Test Suite Design for the ISDN D-channel Q.931 signaling protocol from an Estelle specification

Mokhtar Amalou and Gregor v. Bochmann
Université de Montréal
Département d'informatique et de recherche opérationnelle
Case Postale 6128 Succursale "A"
Montréal, Québec, H3C 3J7

Abstract: We describe in this paper the design of a test suite for the user side ISDN D-channel signaling protocol. A semi-automatic tool is used to generate control, data flow graphs and unparametrized test cases for each function identified by the user using a formal specification of the protocol. The test sequences generated cover the full range of the protocol as specified in the CCITT 1984 I.451 recommandations. The problems of optimising test suites including parameter testing is discussed.

1. Introduction

In recent years, the increasing demand for a wide variety of communication services (e.g. voice, text, facsimile, etc.) made it necessary to look for an economic and flexible way to satisfy these needs. So, as an alternative to various networks dedicated to one single type of service, the concepts of Integrated Services Digital Network (ISDN) has been developped. ISDN will be defined by the standardization of user interfaces and will be implemented as a set of digitals switches and paths supporting a broad range of traffic type [CCITT 84, Stal 88].

To define the requirements for ISDN Basic access on the S interface Fig. 1.1, such an ISDN will provide two circuits switched B-channels at a speed of 64 kbit/s each for user information, plus a separate D-channel at a speed of 16 kbits/s. The D-channel offers enhanced signaling capabilities to new ISDN services, since the exchange of signaling information does not interfere with the transport of user information on the B-channel.

The developpement of standards for the ISDN includes, of course, the developpement of protocols for the user-network interactions. As a result, an architecture has been defined to structure the protocols of ISDN into various layers. This architecture is called *CCITT Reference Model* [CCITT 84] and illustrated in Figure 1.2, where the signaling on the D-Channel is ruled by a layered protocol. Layering and layers have been defined in CCITT I.400 recommandations.

Since protocols and services of ISDN are complex, they have to be thoroughly tested before being incorporated in a system. An important activity is conformance testing in respect to the protocol specification [Rayn 87]. The design of a suitable test suite is a complex process. However, this process can be partially automated through the use of tools based on formal specification techniques (FDTs) with which protocos and services can be specified.

This paper discusses the test suite design and test cases generation for the user side ISDN D-channel network layer Signaling protocol which (thereafter S protocol). Test case generation is based on a formal specification of the protocol, written in Estelle, and follows a test design methodology supported by the tool CONTEST-ESTL. After a short overview of the protocol itself, Section 3 describes the formal specification in Estelle of this protocol. and the design of test sequences from Estelle specification using the CONTEST-ESTL tool. Section 4, gives a concluding discussion.

2. Q.931 Overview: services and protocol

The layer 3 signaling protocol Q.931, used on the ISDN D-channel, is message oriented and based on a set of signaling messages needed to setup, maintain and clear down 64 kbit/s circuit-switched channels accross ISDN. The process of establishing, maintaining and terminating a call occurs as a result of control signaling messages exchanged between the user and the network over a D channel. The call states and the messages for the S protocol are defined in [CCITT 84]. The use of some of these basic messages is illustrated in fig. 2.1. A common format is used for all messages defined in these recommandations. Three fields are common to all messages which we call signaling parameters:

- Protocol Discriminator (PD): used to distinguish messages for user-network call control from other
  messages types.

- Call Reference (CR): identifies the B-channel call to which this message refers.

- Message Type (MT): identifies which S protocol message is being sent.

The content of the remainder of the message depends on the message type and consist of a sequence of zero or more information elements which we call Control parameters such as:

- Bearer Capability (BC): indicates provision, by the network, of one of the bearer capabilities defined in Recommandation I.201.

- CHannel Identification (CHI): identifies the B channel within the interface for user data.

Among the defined messages that a user interface can receive and send are *Status Inq* and *Status* which can be exchanged any time regardless of the status of the protocol and play an important role for testing purposes. The former message serves to request the status of the protocol (i.e. major state) while the latter may be used to report back this information. Neither of these messages changes the state of the protocol.

3. Test suite design

The main objective of protocol testing is to verify that an implementation of the protocol conforms to the specification of the protocol. The test cases must be derived only on the basis of the protocol specification, no knowledge of a particular implementation structure should be assumed. Recently, a method has been developped for test suite design [Sari 87] based on a formal specification written in the Estelle language [Budk 87].We use this methodology to design our test cases for the S protocol.

