PROTOCOL CONVERSION BETWEEN ADLP 80 AND IEC 870 COMMUNICATION PROTOCOL STANDARDS USING FORMAL METHODS

Sufyan T. Faraj **Prof. Saleem M. R. Taha** **Mustafa M. Abd Al-jabbar**
College of Engineering-University of Baghdad

ABSTRACT
In this work, a protocol converter had been designed using formal methods in order to make a controlling station that supports ADLP 80 (ASEA) protocol standard able to communicate with a controlled station that utilizes another communication protocol called IEC 870 protocol standard, in a central control system. The two protocols are modeled using communicating finite state machine (CFSM) models. For simplicity, a converter is constructed for each application function using a conversion algorithm. This algorithm adopts a protocol data unit approach to derive the converter machine.

الخلاصة
الهدف من هذا العمل هو لتصميم محول بروتوكول باستخدام الطرق المعتمدة من أجل جعل محطة سـيطرة تستخدم بروتوكول ADLP 80 قادرة على الاتصال بمحطة طرفية تستخدم بروتوكول أخر يـدعى IEC 870 في منظومة سيطرة مركزية. تم بناء موديل رياضي باستخدام مكائن الاتصالات المنتهية الحالة. لكون هـذه البروتوكولات متعددة الوظائف، تم تصميم محول لكل وظيفة مشتركة بين هذين البروتوكـولين باسـتخدام خوارزمية تحويل. هذه الخوارزمية تتبنّى طريقة وحدة بيانات البروتوكول في بناء المحول.

KEY WORDS
Protocol conversion, ADLP 80, ASEA, IEC 870, CFSM, protocol data unit, service level, legal trace.

INTRODUCTION

It is often necessary to build a heterogeneous computer network, one made from components that were not originally designed to work together, and when this is attempted there arise at certain points in the system protocol mismatches, disparities between details of the two separate network architectures [Green 1986]. The problem of overcoming the protocol mismatches without an extensive modification of the existing systems is called the protocol conversion problem [Calvert and Lam 1989].

Protocol conversion is a complex problem since multiple protocols are considered. It is difficult to design a correct protocol converter by informal, heuristic methods. A formal approach is a reasonable choice in this area, which may minimize design errors and simplify design procedure [Tao, et al 95].

Two major concerns are being addressed in protocol conversion synthesis. First, the architectural concerns which deal with the identification of the layer at which a converter must exist, and second,

the behavioral concerns, which deal with the reconciliation between the different behaviors of protocols that result from the different protocol message formats and their orderings. According to the second concern, the protocol converter can be designed in one of two architecturally and fundamentally distinct design approaches, namely, the service level and the protocol level. Recently, a third approach has appeared which merges the last two approaches. This approach is called the hybrid approach [Saleh and Jaragh 1998].
The problem of protocol conversion can be formalized as follows:
Given two protocols $A=(A_0, A_1)$ and $B=(B_0, B_1)$, as shown in **Fig.(1)**, each protocol is schematically represented as a pair of communication finite state machines (CFSM). A communicating finite state machine is an abstraction of a process that communicates, with the environment or with other processes, by sending and receiving messages over unbounded, one-directional, FIFO channels. The abstraction is achieved by ignoring the internal data structures and internal operations of the process, and representing the process only by its external behavior, i.e., by all possible sequences of its sending and receiving operations [Gouda 1984].
Now, Machine A_0 is designed to communicate with machine A_1 according to protocol A, while B_0 and B_1 are designed to communicate according to protocol B. In each protocol system, message interchanging between one machine and the other is done through a two one-way, perfect nonlossy, infinite-capacity FIFO (first in first out) channels.
If machine A_0 needs to interact with machine B_1, a protocol mismatch exists. The aim of the conversion is to allow the components of one architecture to communicate with those of another architecture.

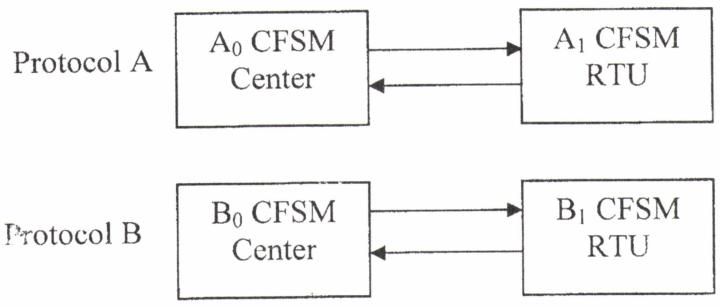

Fig. (1) Model of the two protocols A and B.

The approach in this work is to place a communicating finite state machine H between A_0 and B_1, as shown in **Fig. (2)**. Ideally, we would like H to be able to communicate with the CFSM A_0 as if were the CFSM A_1 and with the CFSM B_1 as if were the CFSM B_0 [Rajagopal and Miller 1991].

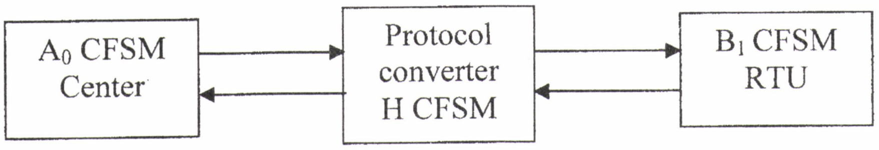

Fig. (2) Model for the protocol conversion problem.

THE CONVERSION ALGORITHM REQUIREMENTS

One of the requirements of conversion algorithm is the construction of protocol models using a special type of finite state machines called communicating finite state machines (abbreviated as CFSMs).

The protocol model is defined formally as follows [Brand and Zafiropulo 1983]:
Definition 1: A protocol is a quadruple (S_i, M_{ij}, O_i, τ) (1)

for i,j = 1 to N, where:

1- N is a positive integer representing the number of processes.
2- S_i is N disjoint finite sets (S_i represents the set of states of process i).
3- Each O_i is an element of S_i representing the initial state of process i.
4- M_{ij} are N^2 disjoint finite sets with M_{ii} empty for all i (M_{ij} represents the messages that can be sent from process i to process j).
5- τ : is a partial function mapping for each i and j,

$$\tau : S_i \times M_{ij} \rightarrow S_i \quad \text{and} \quad S_i \times M_{ji} \rightarrow S_i$$

where $\times$ is a cross product operation. In the CFSM model of protocols, each transition corresponds to a message sent or received is abbreviated as –m or +m, respectively.

Definition2: Progress properties of protocols are the properties that can be stated in terms of error conditions that may arise in the execution of the protocol. A protocol that is error free during execution is said to have the required progress properties.

The converter H is constructed by algorithm conversion using input specification M_k and S_k (defined later). Algorithm conversion is a high level algorithm that utilizes other algorithms; Composite, Trace, Synthesize, and Verify for this purpose. Algorithm Composite constructs a state space C formed by the cross product of the state space A_1 and B_0. Algorithm Trace then searches this state space C to construct a set of legal traces T_H. Algorithm Synthesize combines the trace set T_H to form the state machine H. Finally, algorithm Verify verifies that H is a valid converter i.e., it is free from errors. Below, the basic steps of the algorithm are listed. For more details about Composite, Trace, Synthesize, and Verify algorithms, see [Rajagopal and Miller 1991].

1- Input
 a) Protocols A= (A_0, A_1); B= (B_0, B_1) and
 b) Specifications S_k and M_k.
2- Apply algorithm Composite to obtain:
 a) CFSM C= A_1 X B_0 and
 b) State transition table C.
3- Apply algorithm Trace to obtain a trace set T_H
4- If T_H= empty then stop (single state converter).
5- Apply algorithm Synthesize to obtain a CFSM H and table H.
6- Apply algorithm Verify to CFSM H (Test is set to F (false) if H is found to contain errors).
7- If Test = T then H is the required converter in protocol X= (A_0, H, B_1).
8- Stop.

Where M_k and S_k are the converter and semantic specifications, respectively. Given protocols A=(A_0, A_1) and B=(B_0, B_1), a converter specification M_k for CFSM C in protocol G=(A_0, C, B_1) consists of message sets R, S, and N. Where R and S represent the sets of reception and send messages, respectively, that are considered significant for the conversion process. While, N represents the set of non-convertible send and reception messages. The sets must satisfy the following conditions:

$$R \cap S = \phi;\ S \cap N = \phi;\ N \cap R = \phi$$

The specification of message sets R, S, and N is entirely left to the designer.

Definition 3: Given two protocol messages $+m_{Ai}$ (protocol A) and $-m_{Bi}$ (protocol B), a message relationship function Ψ can be defined to map the message $+m_{Ai}$ to the message $-m_{Bj}$, i.e.,

$\Psi: +m_{Ai} \rightarrow -m_{Bj}$

if m_{Ai} is convertible or partially convertible to m_{Bj} .
The semantic specification S_k specifies a sequence of protocol functions that is a design choice in the overall protocol execution. For example, the protocol conversion may be designed to provide end-to-end or local acknowledgment depending on how S_k is specified.
Definition 4: A legal trace t is defined as an executable trace of length ≥ 1 in protocol $G=(A_0, C=A_1 \times B_0, B_1)$ and machine C provided it satisfies the following conditions:

1- Elements of t belong to the sets R, S, and N and are constrained by specifications M_k and S_k.
2- For every trace element $t(i) \in R$ there exists one and only one element $t(j) = \Psi(t(i))$, $j > i$.
3- For trace elements $t(i_1) \in R$ and $t(i_2) \in R, i_2 > i_1$, there exist elements $t(j_1) = \Psi(t(i_1))$ and $t(j_2) = \Psi(t(i_2)), j_2 > j_1$.
4- No prefix of a legal trace is a legal trace.
5- The first legal trace begins from the initial CFSM State. All legal traces may begin only from a begin or an end trace state.

