Journal of Computer Science 4 (5): 402-407, 2008
ISSN 1549-3636

A Software Agent for Automatic Creation of a PLC Program

Walid Mohamed Aly
Technical and Vocational Institute Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt

Abstract: Using structured design techniques to design a Programmable Logical Control (PLC) program would decrease the time needed for debugging and produces a concise bug free code. This study is concerned with the design of a software agent for automatic creation of code for a PLC program that can be downloaded on a Siemens Step 7 series. The code is generated according to the syntax rules for the AWL Language, AWL is the abbreviation for the germen word Anweisungsliste which means Instruction List The proposed system uses object oriented approach to transfer the design specification into an object that adequately describes the system using the state based design technique. The industrial system specifications are supplied by the user through a simple Graphical User Interface (GUI) environment These specification define the attributes vales of an object oriented class describing the control system, all the functions needed to generate the code are encapsulated in the class.

Key words: Software agent, PLC, automation, AWL, state diagram

INTRODUCTION

The act of design is recognized as an act of intelligence and with the complications of modern engineering systems design, the computer aided design plays a major role in this act.

Industrial control systems were historically designed using contactors and relays and since the 80's, a continuous drift to Programmable Logic Controllers (PLCs) is happening and with the cost reduction- that made a practical PLC[1] costs about 650 L.E(≈100$)- PLC is a rational choice for industrial control systems.

The IEC 61131- formally known as IEC 1131- is a standard for PLC, prepared and published by the International Electro technical Commission (IEC). IEC 61131-3[2] is the third chapter of IEC 61131 and is concerned with the programming languages standards by defining the syntax, semantics and structure for five different languages:

- Ladder diagram
- Function black diagram
- Instruction list
- Structure text
- Sequential function chart

Molina *et al*.[3] suggested the use of this standard in PLC leaning to decrease the knowledge gap between education and industry. Plc vendors like Siemens and Rockwell have their products as IEC 61131 complaint, however PLC products do not have to support the all five languages to be complaint.

The AWL language is adopted by Siemens[4] as a capability of the step 7 PLC series. AWL is the abbreviation for the germen word Anweisungsliste which means Instruction List

AWL resembles the instruction list language and its source code is in asci code format, that can be edited by a simple text editor and then imported using the Micro win or Simatic manager software offered by Siemens.

A software agent is software which is goal oriented and usually is designed to serve a specific purposes, an intelligent software agent would utilize artificial intelligent techniques in its quest towards the goal. Smith *et al*.[5] defined agent as follows:

Let us define an agent as a persistent software entity dedicated to a specific purpose. Persistent distinguishes agents from subroutines……....

This study aims to build a software agent enabling the PLC program designer to design the application at a higher descriptive level and have their code automatically generated by the agent as an AWL code.

The study starts with the current introductory section, followed by a section describing the basic PLC structured design techniques, then the next section declares the proposed automated design methodology followed by an application for this proposed methodology. A final section summarizes and discusses the results.

PLC structured design: Structured design techniques will produce a program that is reliable and predictable and using these techniques decreases the overall development time by reducing the time needed to edit and debug the program. Some of these techniques are[6].

Process sequence bits design: If the control system is sequential in nature and can be described in simple clear steps, process sequence bits can be used. Using this design, each step will be represented with a memory bit and the logic will be split into two main sections, a section that turns on and off the step bits according to specification, the other section is concerned with energizing and deenrgizing the outputs relative to each step.

Timing diagram design: This design method is suitable for system that is highly dependable on time, the timing diagram is drawn and each time value is assigned a timer which is usually an on delay timer. Different outputs will be energized and energized by contacts from these timers.

Flow chart based design: Flow chart based design can be used for sequential control system with simple decision making that might change the flow of control, the design starts with drawing the flow chart of the system, then each block in the flow chart is implemented using the master control relay instruction. Each block of code controls the transfer to the next block.

State diagrams: Design using state diagrams is a very useful design tool that can be applied to problems that have discrete states and transition exist to transfer from one state to another. It is based on dividing the whole design to three smaller design problems as follows:

- Design the relation between the inputs and the transitions
- Design the relation between the transitions and the states
- Design the relation between the different states and the outputs

Sequential Function Charts (SFC): The Sequential Function Charts (SFC) also known as Grafcet- can be used to design more complex systems which can have more than one state active at the same time. SFC is a subsection of the Petri net techniques

Although all of these techniques exist, however, many PLC programmers might ignore these design methods and work on programming based on their prior experience until they are satisfied with program and then start commissioning it.

