# Introducing System Description Language for AX.25 and Related Protocols

Eric L. Scace K3NA
10701 Five Forks Road
Frederick MD 21701
USA

## 0. Summary

This paper is part of a series of papers which provide extended finite state machine representations for AX.25 and related protocols. The state machines are depicted using state description language (SDL) graphic conventions from the 2.100 series of Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT) of the International Telecommunications Union (ITU). An extended finite state machine representation of a communications protocol such as AX.25 avoids the ambiguities associated with prose descriptions. These descriptions also compel the protocol designer to confront many of the error scenarios which arise on a communications path, and simplify the implementor's task of producing correct solutions which will interwork with solutions created by others.

This particular paper introduces the SDL symbols and explains other general aspects of SDL diagrams. It also lays out an organization of extended finite state machines which, together, perform the AX.25 link layer protocol, handle multiple simultaneous links, interact with the radio transmitter and receiver, and accomodate large data units sent by the application.

## 1. Status of Proposal

The ARRL Digital Committee proposes to completely replace the existing AX.25 Revision 2.0 state tables with the following SDL representations. The scope of the SDL embraces not only the AX.25 data link procedure used between two stations (and the intervening digipeaters), but also includes descriptions of related protocol functions:

a) segmentation and reassembly of large data units which exceed the N1 limit of a data link.
b) multiple simultaneous links on a single physical channel.
c) contention resolution for shared physical channels.
d) manipulation of the physical transmitter and receiver.

The following material is still in a draft state. You are invited to review and comment on this material. Comments are desired so that the final publication is as useful as possible to its readers.

## 2. Principles of Extended Finite State Machines

An extended finite state machine models the operation of one party on a communications channel. The condition of the machine is described by its state, a resting condition where the machine awaits input signals from either the application or from remote parties on the communications channel.

Whenever an input signal arises, the machine is triggered to execute a series of operations. These operations may include calculations, the generation of signals to the remote parties on the communications channel, and the generation of signals to the application. The sequence of operations concludes with the machine reaching again a resting state (the same state or a different one).

The entire sequence of operations performed by the machine is *atomic*. For modelling purposes, the sequence is considered to occur instantaneously, and can not be interrupted by any further event. The processing of such further events does not begin until the machine has completed the sequence of operations and has reached a resting state.

Signals may be sent between machines within the same equipment, or via the communications channel(s) to machines in other equipments. These signals are often called "primitives".

Extended finite state machines differ from their cousin, the finite state meachine, in three respects relevant to AX.25:

a) they can maintain internal variables, such as flags, sequence counters, and lists.

b) timers may be set. The expiration of a timer at a later time generates an input signal to trigger the machine to execute a specific series of operations. Timers may be stopped before they expire.

c) internal queues may be maintained. The queues are used to retain input signals (or other information) for processing at a later, more appropriate time.

## 3. Extended Finite State Machine Model for AX.25

The first figure illustrates the organization of extended finite state machines within a single equipment to implement AX.25 Revision 2.1. In accordance with industry conventions (as embodied in, for instance, the Open Systems Interconnection Reference Model), the machines which are closer to the application are shown as "higher layer" machines, whilst those machines closer to the physical channel are shown as "lower layer".

The collection of machines in the figure embraces both the data link layer and the physical layer of the standard Open Systems Interconnection Reference Model.

### 3.1 Segmentor

The segmentor state machine accepts input from the higher layer AX.25 user. If the unit of data to be sent exceeds the limits of an AX.25 I or UI frame, the segmentor breaks the unit down into smaller segments for transmission. Incoming segments are reassembled for delivery to the higher layer AX.25 user. The segmentor passes all other signals unchanged.

One segmentor exists per data link. Since a single piece of equipment may have multiple data links in operation simultaneously (e.g., to support multiple higher layer applications), there can be multiple, independently operating, segmentors within the equipment.

Due to the tight time schedule between the Committee's last working group meeting and the publication deadline for these Proceedings, it was not possible to complete a paper containing the SDL description and encodings for the Segmentor. I hope to be able to provide that paper as a handout at the Conference.

### 3.2 Data Link

The data link state machine is the heart of the AX.25 protocol. It provides all logic necessary to set-up and tear-down connections between two stations and to exchange information in a connectionless (i.e., via UI frames) and connection-oriented (i.e., via I frames with recovery procedures) manner.

One data link state machine exists per data link. Since a single piece of equipment may have multiple data links in operation simultaneously (e.g., to support multiple higher layer applications), there can be multiple, independently operating, data link machines within the equipment.

The data link state machine SDL description is contained in a companion paper found elsewhere in these Proceedings.

### 3.3 Link Multiplexor

The link multiplexor state machine allows one or more data links to share the same physical (radio) channel. It provides the logic necessary to give each data link an opportunity to use the channel, according to the rotation algorithm embedded within the link multiplexor.

One link multiplexor state machine exists per physical channel. If a single piece of equipment has multiple physical channels operating simultaneously, then an independently operating link multiplexor state machine exists for each channel.

The link multiplexor state machine SDL description is contained in a companion paper found elsewhere in these Proceedings.

