Conversion of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams

M. Uzam, A.H. Jones, and N. Ajlouni

Intelligent Machinery Division
Research Institute for Design, Manufacture and Marketing
University of Salford, SALFORD M5 4WT, UK
fax: ++ 44 161 745 59 99
e-mail: m.uzam@aeromech.salford.ac.uk

Abstract - **As automated manufacturing systems become more complex, the need for an effective design tool to produce both high level Discrete Event Control System (DECS) and low level implementation, becomes increasingly more important. Petri nets represent the most effective method for both the design and implementation of DECSs. The conversion of such Petri nets into real-time applications has recently been greatly simplified through the advent of the Token Passing Logic (TPL) methodology. The technique has been developed for normal Petri nets, P-timed Petri nets, T-timed Petri nets and Coloured Petri nets. In this paper the Petri net concepts are extended to deal with Petri net controllers, by including actuators and sensors as formal structures within the Petri net controller. The conversion of such a Petri net controller into ladder logic diagrams is also demonstrated by considering the control of a component sorting manufacturing system.**

I. INTRODUCTION

To-date industry standard Petri net controllers do not exist, rather Programmable Logic Controllers (PLCs) dominated the application domain. In today's modern factory, PLCs have emerged as the mainstay in the execution of automation tasks. Their selection for Discrete Event Control tasks is due to their low-cost, ruggedness and ease of programming. Indeed, the majority of PLCs can be programmed in a graphical symbolic language called *ladder logic*. The very simplicity of the ladder logic programs which makes them so transparent is also their greatest downfall. This is because when developing complex control systems involving parallel tasks which interact periodically the ladder logic programming language offers little in the way of structural constructs to deal with problem. As automated manufacturing systems become more complex, the need for an effective design tool to produce high level Discrete Event Control System (DECS) becomes increasingly more important. Petri nets [1] have appeared as the most promising tool to facilitate such design work. By using Petri net techniques Discrete Event models of the manufacturing systems can be built which can be used for a variety of design studies, including production ratio, deadlock and control design. However, Petri net methods have not been used for the direct design of PLC programs. This is because until the recent advent of Token Passing Logic (TPL) there was no single technique to convert Petri nets into a suitable format for implementation on a PLC. Indeed, there have been many attempts to produce structured methods to convert Petri nets into ladder logic, [2], [3], [4], [5], and [6]. However until the advent of the Token Passing Logic (TPL) no solution has been provided to cope with the full complexity of today's automation needs. The TPL methodology bridges the gap between Petri net analysis and ladder logic. The technique is powerful and yet simple to both understand and implement. Moreover, the thecnique has also been extended to deal with P-timed Petri nets [7], [8], T-timed Petri nets [9], and Coloured Petri nets [10], [11]. The TPL methodology has also been developed to embrace statement lists [12], [13] and knowledge based systems [14], [15]. In order to implement Petri net designs directly as controllers some additional features to ordinary Petri nets have to be defined. This is because Petri nets do not have constructs to adequately deal with actuators and sensors. This deficiency has prompted the advent of a Petri net controller (PNC), which extends the ordinary Petri nets to deal with discrete event control applications. These extensions involve interfacing the Petri net to actuators and sensors.

The purpose of this paper is to establish a set of additional features to ordinary Petri nets which facilitate the addition of sensor readings at transitions and both the impulse and level control of actuators. Finally, a Petri net controller is described for a component sorting manufacturing system.

II. PETRI NET CONTROLLERS AND TOKEN PASSING LOGIC

Ordinary Petri nets do not deal with actuators or sensors. Because of this, it is necessary to define a Petri net controller (PNC) which can embrace both actuators and sensors within the existing Petri net framework. In a PNC sensors are used at transitions. The presence or absence of sensor readings can be used in conjunction with the normal Petri net pre-conditions to enable or fire transitions. In a PNC two types of actuation are considered, namely impulse actions and level actions. Actuators can be associated with either places or transitions. With these additional features, it is possible to construct Discrete Event Control Systems. In order to facilitate the conversion of a PNC into a ladder logic diagram, the recently introduced TPL methodology can be deployed.

