ASTM INTERNATIONAL

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

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

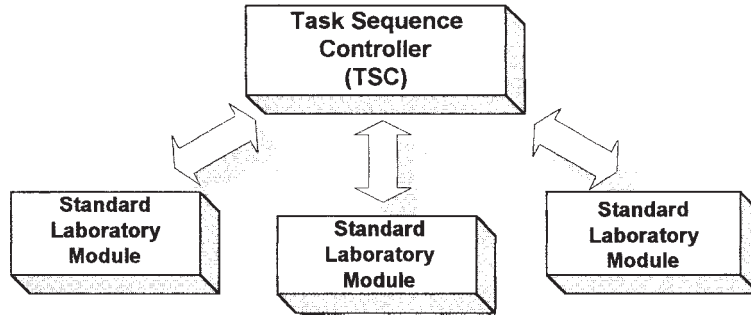|	
<FIELD>	non-terminal string
::=	is defined as
│	choice of either field on each side
[]	optional one or none
{}	0 or more repetitions
{}xy	at least x, maximal y repetitions
(MSB)	this field is most significant bit or byte in multi bit or byte sequence
BOLD	terminal ASCII or hex symbol, in bold
Xx .. 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
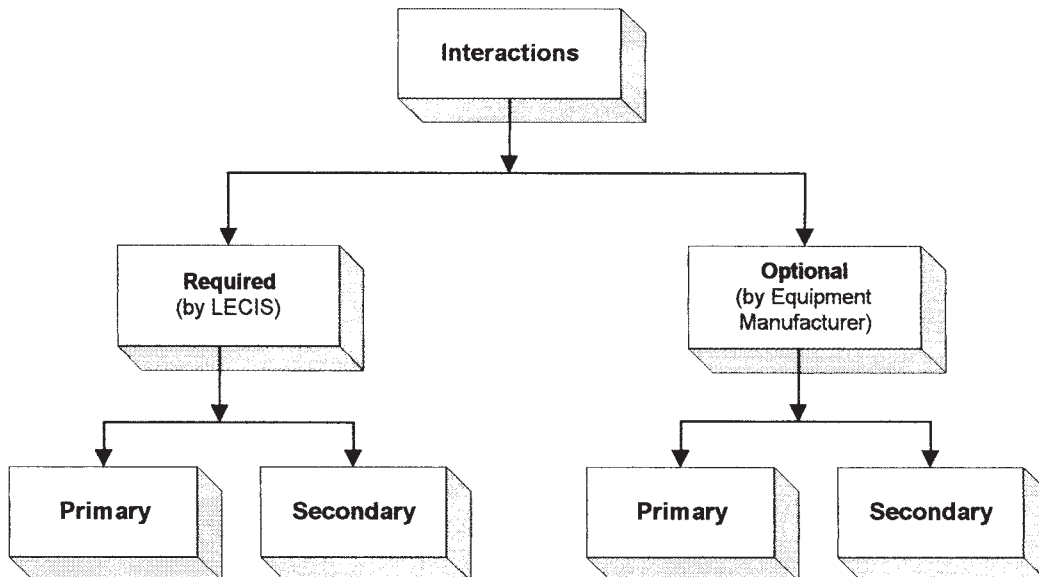

FIG. 2 Interaction Categories

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

TABLE 3 Harel State Chart Symbols

⬭	State
↓	Transition
↘	Default Entry Point
Ⓗ	History Selector

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

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 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.
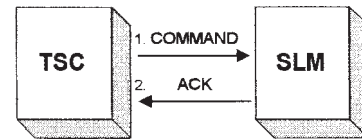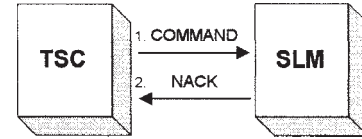

FIG. 3 Command/ACK Exchange

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 date 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.
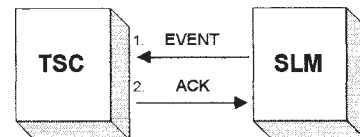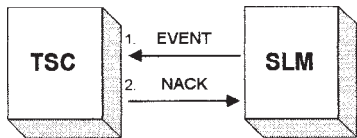

FIG. 5 Event/ACK Exchange

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: yyyymmddh-hmmssss
	  August 10, 1996, 14:22:10.33 ->
	1996081014221033
	or
	Counter:
	  00000000012345
	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 chart for the Control
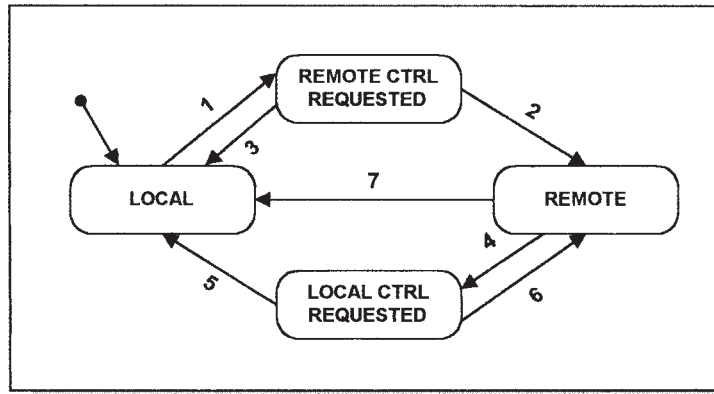
FIG. 7 Local/Remote Control State Chart

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     ::= REMOTE_CTRL_ACCEPTED
MSG>
 <REMOTE CTRL DENIED      ::= REMOTE_CTRL_DENIED [<(> [<RE-
MSG>                       JECT 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**

<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

6.5.1.3 *Message Scenario 1*:
  ⇐ **LOCAL_CTRL_REQ**
  ⇒ **LOCAL_CTRL_GRANTED**

6.5.1.4 *Message Scenario 2*:
  ⇐ **LOCAL_CTRL_REQ**
  ⇒ **LOCAL_CTRL_DENIED** (″CONTROLLER OVERWRITE″)

  Messages in EBNF-Syntax

  <ESTOP MSG>   ::= **ESTOP**

6.5.1.5 *Message Scenario*:
  ⇒ **ESTOP**

6.5.1.6 *Message Scenario*:
  ⇐ **STATE_CHANGED** (, ″ESTOPPED″)

6.6 *Event Management*:

6.6.1 The Next Event Interaction lets the TSC control the flow of event reports from the SLM. The command message initiating this interaction gives the SLM "permission" to place the next event report into the communication channel. Every event report sent by the SLM to the TSC must be sent through the Next Event interaction.

6.6.2 Like every interaction, the Next Event Interaction message exchange is asynchronous. If no event report message is waiting for transmission when the SLM receives the NEXTEVENT command, the SLM simply waits until an event report message is needed and then sends it. In other words, the permission conveyed with a NEXTEVENT command does not expire.

6.6.3 Note that in most of the interactions in this specification, a message transaction signals a state change in one interaction. In this interaction, however, the event report signals two state changes: the state change in the interaction that generated the event *and* the state change from NEXT EVENT REQUESTED to TERMINATED in the Next Event Interaction.

6.7 *Next Event Interaction*:

6.7.1 Fig. 8 shows the Next Event state chart. The messages for the event transfer are described in Table 10.

6.7.2 An SLM should provide an event queue buffer with sufficient capacity to ensure that no event report will be lost while the SLM is waiting for the NEXTEVENT command.

  Messages in BNF-Syntax

  <NEXTEVENT MSG>   ::= NEXTEVENT
  <EVENT MSG>   see 5.4

  Message Scenario

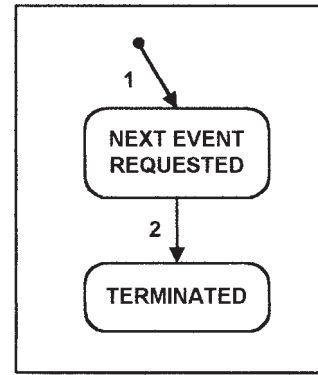 ⇒ **NEXTEVENT**
  ⇐ 08109614221033, 08109614204023, NO_ALARMS

FIG. 8 Next Event State Chart

TABLE 10 Message for Next Event Interaction

Number	Message Exchange	Old State	New State	Comment
1	⇒ NEXTEVENT	None	NEXT EVENT REQUESTED	During the transmission of the event queue additional event reports may have to be queued.
2	⇐ INTERACTION ID, EVENT TIME, EVENT ID, EVENT ARG LIST[A]	EVENT TRANSFER REQUESTED	TERMINATED	The SLM sends the next available event to the TSC.

[A] See 4.4.

7. Operation Management

7.1 *Overview*:

7.1.1 This section describes interactions to set up the SLM, engage in normal operations, and handle exceptions and emergency stops.

7.1.2 The standard interactions presented in this section must accommodate a wide variety of laboratory equipment of varying complexity. Some types of laboratory equipment process one sample at a time while others handle batches of samples. Some equipment operates on a single sample at a time, while others may perform operations on many samples in parallel with each sample at a different stage in the process. Furthermore, some instruments can upload and download programs that define methods while other equipment can perform single or preprogrammed operations only. A fully automated laboratory must integrate a wide variety of SLMs using this LECIS. Thus every SLM must support the interactions presented in this specification, although not all SLMs will have internal operations that correspond to every state.

7.2 *Control Flow Interaction*:

7.2.1 The Control Flow interaction state model is a three-level hierarchical state model and so can be depicted in three state charts: Figs. 9-11. The top level states in the hierarchy, OPERATING and ESTOPPED, are shown in Fig. 9. The OPERATING state is specified as the initial entry state.