### 3.4 Physical

The physical state machine exists at the physical layer of the Open Systems Interconnection Reference Model. The physical state machine manipulates the radio transmitter and receiver.

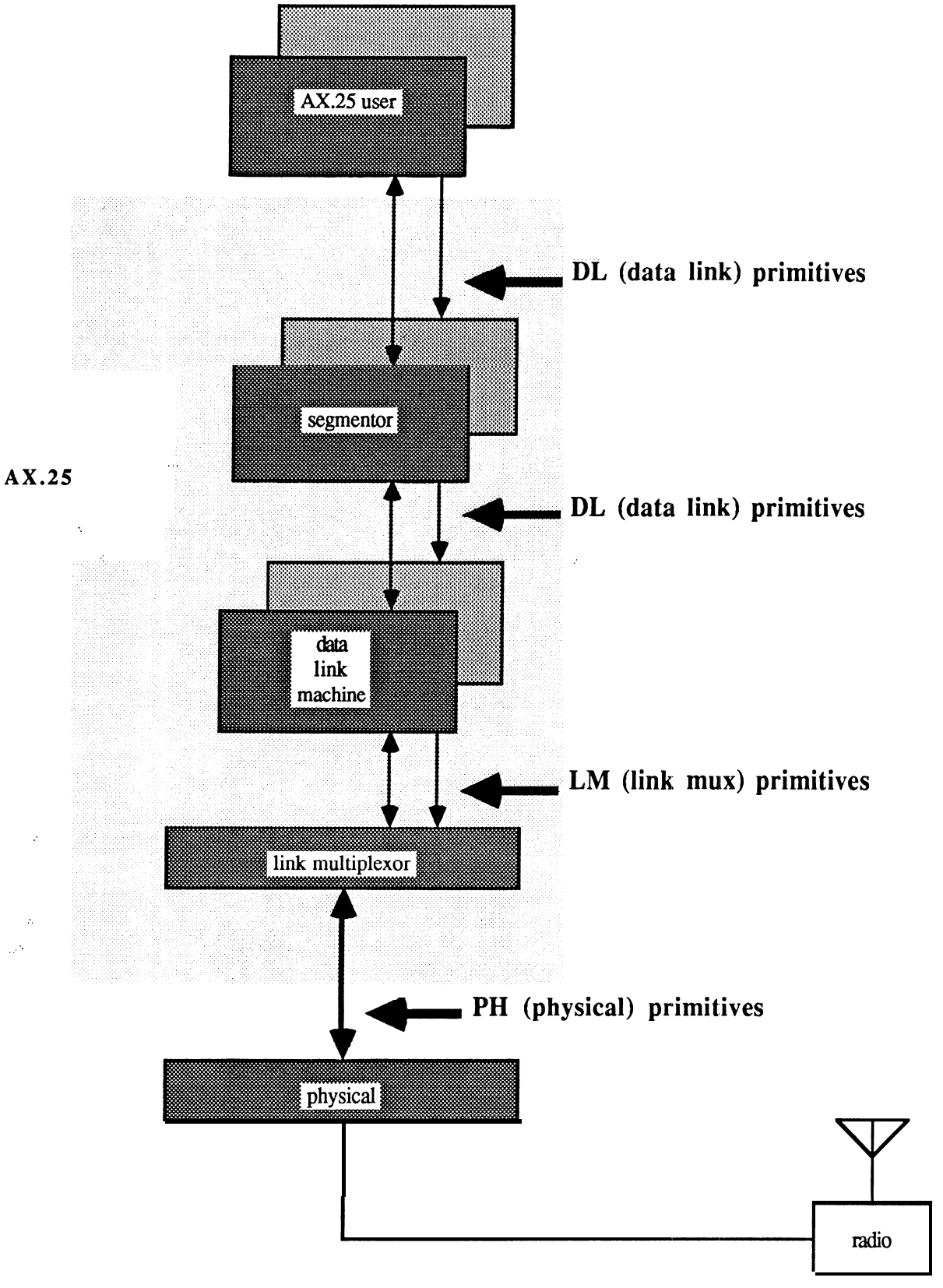
One physical state machine exists per physical channel.

Because different types of radio channel operations are used, the physical state machine comes in different flavors. The intent of each flavor is to hide the peculiar characteristics of each radio channel from the higher layer state machines. Two physical state machines have been defined in SDL:

a) a simplex physical state machine, suitable for use on simplex radio channels (such as 145.01 MHz FM) or on a classical FM repeater. This machine incorporates the CSMA/CD and p-persistence contention mechanisms to permit effective sharing of the radio channel with other stations.

b) a simple full duplex physical state machine. Although full-duplex radio channels are not extensively used to date, the full duplex physical SDL description was developed to illustrate how different flavors of physical channel can work with the other state machines proposed within AX.25 Revision 2.1. This particular full duplex state machine assumes that two stations are assigned a radio channel pair for full-time use (no sharing with other stations), such as would occur between two network backbone switching nodes. Although no sharing is envisioned, intermediate digipeaters and repeaters are allowed

# Overview of AX.25 State Machines & Primitives



for in the SDL description.