The prime feature of the TPL technique is that it facilitates the direct conversion of any Petri net Controllers into control logic. This

0-7803-3685-2/96 $5.00 © 1996 IEEE

is achieved by adopting the Petri net concept of using tokens as the main mechanism for controlling the flow of the control logic. Hence, each place within the Petri net corresponds to a place within the PNC program. The simulated movement of tokens is achieved by deploying separate counters at each logic place, whose capacity is equal to or greater than 1. These counters are then incremented and decremented to simulate token flow. Thus, each logic place within the PNC program has an associated counter, and the current count value of the counter at the logic place represents the number of tokens that would be at the corresponding place within the Petri net. The assignment of a counter to a Petri net place is shown in Fig. 1.(a), where C stands for counter. Finally, to complete the Petri net synergy, if the counter associated with a place is non-zero and a Petri net like transition associated with that place becomes active, then the counter at the place is decremented by one, and the subsequent place linked by the transition is incremented by one. In the case of single capacity places the counters can be replaced by flags. Token flow is then achieved by setting and resetting flags. The assignment of a flag to a Petri net place is also shown in Fig. 1.(b), where F stands for flag. An *on delay timer* can be readily used to model timed transitions and timed places in PNC method.

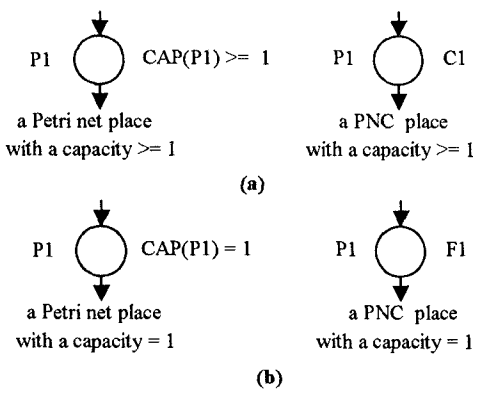

Fig. 1. Petri net places and their equivalent PNC places.

In essence, the Petri net places are represented by logic places, the Petri net tokens are represented by separate counters at each logic place. Moreover, the flow of Petri net tokens is simulated by counting down and counting up the counters or similarly by setting and resetting the appropriate flags at the appropriate places. In the PNC, actions can either be assigned to places or transitions. Places on which actions are assigned are called control places, and transitions on which actions are assigned are called control transitions. If an action is assigned to a place for a finite time this corresponds to a P-timed Petri net feature. However, if an action is assigned to a transition for a finite time, this corresponds to a T-timed Petri net feature.

In theory, the methodology can cope with any number of tokens and provide a visual description of the control program which has all the advantages of a full Petri net analysis. Furthermore, coloured Petri net controllers can also be converted into control logic using this methodology, simply by adding more counters or flags to each place. It is believed that this new technique provides a tool which is a simple, but sophisticated way of developing complex Discrete Event Control Systems. It is these very features which will be vital to the success of agile manufacturing systems.

The Petri Net Controllers are illustrated by considering the following Petri net structures: *Sensor readings at a transition, level action at a place, impulse action at a place, level action at a transition,* and *impulse action at a transition.*

A. Sensor Readings at a Transition

A standard transition with sensor readings is shown in Fig. 2. In Petri net theory, a transition can only be fired if the number of tokens at the input place is non zero and a signal enabling the transition occurs[1]. In a PNC an enabling arc can be used as an additional pre-condition to the Petri net such that if the logic for a sensor reading is high then the pre-condition is satisfied. Similarly, an inhibitor arc can be used as an additional pre-condition to the Petri net such that if the logic for a sensor reading is low then the pre-condition is satisfied. The transition is fired when all the pre-conditions are satisfied, and one or more pre-conditions undergoes a change in logic state. This is shown at the instances a, b, c, d in Fig. 3. Enabling arcs from sensors can fire transitions when the state of the sensor goes from low to high (leading edge - ↑), {instance *a* and *b*}. Inhibitor arcs from sensors can fire transitions when the state of the sensor goes from high to low (trailing edge - ↓), {instance *c*}. If the pre-conditions from the sensor readings are already satisfied and a token enters an input place which previously had no tokens, the presence of the token will also fire the transition (leading edge - ↑), {instance *d*}. In the TPL method, when a transition is fired, it withdraws a token from the current logic place and adds a token to the next logic place.

This is achieved by using a counter at each place to represent the tokens. When a transition is fired, to simulate the passing of a token the input counter is decremented and the output counter is incremented by one. The ladder Logic program for the standard transition with sensor readings shown in Fig. 2 is given in Fig. 4.

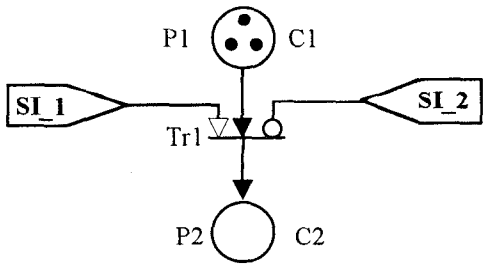

Fig. 2. Sensor readings on a transition in a PNC.

