



Designation: E 1989 – 98

Standard Specification for Laboratory Equipment Control Interface (LECIS)¹

This standard is issued under the fixed designation E 1989; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision. A number in parentheses indicates the year of last reapproval. A superscript epsilon (ε) indicates an editorial change since the last revision or reapproval.

1. Scope

1.1 This specification covers deterministic remote control of laboratory equipment in an automated laboratory. The labor-intensive process of integrating different equipment into an automated system is a primary problem in laboratory automation today. Hardware and software standards are needed to facilitate equipment integration and thereby significantly reduce the cost and effort to develop fully automated laboratories.

1.2 This Laboratory Equipment Control Interface Specification (LECIS) describes a set of standard equipment behaviors that must be accessible under remote control to set up and operate laboratory equipment in an automated laboratory. The remote control of the standard behaviors is defined as standard interactions that define the dialogue between the equipment and the control system that is necessary to coordinate operation. The interactions are described with state models in which individual states are defined for specific, discrete equipment behaviors. The interactions are designed to be independent of both the equipment and its function. Standard message exchanges are defined independently of any specific physical communication links or protocols for messages passing between the control system and the equipment.

1.3 This specification is derived from the General Equipment Interface Definition developed by the Intelligent Systems and Robotics Center at Sandia National Laboratory, the National Institute of Standards and Technologies' Consortium on Automated Analytical Laboratory Systems (CAALS) High-Level Communication Protocol, the CAALS Common Command Set, and the NISTIR 6294 (1-4).² This LECIS specification was written, implemented, and tested by the Robotics and Automation Group at Los Alamos National Laboratory.

1.4 *Equipment Requirements*—LECIS defines the remote control from a Task Sequence Controller (TSC) of devices exhibiting standard behaviors of laboratory equipment that meet the NIST CAALS requirements for Standard Laboratory

Modules (SLMs) (5). These requirements are described in detail in Refs (3, 4). The requirements are:

- 1.4.1 Predictable, deterministic behavior,
- 1.4.2 Ability to be remotely controlled through a standard bidirectional communication link and protocol,
- 1.4.3 Maintenance of remote communication even under local control,
- 1.4.4 Single point of logical control,
- 1.4.5 Universal unique identifier,
- 1.4.6 Status information available at all times,
- 1.4.7 Use of appropriate standards including the standard message exchange in this LECIS,
- 1.4.8 Autonomy in operation (asynchronous operation with the TSC),
- 1.4.9 Perturbation handling,
- 1.4.10 Resource management,
- 1.4.11 Buffered inputs and outputs,
- 1.4.12 Automated access to material ports,
- 1.4.13 Exception monitoring and reporting,
- 1.4.14 Data exchange via robust protocol,
- 1.4.15 Fail-safe operation,
- 1.4.16 Programmable configurations (for example, I/O ports),
- 1.4.17 Independent power-up order, and
- 1.4.18 Safe start-up behavior.

2. Terminology

2.1 *Definitions of Terms Specific to This Standard:*

2.1.1 *command message*—communication from the TSC to the SLM that is being controlled. Receipt of this message causes a state transition in an interaction.

2.1.2 *device capability dataset*—data file that contains all of the SLM-specific information required for the TSC to interact with the SLM (2). This includes definitions of the arguments of the standard commands and events, estimates of processing times in states, and SLM specific interactions. Material input and output ports and support services are also defined.

2.1.3 *error*—infrequent, unplanned event that makes the current goal of the SLM unachievable given the system resources, the state of the system, or the absence of a guaranteed, alternative plan. (This definition is distinct from any other ASTM standard definition of error.)

¹ This specification is under the jurisdiction of ASTM Committee E01 on Analytical Chemistry for Metals, Ores and Related Materials and is the direct responsibility of Subcommittee E01.25 on Laboratory Data Interchange and Information Management.

Current edition approved Oct. 10, 1998. Published March 1999.

² The boldface numbers in parentheses refer to the list of references at the end of this standard.



2.1.4 *event*—change in the operational state of the SLM that must be reported to the TSC.

2.1.5 *event report*—communication from the SLM to the TSC indicating an event.

2.1.6 *exception*—off-normal change in the operational state of the SLM that prevents the goal from being achieved through the normal sequence of state transitions in an interaction. Alternative sequences of state transitions in an interaction are provided as part of the interaction definition in order to handle an exception. Occurrence of an exception invokes an action that is simply an alternative in the course of normal processing. It causes suspension of normal operation and initiates a defined alternative operation.

2.1.7 *interaction*—standard exchange of messages between the TSC and SLM that synchronizes the execution of a series of standard SLM behaviors. State models are used to describe the standard interactions.

2.1.8 *material*—any input being processed by the SLM or output produced by the SLM, including samples, consumables, and data.

2.1.9 *message*—event or command that is passed between a TSC and an SLM.

2.1.10 *message transaction*—synchronized exchange of a message and its acknowledgment. There are two types of message transactions: (1) Command transactions are initiated by a command from the TSC to the SLM. (2) Event reports are initiated by the SLM to the TSC.

2.1.11 *port*—physical or logical location in the SLM that accepts or provides material or data. Physical ports are used to transfer items such as samples and supplies. Logical or data ports are used to transfer data to and from the SLM.

