one command at a time. The only exception is the *RST (reset) command which is executed upon arrival by the SMDP processor (see 6.1 for more information on *RST).

5 Common command and response structure

The SMDP defines 11 common commands and response messages that all protocol compliant virtual machines should implement. In addition, there are five that are optional. These are general in nature and not device-specific in any way. In addition, manufacturers may add private commands as long as they follow the byte and packet level protocols defined in this standard.

Next, the common set is defined and explained. All these message names include an * as part of the name to identify it as a common command. Manufacturer supplied commands should omit the *. Also, any command that is a query for information must include a ? to identify the query operation. Every command that is issued to a VM causes it to send one of three possible responses back to the supervisor. The possible responses are (1) *ATN:OPC, (2) ATN:CMDERR, or (3) *ATN:QRESP (plus response data structure). The ATN prefix signifies that the VM desires the attention of the supervisor. The OPC response indicates that the OPERATION is Complete (no errors either) and the VM may receive another command. The CMDERR response indicates that the last command issued to the VM was errored in some way. The QRESP response indicates that data corresponding to a Query RESPONSE is being sent. The command and response names are case insensitive.

Whenever ASCII string data are returned to the supervisor, any given line should not be longer than 62 characters including the carriage return/line feed combination, CRLF. The CRLF set allows for the convenient printing of returned response data. All string lines, regardless of length, should end with an ASCII CRLF combination (HEX 0D 0A). As many lines as needed may be returned for a query response.

Commands, responses, and data are sent between devices using packets. In general, packets to and from a VM are composed as follows. The braces below are used to delimit optional parameters and are not part of the actual command format.

Simplified packet for an issued command with data:

```
command_keyword{:command_keyword...} {?} { parameter,parameter...} ; data field
```

Simplified packet for an issued response with data:

```
response_keyword{:response_keyword...} { parameter,parameter...} ; data field
```

Commands, responses, and optional parameters go into the command/response field. A command is one or more colon-separated command_keywords. Likewise for a response. Associated data structures, if any, go into the data field. Values within braces are optional. Generally, commands go from the supervisor to the VM and responses go from the VM to the supervisor, but there are exceptions. The keywords and parameters must be printable strings. The following should be observed:

- 1) Keywords may be concatenated if desired. This is a hierarchical method of command/response naming. Keywords must be separated by colons (:). No colon is needed after the last keyword.
- 2) Any optional parameters must be comma (,) separated. There is no trailing comma after the last parameter. There must be a space between the last keyword and any parameters to follow.
- 3) The command/response field is always terminated with a semicolon (;). No other embedded semicolons are allowed in the command/response field. A ? is appended to the compound command name if the command is a query.

The command/data and response/data form a message set. For every command/data message sent to a receiver, a response/data message is returned. The response is, in effect, a termination for the transaction.

Table 1 summarizes the message names used in SMDP. The classification is as follows: C indicates that the name is a command message to the VM. R indicates that the name is a response message from the VM. CSUP indicates the name is a command message to the supervisor from the VM. If a \$ appears next to the indicator, then command implementation is optional.

6 Command messages to the VM

6.1 Command name: *RST (reset)

Description: Reset the SMDP processor in the VM upon receipt of this command. This causes a termination of any pending or existing VM SMDP operations and it returns to the idle state. All error and message queues are flushed. Any current packet transmissions to the supervisor are cancelled. However, the *ATN:OPC is sent in response. When the VM powers on, the SMDP processor should be reset to the idle state. If the VM is attached to a multidrop link, or equivalent, this command causes the VM to unaddress itself (see annex B for more information.)

Table 1 – Command summary

	Command or response message name	Full name	Classification
1	*RST	Reset	C
2	*IDN?	Identify query	C
3	*TST	Test	C
4	*TST?	Test query	C
5	*FLAGS?	Flag query	C
6	*STATUS?	Status query	C
7	*CMDERR?	Command error query	C
8	*MSG?	Message register query	C
9	*PIPE	Pipe	\$C
10	*UPLOAD?	Upload query	\$C
11	*ADDSEL	Select subaddress for VM	\$C
12	*ATN:OPC	Attention:Operation complete	R
13	*ATN:CMDERR	Attention:Command error	R
14	*ATN:QRESP	Attention:Query response	R
15	*ATN:PIPE	Attention:Pipe	\$CSUP
16	*ATN	Attention:Supervisor	\$CSUP

Response of the VM: Return the *ATN:OPC response when the reset is complete.