3.1 Developing a formal protocol specification

We have written an Estelle specification of the S protocol. We subdivide the behaviour of the S protocol entity into two modules, Fig. 3.1. The *Signal* module interfaces with the Link Layer, namely the LAP-D protocol entity, via Abstract Service Primitives, and handles the peer-to-peer communication with the network side. The *Control* module interfaces with Signal module via a set of Internal Service Primitives. It also handles telecommunication service control such as checking that the bearer service, offered by the network in the BC parameter, matches the bearer service that the user is able to support. The messages exchanged between the S entity and some application entity (e.g. a user) are not explicitly defined in [CCITT 84]. Explicitly defining communication

with an application entity is deliberately avoided in the specification since it may be interpreted as an implementation requirement to the S protocol entity . In [Amal 90], we have defined a Abstract User Entity (AUE) simulating an application entity and a set of abstract user messages which may be exchanged between the AUE and the Control module of the S entity. This permits a great test control for the S entity and the use of remote test architecture with test sequences being specified in term of the interactions at the user and the network interfaces.

The set of parameters defined within the S protocol entity are distributed on the two modules. The signaling parameters belong to the Signal module and the control parameters belong to the Control module.

The Control module body is not defined in our specification. However, the specification of the Signal module is completely defined. Its specification in Estelle was obtained by translating the transition table which is provided in the CCITT document.

The standard also defines by which link service primitives the S entity PDUs will be carried. Two services primitives are used: the *DL_DATA_REQ* and the *DL_DATA_IND* .

3.2 Test design methodology

The test design methodology that we consider in this paper is based on an Estelle specification of the protocol. The method uses the specification as a basis for constructing two graphs. First, the specification is transformed into a simpler form consisting of normal form transitions . It can then be modelled by a control graph (CG) and a data flow graph (DFG). The CG represents the behaviour of the finite state automaton as described by the major state transitions of the protocol, such as the one described in Fig. 2.1. The DFG is a model of the data flow between the interaction parameters and additionnal state variables within the protocol machine. Test sequences are transition subtours derived from the CG to cover the various data flow functions identified from the DFG. This methodology is partially implemented in CONTEST-ESTL tool [Kouk 88].

3.3 Application to ISDN

We have applied this methodology to the S protocol using the CONTEST-ESTL tool and the Estelle specification described in Section 3.1. We have identified the following propreties for both the CG and the DFG:

CFP1: The CG is directed, strongly-connected and completely specified;

CFP2: Except for a very few exceptions, *Provided* clauses do not contains reference to context variables. This independence is due to the absence of certain typical functionnalities which require the use of contexte variables, such as data transfer, data flow control, multiplixing/demultiplixng.

CFP3: Transitions tours do not contains any synchronisation problems [SaBo 84]. Such problems arise for certains protocols in the distributed test architecture.

DFP1: Blocks of the DFG exhibit a simple structure, in particular, most values are transferred from the I-nodes to the O-nodes without internal transformations. In some cases, some values of O-nodes are determined from constant D-nodes. This property is the consequence of certain simplifications. In fact, we have not considered the control module body.

DFP2: We have identified three blocks in the data flow graph:

1- Block *Info_SIGNAL* which defines the data flow for signaling parameters

2- Block *Info_Timers* which defines the mechanisms of starting and stopping of different timers defined in the specification;

3- Block *Info_Control* which exhibits the exchange of control parameters between the signal module and the control module.

From the CG and the DFG graphs, a number of control functions and data flow functions are identified.

3.3.1 Control flow functions

Control flow function corresponds to control phases of communication protocols. The CG supports control functions, for instance, the following are identified for the S protocol:

- Successful call establishment
- Call establishment failure
- Successful call suspension
- Call suspension failure
- Successful call resume
- Call resume failure

- Call release

Control functions are tested by means of transition subtours, enhanced by the distinguishing sequence <Status-Inquiry, Status> which identifies the reached state. This pair of interactions can be used to improve the observation capability of the transition tour method. Thus, given a transition subtour to be applied to the implementation, the tester should add a STATE-TEST subsequence after each test step as shown in the following:

STATE-TEST

IUT		TESTER
	♦	DL_DATA_IND: PD=8; CR; MT= status-inquiry
	∅	DL_DATA_REQ: PD=8 ; CR; MT= status; CA; CS=< Current-State >

CA and CS stand for CAuse and Call State control parameters, respectively.