2.1.12 *port index*—ports may have multiple internal locations. These internal locations are tracked with the Port Index.

2.1.13 *standard laboratory module (SLM)*—laboratory equipment that satisfies the NIST CAALS physical and behavior requirements (5).

2.1.14 *state*—standard logical configuration of the SLM for the purposes of remote control that may correspond to a logical or hardware configuration or both. Specific behaviors and operations of the SLM are allowed in each state.

2.1.15 *state model*—graphical illustration of the states in the standard interactions. Transitions between states in the interaction, indicated by arrows, are initiated and tracked with messages.

2.1.16 *task sequence controller (TSC)*—software system that orchestrates the execution of tasks (steps in sample processing) by assigning them to the appropriate SLM and coordinating material and data movement to and from the SLM selected for particular tasks.

3. Notation and General Message Syntax

3.1 *Notation*—In the discussions of the messages between the TSC and SLM, the following arrow symbols are used to indicate the nature of the message. These symbols are provided as a guide to the reader; they are *not* part of the message definition.

Symbol	Description
⇒	Command from the TSC
←	Event from the SLM

→	Event message acknowledgment from the TSC to the SLM
←	Command message acknowledgement from the SLM to the TSC

3.2 *General Message Syntax*—The messages in this specification are defined in *Extended Backus Naur Form (EBNF)* (6). A description of the symbols used in the message definition is provided in Table 1. The EBNF definition of all data types used with the message definitions is provided in Annex A1. The message definitions in Annex A1 use a Arial font to denote definitions in EBNF data types and bold Courier font to denote the exact ASCII string that is used in the message. Throughout the text of this specification, EBNF data types appear in the text font. Boldface is used to denote the exact ASCII string. For example, the <REMOTE CTRL ACCEPTED MSG> EBNF data type is defined to be the **REMOTE_CTRL_ACCEPTED** string.

3.3 Although this specification defines commands and event reports in upper case characters, the TSC and SLM message handlers should treat messages as *case-insensitive*. For example, this specification defines the emergency stop command, transmitted from the TSC to the SLM, as ESTOP. However, the messages “EStop,” “Estop,” “estop,” or any other combination of upper and lower case should be interpreted as ESTOP. Message parameters, however, are *case-sensitive*.

4. Control Paradigm

4.1 *Control Principle*—Fig. 1 illustrates the laboratory automation control architecture to which this specification is designed (7). An SLM can only be controlled by one TSC at any given time. An SLM may not communicate directly with another SLM. The TSC sends command messages to the SLM, and the SLM sends event reports to the TSC. The communication channel between the TSC and SLM is logically separate for each SLM but need not be physically separate. For example, a TCP/IP network will allow multiple devices on a single physical link. A single logical link between the TSC and SLM is also not required in this specification. The (single or multiple) physical links between the TSC and SLM can be multiplexed into multiple logical channels.

4.2 *Communication Requirements*—The interface described here is independent of the physical communication link and message exchange protocol. However, the communication link chosen shall meet the following requirements in order to be compliant with this LECIS.

4.2.1 The physical communication links must ensure accurate messaging by providing a low-level communication check of the accuracy of message transmission (parity checking, checksums, etc.).

TABLE 1 EBNF Message Definition Symbols

<FIELD>	non-terminal string
::=	is defined as
	choice of either field on each side
[]	optional one or none
{ }	0 or more repetitions
{ } ^x _y	at least x, maximal y repetitions
(MSB)	this field is most significant bit or byte in multi bit or byte sequence
BOLD	terminal ASCII or hex symbol, in bold
Xx .. yy	terminal range from xx to yy
"STRING"	literal insertion of string (without quotes)

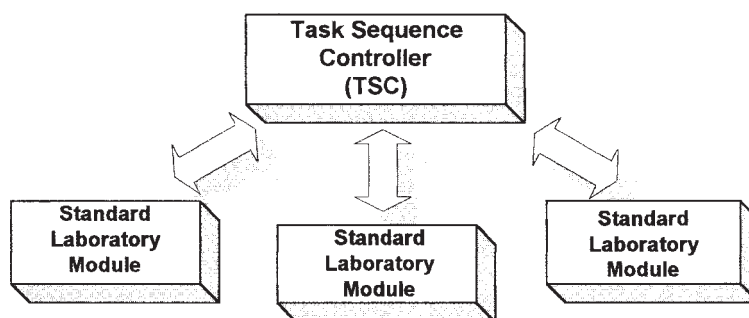


FIG. 1 Simplified, Local Control Architecture

4.2.2 The communication link cannot be blocked while the SLM performs a task. To prevent blocking, the SLM may provide channel multiplexing or separate communication links.

4.2.3 Both the TSC and SLM must periodically validate communication integrity.

4.2.4 Messages from and to instruments must be uniquely identifiable.

4.3 Interaction:

4.3.1 All communications between the TSC and an SLM in an automated laboratory are modeled as interactions. These interactions define the standard behaviors of SLMs and the standard message exchange needed to control the behaviors.

4.3.2 LECIS distinguishes between *required* and *optional* interactions and defines the required interactions. The required interactions provide basic remote control functionality for any type of laboratory equipment. SLMs shall implement the required interactions. Additional optional interactions can be defined by the SLM manufacturer to provide remote control of SLM-specific behaviors. These interactions are not the subject of this specification. Fig. 2 shows a breakdown of the interaction classes and types.