6.2 Command name: *IDN? (identify query)

Description: Query the VM for device-specific information. The format of the returned information is a list of parameter identifier names, a colon, and the actual string parameter followed by the carriage return and line feed combination (crLf). The parameters with capital letters are user-defined values. There are seven predefined return types as shown below. These must be returned with values for all seven parameters. Use a default string value of NONE when no parameter is defined. Also, users may add custom parameters as needed. Insert these starting after the last row. Some VMs may have several associated serial numbers (SERIALNUM) or software versions (VERSION). It is the responsibility of the manufacturer to clarify these possible ambiguities in the VM's user manuals. The return data structure order and form is as follows:

```

Manufacturer:NAME[crLf]
Model:MODEL[crLf]
Device ID:DEVICEID[crLf]
Serial number:SERIALNUM[crLf]
Software version:VERSION[crLf]

```

Virtual machine type:TYPE[crLf]
 VM subaddress:SUBADD[crLf]
 (User defined parameters start here.)

A few of the parameters require some added explanation. The DEVICEID parameter is provided by the VM as a unique number or name identifier for that VM. The DEVICEID is useful for locating devices by name, but should not be considered an address for the VM. The manufacturer should define how to set the DEVICEID value for the VM. The virtual machine type parameter, TYPE, identifies what virtual machine control dialect is used by the VM. For example, SMPTE RP 170 gives the TYPE for video tape recorders as 02. TYPE values have been defined for several different classes of equipment by SMPTE. TYPE is a string value of a HEX number. The subaddress (SUBADD) for the VM defines which VM is selected when more than one device is attached to a single TIA-232 interface port. This situation may occur whenever a multidrop, or equivalent, configuration is implemented. If the VM is not configured in this fashion, use the string NONE as the address value (see 6.11 and annex B for more information.)

Response of the VM: Return the ASCII string *ATN:QRESP identifier immediately followed by the data structure, with parameters, shown above.

Example: A sample returned data structure from a VTR that was issued the *IDN? command.

Manufacturer:Video Things Inc [crLf]
 Model:VTI-100 [crLf]
 Device ID:VTR37 [crLf]
 Serial number:1000512A [crLf]
 Software version:2.1 [crLf]
 Virtual machine type:02[crLf]
 VM subaddress:NONE [crLf]

6.3 Command name: *TST TESTNUM (test)

Description: This command instructs the VM to execute numeric string TESTNUM. The test is run according to the manufacturer's specification for that test. Test number 0 is reserved. Tests range from 1 to FFFFFFFF. Test number 1 instructs the VM to perform a power-on test if one exists. All other tests are user defined. The results of each test are stored in a buffer that may be read by the *TST? command. The buffer's size is not specified by this standard.

Response of the VM: The VM returns a *ATN:OPC when the requested test has been executed.

Example: *TST 23 (Run test number 23 and put the results in the VM's test results buffer)

6.4 Command name: *TST? (test query)

Description: A query command to read the test results buffer (see *TST command). This buffer contains printable ASCII characters that describe the results from previously executed tests. The test results are retrieved in a first executed, first returned basis. If the result buffer overflows as a consequence of executing too many tests, the last test's results are lost. The only way to empty the results buffer is by reading it using *TST? or issue a *RST. The actual returned data is a string (RESULT) containing the results. The string is preceded by the test number. If more than one line is needed, separate lines with a crLf combination.

Test:TESTNUM:RESULT [crLf]

Response of the VM: The response is in the form *ATN:QRESP + data structure. If there are no test results in the buffer, the returned data structure is: Test:0[crLf]. If several different test results are in the buffer, each time *TST? is executed, the results buffer will empty another response set until finally the Test:0 response is returned.

Example: Assume that test numbers 4 and 7 (used here by example only) have been successfully executed using *TST. Then issuing *TST? will return the data structure (RESULT).