3.3.2 Data flow functions

-The signaling parameters do not interact within the *Info_Signal* Block. Each of them may be varied independetly of the others. The value assignements follow the definitions given previously for these parameters.

-The control parameters data flow is not generated in our work. Thus, explicitly defining the corresponding data flow functions is not possible. The exact value assignement for these parameters is mainly implementation dependant (various options to choose). However, some errors exceptions are defined in the [CCITT 84].Therefore, given a test sequence to apply to the IUT, for instance a subtour, we may identify all the relevant control parameters for the alteration process.

- Each timer defined in the Estelle specification may be tested based on the following scenario: a given timer T is started by means of the *Start_timer* procedure prior of sending a specific PDU P_1 to the tester. The same timer is stopped by means of the *Stop_timer* procedure when another specific PDU P_2 is received from the tester. The expiration of the timer T is usually indicated by means of an internal message *Timeout* followed by the sending of a PDU P_3 to the tester. Thus, we

can test each of the defined timers, by observing the flow of associated PDUs exchanged between the tester and the IUT. This process will necessitate the availability of reference timers in the tester side.

3.4. Test case preparation and notation

The data and control flow functions so far identified should be tested by means of sequencing and parameter variation tests. Transition subtours will be associated with test purposes which describe their meaning. Test purposes are split onto two classes: the proper test purposes for the correct behaviour of the protocol and the improper test purposes for error handling including inopportune/invalid messages received.

We have used a particular notation for our test cases specification. Our format is a slight variation of what is proposed in [Rath 87]. It is expressed in terms of <Subtour-header> and <Subtour_body>. The first part define the objective of the test sequence, the cause of failure or succes of the test and some comments related to the standard.

The second part is expressed in terms of protocol messages including all mandatory parameters. The messages received from the IUT are preceeded by the horizontal arrow ∅. The messages sent by the tester are preceeded by the arrow ♦. Timing requirements are represented by the timer name and the vertical arrow between the messages that start and stop the timer. As an example, the test specification for the transition subtour of Table I is given in Table II.

4. Discussion on test suite optimization in the presence of interaction parameters

We have described above the application of a test design methodology, which is partly automated, to the development of a test suite for the Q.931 ISDN protocol. While a standard test suite ISDN conformance testing is being elaborated within CCITT, a test suite as described in this paper has the advantage that it is developed based on a given set of coverage criteria. The coverage criteria on which our methodology is based has two aspects: control flow and data flow.

From the point of view of control flow, the test selection is based on the finite state machine model of the protocol specification, and two types of faults are considered: (1) output fault, where a given transition generates a wrong output, and (2) transfer fault, where a given transition leads to wrong state.

Various test case selection methods have been developed which all cover the detection of all output faults, and most transfer faults. Several methods also guarantee the detection of transfer faults under certain assumptions [Fuji 91]. The weakest method is the so-called Transition Tour, which executes all transitions at least once. It finds all output errors, however, its fault detection capability for transfer faults is limited. Therefore many people suggest the use of other more complex methods which are usually based on an approach where for each transition, not only the output is observed, but also a "distinguishing" sequence is appended which identifies the reached state and could therefore detect any transfer fault. In the case of the ISDN protocol, this is not necessary, since the protocol contains a specific facility, the status exchange, by which the test system can obtain information about the present state of the protocol entity. Under the assumption that this facility is correctly implemented, it is therefore sufficient to use a transition tour and add the status exchange after each tested transition in order to check for a transfer fault. The status exchange is quite efficient, compared with the approach using "distinguishing" sequences. We conclude that the existence of the status exchange in ISDN makes the protocol more easily testable.

From the point of view of data flow, the test case development is based on an analysis of the data flow in the protocol specification, considering the flow from input parameters and internal variables through functions to other variables and output parameters. The types of faults that are considered are the following:

(1) The data flow path leading to the definition of a new value of an internal variable is wrong, depending on input parameters, other variables, and possibly some functions.

(2) The data flow path that serves the usage of a value stored in a variable is wrong, which leads to a new (faulty) value of another variable or an output parameter which is observable.