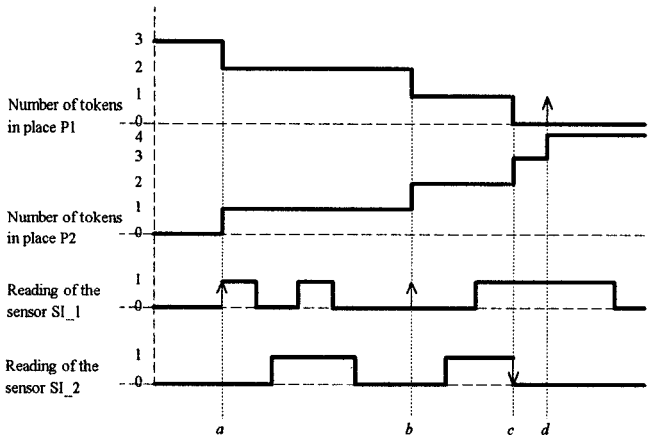

Fig. 3. Firing of the PNC shown in Fig. 2.

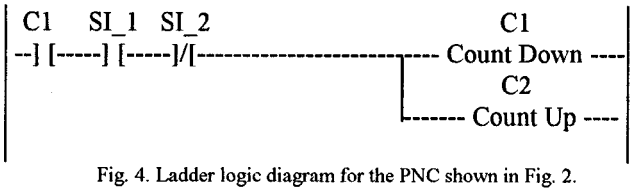

Fig. 4. Ladder logic diagram for the PNC shown in Fig. 2.

B. Level Action at a Place

In a PNC, *actions* can be assigned to places called control places. If there is a condition on the output transition, then the action is called *level action* because when a token is deposited in a control place, actions are enabled up to the moment when the token is removed from the place. Fig. 5.(a) shows the Petri net place P2 on which a level action is assigned. A level action at a given place within a Petri net occurs only if the number of token at the place is non-zero. Fig. 5.(b) shows a PNC place P2 on which a level action is assigned. In a PNC, a level action at a place is controlled by the counter or flag at the place. If the count value of a counter at the control place is greater than zero or the related flag is set then any actions associated with the place are enabled. The firing of the Petri net shown in Fig. 5.(a) is given in Fig. 6. Also, the ladder logic diagram for the PNC shown in Fig. 5.(b) is given in Fig. 7.

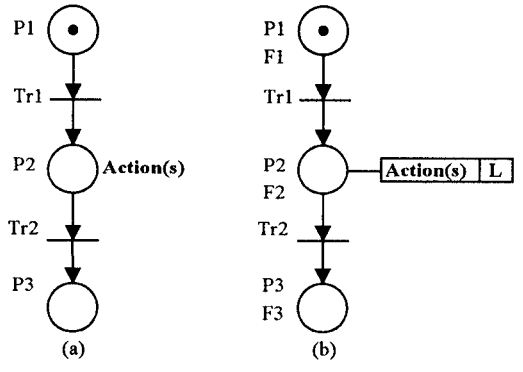

Fig. 5.(a). Level action(s) assigned on a place in Petri nets.
(b). Level action(s) assigned on a control place in a PNC.

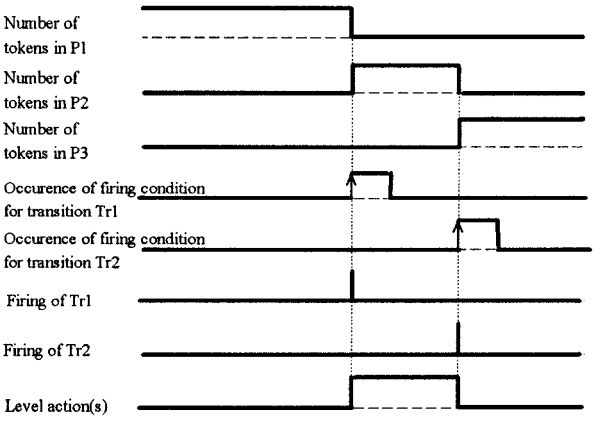

Fig. 6. Firing of the Petri net shown in Fig. 5.(a).

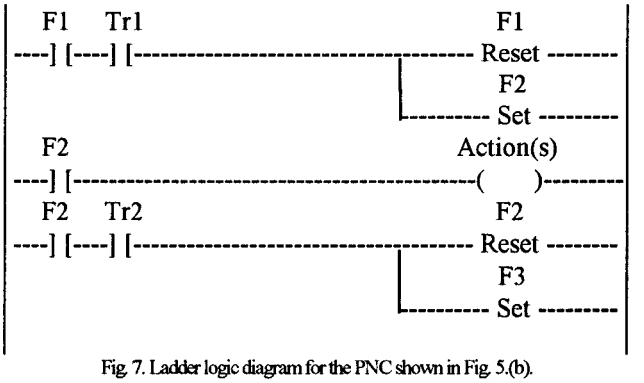

Fig. 7. Ladder logic diagram for the PNC shown in Fig. 5.(b).