7.2.2 Within the OPERATING parent state there are three substates, as shown in Fig. 10. The CONTROL FLOW substate is specified as the initial entry substate of the interaction in the

TABLE 11 Control Flow State Transitions

Number	Old State	Transition Event	Command/Event	New State	Comments
0	POWER-ED UP	initialize command received	INIT	INITING	SLM starts to initialize
1	INITING	initialization complete	STATE_CHANGED	IDLE	Initialization phase is completed
2	IDLE	set up command received	SETUP	CONFIG-URING	SLM starts to configure
3	CONFIG-URING	configuration complete	STATE_CHANGED	NORMAL OPERA-TION	Configura-tion phase is completed
4	NORMAL OPER-ATION	clear command received	CLEAR	CLEARING	SLM start to reset its configur-ation
5	CLEAR-ING	clearing completed	STATE_CHANGED	IDLE	SLM config-uration is reset
6	any Control Flow state with PAUSED available	pause command received	PAUSE	PAUSING	SLM starts to suspend current operations
7	any Control Flow state with PAUSED available	communica-tion lost or an off-normal internal event occurs	STATE_CHANGED	PAUSING	SLM starts to suspend current operations to prevent data or material loss
8	PAUS-ING	a resumable internal condition reached	STATE_CHANGED	PAUSED	all operations are suspended
9	PAUSED	resume command received	RESUME	state from which PAUSED was entered	Previous state is indicated by the history selector.
10	any Control Flow state	emergency stop command received	ESTOP	ESTOPPED	TSC causes emergency stop
11	any Control Flow state	internal condition detected that requires the SLM to perform an emergency stop	STATE_CHANGED	ESTOPPED	SLM detects an emergency situation

TABLE 12 Default Message for Interaction State Change Reporting

Number	Message Exchange	Old State	New State
1, 3, 5, 7, 8, 11	⇐ STATE_CHANGED (OLD STATE,[A] NEW STATE)	state defined by first parameter	state defined by second parameter

[A] Optional.

TABLE 13 Messages for SLM Initialization

Number	Message Exchange	Old State	New State	Comment
0	⇒ INIT	POWERED UP	INITING	TSC instructs the SLM to initialize itself
1	⇐ STATE_CHANGED ("INITING", "IDLE")	INITING	IDLE	SLM indicates the completion of the initialization process

TABLE 14 Messages for SLM Configuration

Number	Message Exchange	Old State	New State	Comment
2	⇒ SETUP (CONFIG ID,[A] CONFIG PARAMETER*)	IDLE	CONFIG-URING	TSC configures the SLM; configuration data must be described in the SLM's capability dataset **(2)**
3	⇐ STATE_CHANGED ("CONFIGURING", "NORMAL OPERATION")	CONFIGURING	NORMAL OPERATION	SLM signals completion of the configuration process

[A] Optional.

TABLE 15 Messages for Resetting SLM Configuration

Number	Message Exchange	Old State	New State	Comment
4	⇒ CLEAR (CLEAR TYPE[A])	NORMAL OPERATION	CLEARING	The TSC instructs SLM to return to the default initial configuration
5	⇐ STATE_CHANGED ("CLEARING", "IDLE")	CLEARING	IDLE	The SLM indicates that it has returned to the default initial configuration.

[A] Optional.

7.2.4 To simplify the definition of the SLM behavior and message exchange in the Control Flow interaction, the three state charts of the Control Flow interaction state model are depicted as a single state chart in Fig. 12 and called simply the Control Flow State Chart. The parent states are drawn as dotted lines encompassing their substates.

7.3 *Behavior in the Control Flow Interaction*:

7.3.1 Once the SLM performs the initial steps to establish communications with the TSC, the SLM is expected to be in POWERED UP in the Control Flow interaction. Transitions from POWERED UP and between all other states in these models must be the result of a command from the TSC or signaled to the TSC by an event report.

7.3.2 The primary Control Flow state model has two states that allow for halting operations—PAUSED and ESTOPPED.

OPERATING parent state. This CONTROL FLOW substate is itself a parent state. Transitions from PAUSED to the correct substate within the CONTROL FLOW parent state are controlled by the history selector. The history selector is set to be the substate within CONTROL FLOW that was active upon the transition from CONTROL FLOW to PAUSING.

7.2.3 The substates of CONTROL FLOW are shown in Fig. 11. The POWERED UP substate is specified as the initial entry substate of CONTROL FLOW.

TABLE 16 Messages for Pausing the SLM

Number	Message Exchange	Old State	New State	Comment
6	⇒ PAUSE	any state	PAUSING	TSC instructs the SLM to pause operations
7	⇐ STATE_CHANGED (any state, "PAUSING")	any state	PAUSING	SLM detects an internal condition that requires pausing
8	⇐ STATE_CHANGED ("PAUSING", "PAUSED")	PAUSING	PAUSED	SLM signals that it suspended operations

TABLE 17 Messages for Resuming Operations

Number	Message Exchange	Old State	New State	Comment
9	⇒ RESUME	PAUSED	state from which PAUSING was entered	TSC instructs the SLM to resume operations

TABLE 18 Message for Emergency Stopping the SLM

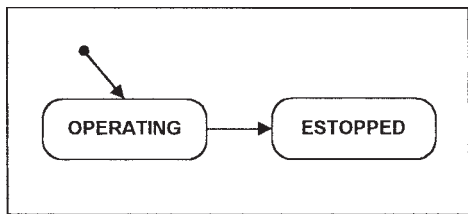| Number	Message Exchange	Old State	New State	Comment
10	⇒ ESTOP	any state	ESTOPPED	TSC instructs the SLM to perform an emergency stop.
11	⇐ STATE_CHANGED ("ESTOPPED")	any state	ESTOPPED	The SLM indicates that it performed an emergency stop.

FIG. 9 SLM Top-Level States

ESTOPPED is only used for emergency situations. The SLM is neither required nor expected to reach ESTOPPED in a configuration that preserves material being processed or from which the SLM can resume operation without manual intervention. The states PAUSING and PAUSED are provided for a graceful halt to operations.

7.4 *Description of States in the Control Flow Interaction*:

7.4.1 **POWERED UP**:

7.4.1.1 Once the SLM establishes communication with the TSC, the SLM enters the Control Flow interaction in POWERED UP. To indicate this state change the instrument must send the appropriate state change event to the TSC (see 7.5). In the POWERED UP state the SLM awaits the command INIT to starts its initialization.

7.4.1.2 Equipment initializations such as loading the operating system, warming up, or self-tests may be conducted prior to or within POWERED UP. However, the SLM may not move

any parts which are considered to be access points (ports) or do anything that will impact the environments of neighboring SLMs or laboratory personnel.

7.4.2 **INITING**—The SLM may perform power-up initializations, such as self-tests or clearing input and output buffers, that have an impact on public resources and require secondary interactions. The initialization activity is specific to each SLM. At the conclusion of the initialization, the SLM should be in its default configuration and transition to IDLE with an event report.

7.4.3 **IDLE**—The SLM has performed its power-up initializations and is waiting for the command SETUP to start the configuration process.

7.4.4 **CONFIGURING**:

7.4.4.1 The SLM is set up for operation based on the SETUP command from the TSC. The SLM configures its internal workspace for the required work. The TSC or the SLM may initiate secondary interactions in CONFIGURING.

7.4.4.2 For very simple SLMs, the CONFIGURING state may be a fall through state. More complex SLMs may have equipment parameters and operation scripts specified in the arguments to the SETUP command or the SETUP command may specify a source of these parameters that is downloaded separately from the LECIS communication.

7.4.4.3 LECIS provides two alternatives to configure the SLM: SETUP in the Control Flow state model and the RUN_OP command, described in 8.2. Arguments of the SETUP command can be used to specify equipment parameters or equipment programs, such as method or set up files. It is also possible for arguments of the RUN_OP command to specify parameters or select an instrument method file to be run that differs from the method selected or downloaded in the SETUP command. Equipment parameters or methods common to a batch of samples should be defined with the SETUP command while parameters or methods that vary from sample to sample should be defined with the arguments of the RUN_OP command.

7.4.5 **NORMAL OPERATION**—In this state, the SLM can accept a sample, perform operations on it, generate results, and create products. A normally operating SLM is expected to remain in this state indefinitely, processing until it is commanded to clear itself or receives an ESTOP or PAUSE command. Any secondary interaction can be initiated from within this state.

7.4.6 **CLEARING**:

7.4.6.1 The TSC may command the SLM to clear itself to recover from an off-normal condition, to shut down, or to return to IDLE and reconfigure with the SETUP command. In CLEARING, the SLM performs any activities required to clear itself and return to IDLE.

7.4.6.2 Normal behavior in CLEARING preserves the integrity of samples in the SLM. However, the SLM manufacturer may elect to provide an argument to the CLEAR command to control the clearing behavior. For example, one implementation of SLM clearing behavior may direct the SLM to clear without preserving sample integrity. A second implementation of clearing behavior may allow the SLM to complete
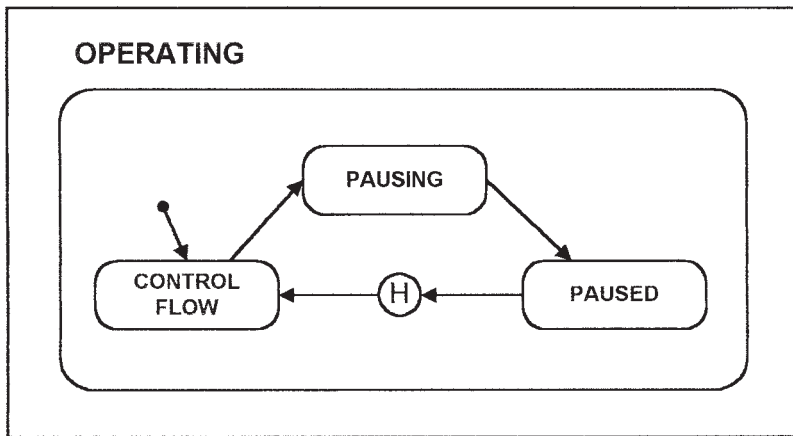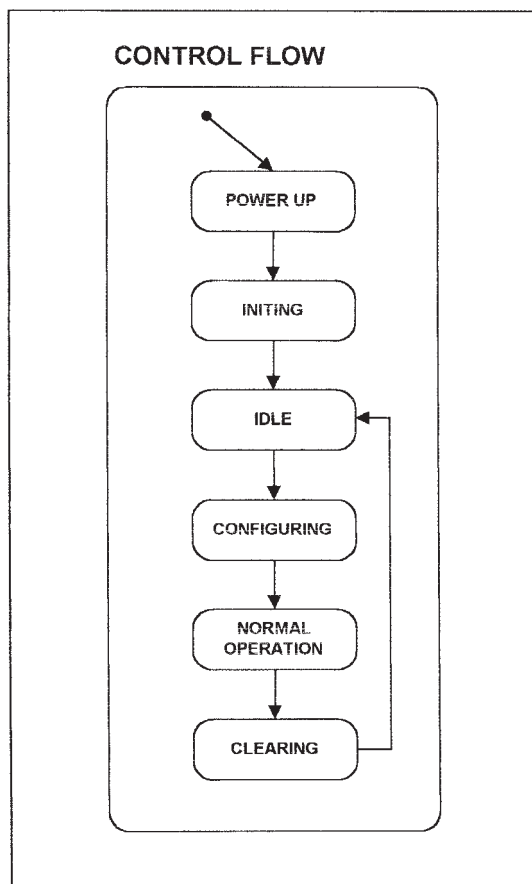
FIG. 10 Substates of Top-Level State OPERATING

FIG. 11 Substates of CONTROL FLOW State

processing to preserve the integrity of the sample or samples loaded (the normal behavior).