4.3.3 There are two types of interactions—*primary* and *secondary*. Primary interactions are permanently active and there can only be one instance of each primary interaction active at any time. Secondary interactions, by contrast, are

created and terminated as needed. For example, an instance of the Processing secondary interaction is created when the TSC wants the SLM to perform an operation on an input material. The interaction instance is terminated once the operation is completed. There is no limit in this specification to the number of instances of any secondary interaction that can be active simultaneously. Note that creating multiple instances of a secondary interaction is a way to implement command stacking. Table 2 lists the required primary and secondary interactions.

4.4 Interaction Identifier:

4.4.1 The interaction identifier allows the TSC and SLM to identify a specific instance of an interaction. A session-unique interaction identifier is required since multiple instances of (secondary) interactions may exist simultaneously. The interaction identifier is attached to all command and event report messages. This enables the TSC and SLM to associate the commands and event reports with the appropriate interaction instance. The interaction identifier generated by a specific SLM is only required to be unique for that SLM.

4.4.2 The interaction identifier is generated for the first message in the interaction. If the first message is a command, the TSC generates the interaction identifier. If the first message is an event report, the SLM generates the interaction identifier. Once created, the interaction identifier remains the same

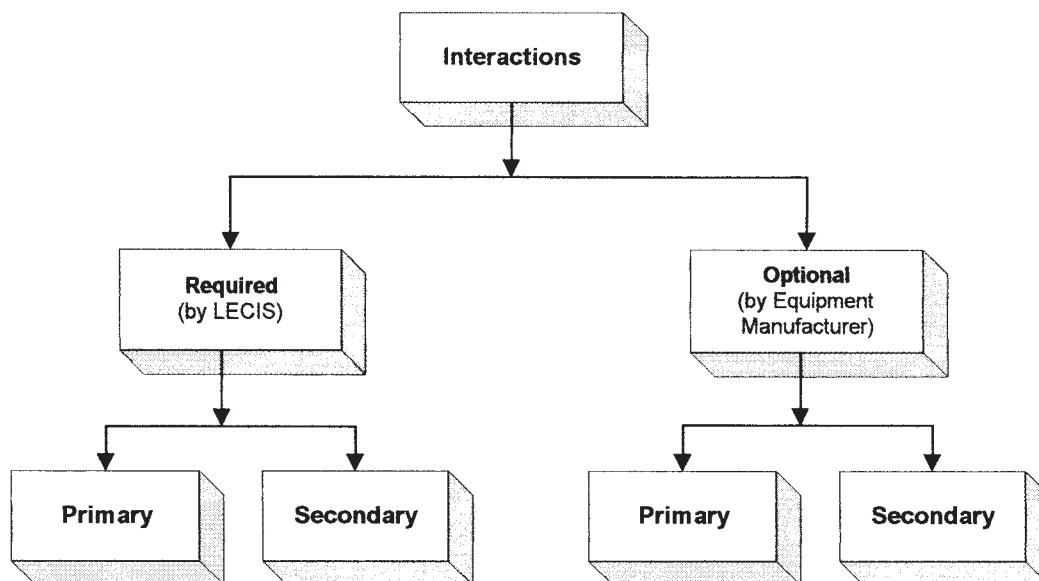


FIG. 2 Interaction Categories



TABLE 2 Required Interactions

Type of Interaction	Interaction
Primary	Local/Remote Control Control Flow
Secondary	Processing Status Lock/Unlock Abort Alarm Item Available Notification Next Event

throughout the lifetime of the interaction instance and is used to label each additional message in that interaction. This specification does not place a requirement on how the interaction identifier is generated. Any unique integer identifier can be used as an interaction identifier. We suggest that a unique interaction identifier be generated from a date and time combination. For example, from August 10, 1996, 14:22:10.33 we could generate the following interaction identifier: 1996081014221033 (YYYYMMDDHHMMSSSS). However, one could also generate an integer-based identifier using a simple counter mechanism if no real-time clock is available on the instrument.

4.5 State Models:

4.5.1 Interactions are defined by state models. Standardized state models and message exchange ensure that both the TSC and the SLM are aware of each other's state during interactions. Specific, discrete SLM behaviors and operations are allowed in the appropriate standard states. State transitions, following the arrows in the state model, occur as a result of the conclusion of internal operation by the SLM or as a result of commands from the TSC and are signaled by (and synchronous with) a message transaction. Message transactions are modeled as instantaneous in the state models. A state model consists of a graphic state chart, a text description of the states, a table of state transitions describing the command or event that cause the transition, and table(s) defining the corresponding messages for each state transition. Numbered rows in the table of state transitions correspond to numbered transitions between the states in the state chart.

4.5.2 The messages are completely standardized by this specification. The specification is customized to the control of vendor-specific equipment using the arguments of the commands and events. The message arguments are described in the SLM's capability dataset (2) that must be provided by vendors with their equipment.