C. Impulse Action at a Place

If there is no condition on the output transition from the control place, then a token will only remain in the place for a very short time (↑). The action assigned to this control place is then called an *impulse action*. In this case, when a token is deposited in a control place the actions are enabled and immediately disabled as the token is removed from the place, hence creating an impulse action. Fig. 8.(a) shows the Petri net place P2 on which an impulse action is assigned. An impulse action at a given place within a Petri net occurs only if the number of token at the place is non-zero. Fig. 8.(b) shows a PNC place P2 on which an impulse action is assigned. In a PNC, an impulse action at a place is controlled by the counter or flag at the place. If the count value of a counter at the control place is greater than zero or the related flag is set then any actions associated with the place are enabled. The firing of the Petri net shown in Fig. 8.(a) is given in Fig. 9. Also, the ladder logic diagram for the PNC shown in Fig. 8.(b) is given in Fig. 10.

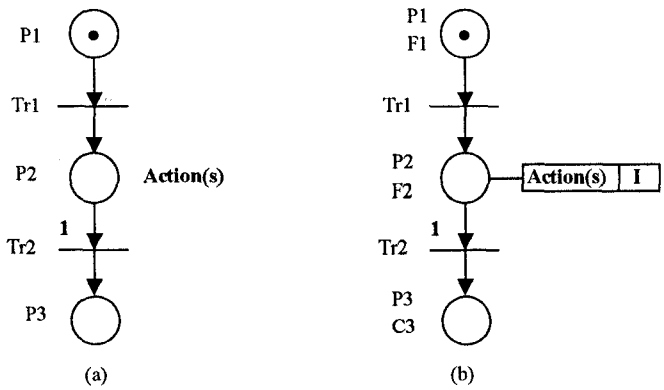

Fig. 8. (a). Impulse action(s) assigned on a place in Petri nets.
(b). Impulse action(s) assigned on a control place in a PNC.

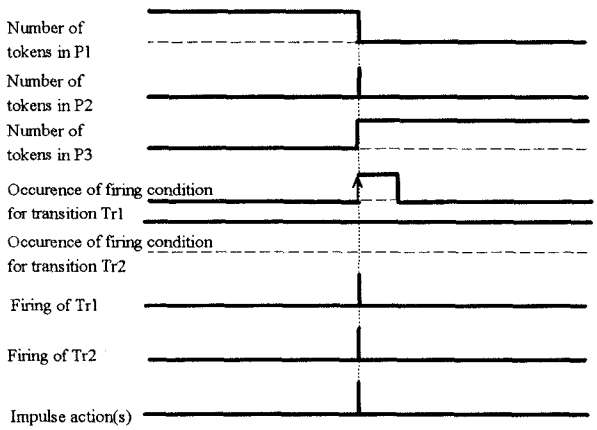

Fig. 9. Firing of the Petri net shown in Fig. 8.(a).

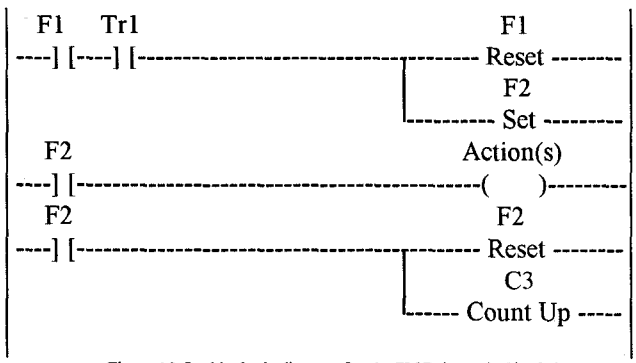

Figure 10. Ladder logic diagram for the PNC shown in Fig. 8.(b).

D. Level Action at a Transition

When considering a level action, it is important to realise that two signals are required to trigger the level action, one signal to enable the action and one signal to disable the action. This requirement for two signals is easily achieved at a level action at a place as shown in Fig. 5.(b). However, in the case of a level action at a transition this can not be achieved, unless a hierarchical transition is deployed.