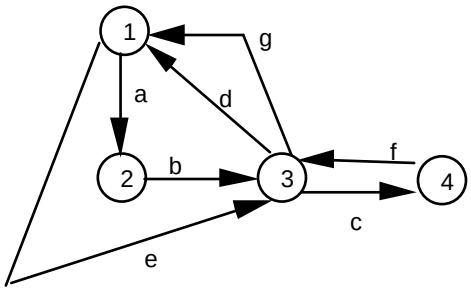
(3) Finally, one of the functions may be wrongly implemented.

For systematically testing a function, it is important to repeat tests with varying input parameters [Howd 80b]. A automated tool for this purpose is described in [Boch 90x]. Parameter variation is also often used to systematically verify that the data flow functions (point (1) and (2) above) are correctly implemented. The methodology described in Section 3.2, systematically covers the different data flow paths for definition and usage of internal variables and direct flow between input and output parameters within the same transition.

The test case development tool which we used combines, in a sense, the control flow and data flow aspects of the test suite. In fact, it uses a single transition tour, which is decomposed into a number of separate subtours, which go from the initial state back to the initial state, and use these subtours in the developement of the data flow test cases. If certain subtours are not used for the testing of data flow, then additional tests will be performed in order to cover these transitions.

However, this close relation between control flow subtours and data flow testing is not not optimal in many cases. The following example will demonstrate this. We consider the control flow shown by the state transition diagram below, for which two transition tours may be considered:

TT1: a.b.d, e.g, e.c.f.d
TT2: a.b.c.f.g, e.d



The second tour is miminal, but the subtours of the first one are shorter, on average. We assume that three data flow functions f1, f2 and f2 are to be tested, and that the data flow paths of f1 are associated with transitions e and g, the paths of f3 with a and b, and the path of f3 with a and g. We may consider several combinations of subtours for testing the data flow functions, as shown in the following table:

Subtours selected for testing

No.	Subtours used	f1(e,g)	f2(a,b)	f3(a,g)
(1)	from TT1	e.g	a.b.d	a.b.d, e.g
(2)	from TT2	e.d, a.b.c.f.g	a.b.c.f.g	a.b.c.f.g
(3)	ad hoc selection	e.g	a.b.d	a.b.g

This example shows that the subtours of the minimal transition tour TT2 are not very suitable for the testing of these data flow functions. In fact, the best choice of subtours is done independently of any given transition tours, as indicated by the third alternative, which introduces a new subtour a.b.g. The optimisation of the subtours used for a given data flow function is specially important when the tests are repeated for many different parameter values.

In fact the problem is even more difficult, since for the function f1 in our example, for instance, the value of an internal variable set during the transition e and to be tested through the output parameter provided by transition g, may be reset to zero by the transition d. If that is the case the subtours of TT2 can not be used at all for verifying the data flow of function f1, since the transition d will necessarily occur between the transitions e and g, as indicated in the table above.

We conclude that for the test suite development methodology based on a combined fault model of finite state machine (FSM) control flow and data flow control flow, it is convenient to (initially) generate separate subsuites for the testing of the control flow and data flow aspects, respectively. The optimization of each subsuite follows its own rules. Much work on the optimization of FSM test suites has already been performed, but little is known about the optimization of data flow testing.

Finally, it may be adequate to combine different test cases for different data flow functions or test cases relating to control flow and data flow, respectively. However, optimization often leads to less clear test purposes for the individual test cases, and their fault location power diminishes.

5. References

[Amal 90] M. Amalou, " Developpement de test pour le protocole de Signalisation Q.931 du RNIS basé sur une spécification formelle", Master Thesis in Preparation, Université de Montréal, Canada.

[Boch 90x] G. v. Bochmann, S. Desmarais, P. Gamache, B. Lefebvre and J. Vaucher, *Simulation and testing tools for the MAP MMS protocol*, Deliverable D5, CRIM-IBM research project "Manufacturing Message Specification (MMS)", August 1990.

[Budk 87] S. Budkowski, P.Dembinsk, " An Introduction to Estelle: A specification language for distributed systems", Computer Networks ans ISDN systems, 1988.

[CCITT 84] " Integrated Services Digital Network ", Recommandations of the Series I, 1984.

[Fuji 90] S. Fujiwara, G. v. Bochmann, F. Khendek, M. Amalou and A. Ghedamsi, *Test selection based on finite state models*, to be published in IEEE Tr. SE.