The two communication protocol standards; ADLP 80 of ASEA company and IEC870; are dedicated for remote monitoring and supervisory control. Since such applications require particularly short reaction times with reduced transmission bandwidths, these protocols use only three layers, namely the physical, the link, and the application layer. Because all the layers of the two protocols are not compatible, the conversion will happen on the top of the application layer.
In fact, ASEA and IEC 870 protocol standards are multifunction, non-trivial, realistic protocols. Therefore, a modular approach has been concluded. Instead of designing one big protocol converter, a converter was designed for each application function (divide and conquer). It can be recognized that ASEA protocol consists of eight basic application functions. While IEC870 protocol is comprised of seven application functions. The application function that is missing (not implemented) in IEC 870 protocol is the selective data acquisition application function. There are four protocol conversion principles related to gateways [Zoline and Lidinsky 85]:

1- Protocol conversion can be effectively used to support communication between two entities belonging to different protocol architectures, if and only if, the protocols pertaining to these entities carry the same semantics and are therefore functionally compatible.
2- When two protocols belonging to different architectures support different sets of functions, protocol conversion can be applied only to the functional subset that is common to both protocols.
3- When two protocols are not convertible, the only solution for communicating with the other network entity is to actually implement the other entity's protocol.
4- When two protocols differ only in some respects, a mixed solution can be employed. Protocol conversion can be appreciated to the common subset and this in turn can be complemented by implementing the missing function(s) on the corresponding side.

In accordance with the above approaches, a converter for each compatible application function in both protocols is designed as described in the following sections. Except for initialization and re-start up application functions which are further divided into two distinct functions. Therefore, two converters are derived for each application function. Moreover, a converter for selective data acquisition application function is not derived, since this function exists only in **ASEA** protocol standard. Complementing this function in **IEC870** protocol standard is out of this work.

PROTOCOL CONVERTER FOR INITIALIZATION APPLICATION FUNCTION

Since initialization application function in ASEA protocol consists of two functions; namely link establishment and general interrogation, a distinct converter will be constructed for each function.

Link Establishment Converter

Fig. (3) shows the CFSM model for this function in ASEA protocol. This function is started when the controlling station (control center) sends SCI (Status_Check_Instruction) to the faulty controlled station (RTU). If the RTU is ready, it responds with EXR (Executed_Response) message. At this point, the link between control center and RTU is reestablished.

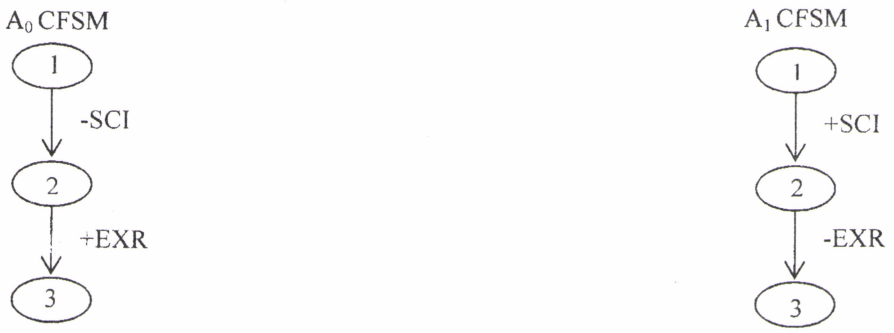

Fig. (3) CFSM models for connection establishment phase of ASEA protocol.

Fig. (4) shows the CFSM model for this function in IEC870 protocol. This function is started when the control center requests the RTU for status of link (Req.SOL). If the RTU is ready, it responds with Res.SOL. Then, the control center sends Send.RSOL (reset of link) command to the RTU, which responds by Rep.Ack (acknowledgment). Thereafter, the control center again requests the status of the link, to which is responded by Res.SOL. At this point, the control center starts to request information class 1. The RTU response is M.AA (application layer available) until the RTU is completely initialized (MEI).

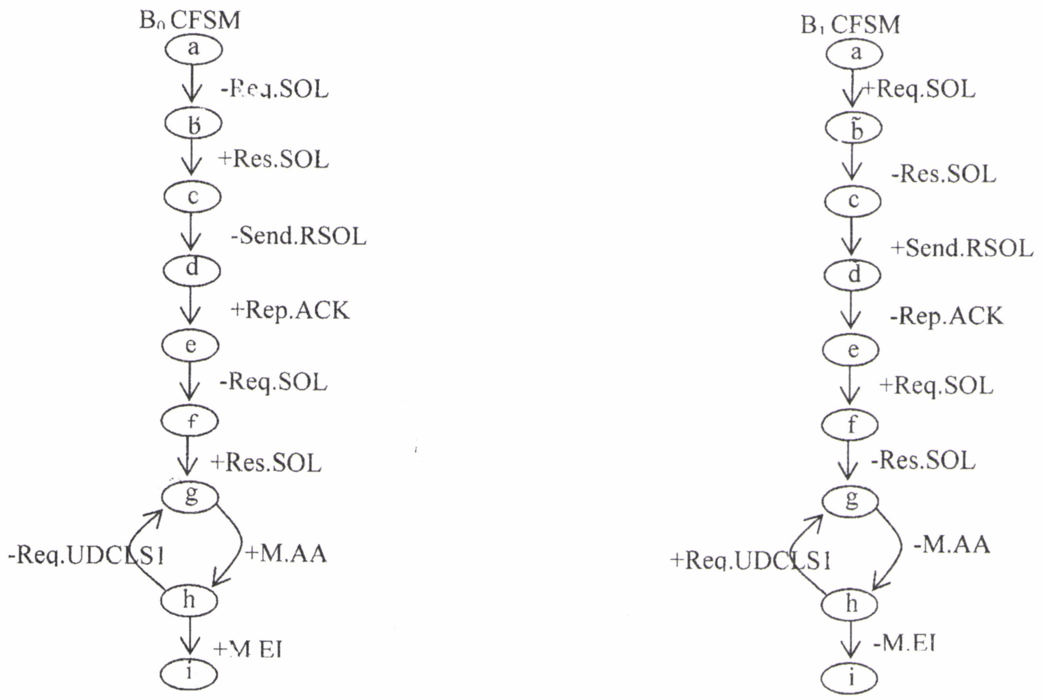

Fig. (4) CFSM model for initialization application function of IEC 870 protocol.

Semantic Specifications: When the converter receives SCI from the control center, it sends Req.SOL to the RTU. When the converter receives M.EI from the RTU, it sends EXR message to the control center.

Converter Specifications:
R= {SCI, M.EI} ≡ Convertible set of receives.
S= {Req.Sol, EXR} ≡ Converted set of sends.
N= { Res.Sol, Req^*.Sol, Send.Rsol, Rep.Ack, Req.UDCLS1, M.AA } ≡ Nonconvertible set of sends and receives

Now, the message relationship function is defined:
$\Psi: R \rightarrow S$ such that:
Ψ [+SCI]= -Req.Sol Ψ [+M.EI]= -EXR.

After applying Trace algorithm to the trivial converter $C=A_1 \times B_0$, the resulting legal traces are shown below:

T_1 = (1a, +SCI, 2a, -Req.sol, 2b)
T_2 = (2b, +Res.sol, 2c)
T_3 = (2c, -Send.Rsol, 2d)
T_4 = (2d, +Rep.Ack, 2e)
T_5 = (2e, -Req^*.Sol, 2f)
T_6 = (2f, +Res.Sol, 2g)
T_7 = (2g, -Req.UDCLS1, 2h)
T_8 = (2h, +M.AA, 2g)
T_9 = (2g, +M.EI, 2i, -EXR, 3i)

After applying Synthesize algorithm, **Fig. (5)** shows the optimized converter machine H.

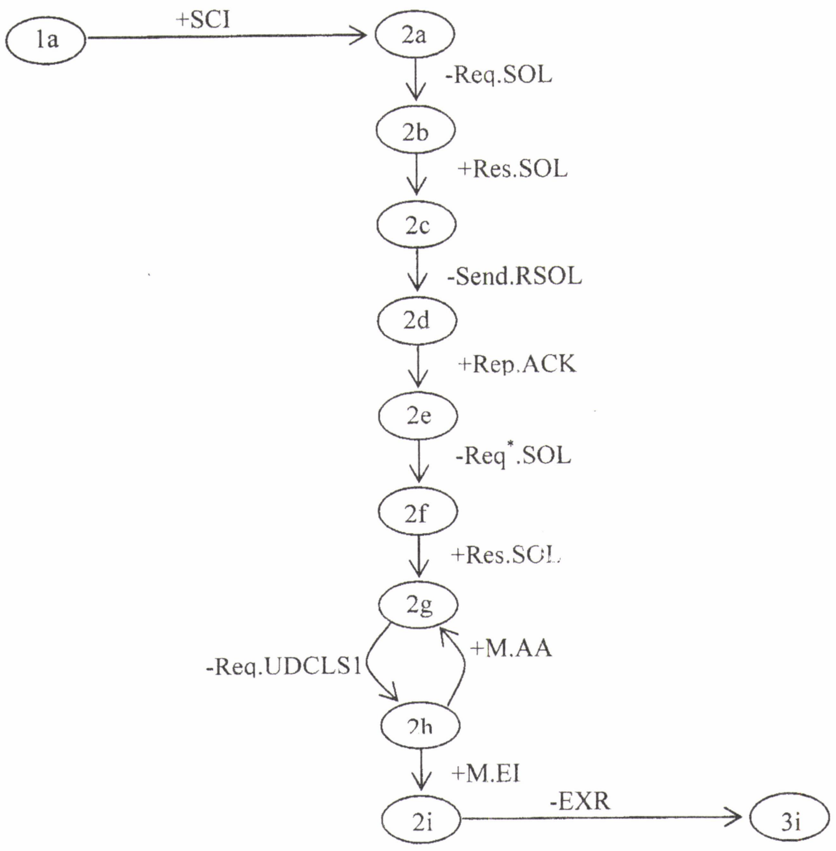

Fig. (5) converter H for link establishment.

General Interrogation Converter

Fig. (6) shows the CFSM model for this function in ASEA protocol. After link establishment, a general interrogation function is initiated. The control center starts to interrogate the RTU by sending RA (request data priority 1) and RB (request data priority 1, 2, or 3).

Fig. (7) shows the CFSM model for this function in IEC 870 protocol. This function starts when the control center sends C.IC.ACT (interrogation command) to the RTU. The RTU responds by sending C.IC.ACTCON (interrogation command confirmation) and the interrogated information

(Data1 and Data2). When the control center receives C.IC.ACTTERM (function termination) this function is finished.