The transition Tr1 shown in Fig. 11.(b) is a *hierarchical transition*. There are two conditions for the transition Tr1. The first one is an enabling event ($Tr1_e$) which is used to enable the action(s). The second condition for transition Tr1 is a disabling event ($Tr12_d$) which is used to disable the action(s). When the firing condition $Tr12_e$ occurs, the number of tokens at the input place P1 is decremented by 1 and the number of tokens at the hidden place within the hierarchical transition Tr1 is incremented by 1. A level action at a hierarchical transition within a Petri net occurs only if the number of token at the place in the hierarchical transition is non-zero. When the firing condition $Tr12_d$ occurs, the number of tokens at the hidden place is decremented by 1 and the number of tokens at the output place P2 is incremented by 1. To achieve these effects in a PNC, a counter has to be assigned to the transition. When $Tr12_e$ occurs, flag F1 associated with place P1 is reset and counter C12 is incremented by 1. Hence, when the count value of C12 is non-zero it enables the related actions. When $Tr12_d$ occurs, counter C12 is decremented and counter C2 associated with place P2 is incremented by 1. It is evident that this hierarchical transition arrangement is required since there is no such feature in ordinary Petri nets. During the time between the conditions enabling event $Tr12_e$ and disabling event $Tr12_d$, action(s) will be enabled. The firing of the Petri net shown in Fig. 11.(a) is given in Fig. 12. Also, the ladder logic diagram for the Petri net shown in Fig. 11.(b) is given in Fig. 13.

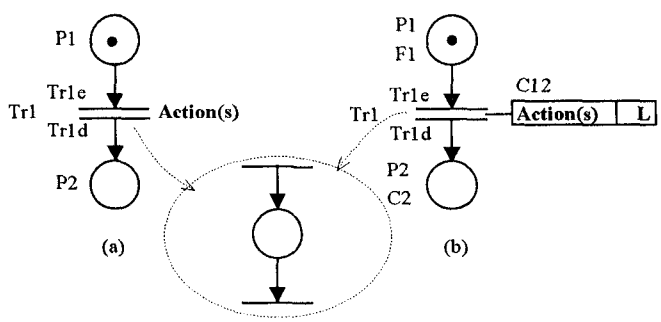

Fig. 11. (a). Level action(s) assigned on a transition in Petri nets.
(b). Level action(s) assigned on a control transition in a PNC.

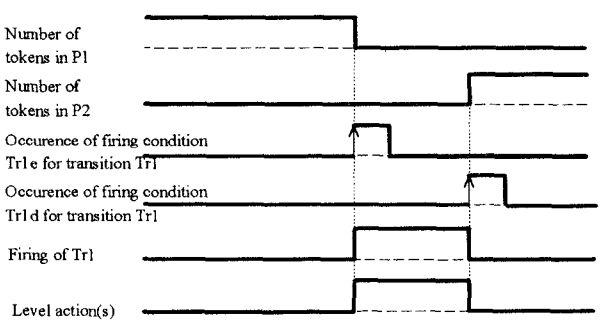

Fig. 12. Firing of the Petri net shown in Fig. 11.(a).

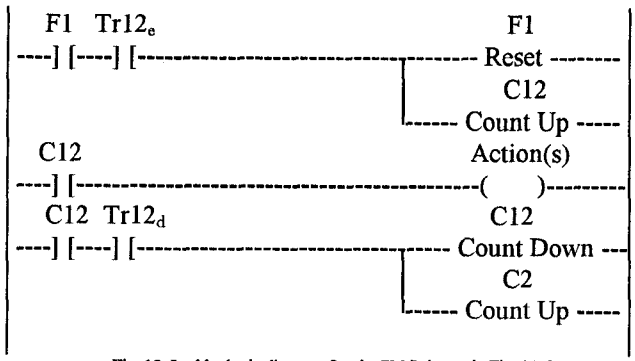

Fig. 13. Ladder logic diagram for the PNC shown in Fig. 11.(b).

E. Impulse Action at a Transition

In a PNC, *impulse actions* can be assigned to the transitions. In this case, the actions are *impulse actions* because the action associated with transition is carried out as the transition is fired. Fig. 14.(a) shows the Petri net transition Tr1 on which an impulse action is assigned. If there is a token in place P1 and the firing condition for transition Tr1 occurs then a token is removed from place P1 and put in place P2. The impulse action at a transition Tr1 occurs only as the transition is fired. Fig. 14(b) shows the PNC transition Tr1 on which an impulse action is assigned. In the PNC, impulse actions are carried out when the input flag F1 is reset and the output flag F2 is set. In the case of counters this corresponds to counting down and counting up the counters. The firing of the Petri net shown in Fig. 14.(a) is given in Fig. 15. Also, the ladder logic diagram for the PNC shown in Fig. 14.(b) is given in Fig. 16.