[Howd 80b] W.E. Howden, " Functionnal Program Testing" IEEE trans. on Software Engineering, vol. SE-6, No. 2 , March 1980.

[Kouk 88] V. Koukoulidis, "Complete Implementation of a Protocol Test Design Methodology" M.sc Thesis, Concordia University.

[Rath 87] Erwin P. Rathgeb, " Protocol Testing for the ISDN D-Channel Network Layer ", in Protocol Specification, Testing and Verification, Zurich, May 5-8, 1987.

[Rayn 87] D. Rayner, " Standardizing Conformance Testing for OSI", Computer Network and ISDN systems, Vol. 14, Number 1, 1987.

[Sari 84] B. Sarikaya, G.v. Bochmann, "Synchronization and Specification Issues in Protocol Testing", IEEE Trans, on Comm., COM-32, No.4, Avril 1984, pp.389-395.

[Sari 87]: Sarikaya, B., Bochmann, G.v., Cerny, E. " A Test Design Methodology for Protocol Testing " IEEE Transactions on Software Engineering, vol. SE-13, no. 5, May 1987, pp. 518-531.

[Stal 88] W. Stallings " ISDN, an Introduction " Macmillan Publishing Company, NY. 1988.

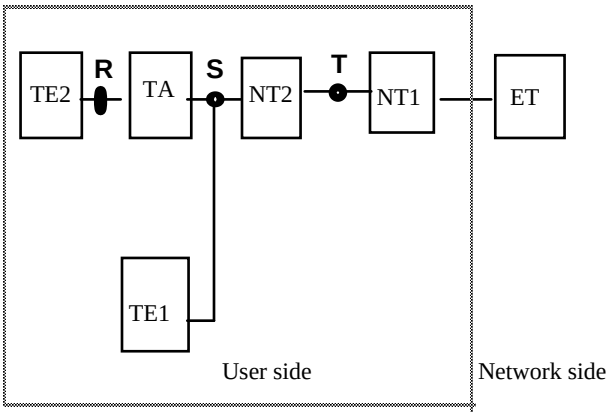
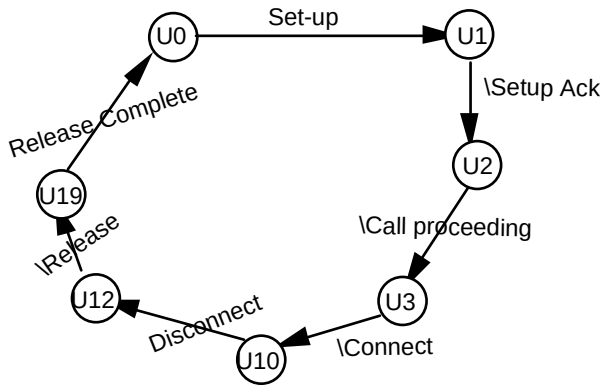
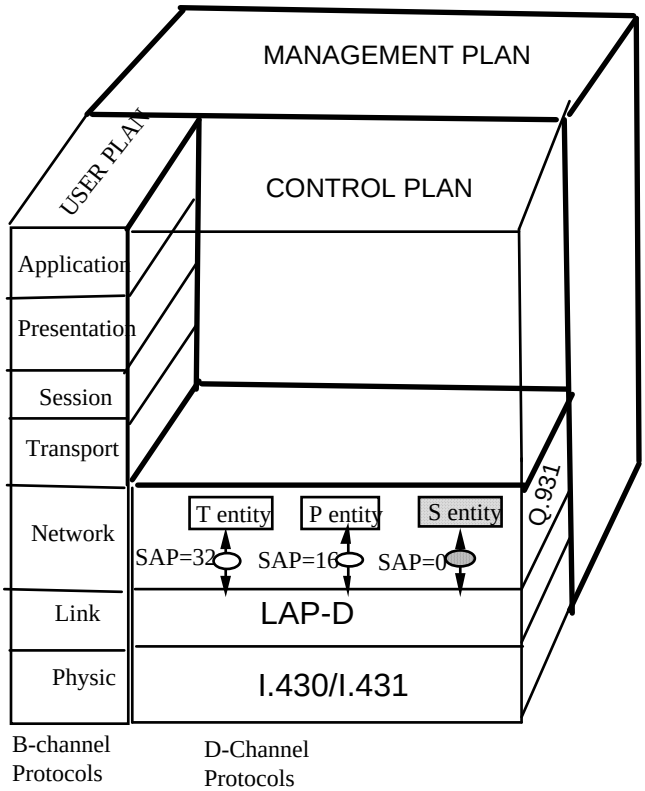