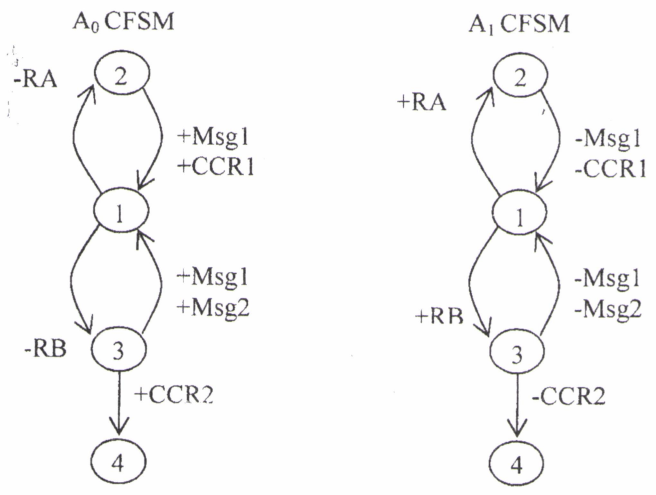

Fig. (6) General interrogation function of ASEA protocol

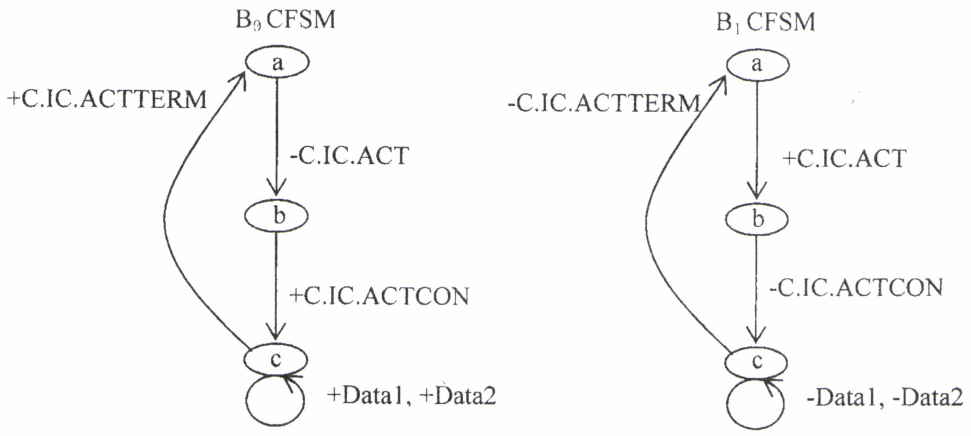

Fig. (7) CFSM model for general interrogation application function of IEC870

Semantic Specifications: When the converter receives RA or RB from the control center, it sends C.IC.ACT to the RTU and waits for confirmation and interrogated information. When the converter receives C.IC.ACTTERM from the RTU, it sends CCR2 message to the control center.
Converter Specifications:
R={C.IC.ACTTERM, Data1, Data2} ≡ Convertible set of receives.
S={Msg1, Msg2, CCR2} ≡ Converted set of sends.
N= {RA, RB, CCR1, C.IC.ACTCON, C.IC.ACT} ≡ Nonconvertible set of receives and sends.
Now, the message relationship function is defined:
Ψ [+Data1]= -Msg1, Ψ [+Data2]= -Msg2,
Ψ [+C.IC.ACTTERM]= -CCR2.
After applying Trace algorithm to the trivial converter C=$A_1 \times B_0$, the resulted legal traces are shown below:

T_1 = (1a, +RA, 2a)

T_2 = (2a, -C.IC.ACT,2b)
T_3 = (2b, +C.IC.ACTCON, 2c)
T_{41} = (2c, +Data1, 2c, +Data2, 2c, +C.IC.ACTTERM, 2d, -Msg1, 1d, +RB, 3d, -Msg2, 1d, +RA, 2d, -CCR1, 1d, +RB, 3d, -CCR2,

T_{47} = (1c, +C.IC.ACTTERM, 1d, +RB, 3d, -CCR2, 4d)
T_{48} = (1c, +RB, 3c)
T_{49} = (3c, +Data1, 3c, -Msg1, 1c)
T_{410} = (3c, +Data2, 3c, -Msg2, 1c)

4d)

T_{42} = (2c, +Data1, 2c, -Msg1, 1c)
T_{43} = (1c, +Data2, 1c, +RB, 3c, -Msg2, 1c)
T_{44} = (1c, +RA, 2c)
T_{45} = (2c, -CCR1, 1c)
T_{46} = (1c, +Data1, 1c, +RB, 3c, -Msg1, 1c)

T_{411} = (3c, +C.IC.ACTTERM, 3d, -CCR2, 4d)
T_5 = (1a, +RB, 3a)
T_6 = (3a, -C.IC.ACT, 3b)
T_7 = (3b, +C.IC.ACTCON, 3c)
T_8 = (3c, +Data1, 3c, +C.IC.ACTTERM, 3d, -Msg1 +RB, 3d, -CCR2, 4d)

After applying Synthesize algorithm, **Fig. (8)** shows the optimized converter machine H.

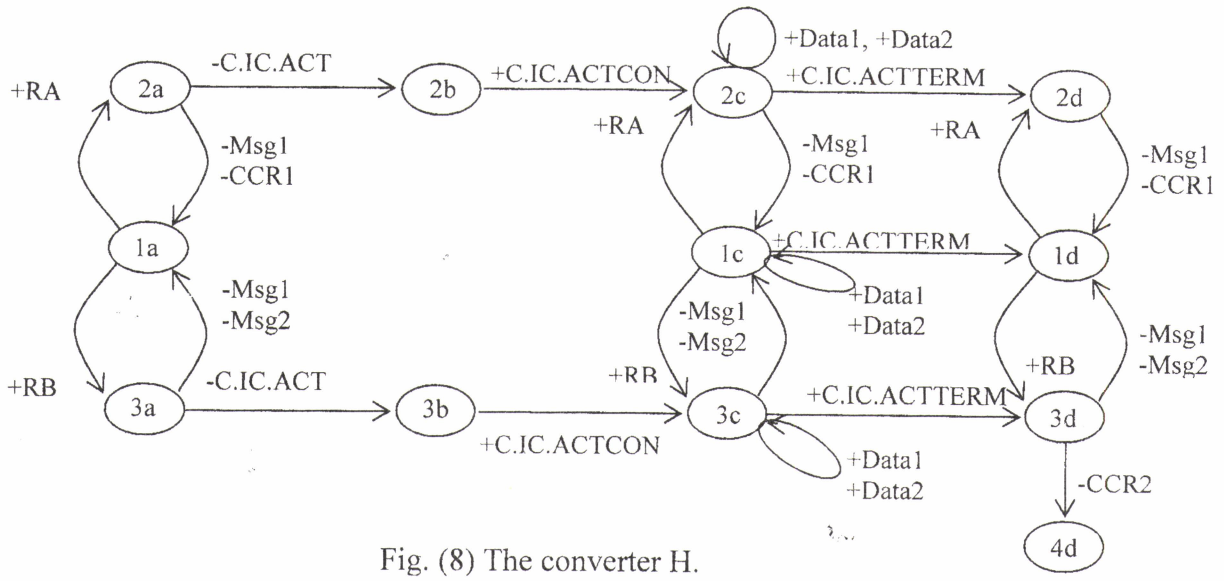

Fig. (8) The converter H.

PROTOCOL CONVERTER FOR POLLING APPLICATION FUNCTION

Fig. (9) shows the CFSM model for this function in ASEA protocol. In ASEA protocol, there are three information priorities; 1,2, and 3. When the RTU receives RA, it sends Msg1 (information priority 1) if exists otherwise it sends CCR1 (information priority 1 finished). When the RTU receives RB, it sends Msg1, Msg2 or Msg3 if exists otherwise it sends CCR2 (information priority 1,2, or 3 are not exist). Whereas, in IEC870 protocol, the data are divided into two priority levels. Fig.10 shows the CFSM model for this function in IEC 870 protocol, where:

Req.UDCLS1 ≡ Request User Data Class 1
Req.UDCLS2 ≡ Request User Data Class 2
Data1 ≡ data class 1
Data2 ≡ data class 2

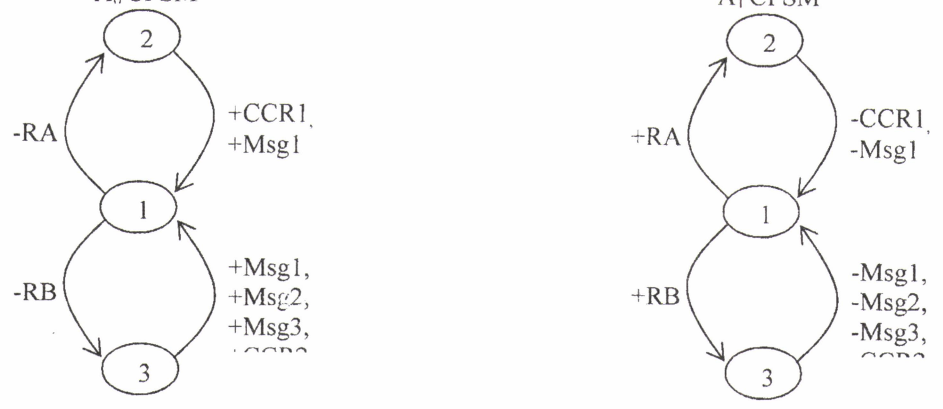

Fig. (9) A_0 and A_1 machines for polling application function in ASEA protocol.

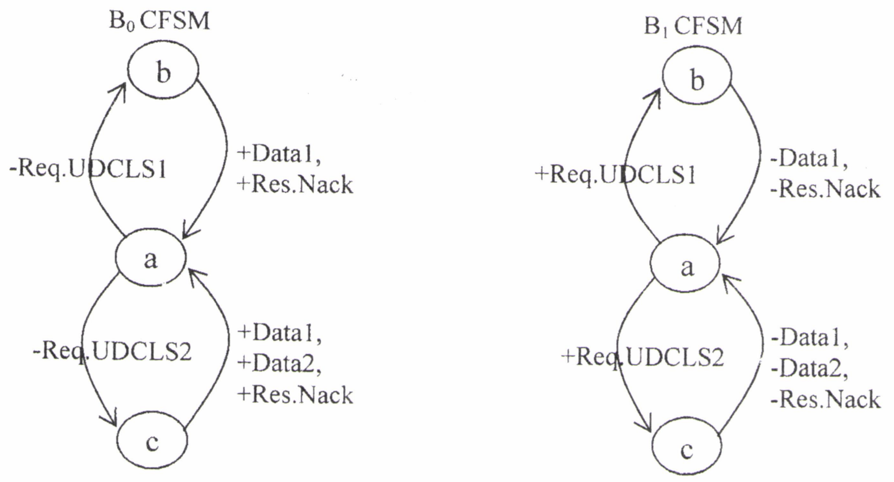

Fig. (10) CFSM model for polling application function of IEC 870 protocol.

