DLadder - an Integrated Environment for Programming PIC Microcontrollers

Viša Tasić[1] , Dragan R.Milivojević[2], Vladimir Despotović[3] , Darko Brodić[4],

Marijana Pavlov[5] and Vladan Miljković[6]

Abstract – **DLadder is a new integrated development environment designed for programming the Microchip PIC microcontrollers. It allows programming using ladder logic, interpretation and compilation of programs into PIC16 and PIC18 native code. Debugger and the real-time simulation are also included, which allows displaying the contents of a microcontroller ports and memory locations in the real-time. Furthermore, the state of the I/O ports and A/D conversion results can be memorized in the database. Benefit of this development environment is the easy way of 'visual' programming in low level code. This paper describes briefly some elements of DLadder development environment.**

Keywords – **microcontroller, programming, ladder, software**

I. INTRODUCTION

Ladder diagram represents a program in ladder logic. Ladder logic is a programming language that represents a program by a graphical diagram based on the circuit diagrams of relay logic hardware. When a programmable logic controller (PLC) is used primarily to replace relays, timers, and counters, it's hard to beat the simplicity and usefulness of ladder diagram programming [1]. The name is based on the observation that programs in this language resemble ladders, with two vertical rails and a series of horizontal rungs between them. The logic in a ladder diagram typically flows from left to right. The diagram can be divided into sections called rungs as shown in Fig. 1.

[1] Viša Tasić is with the Institute of Mining and Metallurgy, Department of Industrial Informatics, Zeleni bulevar 35, 19210 Bor, Serbia, e-mail: visa.tasic@irmbor.co.rs,

[2] Dragan R. Milivojević is with the Institute of Mining and Metallurgy, Department of Industrial Informatics, Zeleni bulevar 35, 19210 Bor, Serbia, e-mail: dragan.milivojevic@irmbor.co.rs,

[3] Vladimir Despotović is with the University of Belgrade, Technical Faculty in Bor, Vojske Jugoslavije 12, 19210 Bor, Serbia, e-mail: vdespotovic@tf.bor.ac.rs,

[4] Darko Brodić is with the University of Belgrade, Technical Faculty in Bor, Vojske Jugoslavije 12, 19210 Bor, Serbia, e-mail: dbrodic@tf.bor.ac.rs,

[5] Marijana Pavlov is with the Institute of Mining and Metallurgy, Department of Industrial Informatics, Zeleni bulevar 35, 19210 Bor, Serbia, e-mail: marijana.pavlov@irmbor.co.rs

[6] Vladan Miljković is with the Institute of Mining and Metallurgy, Department of Informatics, Zeleni bulevar 35, 19210 Bor, Serbia, e-mail: visa.tasic@irmbor.co.rs

Each rung typically consists of a combination of input instructions. These instructions lead to a single output instruction; however, rungs containing function block instructions may be more complicated.

Ladder logic has contacts that make or break circuits to control coils. Each coil or contact corresponds to the status of a single bit in the programmable controller's memory. Unlike electromechanical relays, a ladder program can refer any number of times to the status of a single bit, equivalent to a relay with an indefinitely large number of contacts. So-called "contacts" may refer to physical inputs to the programmable controller from physical devices such as pushbuttons and limit switches via an integrated or external input module, or may represent the status of internal storage bits which may be generated elsewhere in the program.