4.5.3 The Harel notation for state diagrams is used to illustrate the state charts (8). Table 3 illustrates the Harel state chart symbols that are used in this specification. The Harel notation allows for hierarchical states and default initial substates when entering an encompassing parent state. The history selector indicates that the system is to return to the substate that was active at the last transition out of the parent state. Transitions themselves are unidirectional, but separate transitions can be used to toggle between states. Concurrent interactions are independent and, with two exceptions in this specification, do not directly cause transitions in each other, but they do share common context. They can be thought of as

TABLE 3 Harel State Chart Symbols

	State
	Transition
	Default Entry Point
	History Selector

weakly interacting subsystems. The use of concurrent states allows modularity of the model and greatly simplifies the individual state models. All the interactions defined in this specification may be active concurrently. The hierarchical state model that represents the primary Control Flow interaction is discussed in 7.2.

4.5.4 If an error occurs in the execution of a state transition or while the SLM is operating, the interaction *remains* in the originating state. This ensures the SLM and the associated interaction are in a known state. Abnormal termination of interactions is addressed in 9.3.

4.6 Asynchronous Master/Slave Control Model:

4.6.1 The messages that the TSC and SLM use to inform each other of their state transitions are defined to be asynchronous of one another; there is no requirement in this specification for a time interval in which a command follows an event report or an event report follows a command.

4.6.2 This specification follows a master/slave model of equipment control. Commands are issued by the controller and cause action to occur in the SLM. Event reports are generated by the SLM after the action is completed. An event report signals the end of the action by the SLM and may convey information about the results. This is illustrated in the state model by states that are entered by a command and exited by an event report. States that are normally exited by an event report are the states in which the SLM engages in appropriate action commanded by the TSC. If a specific piece of laboratory equipment has no internal operation or behavior that logically corresponds to a standard state, the SLM may "fall through" the state and immediately exit it with an event report after the state is entered.

4.6.3 States that are normally exited by a command are states in which the SLM is waiting for a command from the TSC. While in one of these states the SLM must not engage in any process that requires time to complete. In other words, while in states that are exited by a command message, the SLM must be ready for the command at any time.

4.6.4 There is usually a one-to-one correspondence between command and event reports. Exceptions are the interactions, such as the Processing interaction, that have states both entered and exited with an event report. Other exceptions are interactions that are initiated by an event report, such as the Item Available Notification interaction. Interactions initiated by the SLM should be done so as a result of a command in another interaction. For example, the Item Available Notification interaction is used to signal that the product of an action in the Processing

interaction is available for removal. These interactions are defined separately to increase the versatility of this specification.

4.6.5 LECIS assumes that transitions between states are instantaneous and concurrent with the positive acknowledgment in the message transactions signaling these transitions. In all but two cases, there is a one-to-one correspondence between state transitions and message transactions in this specification. The two exceptions are the required transition to LOCAL with a transition to ESTOPPED (6.5) and the transition to TERMINATED in the Next Event interaction (6.7) concurrent with every event report message transaction.

4.7 *Device Capability Dataset*—The vendor's equipment-specific information necessary to implement this communication standard is described in each SLM's Device Capability Dataset (2). The capability data set defines the arguments of each standard command and event report for that SLM. The Device Capability Dataset also contains information such as the physical characteristics of the SLM, the resources that it uses in its tasks, and definitions of optional interactions.

4.8 *System and Equipment Resources*:

4.8.1 Data and material handling by the SLMs may require the use of resources. Examples of resources are racks, containers, solvent reservoirs, and storage media such as hard disks. There are two types of resources, *public* and *private*. Public resources are owned by the system. Private resources are controlled by the SLM that owns that resource. An SLM's private resources are defined in its capability dataset.

4.8.2 The TSC provides centralized management of the public resources; tracking, allocating, and assigning resources to SLMs as needed. For this reason, an SLM's use of public resources must be negotiated with the TSC. Note that it is possible for private resources to become public and vice versa through negotiations between the SLM and TSC.

5. Message Transactions

5.1 Message transactions have two parts, the initiating message and its acknowledgment. If the message is correct, a positive acknowledgment (ACK) is returned to the sender. If not, a negative acknowledgment (NACK) is returned with, if possible, an indication of the error. The message acknowledgment in this specification is independent of, and in addition to, any acknowledgments required by the physical communication link and message exchange protocol.

5.2 The message recipient must acknowledge each message before a subsequent message can be sent to it. This prevents queuing of messages with the associated synchronization problems. Furthermore, spooling of commands that are sequential within an interaction is not allowed. If possible, prior to acknowledgment, the recipient should check to see if the message is legal for the existing state of the interaction to which the message applies, and if the message syntax and semantics are correct.

5.3 *Command Message Transactions*—Fig. 3 and Fig. 4 show the two possible command message transactions. Fig. 3 illustrates the TSC sending a command to the SLM and the SLM then replying with a positive acknowledgment message. The negative acknowledgment, NACK (Fig. 4), is provided as a means for the SLM to reject a command.

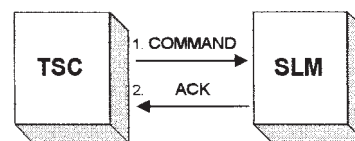


FIG. 3 Command/ACK Exchange

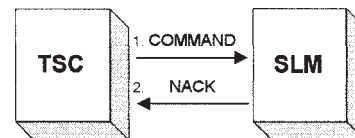


FIG. 4 Command/NACK Exchange

5.3.1 *Command Message Syntax*—A command is composed of an interaction identifier, a command name, and optional command arguments. The interaction identifier is a unique identification number of the interaction to which the command relates. When multiple instances of the same interaction are active, the interaction identifier enables the SLM to associate the command message with the correct interaction. A list of supported commands and their arguments is provided in the SLM's capability dataset (2).