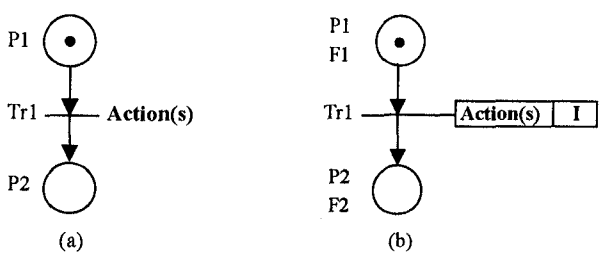

Fig. 14.(a). Impulse action(s) assigned on a transition in Petri nets.
(b). Impulse action(s) assigned on a control transition in a PNC.

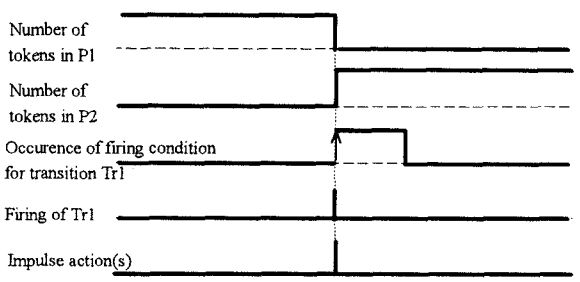

Fig. 15. Firing of the Petri net shown in Fig. 14.(a).

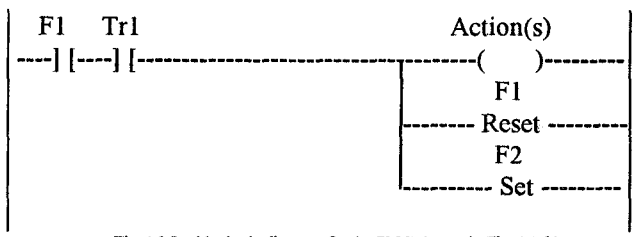

Fig. 16. Ladder logic diagram for the PNC shown in Fig. 14.(b).

III. MANUFACTURING SYSTEM EXAMPLE

The Manufacturing System, shown in Fig. 17, represents a component sorting processes that can be controlled by virtually any PLC.

The conveyor is driven by the conveyor motor A1 (Actuator 1). A random selection of part A and part B are placed on the conveyor. The part As and part Bs need to be identified and separated. This is done by two sensors, a proximity sensor S1 (Sensor 1) and an infra-red reflective sensor S2 (Sensor 2). By using these two sensors a distinction can be made between the part As and part Bs. By means of the sort solenoid A2 (Actuator 2), part As can be ejected into Storage I. Part Bs, meanwhile, continue on the conveyor and are deflected into Storage II. An infra-red emitter/detector S3 (Sensor 3) is used to determine whether or not there is a component in front of the sort solenoid A2. If sensor S3 is active, the sort solenoid A2 can be used to eject either a part A or part B into Storage I. If no action is taken the component is carried into Storage II by the conveyor. In Table 1 sensor readings are explained.

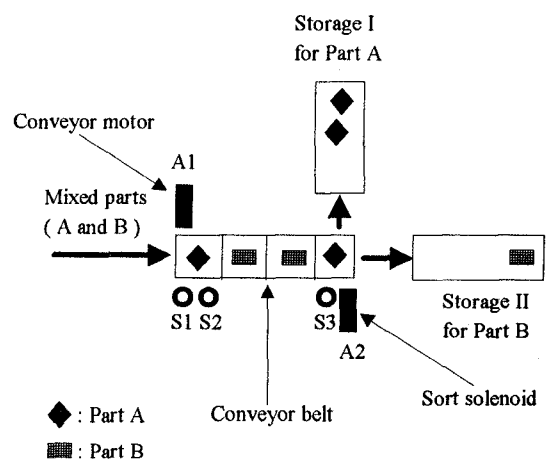

Fig. 17. Manufacturing system.

TABLE I

Sensor readings	Interpretations
S1 & (not S2)	Part A
S1 & S2	Part B
S3	Part A or Part B

IV. PETRI NET CONTROLLER FOR THE MANUFACTURING SYSTEM

A Petri Net Controller (PNC) for the manufacturing system is given in Fig. 17. The controller design is achieved by using a Petri net model of the manufacturing system. A queue structure is deployed for the two types of part by places P1,...P8, where places P1, P3, P5, P7 are used to construct a queue for part As and places P2, P4, P6, P8 are used to construct a queue for part Bs. Note that inhibitor arcs at transitions Tr3,...,Tr8 construct a queue for both part types. In the controller part As need to be put into Storage I and also part Bs need to be put into Storage II. This is achieved by using places P9 and P10. After this is done the sensor readings are added to the PNC. The distinction between part As and part Bs is achieved by using sensor SI_1 and sensor SI_2, at Tr1 and Tr2. Sensor reading SI_3 is used at Tr7 and Tr8 to detect presence of a component and at Tr9 and Tr10 to detect the absence of the component. Finally, the conveyor motor action is assigned to place P11 and the sort solenoid action is assinged to transition Tr9. Once, the PNC has been designed, flags and counters are then assigned to the places as shown in Table II.