7.4.7 **PAUSING**:

7.4.7.1 The states PAUSING and PAUSED are provided to afford a graceful halt to sample processing. When the SLM receives a command to pause, it transitions to PAUSING. The TSC can issue a PAUSE command to the SLM at any time, unless the SLM is in ESTOPPED. In PAUSING, the SLM continues to process all ongoing secondary interactions until each interaction reaches a point where activity may be halted and resumed at a later time. A resumable point in the secondary interaction is normally a command-exit state; however, it may be an event-exit state in which the SLM is performing an operation that can be paused without jeopardizing the process. Once all ongoing interactions have reached a resumable point, the active state in the Control Flow interaction becomes PAUSED.

7.4.7.2 If the activity in some ongoing secondary interactions cannot be halted without compromising the work in progress (for example, if the SLM cannot interrupt an activity in the PROCESSING state), the SLM is expected to complete the interactions before transitioning to PAUSED. If an immediate halt of the SLM operations is required, instead of PAUSE the TSC may use the ABORT or ESTOP commands.

7.4.7.3 If the SLM initiates a transition to PAUSING and PAUSED in response to an off-normal internal event that prevents the SLM from operating properly, the off-normal situation must also be reported to the TSC with an alarm message (see 9.4). The PAUSING and PAUSED states should not be used in the course of normal operation. In particular, the PAUSING and PAUSED states should not be used to address timing issues in SLM control. The LECIS provides command-exit states in every interaction that should be used to synchronize the control of the SLM.

7.4.8 **PAUSED**—Once the SLM has halted every ongoing interaction at a resumable point (for example, when every secondary interaction has reached a command-exit state), the SLM transitions to PAUSED in the Control Flow interaction. Upon receiving the command RESUME the SLM returns to the Control Flow state from which it entered PAUSING (indicated by the history selector in the state chart) and continues processing. Upon receiving the command RESUME, the SLM also resumes the halted activities in the active state in every ongoing interaction. There is no provision to communicate the contents of the history selector to the TSC. The TSC should be implemented such that it remembers the previously active state. Alternately, after a RESUME command, the TSC can query the SLM with a status request (see 8.5).
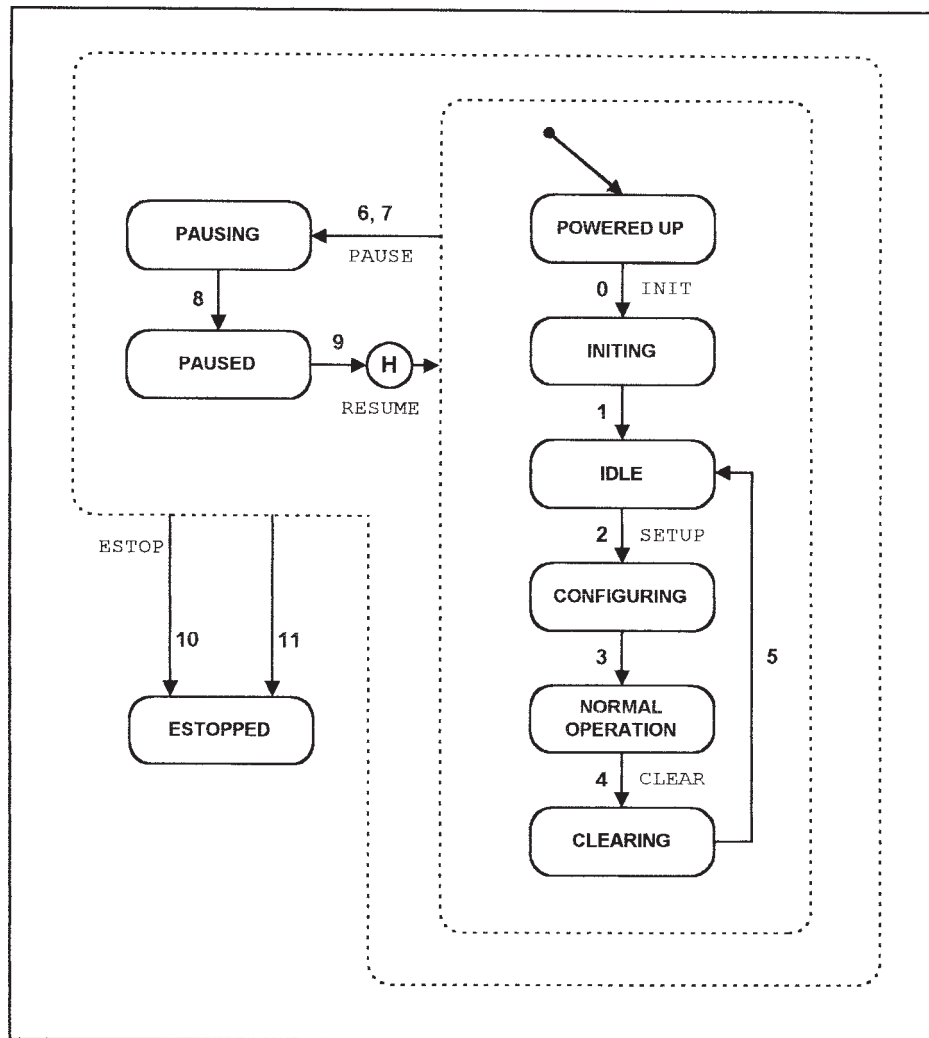
7.4.9 **ESTOPPED**:

FIG. 12 Control Flow State Chart

7.4.9.1 An emergency stop can be initiated by either the TSC or the SLM. The top level of the Control Flow state model provides a one-way transition to ESTOPPED.

7.4.9.2 In case of an emergency, whether self-detected or indicated by the TSC, the SLM must immediately stop processing. Before the SLM transitions to ESTOPPED state, it should send an appropriate alarm message (9.4) indicating the error condition if one is available. Receipt of an ESTOP command or issuance of an event report of a state change to ESTOPPED requires every secondary interaction to be terminated immediately. In the ESTOPPED state, the SLM must be in a configuration that does not pose danger to current and future activities. A transition to ESTOPPED also requires a concurrent, non-negotiated state change to local control (6.5).

7.4.9.3 LECIS does not allow SLMs to resume operation from ESTOPPED. For safety reasons, ESTOP was designed to function system-wide. In other words, all instruments that are in the same workcell as an estopped instrument must be estopped as well. The American national Standard for Industrial Robots and Robot Systems Safety Requirements stipulates that restarting from an emergency stop requires a manual,

deliberate start-up procedure (**9**). Operator intervention is required and the Control Flow interaction must be re-entered in POWERED UP.

7.5 *Default Event Report Message for Informing the TSC of a State Change*:

7.5.1 The SLM must send an event report to the TSC when the active state in an interaction changes due to an action by the SLM. In some interactions, specific event report syntax is defined. For example, the transition from ALARM terminating the Alarm interaction results in the ALARM_OFF event report. When the event report syntax is not defined, the default event message, STATE_CHANGED is used with arguments describing the original and new state. For example, the transition from PAUSING to PAUSED must be reported to the TSC with a STATE_CHANGED ("PAUSING", "PAUSED") event report message. When the SLM initiates a secondary interaction, the STATE_CHANGED message is the first message in the interaction and so there is no originating state in the STATE_CHANGED event message arguments. This default event report message format is used both in primary and secondary interactions.

14

Message in EBNF-Syntax

<STATE CHANGED MSG>	::= **STATE_CHANGED** <(> [<OLD STATE> <,>] <NEW STATE> <)>

Message Parameter

<NEW STATE>	new state data type: <STRING PARAMETER> range: pre-defined states from LECIS state models and states from optional interactions
<OLD STATE>	old state data type: <STRING PARAMETER> range: pre-defined states from LECIS state models and states from optional interactions

7.5.1.1 *Message Scenario 1*:
⇐ **STATE_CHANGED** (″CONFIGURING″, ″NORMAL OPERATIONS″)

7.5.1.2 *Message Scenario 2*:
⇐ **STATE_CHANGED** (, ″ESTOPPED″)

7.5.2 *Initializing the SLM through the INITING State*—In POWERED UP, the SLM waits for the command INIT to start the initialization process. Upon receiving INIT, the SLM transitions from POWERED UP to INITING. Once the initialization process is completed, the SLM transitions from INITING to IDLE.