Semantic Specifications: As obvious, all the messages of the two protocols are convertible except for Msg3.
Converter Specifications:
R={RA, RB, Data1, Data 2, Res.Nack} ≡ Convertible set of receives.
S={Req.UDCLS1, Req.UDCLS2, Msg1, Msg2, CCR1, CCR2} ≡ Converted set of sends.
N={Msg3}
Now, the message relationship function is defined:
Ψ: R $\rightarrow$ S such that:
Ψ [+RA]= -Req.UDCLS1
Ψ [+RB]= -Req.UDCLS2
Ψ [+Data1]= -Msg1
Ψ [+Data2]= -Msg2
Ψ [+Res.Nack]= [-CCR1, -CCR2]

After applying Trace algorithm to the trivial converter C=$A_1 \times B_0$, the resulted legal traces are shown below:
T_1 = (1a, +RA, 2a, -Req.UDCLS1, 2b)
T_2 = (2b, +Data1, 2a, -Msg1, 1a)
T_3 = (1a, +RB, 3a, -Req.UDCLS2, 3c)
T_4 = (1a, +Data1, 3a, -Msg1, 1a)
T_5 = (2b, +Res.Nack, 2a, -CCR1, 1a)
T_6 = (3c, +Data2, 3a, -Msg2, 1a)
T_7 = (3c, +Res.Nack, 3a, -CCR2, 1a)
After applying Synthesize algorithm, **Fig. (11)** shows the optimized converter machine H.

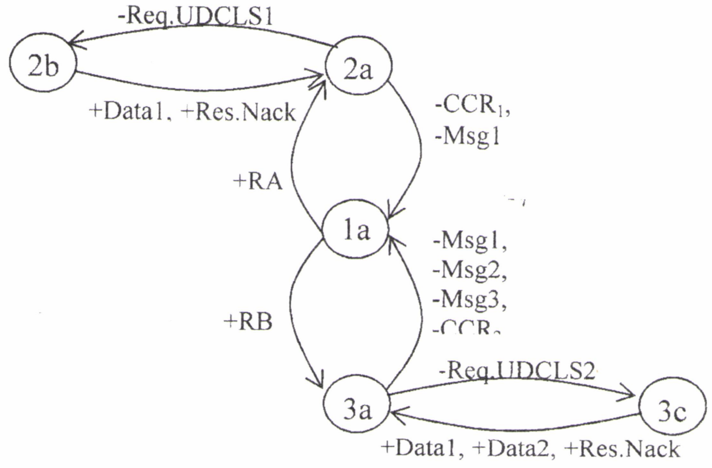

Fig. (11) The converter machine H for the polling application function.

PROTOCOL CONVERTER FOR COMMAND TRANSMISSION APPLICATION FUNCTION

In both protocols, there are two approaches to implement this application function; namely; the single step and double step command transmission. **Fig. (12)** and **Fig. (13)** show the CFSM models for this function in ASEA and IEC870 protocols, respectively. In the first approach, command activation is directly transmitted to the RTU. After command implementation, a confirmation response message is returned. While in the second approach, a select command is firstly transmitted to the RTU. Then, if the selection response is positive, command activation is transmitted to the RTU.

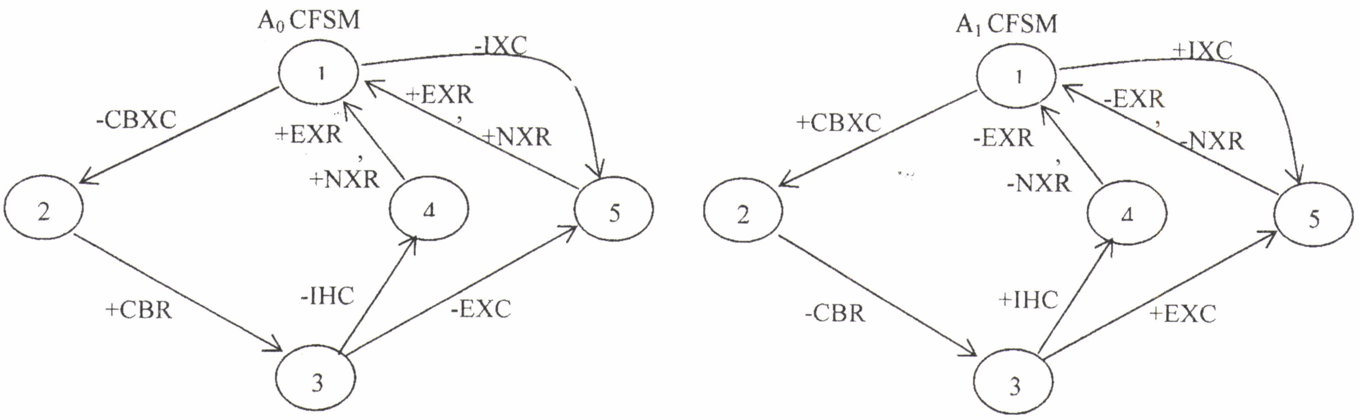

Fig. (12) CFSM models for the command transmission application function in ASEA protocol.

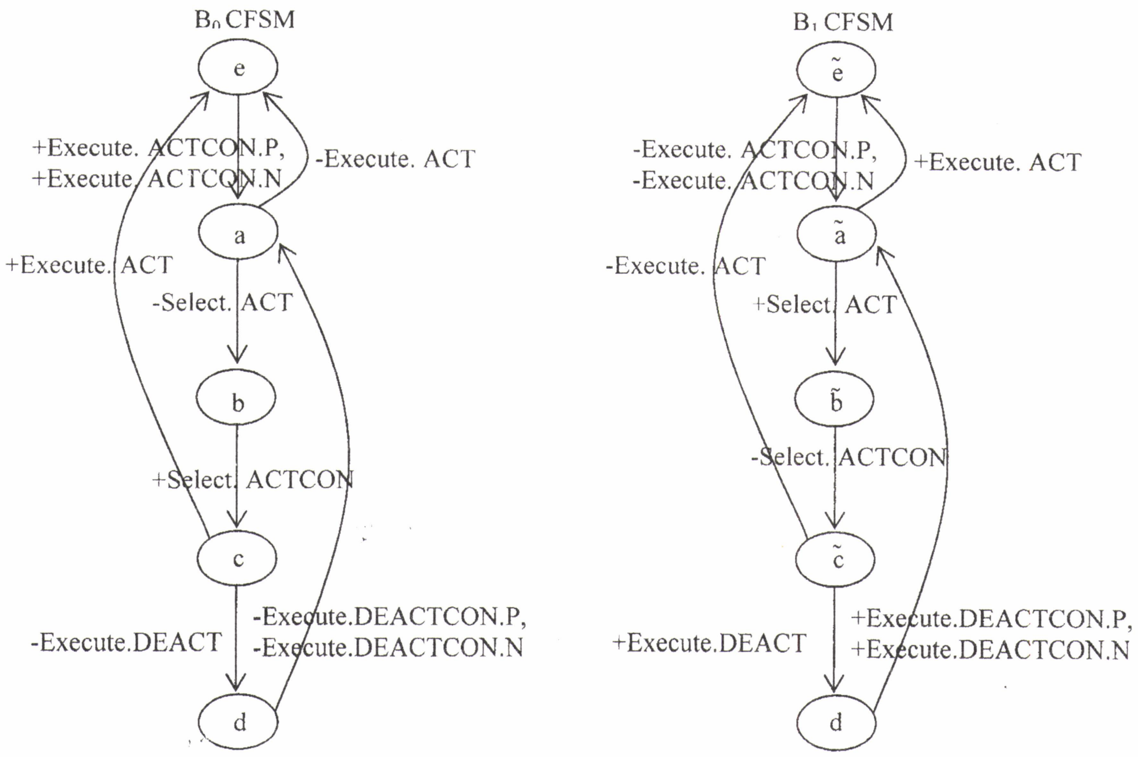

Fig. (13) CFSM model for command transmission application function in IEC870 protocol.

Where the abbreviations are defined as follow:

CBR ≡ check back response
CBXC ≡ check back before execute command
IXC ≡ immediate execute command

EXC ≡execute command
Execute.ACT ≡ execute command
Execute.ACTCON.P ≡ positive execute confirmation
Execute.ACTCON.N ≡ negative execute confirmation
Execute.DEACT ≡ inhibit command
Execute.DEACTCON.P ≡ positive inhibit confirmation
Execute.DEACTCON.N ≡ negative inhibit confirmation
Select.ACT ≡select command
Select.ACTCON ≡select confirmation.
Semantic Specifications: As obvious, all the messages of the two protocols are convertible.
Converter specifications:
R={CBXC, IXC, IHC, Execute.ACTCON.N, Execute.ACTCON.P, EXC,Execute.DEACTCON.N,Execute.DEACTCON.P, Select.ACTCON} ≡ Convertible set of receives.
S={EXR, NXR, CBR, Select.ACT, Execute.ACT,Execute.DEACT} ≡ Converted set of sends.
N= {}
Now, the message relationship function is defined:
Ψ: R → S such that:
Ψ [+CBXC]= -Select.ACT, Ψ [+IXC]= -Execute.ACT,
Ψ [+IHC]= -Execute.DEACT, Ψ [+EXC]= -Execute.ACT,
Ψ [+Execute.ACTCON.P]= -EXR, Ψ [+Execute.ACTCON.N]= -NXR,
Ψ [+Execute.DEACTCON.N]= -NXR, Ψ [+Execute.DEACTCON.P]= -EXR,
Ψ [+Select.ACTCON]= -CBR.
After applying Trace algorithm to the trivial converter C=A1 × B0, the resulted legal traces are shown below:
T_1 = (1a, +CBXC, 2a, -Select.ACT, 2b)
T_2 = (2b, +Select.ACTCON, 2c, -CBR, 3c)
T_3 = (3c, +IHC, 4c, -Execute.DEACT, 4d)
T_4 = (4d, +Execute.DEACTCON.P, 4a, -EXR, 1a)
T_5 = (1a, +IXC, 5a, -Execute.ACT, 5e
T_6 = (5e, +Execute.DEACTCON.N, 5a, -NXR, 1a)
T_7 = (3c, +EXC, 5c, -Execute.ACT, 5e)
After applying Synthesize algorithm, **Fig. (14)** shows the optimized converter machine H.