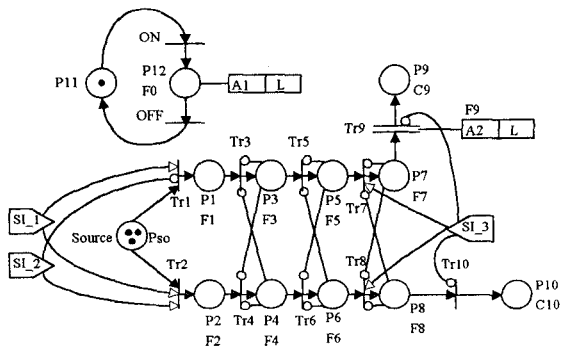

Fig. 18. Petri Net Controller for the manufacturing system.

TABLE II

PNC Places	Interpretation
Pso	Coming components
P1	first place of conveyor belt for part A
P2	first place of conveyor belt for part B
P3	second place of conveyor belt for part A
P4	second place of conveyor belt for part B
P5	third place of conveyor belt for part A
P6	third place of conveyor belt for part B
P7	fourth place of conveyor belt for part A
P8	fourth place of conveyor belt for part B
P9	place for part As in storage I
P10	place for part Bs in storage II
P11	Conveyor motor is off
P12	Conveyor motor is on

V. LADDER LOGIC DIAGRAM FOR THE PETRI NET CONTROLLER

The ladder logic diagram, shown in Fig. 19, is obtained for the Petri Net Controller given in Fig. 18. The LLD is achieved by using direct mapping form PNC to LLDs. The ladder logic symbols are defined in Table III. The ladder logic program has been structured in such a way that rung 0 initialises the system, rungs 1 to 10 represent the transitions Tr1 to Tr10 respectively. Finally, rung 11 represents level action A1 (conveyor motor operation) at the place P0 and rung 12 represents level action A2 (sort solenoid operation) at the transition Tr9. By adopting this concept further clarity can be added to the system documentation.

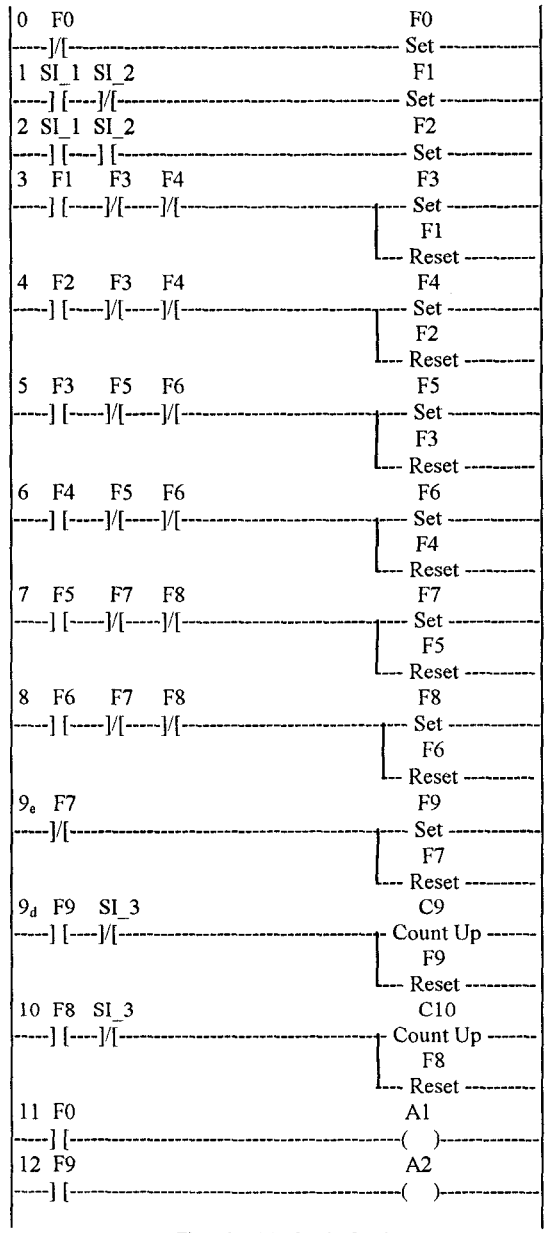

Fig. 19. LLD for the PNC.