Message in EBNF-Syntax

<INIT MSG>	::= **INIT**

7.5.2.1 *Message Scenario*:
⇒ **INIT**
⇐ **STATE_CHANGED** (″INITING″, ″IDLE″)

7.5.3 *Configuring the SLM through the CONFIGURING State*—Upon receiving SETUP, the SLM transitions from IDLE to CONFIGURING. The argument list of SETUP may directly specify the configuration of the SLM or, for more complex instruments, arguments name a method or configuration file that contains the configuration data. Upon completion of the configuration process, the SLM transitions from CONFIGURING to the NORMAL OPERATION.

Message in EBNF-Syntax

<SETUP MSG>	::= **SETUP** [<(> <CONFIG ID> [<,> <CONFIG PARAMETER>] <)>]

Message Parameter

<CONFIG ID>	configuration identifier data type: <STRING PARAMETER> range: unspecified
<CONFIG PARAMETER>	optional configuration parameter data type: <DATA STREAM> range: unspecified

7.5.3.1 *Message Scenario 1*:
⇒ **SETUP**
⇐ **STATE_CHANGED** (″CONFIGURING″, ″NORMAL OPERATION″)

7.5.3.2 *Message Scenario 2*:
⇒ **SETUP** (″CALIB 1″)
⇐ **STATE_CHANGED** (″CONFIGURING″, ″NORMAL OPERATION″)

7.6 *Clearing the SLM through the CLEARING State*—The CLEAR command causes the SLM to change from NORMAL OPERATION to CLEARING state. After the clearing operation is completed, the SLM transitions from CLEARING to IDLE.

Message in EBNF-Syntax

<CLEAR MSG>	::= **CLEAR** [<(> <CLEAR TYPE> <)>]

Message Parameter

<CLEAR TYPE>	type of clear data type: <MNEMONIC> range: SOFT clears all operations when integrity of the work in progress is secured HARD no consideration is given to preserving the state of the work in progress

7.6.1 *Message Scenario 1*:
⇒ **CLEAR**
⇐ **STATE_CHANGED** (″CLEARING″, ″IDLE″)

7.6.2 *Message Scenario 2*:
⇒ **CLEAR** (SOFT)
⇐ **STATE_CHANGED** (″CLEARING″, ″IDLE″)

7.7 *Pausing the SLM*:

7.7.1 The PAUSE command is used to pause the SLM's operations for an indefinite period of time. After the pausing period is over, the TSC issues a RESUME command (7.8) to the SLM to resume its paused operations.

7.7.2 An SLM should initiate a transition to PAUSING if a communication loss with the TSC occurs. This specification also allows the SLM to pause in other, self-detected, off-normal situations. An SLM-initiated transition to PAUSING must be indicated to the TSC with a STATE_CHANGED event report (7.5). In addition, when an off-normal internal event is detected that requires pausing, the SLM must communicate this to the TSC with an Alarm Interaction (see 9.4) in addition to reporting the transition to PAUSING.

Message in EBNF-Syntax

<PAUSE MSG>	::= **PAUSE**

7.7.2.1 *Message Scenario*:
⇒ **PAUSE**
⇐ **STATE_CHANGED** (″PAUSING″, ″PAUSED″)

7.8 *Resuming Operations from PAUSED*—From the PAUSED state the TSC can command the SLM to resume processing with the RESUME command. This command causes the SLM to return to the previous Control Flow state from which it entered PAUSING (indicated by the history selector in the state chart). Resumption of the active state in the Control Flow interaction indicates that every secondary interaction also resumes. If an error occurs in the process of resuming operation in any interaction, the error should be reported to the TSC from the re-entered active state using an alarm message (9.4), not the PAUSED state.

Message in EBNF-Syntax

<RESUME MSG>	::= **RESUME**

7.8.1 *Message Scenario*:
⇒ **RESUME**

7.9 *Emergency Stopping the SLM*:

7.9.1 An emergency stop can be initiated by either the TSC or the SLM. The TSC causes an emergency stop of the SLM's operations by issuing an ESTOP command. The SLM performs an emergency stop if it detects an emergency situation and signals the transition with the default event report (see 7.5). If the SLM detects an emergency, it should send an appropriate alarm message (9.4) describing the emergency prior to transitioning to ESTOPPED.

7.9.2 A transition to ESTOPPED also requires the SLM change from remote control to local control (see 6.5). Only the state change to ESTOPPED must be indicated to the TSC; the change to LOCAL is implied. A transition to ESTOPPED also requires that every secondary interaction be terminated, which is also not reported separately to the TSC. While the SLM is in ESTOPPED, it shall be in local control. Manual intervention is required to exit ESTOPPED.

Message in EBNF-Syntax

<ESTOP MSG>   ::= **ESTOP**

7.9.2.1 *Message Scenario 1*:

⇒ **ESTOP**

7.9.2.2 *Message Scenario 2*:

⇐ **STATE_CHANGED** (, ″ESTOPPED″)

8. Sample Loading and Processing

8.1 SLMs carry out processing operations on their inputs. Separate interactions are defined for discrete steps in the sample loading and processing operations. The Lock/Unlock interaction is used to control access to data or material ports of an SLM. The Lock/Unlock interaction provides explicit synchronization to allow the TSC and SLM to share common physical or logical areas. The Processing interaction is used to initiate processing operations. The *Item Available Notification* interaction is provided for the SLM to indicate the availability of material or data.

8.2 *Processing Interaction*:

8.2.1 The Processing interaction is used to initiate processing operations. The TSC should only initiate this secondary interaction when the SLM is in NORMAL OPERATION. To run multiple operations simultaneously or to convey permission to run a series of samples, the TSC may initiate multiple instances of the Processing interaction. The Processing interaction is terminated once the requested process is completed or aborted. To initiate a processing operation, the TSC provides the SLM with the operation and the necessary arguments in the RUN_OP command that cause the SLM to enter PROCESSING REQUESTED. The SLM can accept or reject the RUN_OP command from PROCESSING REQUESTED. Acceptance is indicated by an event report notifying the TSC that the SLM has entered the PROCESSING state. The SLM notifies the TSC with another event report when the internal processing operation is complete. Messages defining the Processing interaction are described in detail in Table 19.

TABLE 19 Messages for Processing

Number	Message Exchange	Old State	New State	Comment
1	⇒ RUN_OP (COMMAND ID, COMMAND ARG LIST,[A] START TIME,[A] ITEM LIST[A])	None	PROCESSING REQUESTED	TSC instructs the equipment to perform the specified operation
2	⇐ OP_STARTED	PROCESSING REQUESTED	PROCESSING	The SLM indicates start of operation to the TSC
3	⇐ OP_RESULT (RESULT DATA)	PROCESSING	PROCESSING	The SLM generates results
4	⇐ OP_COMPLETED	PROCESSING	TERMINATED	The SLM indicates completion of operation to TSC
5	⇐ OP_DENIED (REJECT REASON CODE,[A] REJECT REASON TEXT[A])	PROCESSING REQUESTED	TERMINATED	The SLM rejects the operation command

[A] Optional.

8.2.2 The Processing interaction has a path that allows reentry of the PROCESSING state with the OP_RESULT event report. The argument(s) to this event report allows the SLM to provide data to the TSC when operations in PROCESSING generate data that will not be available after the PROCESSING state is exited. SLMs should only use this path through the interaction when the results from tasks in the PROCESSING state cannot be stored and retrieved by the TSC (with the Lock/Unlock interaction) after the Processing interaction is complete. The SLM's Capability Dataset describes the SLM-specific path through the Processing interaction, allowing the TSC to determine if OP_RESULT event reports are expected.
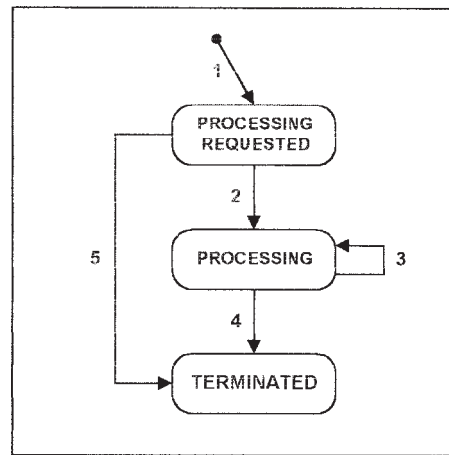

FIG. 13 Processing State Chart

Messages in EBNF-Syntax

<OP COMPLETED MSG>	::= **OP_COMPLETED**
<OP STARTED MSG>	::= **OP_STARTED**
<OP RESULT MSG>	::= **OP_RESULT** <(> <RESULT DATA LIST> <)>
<OP REJECTED MSG>	::= **OP_DENIED** <(> [<REJECT REASON CODE> <,>] [<REJECT REASON TEXT>] <)>
<RUN OP MSG>	::= **RUN_OP** <(> <COMMAND ID> [<,> <COMMAND ARGS>] [<,> [<START TIME>] [<,> <ITEM LIST>]]] <)>
<COMMAND ARG LIST>	::= <(> <COMMAND ARGS> <)>
<RESULT DATA LIST>	::= <(> <RESULT DATA> {<,> <RESULT DATA>} <)>
<ITEM LIST>	::= <(> <ITEM ID> {<,>< ITEM ID>} <)>

Message Parameter

<ITEM ID>	item identifier data type: <PARAMETER> range: unspecified
<COMMAND ID>	Identifier of the command to be started as specified in the SLM's capability dataset data type: <PARAMETER> range: unspecified
<COMMAND ARG>	command argument as specified in the SLM's capability dataset data type: <DATA STREAM> range: unspecified
<REJECT REASON CODE>	code indicating reason for rejection data type: <REASON CODE> range: –00001..–32767
<REJECT REASON TEXT>	text describing reason for rejection data type: <STRING PARAMETER> range: unspecified
<RESULT DATA>	result data, generated during processing data type: <PARAMETER> range: unspecified
<START TIME>	expected start time of operation (date & time stamp with format: yyyymmddhhmmssss) example: August 10, 1996, 14:22:10.33 -> 1996081014221033 data type: <INTEGER NUMERIC PARAMETER> range: unspecified

8.2.2.1 *Message Scenario 1*:

⇒ **RUN_OP** (″MIX″)
⇐ **OP_STARTED**
⇐ **OP_RESULT** (100, ″mg″, 200, ″mg″)
⇐ **OP_COMPLETED**

8.2.2.2 *Message Scenario 2*:

⇒ **RUN_OP** (″DILUTE″, (″100R″, ″21MIN″))
⇐ **OP_STARTED**
⇐ **OP_RESULT** (100, ″mg″)
⇐ **OP_RESULT** (112, ″mg″)
⇐ **OP_RESULT** (114, ″mg″)
⇐ **OP_COMPLETED**

8.2.2.3 *Message Scenario 3*:

⇐ **RUN_OP** (″SCREEN″, , 1996121108342123, (I1))
⇐ **OP_DENIED** (–1999, ″LOW RESOURCES″)

8.3 *Lock/Unlock Interaction*:

8.3.1 The Lock/Unlock interaction is used to control access to data or material ports of an SLM. The Lock/Unlock interaction provides explicit synchronization to allow the TSC and SLM to share common physical or logical areas. Locking transfers ownership of an SLM port to the TSC. The TSC must explicitly unlock the port in order to transfer ownership of it back to the SLM. While the port is locked, the SLM is prohibited from accessing the port physically or logically. Fig. 14 shows the state chart of the Lock/Unlock interaction. Table
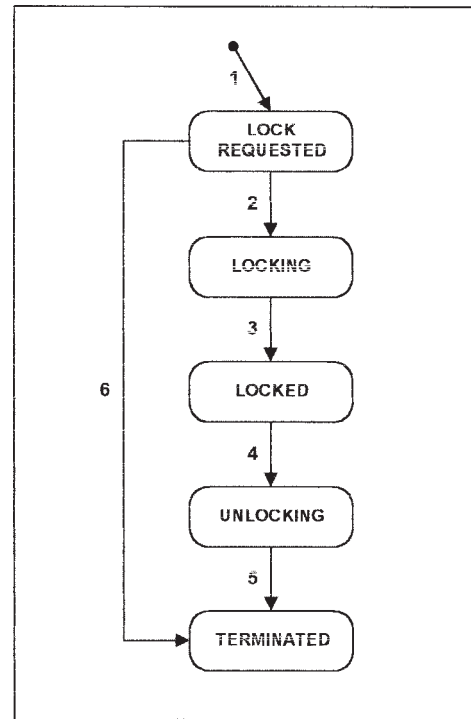

FIG. 14 Lock/Unlock State Chart

17

20 describes the messages for each state transition in the Lock/Unlock interaction.

Messages in EBNF-Syntax

<LOCK REQ MSG>	::=	**LOCK_REQ** <(> <PORT LIST> <)>
<LOCK ACCEPTED MSG>	::=	**LOCK_ACCEPTED**
<LOCK REJECTED MSG>	::=	**LOCK_DENIED** <(> [<REJECT REASON CODE> <,>] [<REJECT REASON TEXT>] <)>
<PORT LOCKED MSG>	::=	**LOCKED**
<PORT UNLOCKED MSG>	::=	**UNLOCKED**
<UNLOCK PORT MSG>	::=	**UNLOCK_REQ**
<PORT LIST>	::=	=< (> <PORT ID> {<,> <PORT INDEX ID>} <)> {<,> <(> <PORT ID> {<,> <PORT INDEX ID>} <)>}

Message Parameter

<PORT ID> port identifier as specified in the SLM's capability dataset
data type: <PARAMETER>
range: unspecified

<PORT INDEX ID> port index identifier as specified in the SLM's capability dataset
data type: <PARAMETER>
range: unspecified

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

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

8.3.2 *Message Scenario 1*:

TABLE 20 Message for Locking/Unlocking Interaction

Number	Message Exchange	Old State	New State	Comment
1	⇒ LOCK_REQ (PORT LIST)	None	LOCK REQUESTED	TSC requests to lock port(s) on the SLM
2	⇐ LOCK_ACCEPTED	LOCK REQUESTED	LOCKING PORT	SLM accepts the lock request and starts locking the specified ports
3	⇐ LOCKED	LOCKING	LOCKED	SLM signals that the specified port(s) are locked
4	⇒ UNLOCK_REQ	LOCKED	UNLOCKING	TSC requests to unlock the previously locked port(s)
5	⇐ UNLOCKED	UNLOCKING	TERMINATED	SLM signals the completion of the unlocking process
6	⇐ LOCK_DENIED (REJECT REASON CODE,[A] REJECT REASON TEXT[A])	LOCKING REQUESTED	TERMINATED	SLM rejects the lock request from the TSC

[A] Optional.

TABLE 21 Messages for the Item Available Notification Interaction

Number	Message Exchange	Old State	New State	Comment
1	⇐ ITEM_AVAILABLE (PORT ID, ITEM ID, ITEM CLASS,[A] ITEM QUANTITY[A])	NONE	TERMINATED	SLM notifies the TSC that an item is available

[A] Optional.

⇒ **LOCK_REQ** ((P1, 1, 2), (P2, 3, 5))
⇐ **LOCK_ACCEPTED**
⇐ **LOCKED**
⇒ **UNLOCK**
⇐ **UNLOCKED**

8.3.3 *Message Scenario 2*:

⇒ **LOCK_REQ** ((P1, 1, 2), (P2, 3, 5))
⇐ **LOCK_DENIED** (–3000, ″PORT ALREADY LOCKED″)

8.4 *Item Available Notification Interaction*—The SLM may notify the TSC that an item is available. This interaction is distinct from the Processing interaction to allow item to be moved asynchronously with the processing operations. This interaction involves a single message exchange and, thus, the initiating message transaction also terminates the interaction. Fig. 15 illustrates the state chart for the interaction.

Messages in EBNF-Syntax

<ITEM AVAILABLE MSG>	::=	**ITEM_AVAILABLE** <(> <PORT ID> <,> <ITEM ID> [<,> [<ITEM CLASS>] [<,> <ITEM QUANTITY>]])>

Message Parameter

<ITEM ID> item identifier
data type: <PARAMETER>
range: unspecified

<PORT ID> port identifier as specified in the SLM's capability dataset
data type: <PARAMETER>
range: unspecified

<PORT INDEX ID> port index identifier as specified in the SLM's capability dataset
data type: <PARAMETER>
range: unspecified

8.5 *Status Interaction*—The Status interaction is used by the TSC to retrieve status information from the SLM. For example, it allows the TSC to get information on the status of data and material ports, equipment inventory, active alarms, interactions, and interaction states. The SLM must process a status
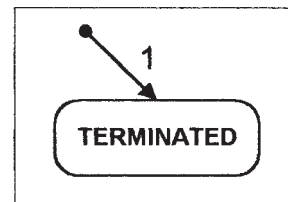

FIG. 15 State Chart for the Item Available Notification Interaction

interaction from any defined state, including the ESTOPPED state. The Status interaction is a secondary interaction. Fig. 16 shows the state chart of the Status interaction.

TABLE 22 Messages for Status Request

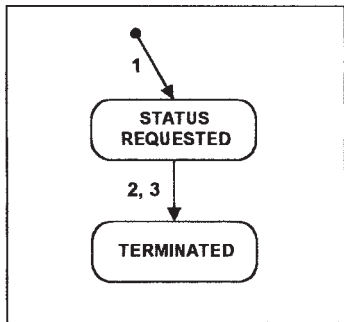| Number	Message Exchange	Old State	New State	Comment
1	⇒ STATUS_REQ (INTERACTION, INTERACTION LIST[A]) or ⇒ STATUS_REQ (INVENTORY, INVENTORY ITEM LIST[A]) or ⇒ STATUS_REQ (PORT, PORT LIST[A]) or ⇒ STATUS_REQ (ALARM, ALARM ID LIST[A])	None	STATUS REQUESTED	TSC requests the status of interactions, inventory, or ports
2	⇐ STATUS (INTERACTION STATUS LIST) or ⇐ STATUS (INVENTORY ITEM STATUS LIST) or ⇐ STATUS (PORT STATUS LIST) or ⇐ STATUS (ALARM ID LIST)	STATUS REQUESTED	TERMINATED	The SLM provides the requested status information to the TSC
3	⇐ NO_STATUS	STATUS REQUESTED	TERMINATED	SLM signals that there is no status information available for that request.

[A] Optional.

FIG. 16 Status Interaction State Chart

TABLE 23 Messages for Aborting Operations

Number	Message Exchange	Old State	New State	Comment
1	⇒ ABORT_REQ (INTERACTION ID)	None	ABORT REQUESTED	ABORT allows the TSC to abort a complete interaction
2	⇐ ABORT_ACCEPTED	ABORT REQUESTED	ABORTING	The SLM starts aborting the interaction
3	⇐ ABORT_COMPLETED	ABORT REQUESTED	TERMINATED	The SLM indicates completion of abort
4	⇐ ABORT_DENIED (REJECT REASON CODE,[A] REJECT REASON TEXT[A])	ABORT REQUESTED	TERMINATED	The SLM rejected the abort request

[A] Optional.