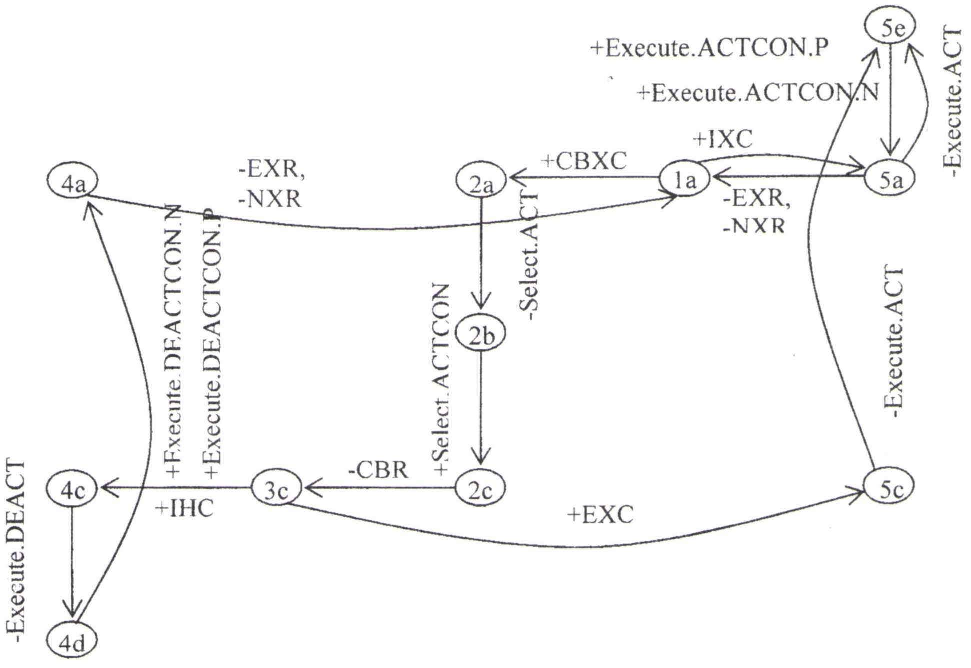

Fig. (14) The converter H for the command transmission application function.

PROTOCOL CONVERTER FOR CLOCK SYNCHRONIZATION APPLICATION FUNCTION

Fig. (15) shows the CFSM model for this function in ASEA protocol. The control center sends TSI (Time_Synchronizing_Instruction) to the RTU, which in turn responds by sending EXR message. Similarly, **Fig. (16)** shows the CFSM model for this function in ASEA protocol. The control center sends C.CS.ACT (clock synchronization activation) to the RTU, which in turn responds by sending C.CS.ACTCON (clock synchronization confirmation) message.

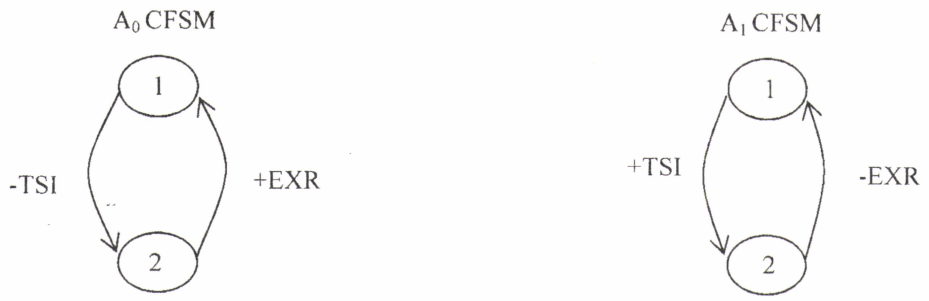

Fig. (15) CFSM models for the clock synchronization application function in ASEA protocol.

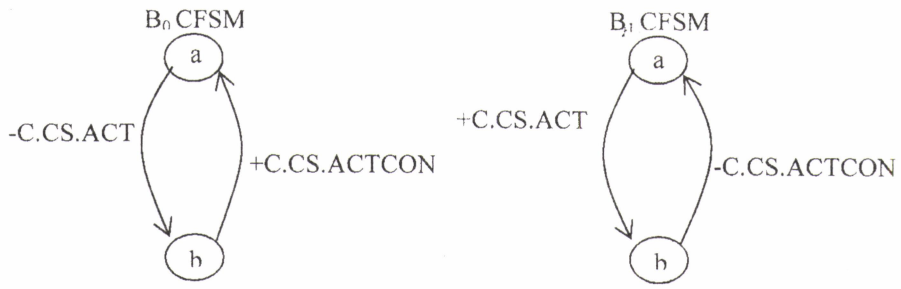

Fig. (16) CFSM model for clock synchronization application function in IEC870

Semantic Specifications: As obvious, all the messages of the two protocols are convertible.
Converter Specifications:
R={TSI, C.CS.ACTCON} ≡ Convertible set of receives.
S={EXR, C.CS.ACT} ≡ Converted set of sends.
N= {}
Now, the message relationship function is defined:
Ψ: R → S such that:
Ψ [+TSI]= -C.CS.ACT, Ψ [+C.CS.ACTCON]= -EXR.
After applying Trace algorithm to the trivial converter C=$A_1 \times B_0$, the resulted legal traces are shown below:
T_1 = (1a, +TSI, 2a, -C.CS.ACT, 2b) T_2 = (2b, +C.CS.ACTCON, 2a, -EXR, 1a)
After applying Synthesize algorithm, **Fig. (17)** shows the optimized converter machine H.

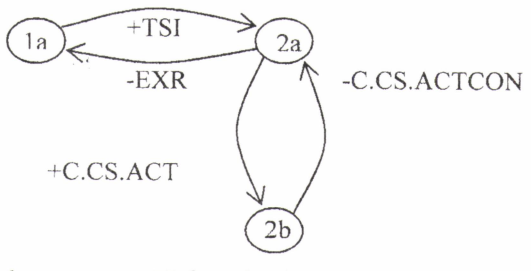

Fig. (17) The protocol converter H for clock synchronization application function.

PROTOCOL CONVERTER FOR TRANSMISSION OF INTEGRATED TOTAL APPLICATION FUNCTION

Fig. (18) shows the CFSM model for this function in ASEA protocol. The control center initially sends FCI (Freeze_Counter_Instruction) to the RTU. Then, the pulse counter values are acquired through the polling routine.

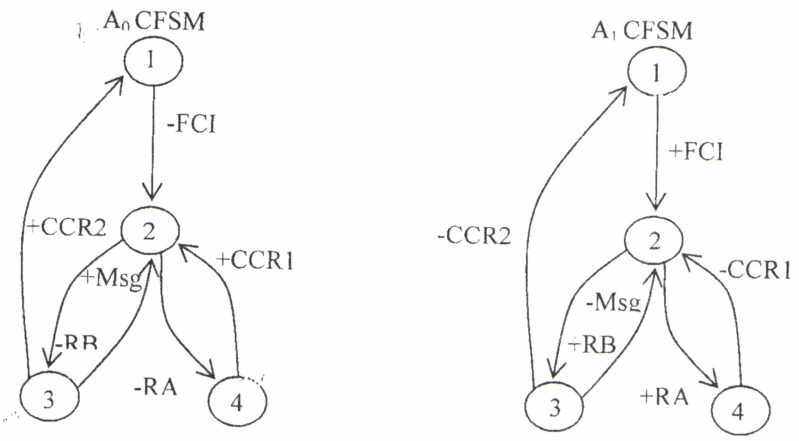

Fig. (18) CFSM model for Transmission of integrated total application function in ASEA protocol.

Fig. (19) shows the CFSM model for this function in IEC870 protocol. The control center initially sends Memorize.ACT (memorize counter contents) to the RTU which in turn responds by

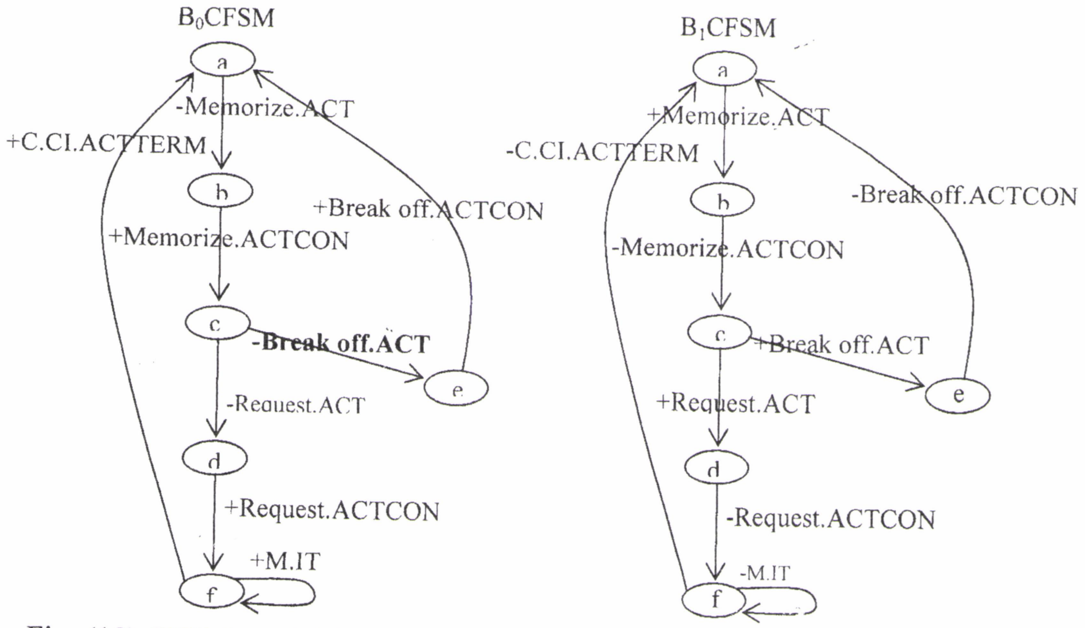

Fig. (19) CFSM model for integrated total application function in IEC870 protocol.