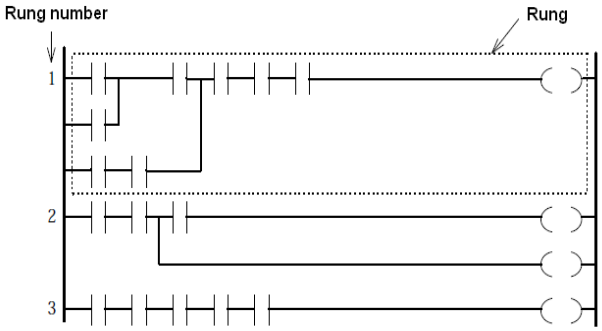

Fig. 1. The basic elements of the ladder diagram

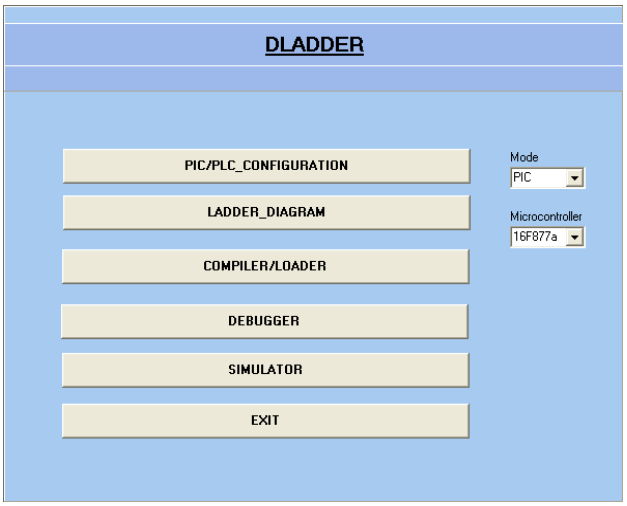

Fig. 2. DLadder main window

Ladder notation is best suited to control problems where binary variables are only required and interlocking and sequencing of binary are the primary control problems. Analog quantities and arithmetical operations are not suitable to express in ladder logic and each manufacturer has different ways of extending the notation for these problems. As microprocessors have become more powerful, notations such as sequential function charts and function block diagrams can replace ladder logic for some limited applications. Very large programmable controllers may have all or part of the programming carried out in a dialect that resembles BASIC or C or other programming language with bindings appropriate for a real-time application environment.

II. DLADDER INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

DLadder is a new integrated development environment (IDE) developed in Borland Delphi 7, and designed for certain Microchip PIC microcontrollers [2], mainly for the purpose of development of applications for control and data acquisition of industrial processes. It allows programming using ladder logic, and compilation of programs into PIC16 and PIC18 native code. However, the PIC microcontroller with this software is able to perform the PLC functions, by some additional features and predefining functional blocks. A PLC is a modular device programmable by using ladder diagrams. It internally uses a microcontroller to handle all input, output and logic scans.

DLadder allows writing programs for microcontroller basic configuration only, or for expanded configurations with a number of additional new input or output modules. Debugger and the real-time simulator are also designed, which allows displaying the state of a microcontroller ports and contents of memory locations in the real-time. Thus, DLadder offers an easy way of 'visual' programming in low level code. DLadder's main window is shown in Fig. 2. It consists of several modules that will be briefly described below.

The main difference between PLC and microcontrollers is only the way of programming. Therefore, in the very beginning the mode of operation (basic configuration – PIC mode; expanded configuration – PLC mode) should be selected. Afterwards, parameters of the PIC/PLC_CONFIGURATION are configured based on the selected mode, as shown in Fig. 3.

A. Ladder Diagram Editor

The option LADDER_DIAGRAM opens the editor for drawing ladder diagrams. Writing a ladder diagram is performed by selecting the required elements from the object toolbar (shown in Fig. 4) and setting them in the appropriate position on the screen. Each window in ladder editor consists of eight rows and 11 columns. The last column in each row is reserved for placing the coils and function objects.

The function objects supporting the arithmetic operations, logic operations, etc. Most objects do not require any special settings. However, it is necessary to enter the memory location to which the object is referenced. An example of a simple ladder program simulating the operation of the traffic lights (semaphore) is shown in Fig. 4. An appropriate electric wiring diagram for the given case realized with the PIC16F877A microcontroller [3] is given in Fig. 5.

Fig. 3. The PIC/PLC configuration window

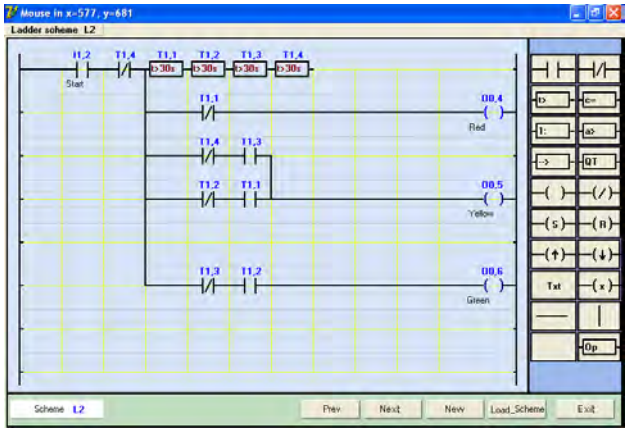

Fig. 4. The ladder diagram created in DLadder editor

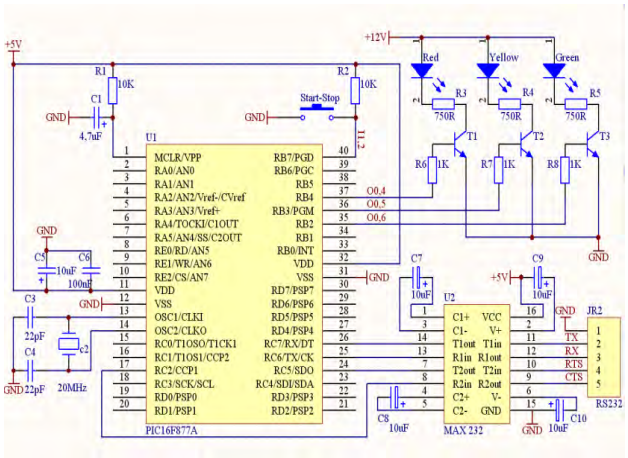

Fig. 5. The semaphore electric scheme realized with the microcontroller PIC16F877A