Messages in EBNF-Syntax

<STATUS REQ MSG	::= **STATUS_REQ** <(> (INTERACTION [<,> <INTERACTION ID LIST>]) ∣ (INVENTORY [<,>< INVENTORY ITEM LIST>]) ∣ (PORT [<,> <PORT LIST>]) ∣ (ALARM [<,> <ALARM ID LIST>]) <)>
<STATUS MSG>	::= **STATUS** <(> (<INTERACTION STATUS LIST>) ∣ (<INVENTORY ITEM STATUS LIST>) ∣ (<PORT STATUS LIST>) ∣ (<ALARM ID LIST>) <)>
<NO STATUS MSG>	::= **NO_STATUS**
<ALARM ID LIST>	::= <ALARM ID> {<,>< ALARM ID> }
<INTERACTION ID LIST>	::= <INTERACTION ID> {<,> <INTERACTION ID> }
<INTERACTION STATUS>	::= <INTERACTION ID> <,> <INTERACTION TYPE> <,> <INTERACTION STATE> [<,><OPERATION STATE>]
<INTERACTION STATUS LIST>	::= <(> <INTERACTION STATUS> <)> {<,> <(> <INTERACTION STATUS> <)>}
<INVENTORY ITEM>	::= [<ITEM CLASS>] [<,> [<ITEM ID>]]
<INVENTORY ITEM LIST>	::= <(> <INVENTORY ITEM> <)> {<,> <(> <INVENTORY ITEM> <)>}
<INVENTORY ITEM STATUS>	::= [<ITEM CLASS>] [<,> [<ITEM ID>] [<,> [<ITEM QUANTITY>]]]
<INVENTORY ITEM STATUS LIST>	::= <(> <INVENTORY ITEM STATUS> <)> {<,> <(> <INVENTORY ITEM STATUS> <)>}
<ITEM QUANTITY>	::= <VALUE> [<,> <UNIT>]
<PORT>	::= <PORT ID> {<,> <PORT INDEX ID>}
<PORT CONTENTS>	::= [<ITEM CLASS>] <,> [<ITEM ID>] <,> [<ITEM QUANTITY>]
<PORT LIST>	::= <(> <PORT> <)> {<,> <(> <PORT> <)>}
<PORT STATUS>	::= <PORT> <,> [<PORT LOCK STATE>] <,> [<PORT CONDITION>] <,> [<PORT CONTENTS>]
<PORT STATUS LIST>	::= <(> <PORT STATUS> <)> {<,> <(> <PORT STATUS> <)>}

Message Parameter

19

<ALARM ID>	alarm identifier as specified in the SLM's capability dataset data type: <REASON CODE> range: –00001..–32767
<INTERACTION ID>	unique interaction identifier data type: <INTEGER NUMERIC PARAMETER> range: unspecified (see 4.4)
<INTERACTION STATE>	current processing state of the interaction data type: <STRING PARAMETER> range: unspecified
<INTERACTION TYPE>	type of interaction data type: <STRING PARAMETER> range: unspecified
<ITEM CLASS>	describe item class data type: <PARAMETER> range: unspecified
<ITEM ID>	item identifier data type: <PARAMETER> range: unspecified
<OPERATION STATE>	status of an operation within an interaction data type: <MNENOMIC> range: ABORTED BUSY DEFERRED DONE FAILED PENDING POSTRUN PRERUN READY2RUN RUNNING SUSPENDED
<PORT CONDITION>	condition of port data type: <MNEMONIC> range: OK ERROR
<PORT ID>	port identifier data type: <PARAMETER> range: unspecified
<PORT INDEX ID>	port index identifier data type: <PARAMETER> range: unspecified
<PORT LOCK STATE>	current lock state of port data type: <MNEMONIC> range: LOCKED UNLOCKED
<UNIT>	measurement unit data type: <STRING PARAMETER> range: unspecified
<VALUE>	item value data type: <PARAMETER> range: unspecified

8.5.1 *Message Scenario 1*:

⇒ **STATUS_REQ** (INVENTORY)
⇐ **STATUS** ((SUPPLY, I101, 100, ″ml″), (SUPPLY, I102, 33, ″ml″))

8.5.2 *Message Scenario 2*:

⇒ **STATUS_REQ** (ALARM)
⇐ **NO_STATUS**

9. Error and Exception Handling

9.1 The LECIS makes several provisions to handle off-normal situations in the SLM by remote control. The Alarm interaction and Abort interaction are described in this section. The Abort interaction shall only be used for error recovery when the detected error does not jeopardize personnel safety and cannot result in equipment damage. An emergency stop (ESTOP) must occur when personnel safety is jeopardized or equipment damage is possible and may occur in other circumstances if necessary. The transition to ESTOPPED is described in 7.7.

9.2 In addition to these separate error handling mechanisms, many secondary interactions have a defined path (and event report) to TERMINATED from the entry state in the interaction. These defined alternative paths are available for the SLM to handle exceptions. They are especially useful to allow the SLM to terminate an active interaction that has been created by the TSC and is "stacked" waiting for execution.

9.3 *Abort an Interaction*:

9.3.1 The Abort interaction in this section allows the TSC to terminate any active secondary interaction from any active state. The secondary interactions are listed in Table 23. The Abort interaction effectively provides a command path from every state in each secondary interaction to TERMINATED. This interaction is provided for automated error recovery short of an emergency stop. There is no requirement that the SLM preserve the integrity of any material being processed in an operation tied to a state of the terminated interaction.

9.3.2 The SLM shall terminate the target of the abort command as soon as possible without endangering laboratory personnel or equipment. The SLM must perform an immediate transition to the TERMINATED state of the interaction from whatever state was active when the command was received. In addition to concluding the Abort interaction, the SLM must signal the state change in the aborted interaction to the TSC with an event report.

TABLE 24 Alarm State Transitions

Number	Old State	Transition Event	New State	Comments
1	None	Alarm condition is detected by the SLM	ALARM	SLM has a set of alarms that it can detect
2	ALARM	Alarm conditions are no longer detected by the SLM	TERMINATED	SLM should periodically check the existence of the alarm condition. The SLM must check the alarm condition upon receipt of a status request (8.5 Status Interaction)

TABLE 25 Messages for Alarm Indication

Number	Message Exchange	Old State	New State
1	⇐ ALARM_ON (ALARM ID, ALARM TEXT[A])	None	ALARM
2	⇐ ALARM_OFF (ALARM ID)	ALARM	TERMINATED

[A] Optional.

TABLE 26 Standard Alarm ID Codes

Alarm ID	Description
–00001	invalid state
–00002	command not supported
–00010	general operation failed
–00020	invalid configuration
–00030	invalid command format
–00031	missing command
–00032	invalid command argument
–00033	command argument out of range
–00034	missing command argument
–00035	invalid number of command arguments
–00036	invalid argument separator
–00037	invalid data
–00038	invalid data type

FIG. 17 Abort State Chart

Message in EBNF-Syntax

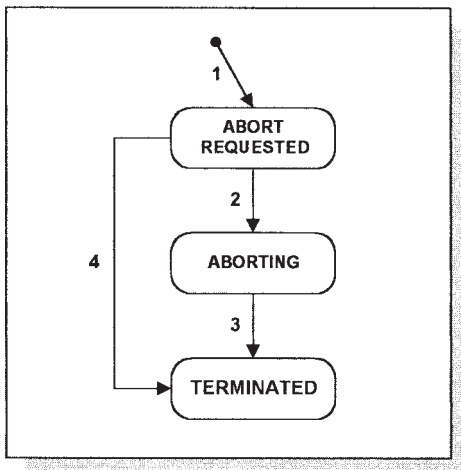|		
<ABORT REQUEST MSG>	::=	**ABORT_REQ** <(> <INTERACTION ID> <)>
<ABORT ACCEPTED MSG>	::=	**ABORT_ACCEPTED**
<ABORT COMPLETED MSG>	::=	**ABORT_COMPLETED**
<ABORT DENIED MSG>	::=	**ABORT_DENIED** <(> [<REJECT REASON CODE> <,>] [<REJECT REASON TEXT>] <)>

Message Parameter

<INTERACTION ID>	unique interaction identifier data type: <INTEGER NUMERIC PARAMETER> range: unspecified

(see 4.4)

<REJECT REASON CODE>	code indication reason for rejection data type: <REASON CODE> range: –00001..–32767
<REJECT REASON TEXT>	text describing reason for rejection data type: <STRING PARAMETER> range: unspecified

9.3.2.1 *Message Scenario*:

⇒ **ABORT_REQ** (1996121108342123)
⇐ **ABORT_ACCEPTED**
⇐ **ABORT_COMPLETED**
⇐ **STATE_CHANGED** (″LOCKING″, ″TERMINATED″)

9.3.2.2 *Message Scenario 2*:

⇒ **ABORT_REQ** (1996121108342123)
⇐ **ABORT_DENIED** (–2000, ″PROCESSING STARTED ALREADY″)

9.4 *Alarm Interaction*:

9.4.1 The Alarm interaction is used to inform the TSC when an off-normal condition has occurred in the SLM, whether or not it results in an exception, Abort interaction, or emergency stop. Every SLM should be able to detect off-normal conditions that can have an adverse impact on the safety of personnel, affect the quality of processing, prevent the completion of activity in a state, or damage equipment. If the SLM detects a condition that requires an emergency stop, it should send an ALARM_ON event report to the TSC to indicate the detected condition and transition to the ALARM state before the transition to ESTOPPED. However it is also possible for the SLM to send an ALARM_ON event report from ESTOPPED.

9.4.2 Fig. 18 shows the Alarm interaction state model. One Alarm interaction is active for each alarm condition that has been reported to the TSC. Each detectable alarm condition has a unique identification, ALARM ID, which is reported as an argument of the ALARM_ON event report. Standard alarm identifier codes are defined in 9.5. Each off-normal condition must only be reported to the TSC once to avoid creating redundant instances of the Alarm interaction and a cascade of messages that obstructs the communication channel. The Alarm interaction is terminated when the SLM detects that the alarm condition no longer exists and issues an ALARM_OFF event report. If the SLM sensors do not automatically detect when an alarm condition has been removed, the SLM should trigger a check of the condition when it processes a Status interaction.
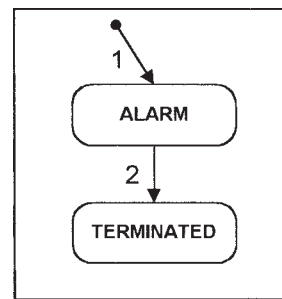

FIG. 18 Alarm State Chart

Message in EBNF-Syntax

<ALARM ON MSG>	::= **ALARM_ON** <(> <ALARM ID> [<,> <ALARM TEXT>] <)>
<ALARM OFF MSG>	::= **ALARM_OFF** <(> <ALARM ID> <)>

Message Parameter

<ALARM ID>	alarm identifier as specified in SLM capability dataset data type: <REASON CODE> range: –00001..–32767
<ALARM TEXT>	text describing the alarm data type: <STRING PARAMETER> range: unspecified

9.4.2.1 *Message Scenario*:

⇐ **ALARM_ON** (–911, ″OUT OF H20″)
⇐ **ALARM_OFF** (–911)

9.5 *Alarm ID Codes*— Alarm codes are used as arguments in the ALARM_ON event report. This Specification reserves the alarm ID codes in the range of –1 to –10000. Alarm ID codes in this range are defined to describe common conditions that can be applied to a wide variety of laboratory equipment. If an SLM is able to detect the appropriate error condition, it must use the standard alarm ID code in the Alarm interaction. The definition of standard alarm codes appears in Table 26. SLM manufacturers should define equipment-specific alarm ID codes outside this range.