Memorize.ACTCON (memorize command confirmation). The control center sends Request.ACT to the RTU, which in turn responds by sending Request.ACTCON and the frozen values (M.IT). The end of this function is indicated whenever the RTU sends C.CI.ACTTERM (function termination). Semantic Specifications: When the converter receives FCI, it sends Memorize.ACT. Then, the collected data (M.ITs) are transferred to the control center. When the converter receives C.CI.ACTTERM, it sends CCR2 to the control center.

Converter Specifications:

R={FCI, C.CI.ACTTERM, M.IT} ≡ Convertible set of receives.

S={Memorize.ACT, CCR2, Msg} ≡ Converted set of sends.
N={RB,Memorize.ACTCON,RA,Request.ACT, Request.ACTCON,Break off.ACT, Break off.ACTCON,CCR1}
Now, the message relationship function is defined:
Ψ: R → S such that:
Ψ [+FCI]= -Memorize.ACT, Ψ [+C.CIACTTERM]= -CCR2 Ψ [+M.IT]= -Msg.
After applying Trace algorithm to the trivial converter C=A1 × B0, the resulted legal traces are shown below:

T_1 = (1a, +FCI, 2a, +RA, 4a, -CCR1, 2a, +RB, 3a, -Memorize.ACT, 3b)
T_{12} = (1a, +FCI, 2a, -Memorize.ACT, 2b)
T_{13} = (2b, +RA, 4b)
T_{14} = (4b, -CCR1, 2b)
T_{15} = (2b, +RB, 3b)
T_2 = (3b, +Memorize.ACTCON, 3c)
T_3 = (3c, -Request.ACT, 3d)
T_4 = (3d, +Request.ACTCON, 3f)
T_5 = (3f, +M.IT, 3f, +C.CI.ACTTERM, 3a, -Msg, 2a, +RA, 4a, -CCR1, 2a, +RB, 3a, -CCR2, 1a)
T_{52} = (3f, +M.IT, 3f, -Msg, 2f)
T_{53} = (2f, +RA, 4f)
T_{54} = (4f, -CCR1, 2f)
T_{55} = (2f, +C.CI.ACTTERM, 2a, +RB, 3a, -CCR2, 1a)

T_6 = (2b, +Memorize.ACTCON, 2c)

T_7 = (2c, +RA, 4c)
T_8 = (4c, -CCR1, 2c)

T_9 = (2c, +RB, 3c)
T_{10} = (2c, +Request.ACTCON, 2d)
T_{11} = (2d, +RA, 4d)
T_{12} = (4d, -CCR1, 2d)
T_{13} = (2d, +RB, 3d)
T_{14} = (4d, +Request.ACTCON, 4f)
T_{15} = (2d, +Request.ACTCON, 2f)
T_{16} = (2f, +M.IT, 2f, +RB, 3f, -Msg, 2f)

T_{162} = (2f, +RB, 3f)
T_{17} = (4b, +Memorize.ACTCON, 4c)
T_{18} = (4c, +Request.ACTCON, 4d)
T_{19} = (4f, +M.IT, 4f, +C.CI.ACTTERM, 4a, -CCR1, 2a, +RB, 3a, -Msg, 2a, +RB, 3a, -CCR2, 1a)
T_{192} = (4f, +C.CI.ACTTERM, 4a, -CCR1, 2a, +RB, 3a, -CCR2, 1a)

After applying Synthesize algorithm, **Fig. (20)** shows the optimized converter machine H.

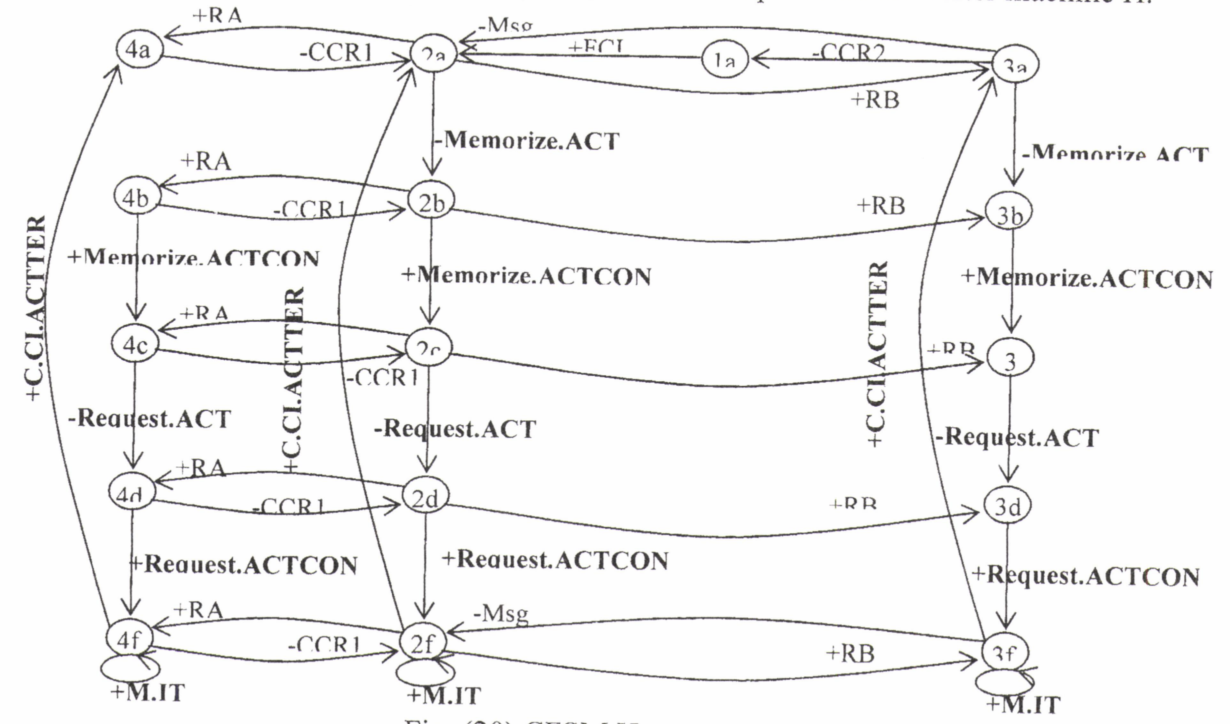

Fig. (20) CFSM H.

PROTOCOL CONVERTER FOR GENERAL INTERROGATION APPLICATION FUNCTION

Fig (.21) and **Fig. (7)** show the CFSM models for this function in ASEA and IEC870 protocols, respectively.

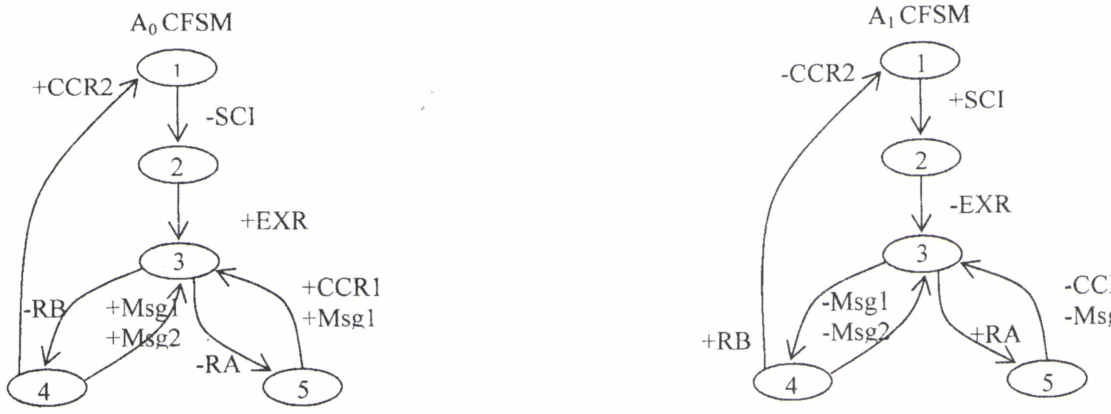

Fig. (21) CFSM model for the general interrogation application function in ASEA protocol.

In ASEA protocol, this function is initiated when the control center sends SCI command to the RTU. Whenever the control center receives EXR from the RTU, polling routine is resumed to acquire the interrogated information. For IEC870 protocol see the second converter of initialization application.

Semantic Specifications: When the converter receives SCI from the control center, it sends C.IC.ACT to the RTU. Then, when the converter receives C.IC.ACTCON from the RTU, it sends EXR to control center.

Converter Specifications:

R={SCI, C.IC.ACTTERM, C.IC.ACTCON, Data1, Data2} ≡ Convertible set of receives.

S={EXR, Msg1, Msg2, CCR2, C.IC.ACT} ≡ Converted set of sends.

N= {RA, RB, CCR1} ≡ Nonconvertible set of receives and sends.

Now, the message relationship function is defined:

Ψ: R $\rightarrow$ S such that:

Ψ [+SCI]= -C.IC.ACT, Ψ [+C.IC.ACTCON]= -EXR,

Ψ [+Data1]= -Msg1, Ψ [+Data2]= -Msg2,

Ψ [+C.IC.ACTTERM]= -CCR2.