5.3.1.1 The standard interactions in Sections 6-9 define specific command names (<COMMAND ID>) and optional command arguments (<COMMAND ARGS>). For clarity, the interaction identifier necessary to make up a complete command message is omitted from the examples provided for each message.

Command Message in EBNF-Syntax

```
<COMMAND MSG> ::= <INTERACTION ID> <,> <COMMAND ID>
                  [<COMMAND ARGS>]
```

Message Parameter

<COMMAND ARGS>	command arguments as specified in the SLM's capability dataset data type: <DATA STREAM> range: unspecified
<COMMAND ID>	command identifier as specified in the SLM's capability dataset data type: <PARAMETER> range: unspecified
<INTERACTION ID>	unique identifier representing the interaction instance in which the command is sent data type: <INTEGER NUMERIC PARAMETER> range: unspecified (see 4.4)

5.3.2 *Command Message Acknowledgment*—A positive acknowledgment message (ACK) indicates that the message has been received and is acceptable at whatever level of syntax and semantic checking is implemented by the SLM. An ACK from the SLM indicates that the SLM has changed state (message transactions are synchronous with the state changes).

5.3.2.1 The negative acknowledgment, NACK, is provided as a means for the SLM to reject a command. If a command message transaction is concluded with a NACK, the command is rejected and the interaction remains in the original state. This leaves the SLM and the associated interaction in a known state.

5.3.2.2 When the tables describing the command transactions and event report transactions in this specification do not

contain an explicit acknowledgment, the default is specified. The default standard ACK/NACK message is described in the SLM's capability dataset (2).

5.3.2.3 Like the messages themselves, event acknowledgments contain interaction identifiers. This enables the TSC and SLM to map the acknowledgment to the originating message and interaction.

ACK Message in EBNF-Syntax

<ACK MSG> ::= <INTERACTION ID> <,> ACK

Message Parameter

<INTERACTION ID> unique identifier that represents the interaction instance in which the command/event to be acknowledged was sent
data type: <INTEGER NUMERIC PARAMETER>
range: unspecified
(see 4.4)

NACK Message in EBNF-Syntax

<NACK MSG> ::= <INTERACTION ID> <,> NACK [<(> <ERROR ID> [<,> <ERROR ARG LIST>]<)>]

<ERROR ARG LIST> ::= <(> <ERROR ARG> [<,> <ERROR ARG>]<)>

Message Parameter

<ERROR ARG> error (event) argument
data type: <PARAMETER>
range: unspecified

<ERROR ID> error (event) identifier
data type: <PARAMETER>
range: unspecified

<INTERACTION ID> unique identifier that represents the interaction instance in which the command/event to be acknowledged was sent
data type: <INTEGER NUMERIC PARAMETER>
range: unspecified
(see 4.4)

5.3.3 *Standard NACK Messages*—This specification defines standard NACK error messages below. These messages should be implemented on SLMs that perform the syntax and semantic checking that would lead to the relevant error condition. SLM manufacturers are free to specify additional NACK messages to report error conditions that are not encompassed by the standard NACK messages below.

<ARG OUT OF RANGE MSG> ::= **ARG_OUT_OF_RANGE** [<(> <ARG RANGE LIST>]<)>

<CMD NOT SUPPORTED MSG> ::= **CMD_NOT_SUPPORTED** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<GENERAL OP FAILED MSG> ::= **GENERAL_OP_FAILED** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<INVALID ARG MSG> ::= **INVALID_ARG** [<(> <ARG INDEX>]<)>

<INVALID CMD MSG> ::= **INVALID_CMD** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<INVALID CONFIG MSG> ::= **INVALID_CONFIG** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<INVALID DATA MSG> ::= **INVALID_DATA** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<INVALID DATA TYPE MSG> ::= **INVALID_DATA_TYPE** [<(> <ARG INDEX>]

[<,> <EXPECTED DATA TYPE>]<)>

<INVALID STATE MSG> ::= **INVALID_STATE** [<(> [<CURRENT STATE> [<,> <EXPECTED STATE>]<)>]

<MISSING ARG MSG> ::= **MISSING_ARG** [<(> <EXPECTED NUM ARGS>]<)>

<MISSING CMD MSG> ::= **MISSING_CMD** [<(> <ERROR CODE> [<,> <ERROR TEXT>]<)>]

<ARG RANGE> ::= <ARG INDEX> [<(> [<ARG MIN>] [<,> <ARG MAX>]<)>]

<ARG RANGE LIST> ::= {<(> <ARG RANGE>}<)>}

Message Parameter

<ARG INDEX> argument index (>=1)
data type: <INTEGER NUMERIC>
range: unspecified

<ARG MAX> maximum argument value
data type: <PARAMETER>
range: unspecified

<ARG MIN> minimum argument value
data type: <PARAMETER>
range: unspecified

<CURRENT STATE> current interaction state
data type: <STRING PARAMETER>
range: unspecified

<ERROR CODE> code representing the type of error
data type: <REASON CODE>
range: -00001...-32767