Fig. 1.1: The ISDN Basic access configuration



\ stands for network to user direction
U0 : Null **U1** : Call-Init **U2** :Overlap-sending
U3 : Outgoing-Call-Proceeding **U10** : Active
U12 : Disconnect-Request
U19 : Release-Request

Figure 2.1: Sample call establishement and release fucntions for S protocol



SAP: Service Access Point

Fig. 2.2. ISDN Protocols: Reference Model

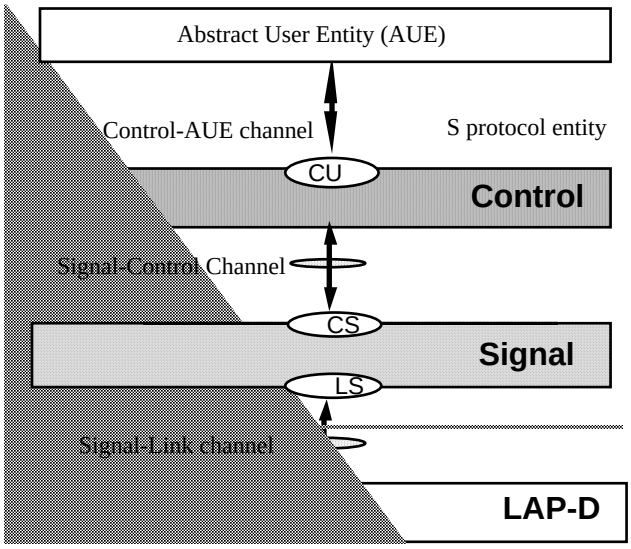


Fig. 3.1: Estelle Structure of S protocol entity

State	Input	Output	No. Trans.
null	-	Dl_data_req	1
call_init	Dl_data_ind	-	5
overl_send	-	Dl_data_req	20
overl_send	Dl_data_ind	-	21
out_call_p	-	Dl_data_req	16
disc_req	Dl_data_ind	-	56
release_req	Dl_data_ind	-	69
null			

TABLE I: Control subtour example

Table II

Objective: Call establishment failure; Test timers T303 T304 T305 T308 T310

Cause: Expiration of timer T310

Comments: Address of B-Subscriber is in Setup (DAD-1) and Information (DAD-2) messages

Initial-State= NULL

IUT **TESTER**

Start_timer(T303); Next-State=CALL-INIT

Ø DL_DATA_REQ: PD=8; CR=(0,crv);
MT= setup; BC; DAD-1

STATE-TEST

delay ≤ retrans_T303¬

♦ DL_DATA_IND: PD=8; CR = (0, crv) ;
MT= setup ack; CHI;

Stop_timer(T303)**;** Next-State=OVERLAP-SENDING

STATE-TEST

Start_timer(T304); Next-State=OVERLAP-SENDING

Ø DL_DATA_REQ: PD = 8; CR = (0, crv);
MT= info; DAD-2;

STATE-TEST

delay ≤ retrans_T304¬

♦ DL_DATA_IND: PD= 8; CR = (0, crv);
MT=call proceedimg

Stop_timer(T304);

Start_timer(T310); Next-State= OUTGOING-CALL-PROCEEDING;

STATE-TEST

delay > retrans_T310¬

Start_timer(T305); Next-State=DISC-REQUEST

Ø DL_DATA_REQ:PD=8; CR=(0, crv);

MT=disconnect; CA = timeout

STATE-TEST

delay ≤ retrans_T305¬

- DL_DATA_IND: PD=8;
  CR=(0, crv); MT=disconnect; CA ;

Stop_timer(T305); Start_timer(T308);

Next-State=RELEASE-REQUEST

- DL_DATA_REQ:PD=8;CR=(0,crv);
  MT=release

STATE-TEST

delay ≤ retrans_T308¬

- DL_DATA_IND: PD=8; CR=(0,crv);
  MT=release complete;

Stop_timer(T308); Next-State= NULL;