After applying Trace algorithm to the trivial converter C=A1 × B0, the resulted legal traces are shown below:

T_1 = (1a, +SCI, 2a, -C.IC.ACT, 2b) T_{27} = (5c, -CCR1, 3c)

T_2 = (2b, +C.IC.ACTCON, 2c, +Data1, 2c, Data2, 2c,+C.IC.ACTTERM, 2a, -EXR, 3a, +RA, 5a, -Msg1, 3a, +RA, 5a, -Msg1, 3a, +RA, -CCR1, 3a, +RB, 4a, -Msg2, 3a,RB, 4a, -CCR2, 1a) T_{28} = (3c, +RB, 4c)

T_{22} = (2b, +C.IC.ACTCON, 2c, -EXR, 3c) T_{29} = (4c, +Data1, 4c, +Data2, 4c, -Msg1, 3c, +RB, 4c, -Msg2, 3c)

T_{23} = (3c, +Data2, 3c, +RB, 4c, -Msg2, 3c) $T_{2,10}$ = (3c, +C.IC.ACTTERM, 3a, +RB, 4a, -CCR2, 1a)

T_{24} = (3c, +Data1, 3c, +RA, 5c, -Msg1, 3c) $T_{2,11}$ = (4c,+Data1,4c, +C.IC.ACTTERM, 4a,-Msg1,3a,+RB,4a, -CCR2, 1a)

T_{25} = (3c, +Data1, 3c, +RB, 4c, -Msg1, 3c) $T_{2,12}$ = (5c, +Data1, 5c, +Data2, 5c,+C.IC.ACTTERM,5a, -Msg1, 3a, +RB, 4a, - Msg2, 3a, +RB, 4a,-CCR2,1a)

T_{26} = (3c, +RA, 5c)

After applying Synthesize algorithm, **Fig. (22)** shows the optimized converter machine H.

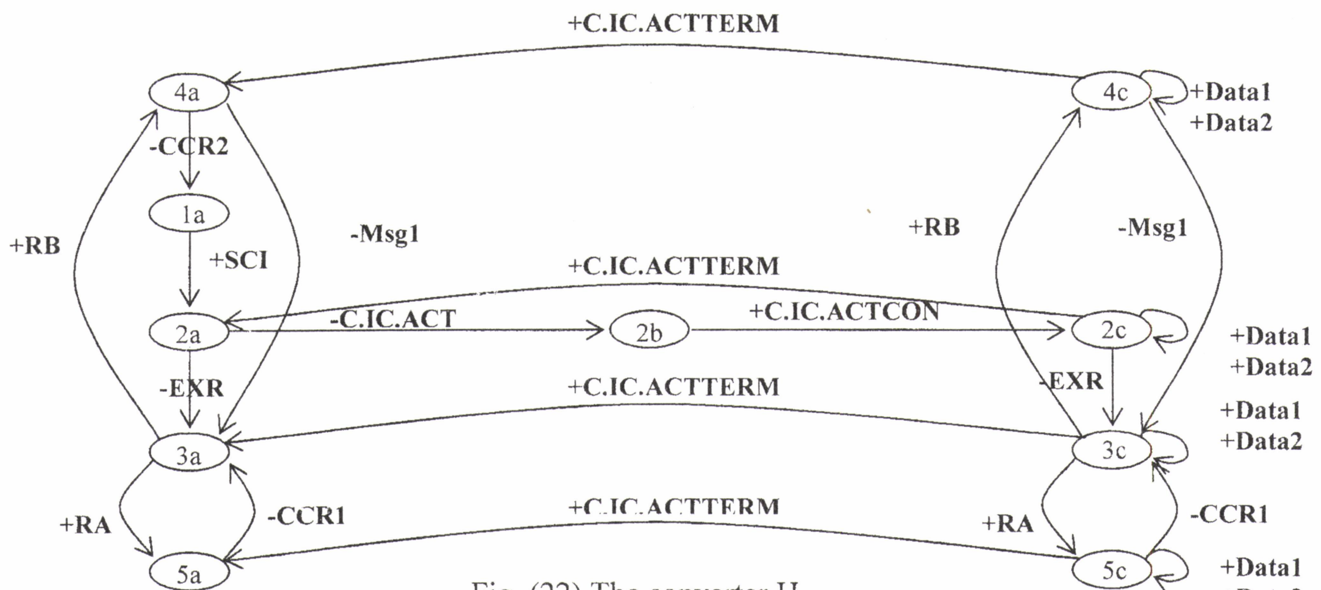

Fig. (22) The converter H.

PROTOCOL CONVERTER FOR RE_START_UP APPLICATION FUNCTION

This function is comprised of two sub functions, cold start and link establishment. For simplicity, instead of designing one large converter, two simple converters can be constructed to achieve the same application function.

Cold Start Converter

Fig. (23) shows the CFSM model for this function in ASEA protocol. When the control center wants to restart the RTU, it just sends CLD.STR (cold start command) to the RTU. **Fig.24** shows the CFSM model for this function in IEC8790 protocol. When the control center wants to restart the RTU, it just sends C.RP.ACT (reset process command) to the RTU, which in turn responds by sending C.RP.ACTCON (reset process confirmation).

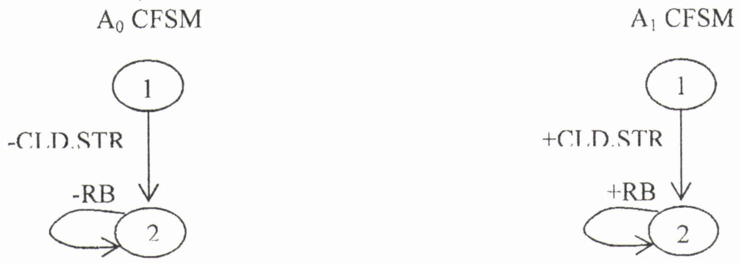

Fig. (23) CFSM model of cold start function for re-initialization application function of ASEA protocol.

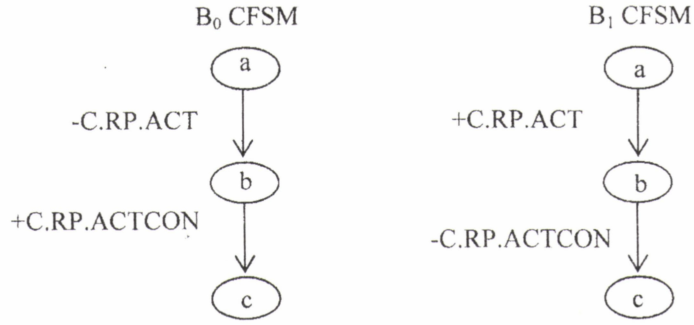

Fig (24) CFSM model for warm start function of re-start up application function in IEC870 protocol.

Semantic Specifications: When the converter receives CLD.STR from the control center, it sends C.RP.ACT to the RTU.
Converter Specifications:
R= {CLD.STR} ≡ Convertible set of receives.
S= {C.RP.ACT} ≡ Converted set of sends.
N= {RB, C.RP.ACTCON} ≡ Nonconvertible set of sends and receives
Now, the message relationship function is defined:
Ψ: R → S such that:
Ψ [+CLD.STR]= -C.RP.ACT.
After applying Trace algorithm to the trivial converter C=A1 × B0, the resulted legal traces are shown below:
T_{11}=(1a, +CLD.STR, 2a, +RB, 2a, -C.RP.ACT, 2b) T_2 = (2b, +C.RP.ACTCON, 2c)
T_{12}=(1a, +CLD.STR, 2a, -C.RP.ACT, 2b) T_3 = (2c, +RB, 2c)
T_{13}=(2b, +RB, 2b)
After applying Synthesize algorithm, **Fig. (25)** shows the optimized converter machine H.

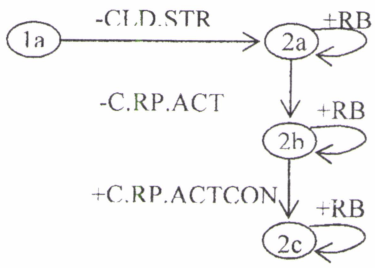

Fig. (25) CFSM H.

Link Establishment Converter
Fig. (26) shows the CFSM model for this function in ASEA protocol. At first, the control center sends SCI to the RTU. If the RTU is ready, it responds by returning EXR. Then, the control center requests the RTU by transmitting RB. The RTU answer is that the data base contents (function tables) are missing (M.F.T.). Then, the control center begins to transmit the function tables (FTAB) to the RTU. At last, the control center activates the RTU by sending ACT.RTU, which is confirmed by EXR message.

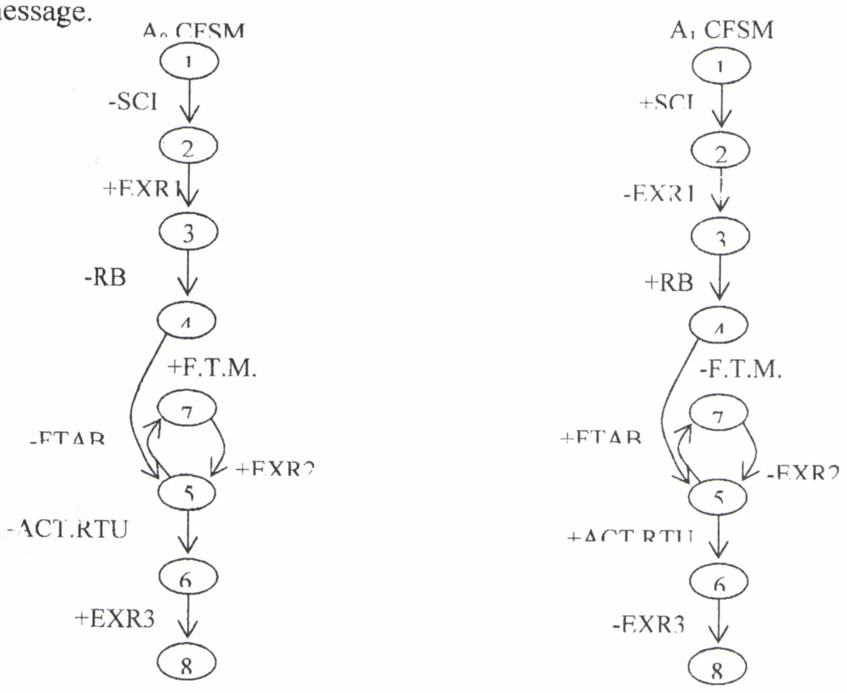

Fig. (26) CFSM model of link establishment function in ASEA protocol.

Fig. (27) shows the CFSM model for this function in IEC870 protocol. At first, the control center requests the RTU for the status of link (Req,SOL). The RTU responds by sending Res.SOL to the control center. Then the control center begins to send Req.UDCLS1 to the RTU, which is confirmed by M.AA or by M.EI when the function is terminated.