The simplex physical state machine SDL description is contained in a companion paper found elsewhere in these Proceedings. Time did not permit the completion of the paper describing the full duplex physical state machine, but I hope to have it available as a handout at the Conference.

## 4. SDL Symbol Definition

The next figure defmes the symbols used in all of the SDL graphic descriptions. You may fmd it helpful to review the text below and the figure along with an actual SDL description from one of the companion papers. The SDL descriptions combine together the operations described by the various symbols into a sequence which is read ***down the page.***

The state symbol denotes the resting states of the extended **finite** state machine. Each state is numbered and named. The sequence number simply indicates the order in which the states are drawn in the SDL. All the permitted sequences of operations **from** a given state originate below the corresponding state symbol. For convenience, each SDL machine is accompanied by **a** summary page which lists, among other things, all of the state names and their corresponding numbers.

Input signal reception (primitive) symbols have notches on either the left or right side. By convention, inputs with the notch on the left are from higher layer (or equal layer) state machines; inputs with the notch on the right are from the lower layer state machine. The name of the input primitive is labeled within the symbol. The SDL machine summary page lists all of the input primitives by name and source.

In addition, the left-notch input signal symbol is used for timer expiration. The number of the expired timer is written inside the symbol. All timers are numbered, by convention, with indications beginning **"T"** and then (usually) a three-digit number. The "hundreds" digit indicates the Open Systems Interconnection Reference Model level number at which the state machine resides; e.g., Tlxx timers are physical layer, **T2xx** are data link layer timers, etc. However, in order to prevent confusion, the present indicators **T1** and T3 are used for AX.25 timers. The SDL machine summary page lists all of the timers by their indicator, and gives a brief description of the purpose of each timer.

Similarly, output signal reception (primitive) symbols have pointers on either the left **or** right side. Output symbols pointing to the left are outputs to the higher layer (or equal layer) state machines; outputs pointing to the right are to the lower layer state machines. The name of the output primitive is written within the symbol. The SDL machine summary page lists all of the output primitives by name and destination.

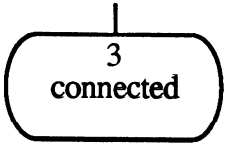
Internal signal symbols are used to post items onto queues (points to left) and to trigger the state machine when something is waiting on the queue to be popped off (notch on left). Each internal signal has **a** description label identifying which queue is involved, and what material is being posted or popped. The SDL machine summary page describes each internal queue used by the state machine.

The save symbol is used to indicate that a particular input event does ***not*** cause operations to be done in the present state. Instead, that particular input event is "saved" until the state machine (triggered by other events) has reached a new and different state.

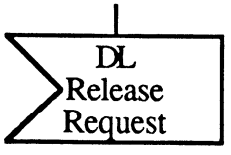
***The*** processing description symbol contains within it a description of internal action(s) executed by the state machine. Examples of these actions are starting and stopping timers, setting and clearing flags, and setting values into variables.

The test symbol is used for branching. The text written within the symbol is posed as a question, and then the appropriate branch is taken.

The subroutine symbol is used to encapsulate **frequently** used sequences of steps; the name of the subroutine is written within the symbol. The expansion of the subroutine is listed at the end of the SDL machine description. Subroutine expansions begin with a subroutine start symbol, flow down the page through the specified sequence of operations, and end with the return-from-subroutine symbol. Note that subroutines are not permitted to contain states, nor are they permitted to branch into different return legs. Each subroutine has a single point of return.

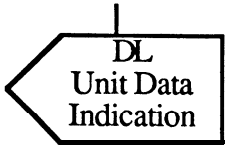


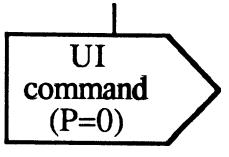

**State**



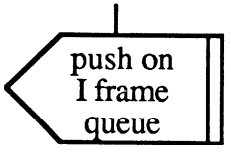

SABM
7

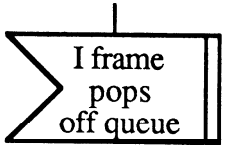
**Signal Reception**



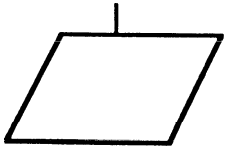




**Signal Generation**

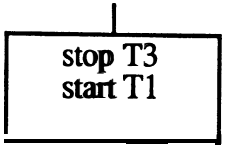


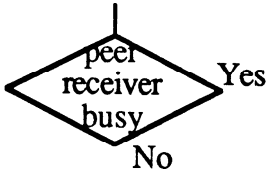

**Internal Signal Generation & Reception**



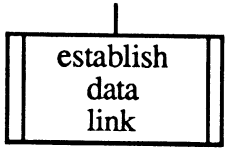
**Save a signal until a new state is reached**



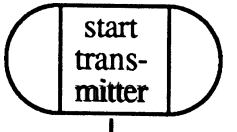

**Processing Description**



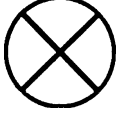

**Test**




**Subroutine Call**




**Subroutine Start**



**Return from Subroutine**