Test:4:Power supply voltages are within range. VCC = 4.95, V12 = 11.8 [crLf]

By issuing *TST? a second time, the returned data structure could be

Test:7:Dynamic memory test passed. 512 kbytes tested. [crLf]

A third (or more) issuing of *TST? will return

Test:0 [crLf] (No buffered test results to return)

6.5 Command name: *FLAGS? (flag query)

Description: This query returns five flags from the VM. Each flag is a string yes or no with an associated flag description. The returned format is

Definition:{ yes no}

Some flags are “sticky,” that is, once the flag is read it resets to “no,” the default state. All five flags are returned per query. A crLf combination is used to separate lines. Additional user-defined flags may be added after the last line. The flags are a quick way to get a simplified picture of certain VM internal states. For more information, the message register (see *MSG?) could be used.

Definition of flags

The returned response from a VM should take the following format. The braces indicate a choice of one selection of the string values “yes” or “no.”

Power cycled on: {yes,no} [crLf]
EDH now: {yes,no} [crLf]
EDH in past: {yes,no} [crLf]
EDA now: {yes,no} [crLf]
EDA in past: {yes,no}[crLf]

Power cycled on flag: Whenever the device power is cycled, this flag is set to “yes.” This is a sticky flag. So, reading it will reset it to “no.”

The following four definitions require SMPTE RP 165 to be observed. If not, set the flag to “no.”

EDH now flag: This flag corresponds to the error detected here flag for the full-field checksum according to SMPTE RP 165. If a full-field checksum mismatch is currently occurring (during the field when the VM is queried), then this bit is set to “yes,” else it is “no.” If there is more than one digital video input that obeys SMPTE RP 165, the EDH now flag is set to “yes” if any of the inputs are generating an EDH fault.

EDH in past flag: If the “EDH now” flag was set at some time in the past (since power on of VM), this flag indicates so by being set to “yes.” This flag is sticky.

EDA now flag: This flag corresponds to the error detected already flag in SMPTE RP 165. This flag is a “yes” if the received full-field EDA bit is a logic 1 at the time of the query of the VM. If there is more than one digital video input that obeys SMPTE RP 165, the EDA now flag is set to “yes” if any of the inputs have an EDA bit set.

EDA in past flag: If the “EDA now” flag was set at some time in the past (since power on of VM), this flag indicates so by being set to “yes.” This flag is sticky.

User-defined flags: The manufacturer may define additional flags as needed to be appended after the fifth flag.

A received *RST command causes all flags to be set to "no."

Response of the VM: The VM returns the *ATN:QRESP identifier and the data structure below.

Example: The returned data structure could look as follows:

```
Power cycled on: yes [crLf]
EDH now: no [crLf]
EDH in past: yes [crLf]
EDA now: no [crLf]
EDA in past: no [crLf]
```

A second *FLAGS? query would return the following strings, assuming no changes in EDA or EDH activity between queries. Note the changed state of the sticky flags.

```
Power cycled on: no [crLf]
EDH now: no [crLf]
EDH in past: no [crLf]
EDA now: no [crLf]
EDA in past: no [crLf]
```

6.6 Command name: *STATUS? (status query)

Description: This is a query for the current operational status of the VM. The result is a printable ASCII string indicating the status. This command should not be confused with the *MSG? command described below. The status information returned by *STATUS? is defined by the manufacturer and should be the predominate operational condition of the VM.

Response of the VM: The VM returns the *ATN:QRESP indicator plus the data structure. The data structure is a string.

Example: A possible returned string could be: VTR in play [crLf]

6.7 Command name: *CMDERR? (command error query)

Description: Every command sent to the VM should be checked for syntax, and parameter errors. If such errors occur, the VM loads the command error buffer with an error string indication. The VM command error buffer holds error strings in a queue and empties successive strings in response to each *CMDERR? received. The buffer is emptied in a first set, first read order. Once the last nonzero error code has been read, error code zero is returned for any further responses. The return codes are defined as given in table 2.

The format of the returned data structure is

```
Command in error -> error code:error description [crLf]
```

or by exception

```
*CMDERR -> No errors in queue [crLf] (No errors to return)
```