<ERROR TEXT> optional text describing the error
data type: <STRING PARAMETER>
range: unspecified

<EXPECTED DATA TYPE> expected data type
data type: <STRING PARAMETER>
range: unspecified

<EXPECTED STATE> expected interaction state
data type: <STRING PARAMETER>
range: unspecified

5.3.4 Command Message Transaction Scenarios:

5.3.4.1 Message Scenario 1:

⇒ 08109614221033, INIT
← 08109614221033, ACK

5.3.4.2 Message Scenario 2:

⇒ 1996121108342123, SETUP ("ANALYSIS 12")
← 1996121108342123, NACK (INVALID_CMD (-122))

5.4 *Event Report Message Transactions*—Fig. 5 and Fig. 6 show the two possible event report message transactions. Upon receiving an event from the SLM, the TSC is expected to acknowledge the event message with either a positive or negative acknowledgment. In the same fashion as 5.3.2, the NACK response is used to signal a messaging error.

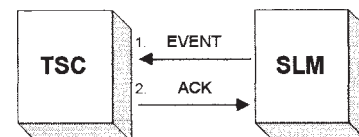


FIG. 5 Event/ACK Exchange

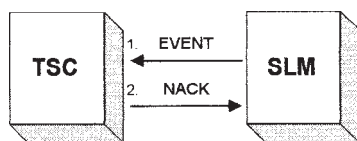


FIG. 6 Event/NACK Exchange

5.4.1 *Event Report Syntax*—Each event report can have up to four parts: interaction identifier, event time, event identifier, and optional event data. The interaction identifier is a unique identification number of the interaction to which the event report relates. When multiple instances of the same interaction are active, the interaction identifier enables the TSC to associate the event report message with the correct interaction. The event time is the time the event report was generated. This facilitates a chronological trace and log of all event report messages. If the equipment implementing this standard does not have a clock, then a counter value can be used in place of the time. This counter value can then be incremented or decremented with each event. The event identifier identifies the type of event report. An event report can also contain a list of event arguments. The event identifier and the event arguments are specified in the SLM's capability dataset (2).

5.4.1.1 The standard interactions in Sections 6-9 define specific event identifiers (<EVENT ID>) and optional event data (<EVENT ARGS>). For clarity, the interaction identifier necessary to make up a complete event message is omitted from the examples provided for each message.

Event Message in EBNF-Syntax

```
<EVENT MSG> ::= <INTERACTION ID> <,> <EVENT TIME> <,>
               <EVENT ID> [ <EVENT ARGS> ]
```

Message Parameter

<EVENT ID>	event arguments as specified in the SLM's capability dataset data type: <DATA STREAM> range: unspecified
<EVENT ARGS>	event identifier as specified by the event entry in the capability dataset data type: <PARAMETER> range: unspecified
<EVENT TIME>	time event was generated; replaced by a simple, 14-digit counter value if no clock is provided Examples: Data & Time stamp of format: yyyyymmddh-hmmssss August 10, 1996, 14:22:10.33 -> 1996081014221033 or Counter: 0000000012345 data type: <INTEGER NUMERIC PARAMETER> range: 00000000000000 - 99999999999999 Note: leading 0's are required
<INTERACTION ID>	unique identifier that represents the interaction instance in which the event occurred. If the event initiates a new interaction, then the SLM must create a new, unique interaction identifier. data type: <INTEGER NUMERIC PARAMETER> range: unspecified (see 4.4)

5.4.2 Event Report Acknowledgment:

5.4.2.1 The meaning of the ACK and NACK event report acknowledgments from the TSC are identical to the definitions in 5.3.2. A positive acknowledgment message (ACK) indicates that the message has been received and is acceptable at whatever level of syntax and semantic checking is implemented by the TSC. The negative acknowledgment, NACK, is provided as a means for the TSC to indicate to the SLM that the event report is in error in some way. If an event report transaction is concluded with a NACK, the interaction remains in the original state. This leaves the SLM and the associated interaction in a known state.

5.4.2.2 Like the messages themselves, event acknowledgments contain interaction identifiers. This enables the TSC and SLM to map the acknowledgment to the originating message and interaction.

5.4.3 Event Report Message Transaction Scenarios:

5.4.3.1 Message Scenario 1:

```
<- 08109614221033, 1996121108342123, NO_ALARMS
-> 08109614221033, ACK
```

5.4.3.2 Message Scenario 2:

```
<- 08109609444300, 1996121108342123, STATE_CHANGED ("INITING",
"IDLE")
-> 08109609444300, NACK (INVALID_STATE ("INITING", "IDLE"))
```

6. Communication Maintenance and Locus of Control

6.1 *Establishing and Maintaining Communications*—The details of preparing for and initializing communications are dependent upon the type of physical communication link and low-level message passing protocol employed between the SLM and TSC. Message passing between the TSC and SLM must be implemented with the following behaviors in order to be compatible with this LECIS (see also 4.2).

6.1.1 When in the POWERED UP state, the SLM must take the necessary steps to initiate communication with the TSC.

6.1.2 After communication has been established, the SLM and TSC should synchronize internal clocks.

6.1.3 After communication has been established, the SLM must allow communication with the TSC at all times, including when the SLM is in the LOCAL state.