Proposed automated design methodology: In this research, we use the state design technique for the automatic creation of the PLC Program. The main steps for the proposed methodology can be summarized as follows:

Stage 1: Enter problem description:
Step 1: The user enters the main date about the problem which is:

- Number of states: States-Num
- Number of inputs together with their names: Inputs-Num, Input-Names
- Number of outputs together with their names: Output-Num, Output-Names

All these data are the values of the attributes of the object created from class State Design

- Timers Data

Numbers of timers: Timers-num,
Timers' period: delay
State related to each timer: enabling-state
The Timers are stored as objects of class Timer

Step 2: The user enters the transitions information that is coded to a Boolean matrix called *Transition*, such that if Transition $_{I\ J}$ is true then Transition can occur from state I to state J.

Step 3: The user enters the information about the status of the outputs at each state, this information is coded to a Boolean matrix called Outputs States, such that if outputs states [I][J] is true, then output number I is enabled at state number J.

Step 4: For each transition, a transition condition is entered as a sum of products expression, resembling the combinational logic equations.

The GUI used for the entire prior steps is shown in Figure 1 with sample data filled.

Figure 2 shows the Unified Modeling Language (UML) class diagrams[7] of class State Design and class Timer.

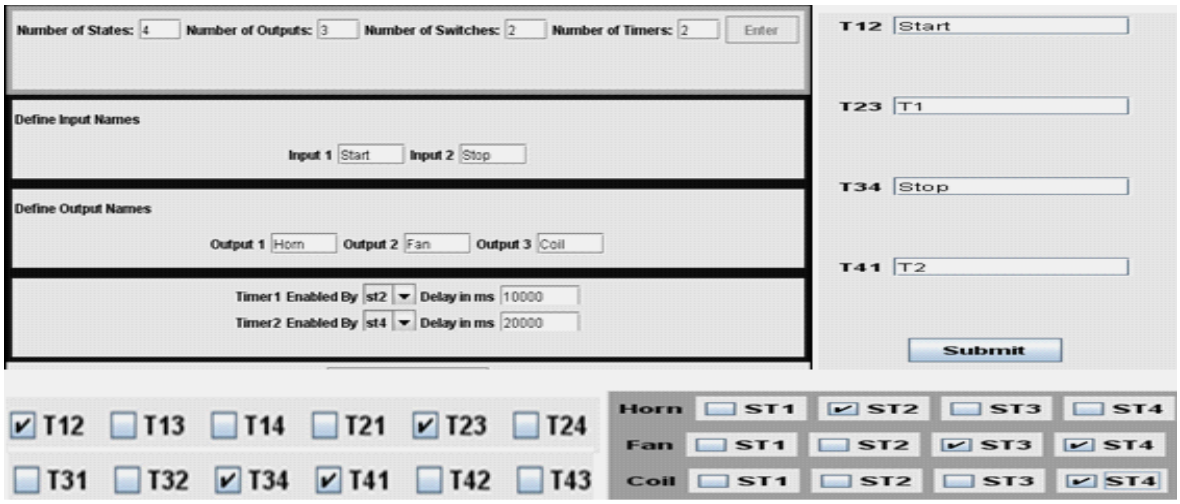

Fig. 1: GUI for entering problem description

Fig. 2: UML class diagrams for class StateDesign and class timer

Stage 2: Apply procedures to generate AWL code: The system will be designed based on state based design techniques three algorithms will be applied in sequence to produce the AWL code. First algorithm outputs the code for the relation between the inputs and the transitions, second algorithm outputs code for the relation between the transitions and the states and finally the third algorithm outputs the code for the relation between the states and the outputs. Each code produced from the algorithms is a subroutine the main method will have to call the three subroutines. Figure 3 shows the relation between states and transition for a four states problem.