TABLE III

LLD Symbols	Definition
F	Flag
C	Counter
SI	Sensor input
A	Action (Output)

VI. CONCLUSIONS

As automated manufacturing systems become more complex, the need for an effective design tool to produce both high level Discrete Event Control System (DECS) and low level implementation, becomes increasingly more important. Petri nets represent the most effective method for both the design and implementation of DECSs. The conversion of such Petri nets into real-time applications has recently been greatly simplified through the advent of the Token Passing Logic (TPL) methodology. The technique has been developed for normal Petri nets, P-timed Petri nets, T-timed Petri nets and Coloured Petri nets. In this paper the Petri net concepts have been extended to deal with Petri net controllers, by including actuators and sensors as formal structures within the Petri net controller. The conversion of such a Petri net controller into ladder logic diagrams has also been demonstrated by considering the control of a component sorting manufacturing system.

VII. REFERENCES

1. R. David, H. Alla, *Petri Nets and Grafcet , Tools For Modelling Discrete Event Systems*, Englewood Cliffs, NJ: Prentice Hall Inc., 1992.
2. M. A. Jafari, T. O. Boucher, "A Rule - Based System For Generating Ladder Logic Control Program From a High Level System Model", *Journal of Intelligent Manufacturing*, 5, 1994, pp. 103 - 120.
3. T. Satoh, H. Oshima, K. Nose, S. Kumagai, "Automatic Generation System of Ladder List Program By Petri Net", *in Proceedings of the IEEE International Workshop on Emerging Technologies on Factory Automation - Technology For The Intelligent Factory*, 1992, pp. 128 - 133.
4. S. Rattigan, "Using Petri Nets to Develop Programs for PLC Systems", *Lecture Notes in Computer Science 616: Application and Theory of Petri Nets*, 1992, pp. 368 - 372.
5. A. Taholakian, W.M.M. Hales, "The Design and Modelling of PLC Programs Using Petri Nets", *in Proceedings of the International Conference on Planned Maintenance, Reliability and Quality Assurance*, Cambridge, England, 6-7 April 1995, pp. 194 - 199.
6. J. Greene, "Petri Net Design Methodology for Sequential Control", *Masuarement and Control*, vol. 22, December/January 1989/1990, pp 288 - 291.
7. A.H. Jones, M. Uzam, A. H. Khan, D. Karimzadgan, S.B. Kenway, "A General Methodology for Converting Petri Nets Into Ladder Logic: The TPLL Methodology", *in Proceedings of the CIMAT'96*, France, May 29-31 1996, pp. 357-362.
8. M. Uzam and A.H. Jones, "Design of a Discrete Event Control System for a Manufacturing System Using Token Passing Ladder Logic", *in Proceedings of the CESA '96 IMACS Multi-Conference, Symp. on Discrete Events and Manufacturing Systems*, Lille, France, July 9-12, 1996, pp. 513 - 518.
9. A.H. Jones, M. Uzam, and N. Ajlouni, "Design of Discrete Event Control Systems for Programmable Logic Controllers Using T-Timed Petri Nets", *in IEEE International Symposium on Computer-Aided Control System Design - CACSD'96*, Michigan, USA, September 15-17, 1996.
10. M. Uzam and A.H. Jones, "Towards a Unified Methodology for Converting Coloured Petri Net Controllers into Ladder Logic Using TPLL : Part I - Methodology", *in International Workshop on Discrete Event Systems, WODES'96*, Edinburgh, UK, August 19-21, 1996.
11. A.H. Jones and M. Uzam, "Towards a Unified Methodology for Converting Coloured Petri Net Controllers into Ladder Logic Using TPLL : Part II - An Application", *in International Workshop on Discrete Event Systems, WODES'96*, Edinburgh, UK, August 19-21, 1996.
12. A.H. Jones and M. Uzam, "Design of Sequential Control Systems is Statement Lists Using TPL: Part I - Token Passing Statement List *to appear in 2nd Portuguese Control Conference - Controlo 96*, Porto, Portugal, September 11-13, 1996.
13. M. Uzam and A.H. Jones, "Design of Sequential Control Systems is Statement Lists Using TPL: Part II - An Application", *to appear in 2nd Portuguese Control Conference - Controlo 96*, Porto, Portugal, September 11-13, 1996.
14. A.H. Jones, M. Uzam and D. Karimzadgan, "Design of Knowledge Based Sequential Control Systems", *in Proceedings of the First International Symposium on Computing for Industry - ISSCI'96*, Motpellier, France, May 27-30 1996.
15. A.H. Jones, M. Uzam and P.B. de Moura Oliveira, "Design of Knowledge Based Sequential Control Systems for Manufacturing Systems", *in Proceedings of the IEEE Mediterranean Symposium on New Directions in Control & Automation - MSCA'96*, Crete, Greece, June 10-14, 1996, pp.764 - 769.