ANNEX

(Mandatory Information)

A1. BASIC DATA TYPES IN EBNF SYNTAX

A1.1 Fig. A1.1 includes the *Extended Backus Naur Form* (EBNF) definitions of all data types that are used with the message definitions in this specification. EBNF descriptions begin with gross definitions. These descriptions are defined in terms of successively finer definitions.

<DATA STREAM>	::=	<ASCII PARAMETER>
		[<ESC> <BINARY DATA>]
<,>	::=	<PARAMETER SEPARATOR>
<ASCII PARAMETER>	::=	[<HEADER SEPARATOR> <PARAMETER>
		{<PARAMETER SEPARATOR> <PARAMETER>}]
		[<WHITE SPACE>]
<PARAMETER>	::=	<MNEMONIC PARAMETER>
		| <INTEGER NUMERIC PARAMETER>
		| <DECIMAL NUMERIC PARAMETER>
		| <NON-DECIMAL-NUMERIC PARAMETER>
		| <STRING PARAMETER>
<PARAMETER SEPARATOR>	::=	[<WHITE SPACE>] <COMMA> [<WHITE SPACE>]
<HEADER SEPARATOR>	::=	<WHITE SPACE>
<MNEMONIC PARAMETER>	::=	<MNEMONIC>
<INTEGER NUMERIC PARAMETER>	::=	[<SIGN>] {<DIGIT>}$_1$
<DECIMAL NUMERIC PARAMETER>	::=	<MANTISSA> [<WHITE SPACE>] [<EXPONENT>]
<NON-DECIMAL NUMERIC PARAMETER>	::=	<#> <HEX REQUEST HEADER>{<UPPER/LOWER HEX DIGIT>}$_1$
		| <#> <OCTAL REQUEST HEADER> {<OCTAL DIGIT>}$_1$
		| <#> <BINARY REQUEST HEADER>{<BINARY DIGIT>}$_1$
<STRING PARAMETER>	::=	<QUOTE>{<QUOTE DELIM CHARACTER>} <QUOTE>
		| <SINGLE QUOTE> {<SINGLE QUOTE DELIM CHARACTER>} <SINGLE
		QUOTE>
<MNEMONIC>	::=	<UPPER/LOWER CASE ALPHA> {<ALPHANUMERIC>}
<MANTISSA>	::=	[<SIGN>] [<DIGIT>]+ <DECIMAL POINT> [<DIGIT>}
		| [<SIGN>] [<DIGIT>]* <DECIMAL POINT> [<DIGIT>}$_1$
<EXPONENT>	::=	<EXPONENT HEADER> [<WHITE SPACE>] [<SIGN>] [<DIGIT>]+
<ALPHANUMERIC>	::=	<UPPER/LOWER CASE ALPHA>
		| <DIGIT>
		| <UNDERSCORE>
<UPPER/LOWER HEX DIGIT>	::=	<DIGIT> | <AF DIGIT>
<UPPER/LOWER CASE ALPHA>	::=	<UPPER CASE ALPHA> | <LOWER CASE ALPHA>
<ASCII DATA>	::=	[<REPLY HEADER> <REPLY HEADER SEPARATOR>]
		[<DATA ITEM>{<DATA ITEM SEPARATOR><DATA ITEM>}]
<DATA ITEM>	::=	<MNEMONIC DATA ITEM>
		| <NR1 NUMERIC DATA ITEM>
		| <NR2 NUMERIC DATA ITEM>
		| <NR3 NUMERIC DATA ITEM>
		| <HEXADECIMAL NUMERIC DATA ITEM>
		| <OCTAL NUMERIC DATA ITEM>
		| <BINARY NUMERIC DATA ITEM>
		| <STRING DATA ITEM>
<DATA ITEM SEPARATOR>	::=	<COMMA>
<MNEMONIC DATA ITEM>	::=	<UPPERCASE MNEMONIC>
<NR1 NUMERIC DATA ITEM>	::=	[<SIGN>] {<DIGIT>}$_1$
<NR2 NUMERIC DATA ITEM>	::=	[<SIGN>]{<DIGIT>}$_1$ <DECIMAL POINT> {<DIGIT>}$_1$
<NR3 NUMERIC DATA ITEM>	::=	[<SIGN>] }<DIGIT>}$_1$ <DECIMAL POINT> {<DIGIT>}$_1$
		<UPPER EXPONENT HEADER> <SIGN>{<DIGIT>}$_1$
<HEXADECIMAL NUMERIC DATA ITEM>	::=	<HEX HEADER>{<HEX DIGIT>}$_1$
<OCTAL NUMERIC DATA ITEM>	::=	<OCTAL HEADER> {<OCTAL DIGIT>}$_1$
<BINARY NUMERIC DATA ITEM>	::=	<BINARY HEADER> {<BINARY DIGIT>}$_1$
<STRING DATA ITEM>	::=	<QUOTE> {<QUOTE DELIM CHARACTER>} <QUOTE>
<UPPERCASE MNEMONIC>	::=	<UPPER CASE ALPHA> {<UPPERCASE ALPHANUMERIC>}

FIG. A1.1 EBNF Definitions

23

\<UPPERCASE ALPHANUMERIC\>	::=	\<UPPERCASE ALPHA\> | \<DIGIT\> | \<UNDERSCORE\>
\<UPPER CASE ALPHA W UNDERSCORE\>	::=	\<UPPER CASE ALPHA\> | \<UNDERSCORE\>
\<HEX DIGIT\>	::=	\<DIGIT\> | \<UPPER AF DIGIT\>
\<SIGN\>	::=	\<PLUS\> | \<MINUS\>
\<QUOTE DELIM CHARACTER\>	::=	\<7 BIT ASCII NO QUOTES\> |\<SINGLE QUOTE\> | \<QUOTE\>\<QUOTE\>
\<SINGLE QUOTE DELIM CHARACTER\>	::=	\<7 BIT ASCII NO QUOTES\> | \<QUOTE\> | \<SINGLE QUOTE\>\<SINGLE QUOTE\>
\<WHITE SPACE\>	::=	{\<WHITE SPACE CHARACTER \>}₁
\<WHITE SPACE CHARACTER\>	::=	\<SPACE\> | \<TAB\>

BINARY DATA (SECS II)

\<BINARY DATA\>	::=	\<BINARY DATA ARRAY\> | \<BINARY LIST\>
\<BINARY LIST\>	::=	\<BINARY LIST HEADER\>{\<BINARY DATA\>}
\<BINARY LIST HEADER\>	::=	\<BINARY LIST FLAG\>\<NUMBER LENGTH BYTES\>\<NUM LIST ITEMS\>
\<NUM LIST ITEMS\>	::=	\<LENGTH BYTE 1 (MSB)\>[\<LENGTH BYTE 2\>[\<LENGTH BYTE 3\>]]
\<LENGTH BYTE 1\>	::=	\<OCTET\>
\<LENGTH BYTE 2\>	::=	\<OCTET\>
\<LENGTH BYTE 3\>	::=	\<OCTET\>
\<BINARY DATA ARRAY\>	::=	\<BINARY ARRAY\> | \<BOOLEAN ARRAY\> | \<ASCII ARRAY\> | \<JIS-8 ARRAY\> | \<UNICODE ARRAY\> | \<JIS X0202 ARRAY\> | \<8-BYTE SIGNED INTEGER ARRAY\> | \<1-BYTE SIGNED INTEGER ARRAY\> | \<2-BYTE SIGNED INTEGER ARRAY\> | \<4-BYTE SIGNED INTEGER ARRAY\> | \<8-BYTE FLOATING POINT ARRAY\> | \<4-BYTE FLOATING POINT ARRAY\> | \<8-BYTE UNSIGNED INTEGER ARRAY\> | \<1-BYTE UNSIGNED INTEGER ARRAY\> | \<2-BYTE UNSIGNED INTEGER ARRAY\> | \<4-BYTE UNSIGNED INTEGER ARRAY\>
\<BINARY ARRAY\>	::=	\<BINARY ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<BINARY ITEM\>}
\<BOOLEAN ARRAY\>	::=	\<BOOLEAN ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<BOOLEAN ITEM\>}
\<ASCII ARRAY\>	::=	\<ASCII ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<ASCII ITEM\>}
\<JIS-8 ARRAY\>	::=	\<JIS-8 ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<JIS-8 ITEM\>}
\<UNICODE ARRAY\>	::=	\<UNICODE ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<UNICODE ITEM\>}
\<JIS X0202 ARRAY\>	::=	\<JIS X0202 ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<JIS X0202 ITEM\>}
\<8-BYTE SIGNED INTEGER ARRAY\>	::=	\<8-BYTE SIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<8-BYTE SIGNED INTEGER ITEM\>}
\<1-BYTE SIGNED INTEGER ARRAY\>	::=	\<1-BYTE SIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<1-BYTE SIGNED INTEGER ITEM\>}
\<2-BYTE SIGNED INTEGER ARRAY\>	::=	\<2-BYTE SIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<2-BYTE SIGNED INTEGER ITEM\>}
\<4-BYTE SIGNED INTEGER ARRAY\>	::=	\<4-BYTE SIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<4-BYTE SIGNED INTEGER ITEM\>}
\<8-BYTE FLOATING POINT ARRAY\>	::=	\<8-BYTE FLOATING POINT ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<8-BYTE FLOATING POINT ITEM\>}
\<4-BYTE FLOATING POINT ARRAY\>	::=	\<4-BYTE FLOATING POINT ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<4-BYTE FLOATING POINT ITEM\>}
\<8-BYTE UNSIGNED INTEGER ARRAY\>	::=	\<8-BYTE UNSIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<8-BYTE UNSIGNED INTEGER ITEM\>}
\<1-BYTE UNSIGNED INTEGER ARRAY\>	::=	\<1-BYTE UNSIGNED INTEGER ITEM FLAG\>\<NUMBER LENGTH BYTES\> \<SIZE ITEM ARRAY\>{\<1-BYTE UNSIGNED INTEGER ITEM\>}