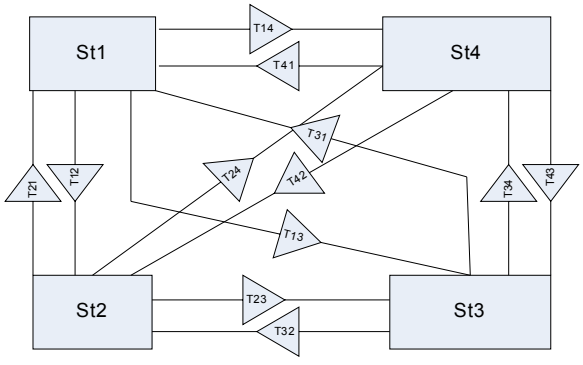

Fig. 3: Relation between states and transition for a four states problem

These three procedures are labeled as follows:

- Coding the relation between the inputs and the transitions
- Coding the relation between the transitions and the states
- Coding the relation between the states and the outputs

The description for these procedures is as follows.

Coding the relation between the inputs and the transitions: This algorithm is based on reading each transition equation as a sum of products equation and coding it using AWL code to a separate network, each network will have parallel branch with each branch consists of bit logic contacts, the OLD, instruction is used to execute the or relation between branches. The generated code will be stored as a String called SOP.

The first networks in this subroutine are used to initialize the on delay timers.

Procedure: Write –subroutine-calculate- transition

- For I =1 to Timers-num
  SOP+ = Network+i
  +LD+Timer[i].enabling-State+TON
  Timer+I +, +Timer[i].delay
  [next i]
- Read the N transition equations
- for I = Timers-num+1 to N+Timers_num
  Call Write-Network (Transition equations [i], i)
- [next i]
- End

Procedure: Write-Network (Transition equation: T, I)

- Start
- K = number of implicants in equation
- Split T based on the+character to subparts S_1, S_2, S_3…..S_K //Ex: A.B.C+D.E will produce S_1:A. B.C and S_2: D. E
- Each subpart produced from previous step is again split based on the '.' character to more subparts. //Ex: A. B.C will produce S_{11}: A , S_{12}: B and S_{13}: C
- Sop = Network + I
- For I = 1 to K
  If S_{i1} is not initially negated SOP + = LD + S_{i1} Else SOP + = LDN + S_{i1}
  L: Number of literals in S_i
  For j = 2 to L
  If S_{ij} is not initially negated SOP+ = A+S_{ij} Else SOP+ = AN+S_{ij}
  [next J]
  Sop+ = OLD
  [next i]

5-Sop+ = T
6-End

Coding the relation between the transitions and the states: The algorithm is based on reading the Transition matrix and coding the subroutine with each state as an output for each network, all the transitions to the state are connected using the or logic operation together as normally open contacts and the connected to normally closed contacts representing the transitions from the state using the AND logic Operation.

Procedure write-subroutine-calculate-states:

- Start
- Loop1: For I = 1 to States-Num
- S = S + Network + i
- S + = LD ST + i
- If I = 1 S + = O First-Scan-On
- Loop2: For j = 1 to States-Num
  If (Transition[j][i) S + = O + T- + j + I;
  [End Loop2]
- Loop3: For j = 1 to States-Num
  If (Transition[i] [j) S + = AN + T- + i + j
  [End Loop3]
- S + = ,= St + i
- [End Loop1]
- End

Coding the relation between the states and the outputs: The algorithm is based on reading the output states matrix and coding the subroutine with each output as the final output for each network, the code is generated simply by connecting the states that would energize an output with the or logic operation

Procedure write-subourtine-update-outputs:

- Start
- Outer Loop : For i = 1 to Outputs-Num
- S + = S+Network+I+LD
- Inner Loop: For j = 1 to Status-Num
  If (output states [i][j]) S+ = St+j+O
  [End Inner Loop]
- Remove O from S
- S+ = Output-Name [i]
- [End Outer Loop]
- End

Practical application: An oven is to be controlled using a PLC problem such that when a start pushbutton is presses the horn is on for 10 sec then both coil and fan are energized, pressing the stop pushbutton will cause the coil to be de-energized immediately and the fan to continue working for more 20 sec, then stops.