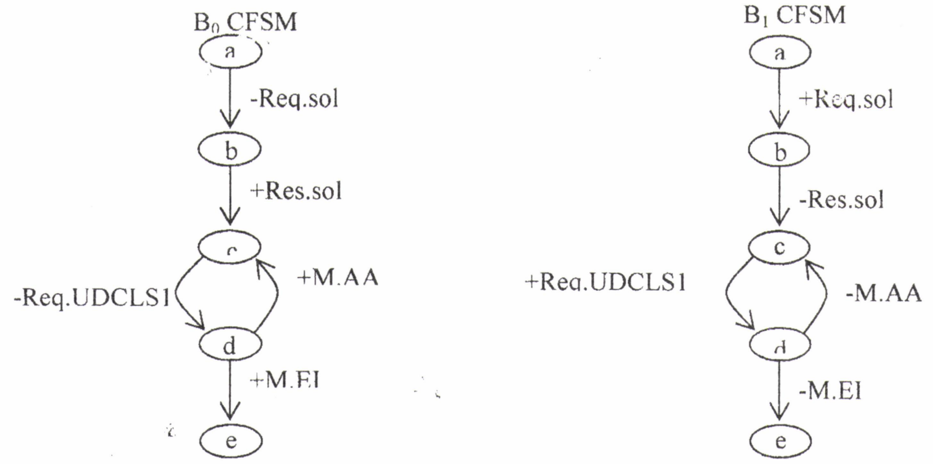

Fig. (27) CFSM model for link establishment function of IEC870 protocol.

Semantic Specifications: When the converter receives SCI from the control center, it sends Req.SOL to RTU. Then, when the converter receives Res.SOL from the RTU, it sends back EXR to the control center.

Converter Specifications:
R={SCI, Res.sol, M.EI} ≡ Convertible set of receives.
S={Req.sol, EXR1, EXR3} ≡ Converted set of sends.
N={RB, F.T.M., FTAB, EXR2, ACT.RTU, Req.UDCLS1, M.AA} ≡ Nonconvertible set of receives and sends.
Now, the message relationship function is defined:
$\Psi: R \rightarrow S$ such that:
Ψ [+SCI]= -Req.sol, Ψ [+Res.sol]= -EXR1, Ψ [+M.EI]= -EXR3.
After applying Trace algorithm to the trivial converter C=A1 × B0, the resulting legal traces are shown below:

T_1 = (1a, +SCI, 2a, -Req.sol, 2b)

T_2 = (2b, +Res.sol, 2c, -EXR1, 3c)
T_3 = (3c, +RB, 4c)
T_4 = (4c, -F.T.M., 5c)
T_6 = (7c, -EXR2, 5c)

T_7 = (5c, +ACT.RTU, 6c)
T_8 = (6c, -Req.UDCLS1, 6d)
T_9 = (6d, +M.AA, 6c)
T_{10} = (6d, +M.EI, 6e, -EXR3, 8e)

T_{11} = (3c, -Req.UDCLS1, 3d)
T_{12} = (3d, +M.AA, 3c)
$T_{13,1}$ = (3d, +M.EI, 3e, +RB, 4e, -F.T.M., 5e, +FTAB, 7e, -EXR2, 5e, +ACT.RTU, 6e, -

$T_{13,3}$ = (4d, +M.EI, 4e, -F.T.M., 5e, +FTAB, 7e, -EXR2, 5e, +ACT.RTU, 6e, -EXR3, 8e)

T_{14} = (64, +M.AA, 4c)
T_{15} = (4c, -Req.UDCLS1, 4d)
T_{16} = (4d, -F.T.M., 5d)
$T_{17,1}$ = (5d, +M.EI, 5e, +ACT.RTU, 6e, -EXR3, 8e)
$T_{17,2}$ = (5d, +ACT.RTU, 6d)
T_{18} = (5d, +FTAB, 7d)
T_{19} = (7d, -EXR2, 5d)
T_{20} = (7d, +M.EI, 7e, -EXR2, 5e, +FTAB, 7e, -EX
+ACT.RTU, 6e, -EXR3, 8e)
T_{21} = (7d, +M.AA, 7c)
T_{22} = (7c, -Req.UDCLS1, 7d)
T_{23} = (5d, +M.AA, 5c)

EXR3, 8e)

$T_{13,2}$ = (3d, +RB, 4d) T_{24} = (5c, -Req, UDCLS1, 5d)

After applying Synthesize algorithm, **Fig. (28)** shows the optimized converter machine H.

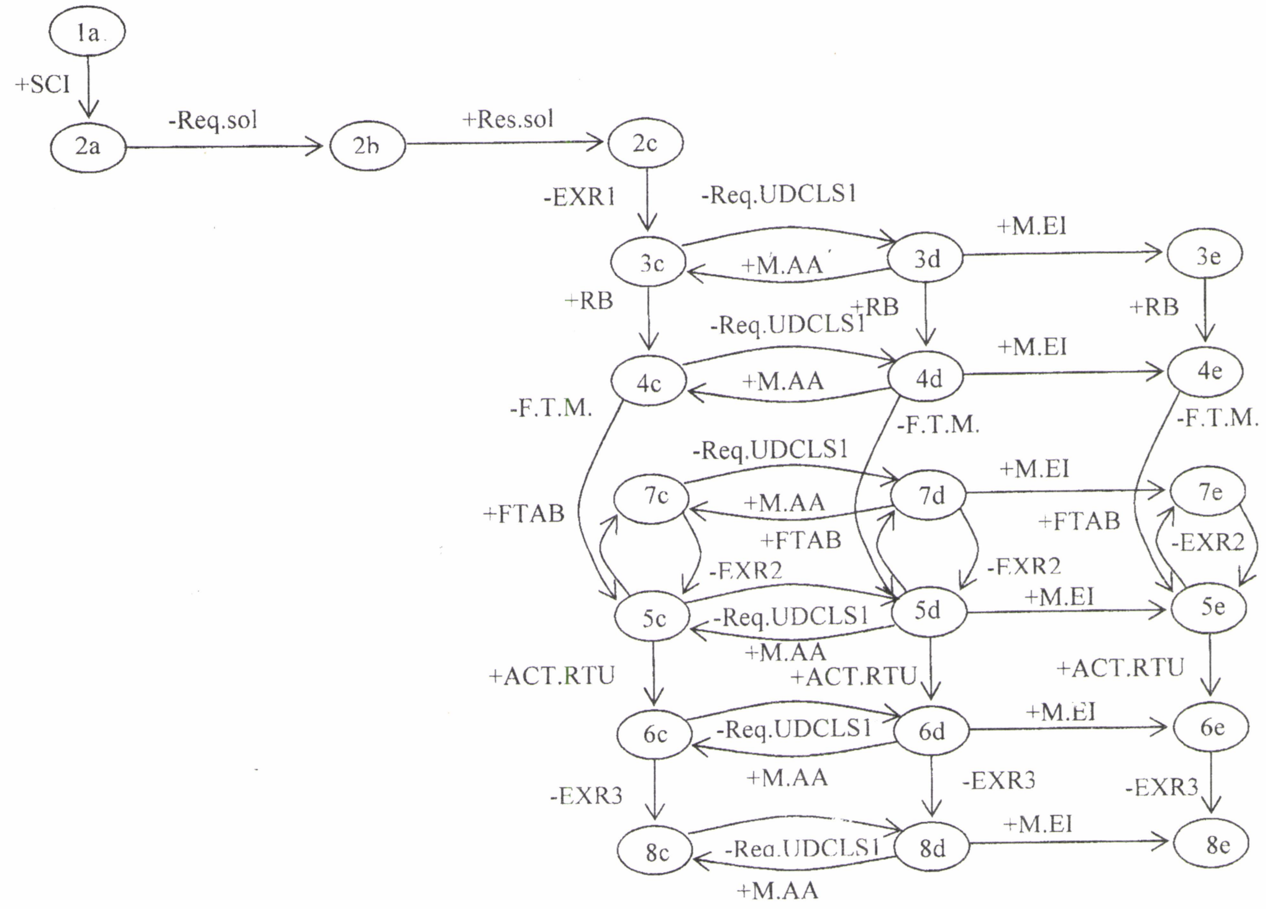

Fig. (28) Converter H.

CONCLUSIONS

In this work, a protocol converter had been designed using formal methods in order to overcome the heterogeneity problem which results from connecting controlled stations that utilize IEC 870 protocol standard to a controlling station that utilizes a different protocol standard of ASEA company.

Using the Conversion algorithm presented in [Rajagopal and Miller 1991], a converter H CFSM is constructed for each application function. The initialization and restart-up application functions are further subdivided into two sub functions and a separate converter was constructed for each sub function. This situation is useful especially if the sub function converters are reusable.

From the results of the conversion process, it is shown that for a two protocols that operate under the same phase such that all the messages are convertible (the nonconvertible set of receives and sends N is empty), the constructed converter are guaranteed to be free of errors provided that both protocols are error free. The number of possible converters that resulted by varying the input specifications is at minimum extent. As the number of messages of set N increased, the number of possible legal traces increased for given semantic specifications. Finally, the conversion algorithm has revealed a high degree of flexibility that depends on the specifications of the converter, which is a designer choice. For more complex protocols, the conversion process will be a tedious task.

REFERENCES

Calvert K., and Lam S. (1989), Deriving a protocol converter: A top-down method, Proc. ACM SIGCOMM'89, pp. 247-258, USA.

Gouda M.G. (1984), Closed Covers: To verify progress for communicating finite state machines, IEEE Trans. Software Eng. 10 (6) 846-855.

Green P., (1986), Protocol conversion, IEEE Trans. Commun. 34 (3) 257-268.

Rajagopal M., and Miller R.E. (1991), Synthesizing a protocol converter from executable protocol traces, IEEE Trans. Comput. 40 (4) 487-499.

Brand D., and Zafiropulo P.,(1983), On communicating finite state machines, JACM 30 (2) 323-342.

Saleh K., and Jaragh M. (1998), Synthesis of protocol converters: an annotated bibliography, Computer Standards & Interfaces 19 105-117.

Tao Z.P., Bochmann G.V., and Dssouli R. (1995), An efficient method for synthesizing optimized protocol converter, an extended version of the paper presented at ICCCN'95, Las Vegas, USA.

Zoline K.O., and Lidinsky W.P. (1985), An approach for Interconnecting SNA and XNS Network, Proc. ACM SIGCOMM'85.