FIG. A1.1 EBNF Definitions *(continued)*

<2-BYTE UNSIGNED INTEGER ARRAY>	::=	<2-BYTE UNSIGNED INTEGER ITEM FLAG><NUMBER LENGTH BYTES> <SIZE ITEM ARRAY>{<2-BYTE UNSIGNED INTEGER ITEM>}
<4-BYTE UNSIGNED INTEGER ARRAY>	::=	<4-BYTE UNSIGNED INTEGER ITEM FLAG><NUMBER LENGTH BYTES> <SIZE ITEM ARRAY>{<4-BYTE UNSIGNED INTEGER ITEM>}

<SIZE ITEM ARRAY>	::=	-- NUMBER OF ITEMS TIMES SIZE EACH ITEM IN OCTETS
<BINARY ITEM>	::=	<OCTET>
<BOOLEAN ITEM>	::=	<OCTET>
<ASCII ITEM>	::=	<OCTET>
<JIS-8 ITEM>	::=	<OCTET>
<UNICODE ITEM>	::=	<OCTET><OCTET>
<JIS X0202 ITEM>	::=	{<OCTET>}
<8-BYTE SIGNED INTEGER ITEM>	::=	<SIGN PLUS 7 BITS><BYTE 7><BYTE 6><BYTE 5> <BYTE 4><BYTE 3><BYTE 2><BYTE 1>
<1-BYTE SIGNED INTEGER ITEM>	::=	<SIGN PLUS 7 BITS>
<2-BYTE SIGNED INTEGER ITEM>	::=	<SIGN PLUS 7 BITS><BYTE 1>
<4-BYTE SIGNED INTEGER ITEM>	::=	<SIGN PLUS 7 BITS><BYTE 4><BYTE 3><BYTE 2><BYTE 1>
<8-BYTE FLOATING POINT ITEM>	::=	<OCTET><OCTET><OCTET><OCTET><OCTET><OCTET><OCTET> <OCTET>
<4-BYTE FLOATING POINT ITEM>	::=	<OCTET><OCTET><OCTET><OCTET>
<8-BYTE UNSIGNED INTEGER ITEM>	::=	<BYTE 8 (MSB)><BYTE 7><BYTE 6><BYTE 5> <BYTE 4><BYTE 3><BYTE 2><BYTE 1>
<1-BYTE UNSIGNED INTEGER ITEM>	::=	<BYTE 1>
<2-BYTE UNSIGNED INTEGER ITEM>	::=	<BYTE 2 (MSB)><BYTE 1>
<4-BYTE UNSIGNED INTEGER ITEM>	::=	<BYTE 4 (MSB)><BYTE 3><BYTE 2><BYTE 1>
<SIGN PLUS 7 BITS>	::=	<SIGN BIT><BIT 6 (MSB)><BIT 5><BIT 4><BIT 3><BIT 2><BIT 1> <BIT 0>
<BYTE 8>	::=	<OCTET>
<BYTE 7>	::=	<OCTET>
<BYTE 6>	::=	<OCTET>
<BYTE 5>	::=	<OCTET>
<BYTE 4>	::=	<OCTET>
<BYTE 3>	::=	<OCTET>
<BYTE 2>	::=	<OCTET>
<BYTE 1>	::=	<OCTET>
<OCTET>	::=	<BIT 7 (MSB)><BIT 6><BIT 5><BIT 4><BIT 3><BIT 2><BIT 1><BIT 0>
<BIT 0>	::=	<BIT>
<BIT 1>	::=	<BIT>
<BIT 2>	::=	<BIT>
<BIT 3>	::=	<BIT>
<BIT 4>	::=	<BIT>
<BIT 5>	::=	<BIT>
<BIT 6>	::=	<BIT>
<BIT 7>	::=	<BIT>

TERMINAL SYMBOLS

<REASON CODE>	::=	-32767..+32767 -- leading zero padded to 5 digits, sign mandatory
<OCTET>	::=	0x00 .. 0xff
<LOWER CASE ALPHA>	::=	a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
<UPPER CASE ALPHA>	::=	A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
<7 BIT ASCII NO QUOTES>	::=	0x00 .. 0x21 | 0x23 .. 0x26 | 0x28 .. 0x7f
<ASCII PRINTABLE>	::=	0x20 .. 0x7e
<DIGIT>	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<AF DIGIT>	::=	A | B | C | D | E | F | a | b | c | d | e | f
<UPPER AF DIGIT>	::=	A | B | C | D | E | F
<OCTAL DIGIT>	::=	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
<BINARY DIGIT>	::=	0 | 1
<BIT>	::=	0 | 1
<HEX HEADER>	::=	<#> H
<OCTAL HEADER>	::=	<#> Q
<BINARY HEADER>	::=	<#> B
<HEX REQUEST HEADER>	::=	H | h

FIG. A1.1 EBNF Definitions *(continued)*

```
<OCTAL REQUEST HEADER>              ::=    Q | q
<BINARY REQUEST HEADER>             ::=    B | b
<EXPONENT HEADER>                   ::=    e | E
<UPPER EXPONENT HEADER>             ::=    E
```

SECS II TERMINALS

```
<NUMBER LENGTH BYTES>                        ::=    1 | 2 | 3
<BINARY LIST FLAG>                           ::=    0x00
<BINARY ITEM FLAG>                           ::=    0x08
<BOOLEAN ITEM FLAG>                          ::=    0x09
<ASCII ITEM FLAG>                            ::=    0x10
<JIS-8 ITEM FLAG>                            ::=    0x11
<UNICODE ITEM FLAG>                          ::=    0x12
<JIS X0202 ITEM FLAG>                        ::=    0x13
<8-BYTE SIGNED INTEGER ITEM FLAG>            ::=    0x18
<1-BYTE SIGNED INTEGER ITEM FLAG>            ::=    0x19
<2-BYTE SIGNED INTEGER ITEM FLAG>            ::=    0x1a
<4-BYTE SIGNED INTEGER ITEM FLAG>            ::=    0x1c
<8-BYTE FLOATING POINT ITEM FLAG>            ::=    0x20
<4-BYTE FLOATING POINT ITEM FLAG>            ::=    0x24
<8-BYTE UNSIGNED INTEGER ITEM FLAG>          ::=    0x28
<1-BYTE UNSIGNED INTEGER ITEM FLAG>          ::=    0x29
<2-BYTE UNSIGNED INTEGER ITEM FLAG>          ::=    0x2a
<4-BYTE UNSIGNED INTEGER ITEM FLAG>          ::=    0x2c
```

ASCII TERMINALS

```
<TAB>                    ::=    0x09
<ESC>                    ::=    0x1b
<SPACE>                  ::=    0X20
<!>                      ::=    0x21
<QUOTE>                  ::=    0x22
<#>                      ::=    0x23
<SINGLE QUOTE>           ::=    0x27
<*>                      ::=    0x2a
<PLUS>                   ::=    0x2b
<COMMA>                  ::=    0x2c
<MINUS>                  ::=    0x2d
<DECIMAL POINT>          ::=    0x2e
<COLON>                  ::=    0x3a
<?>                      ::=    0x3f
<X>                      ::=    0x58
<UNDERSCORE>             ::=    0x5f
<|>                      ::=    0x7c
```

FIG. A1.1 EBNF Definitions (continued)

REFERENCES

(**1**) Griesmeyer, J. Michael, "General Equipment Interface Definition," Appendix A in "Final Report: An Enabling Architecture for Information Driven Manufacturing," SAND97-2076, August 1997. (http://infoserve.library.sandia.gov/sand_doc/1997/972076.pdf)

(**2**) Staab, Torsten A., and Kramer, Gary W., "CAALS Initial Device Capability Dataset," Version 1.7, NIST Internal Report 6294, August 1998. (www.lecis.org)

(**3**) Staab, Torsten A., "CAALS High-Level Communication Protocol (HLCP)," Version 1.11, NIST Internal Report XXXX, April 1996.

(**4**) Staab, Torsten A., Grandsard, Peter, and Kramer, Gary W., "CAALS Common Command Set," Version 1.02, NIST Internal Report XXXX, April 1996.

(**5**) Salit, Marc L., Griesmeyer, J. Michael, "System Ready Behaviors for Integration," Laboratory Robotics and Automation, Volume 9, 1997, pp. 133-118.

(**6**) Wirth, N., "What Can We Do About the Unnecessary Diversity of Notation for Syntatic Definitions," Communications of the ACM, Volume 20, Number 11, 1977, pp. 822-823.

(**7**) Salit, Marc L., Guenther, Franklin R., Kramer, Gary W., Griesmeyer, J. Michael, "Integrating Automated Systems with Modular Architecture," Analytical Chemistry, Volume 66, 1994, pp. 361A-367A.

(**8**) Harel, D., "Statecharts: A Visual Formalism for Complex Systems," Science of Computer Programming, Volume 8, 1987, pp. 231-274.

(**9**) "ANSI/RIA R15.06-1992, Standard for Industrial Robots and Robot Systems Safety Requirements," American National Standards Institute, 11 West 42nd Street, New York, NY 10036.