6.1.4 The SLM must be able to detect loss of communication with the TSC.

6.1.5 Upon loss of communication with the TSC, the SLM must perform the necessary steps to allow communication to be reestablished.

6.1.6 SLMs should have clearly defined behavior at loss of communication that includes a graceful end of processing so that process information is not lost. An SLM-initiated transition to PAUSING should occur when the continued execution of current and pending instrument operations would overflow the event queue.

6.2 Local/Remote Control Interaction:

6.2.1 In an automated laboratory an SLM can either be under local or remote control. Even SLMs whose hardware does not support local operations must provide for a software-based local control state, because the local control state is integral to the specified emergency stop behavior (see 7.7).

6.2.2 The primary Remote/Local Control interaction describes the LOCAL and REMOTE states and the transitions between them. Fig. 7 represents the state transitions for the Control

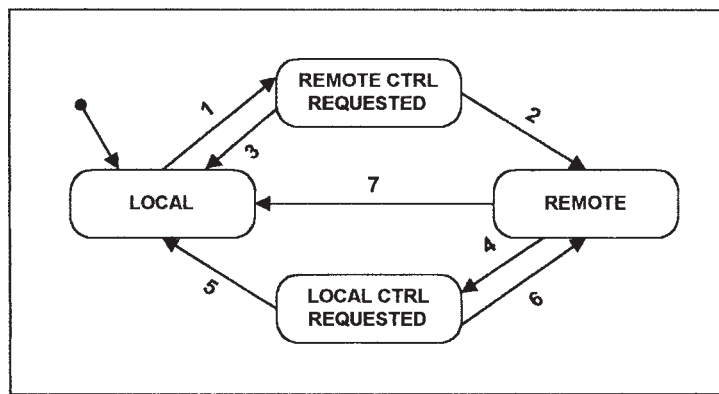


FIG. 7 Local/Remote Control State Chart

Status interaction. The transitions shown in this state chart are described in 6.4 and 6.5. While under remote control, the SLM behavior is governed by LECIS. The SLM must allow all interactions while in remote control. This specification also requires communication and certain interactions while the SLM is under local control.

6.2.3 For safety reasons, the SLM and the TSC must negotiate for changes in the control status. Hand-offs between local and remote control are negotiated through the intermediate states. Neither can take control from the other with the exception that a state change to ESTOPPED in the Control Flow interaction requires a concurrent, non-negotiated state change to LOCAL.

6.2.4 After communication has been established, the initial state at power up is expected to be LOCAL.

6.3 Description of Local/Remote Control Interaction States:

6.3.1 **LOCAL**—The SLM operation is controlled locally. While under local control, the SLM is only allowed to process status and remote control requests or ESTOP commands. The SLM should refuse all other commands by concluding the command message transaction with a NACK message (5.3.2). The TSC may receive event reports from the SLM as the SLM changes states under local control.

6.3.2 **LOCAL CTRL REQUESTED**—The SLM enters this state from the REMOTE state upon issuing or receiving a local control request. In this state, the SLM or the TSC is able to accept the request and transition to LOCAL or reject the request and transition back to REMOTE.

6.3.3 **REMOTE**—The TSC has access to all standard SLM control functions through the standard interactions.

6.3.4 **REMOTE CTRL REQUESTED**—The SLM enters this state from the LOCAL state upon issuing or receiving a remote control request. In this state, the SLM or the TSC is able to accept the request and transition to REMOTE, or reject the request and transition back to LOCAL.

6.4 Transfer to Remote Control:

6.4.1 A transfer from local to remote control can be either initiated by the TSC with a command message or initiated by the SLM with an event report message. Because of this, the message protocol is described twice, in Tables 4 and 5. The required messages for a TSC-initiated transfer to local control through the intermediate REMOTE CTRL REQUESTED state

TABLE 4 Messages for Transferring to Remote Control (TSC-initiated)

Number	Message Exchange	Old State	New State
1	⇒ REMOTE_CTRL_REQ	LOCAL	REMOTE CTRL REQUESTED
2	⇐ REMOTE_CTRL_ACCEPTED	REMOTE CTRL REQUESTED	REMOTE
3	⇐ REMOTE_CTRL_DENIED (REJECT REASON CODE, ^A REJECT REASON TEXT) ^A	REMOTE CTRL REQUESTED	LOCAL

^A Optional.

TABLE 5 Messages for Transferring to Remote Control (SLM-initiated)

Number	Message Exchange	Old State	New State
1	⇐ REMOTE_CTRL_REQ	LOCAL	REMOTE CTRL REQUESTED
2	⇒ REMOTE_CTRL_GRANTED	REMOTE CTRL REQUESTED	REMOTE
3	⇒ REMOTE_CTRL_DENIED (REJECT REASON CODE, ^A REJECT REASON TEXT) ^A	REMOTE CTRL REQUESTED	LOCAL

^A Optional.

are described in Table 4. The required messages for an SLM-initiated transfer to local control are described in Table 5.

6.4.2 While in local control, if the configuration of the SLM has been changed such that the SLM is returning to remote control in a different state in any interaction than when it entered local control, the operator should restart the SLM. Alternately it may be possible for the operator to update the TSC manually to reflect the new state of the SLM when it reenters REMOTE, however this must be done with care.