For the oven control problem, the configuration data is as follows in Table 1 and 2:

Table 1: Basic configuration data

Parameter	Value
No. of states	4: (st1, st2, st3, st4)
No. of inputs/names	2: Start-Stop
No. of outputs/names	3: Horn -fan-coil
No. of timers/enabling state/delay	2: Timer1 enabled by st2 delay 10000 ms Timer2 enabled by st4 delay 20000 ms

Table 2: State and Output Relation

State/output	Horn	fan	Coil
St1	Off	Off	Off
St2	On	Off	Off
St3	Off	On	On
St4	Off	On	Off

According to this problem description, the code generated by the software agent is as follows:

Generated code:

```
**** Subortine Update_Outputs****
Network 1
LD   ST2
= Horn
Network 2
LD   ST3
= Coil
Network 3
LD   ST3
O    ST4
= Fan
```

```
****Subortine Calulate_State****
Network 1
LD   ST1
 O    First_Scan_ON
O    T41
AN   T12
= ST1
Network 2
LD   ST2
O    T12
AN   T23
= ST2
Network 3
LD   ST3
O    T23
AN   T34
= ST3
Network 4
LD   ST4
O    T34
AN   T41
= ST4
```

```
****Subortine_Calculate_Transition****
Network 1
LD    ST2
TON   Timer1, 10000
Network 2
LD    ST4
TON   Timer2, 20000
Network 3
LD  Start
= T_12
Network 4
LD  Timer1
= T_23
Network 5
LD  Stop
= T_34
Network 6
LD  Timer2
= T_41
```

The mapping between the symbols deployed in the code and the real physical inputs and outputs or memory location is as shown in Table 3.

Table 3: Symbol table

Symbol	Address
Start	I0.0
Off	I0.1
ST1	M0.0
ST2	M0.1
ST3	M0.2
ST4	M0.3
T_12	M1.0
T_23	M1.1
T_34	M1.2
T_41	M1.3
Horn	Q0.0
Coil	Q0.1
Fan	Q0.2
Timer1	T32
Timer2	T33

DISCUSSION

Although different well structured techniques exist for creating PLC programs, usually engineers would prefer to code the program based on their intelligence and prior experience. When they feel satisfied with the program, they download and start commissioning the software, this might cause the production of non 100% functional software or creating a soft ware that is not easy to understand except by its programmer.

one of the other problems automatic control engineering might face is that PLC vendors supply the PLC hardware with the program download to C.P.U without giving the control engineers the privilege of having the program itself, nevertheless the engineers

might be totally aware of the system and can adequately describe it.

The automatic PLC program creation agent presented in this research is based only on the description supplied by engineer and would produce the AWL code that is ready to be downloaded to a PLC Siemens station.

The industrial system specifications are supplied by the user through a simple GUI environment either locally and be even done remotely using the remote method invocation supported by the JAVA programming language

The proposed agent is a solution for the mentioned problems as it will create a 100% functional code that would be easy to interpret; this was clearly demonstrated with the code generated for the oven control system mentioned in the research.

A more intelligent PLC software agent might even read the input states and output states and build its own perception of the control system and automatically produce the code without even a need to enter the problem specification.

REFERENCES

1. Bolton, W., 2006. Programmable Logic Controllers. 4th Edn. Newnes Publisher, U.S.A. ISBN-13**:** 978-0750681124
2. Lewis W., 1998. Programming Industrial Control Systems Using Iec 1131-3, Publisher**:** Institution of Electrical Engineers**,** U.S.A. ISBN-13: 978-0852969502
3. Molina, F.J., J.L. C.M. Barbancho and A. Gomez, 2007. Using industrial standards on PLC programming learning, seville control and automation. Proceedings of 15th Mediterranean Conference on Control and Automation (MED '07), June 2007, Greece, pp: 1-6.
4. Tubbs, P., 2007. Programmable Logic Controller (PLC) Tutorial, Siemens Simatic S7-200, 2007. Publisher**:** Stephen Philip Tubbs, Oklahoma, U.S.A. ISBN-13: 978-0965944687
5. Smith, D.C., A. Cypher and J. Spohrer, 1994. KidSim: Programming agents without a programming language. Communications of the ACM, Vol.37,pp. 55-67.
6. Hugh J., 2007. Automating Manufacturing Systems with PLCs, Retrieved from URL: http://www.eod.gvsu.edu/~jackh/books/plcs.
7. Ali, N.H., Z. Shukur and S. Idris, 2007. A design of an assessment system for UML Class Diagram. Proceedings OF Conference of Computational Science and its Applications (ICCSA07), August 2007, Kuala Lumpur, Malaysia, pp: 539-546. DOI: 10.1007/978-3-540-74472-6,URL: http://2007.iccsa.org/