Messages in EBNF-Syntax

```

<REMOTE CTRL ACCEPTED MSG> ::= REMOTE_CTRL_ACCEPTED
<REMOTE CTRL DENIED MSG> ::= REMOTE_CTRL_DENIED [<(> [<REJECT REASON CODE> <,>] [<REJECT REASON TEXT>] <,>]>]
<REMOTE CTRL REQ MSG> ::= REMOTE_CTRL_REQ
  
```

Message Parameter

```

<REJECT REASON CODE> code describing reason for rejection
                      data type: <REASON CODE>
                      range: -00001...-32767

<REJECT REASON TEXT> text describing reason for rejection
  
```



data type: <STRING PARAMETER>
range: unspecified

6.4.2.1 Message Scenario 1:

⇒ REMOTE_CTRL_REQ
⇐ REMOTE_CTRL_ACCEPTED

6.4.2.2 Message Scenario 2:

⇒ REMOTE_CTRL_REQ
⇐ REMOTE_CTRL_DENIED (–1022, "OPERATOR OVERWRITE")

Messages in EBNF-Syntax

<REMOTE CTRL DENIED MSG> ::= REMOTE_CTRL_DENIED [<(> [<REJECT REASON CODE> <,>]
[<REJECT REASON TEXT>] <,>]

<REMOTE CTRL GRANTED MSG> ::= REMOTE_CTRL_GRANTED

<REMOTE CTRL REQ MSG> ::= REMOTE_CTRL_REQ

Message Parameter

<REJECT REASON CODE> code describing reason for rejection
data type: <REASON CODE>
range: –00001...–32767

<REJECT REASON TEXT> text describing reason for rejection
data type: <STRING PARAMETER>
range: unspecified

6.4.2.3 Message Scenario 1:

⇐ REMOTE_CTRL_REQ
⇒ REMOTE_CTRL_GRANTED

6.4.2.4 Message Scenario 2:

⇐ REMOTE_CTRL_REQ
⇒ REMOTE_CTRL_DENIED (–1231, "CONTROLLER OVERWRITE")

6.5 Transfer to Local Control:

6.5.1 A transfer from remote to local control can be either initiated by the TSC or the SLM. Because of this, the message protocol has been described twice, in Table 6 and Table 7. The non-negotiated transfer to local control concurrent with a transition to ESTOPPED is described in Table 8 and Table 9. See 7.5 for message EBNF-Syntax and example.

TABLE 6 Messages for Transferring to Local Control (TSC-initiated)

Number	Message Exchange	Old State	New State
4	⇒ LOCAL_CTRL_REQ	REMOTE	LOCAL CTRL REQUESTED
5	⇐ LOCAL_CTRL_ACCEPTED	LOCAL CTRL REQUESTED	LOCAL
6	⇐ LOCAL_CTRL_DENIED (REJECT REASON CODE, ^A REJECT REASON TEXT) ^A	LOCAL CTRL REQUESTED	REMOTE

^A Optional.

TABLE 7 Messages for Transferring to Local Control (SLM-initiated)

Number	Message Exchange	Old State	New State
4	⇐ LOCAL_CTRL_REQ	REMOTE	LOCAL CTRL REQUESTED
5	⇒ LOCAL_CTRL_GRANTED	LOCAL CTRL REQUESTED	LOCAL
6	⇒ LOCAL_CTRL_DENIED (REJECT REASON CODE, ^A REJECT REASON TEXT) ^A	LOCAL CTRL REQUESTED	REMOTE

^A Optional.

TABLE 8 Message for Transferring to Local Control Concurrent with ESTOP Command

Number	Message Exchange	Old State	New State
7	⇒ ESTOP	REMOTE or LOCAL	LOCAL

TABLE 9 Message for Transferring to Local Control Concurrent with ESTOP Event

Number	Message Exchange	Old State	New State
7	⇐ STATE_CHANGED (, "ESTOPPED")	REMOTE or LOCAL	LOCAL

Message in EBNF-Syntax

<LOCAL CTRL ACCEPTED MSG> ::= LOCAL_CTRL_ACCEPTED

<LOCAL CTRL DENIED MSG> ::= LOCAL_CTRL_DENIED [<(> [<REJECT REASON CODE> <,>]
[<REJECT REASON TEXT>] <,>]

<LOCAL CTRL REQ MSG> ::= LOCAL_CTRL_REQ

Message Parameter

<REJECT REASON CODE> code describing reason for rejection
data type: <REASON CODE>
range: –00001...–32767

<REJECT REASON TEXT> text describing reason for rejection
data type: <STRING PARAMETER>
range: unspecified

6.5.1.1 Message Scenario 1:

⇒ LOCAL_CTRL_REQ
⇐ LOCAL_CTRL_ACCEPTED

6.5.1.2 Message Scenario 2:

⇒ LOCAL_CTRL_REQ
⇐ LOCAL_CTRL_DENIED (–1500, "OPERATOR OVERWRITE")

Messages in EBNF-Syntax

<LOCAL CTRL DENIED MSG> ::= LOCAL_CTRL_DENIED [<(> [<REJECT REASON CODE> <,>]
[<REJECT REASON TEXT>] <,>]

<LOCAL CTRL GRANTED MSG> ::= LOCAL_CTRL_GRANTED