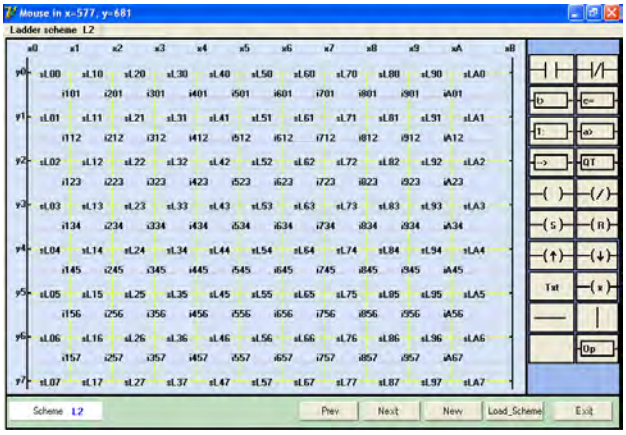

Fig. 6. The fields in DLadder database associated to the first screen of the ladder diagram

Larger ladder diagrams consist of several windows. At the bottom of the editor window there are buttons for scrolling through the ladder diagram windows, for opening of a new diagram or closing the editor. The existed ladder diagram can be loaded and modified by inserting or deleting some objects.

B. DLadder Compiler/Loader

Each ladder diagram is stored in the database as a series of symbols. The database records the position of each object on the ladder diagram, as well as their interrelations. List of fields in the database with their positions on the ladder diagram is shown in Fig. 6. The source code of the program, whether it is written in assembler or other programming language (C, Pascal, and ladder) has to be translated into machine language of the microcontroller. As a result of this operation an object file will be created in hexadecimal format (*. hex), which carries a series microcontroller instructions. Hence, in order to be ready for loading into the PIC microcontroller the ladder diagrams must be compiled and stored into the hex files in the prescribed form. Microchip has published a protocol for transmission of hex file into internal program memory of microcontrollers. An example of the structure of hex file is shown in Fig. 7.

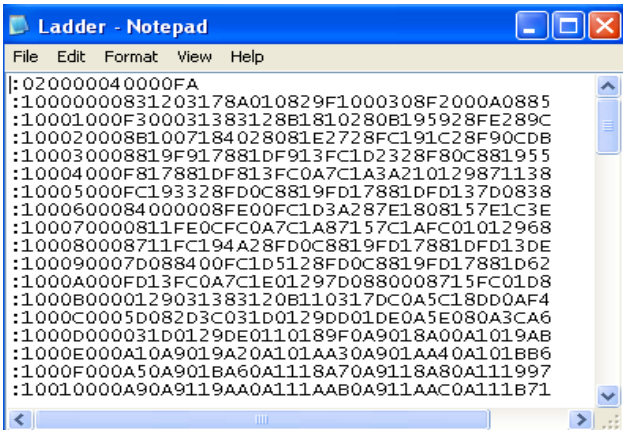

Fig. 7. The structure of the hex file prepared for programming the PIC microcontroller

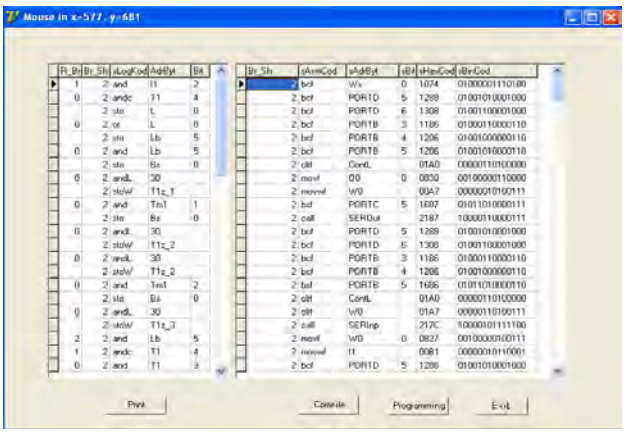

Fig. 8. DLadder's compiler/loader window

The option COMPILER/LOADER in DLadder main window opens compiler/loader window (shown in Fig. 8). Compiling of the ladder diagram starts after pressing the *Compile* button. The compilation of the ladder diagram is executed from left to right. The result of the compilation process is the code written into hex file. In a special case, when the PIC microcontroller on the development board is connected to PC, it is possible to program the microcontroller directly from DLadder. In such a case, by pressing the *Programming* button, the latter compiled program will be immediately loaded into the PIC microcontroller. DLadder uses both the USB and serial connection for file transfer, monitoring and control of the PIC program execution.

C. DLadder Debugger/Simulator

Once the code has been built and checked from the syntax point of view, it needs to be tested. Debugging is a process of finding and reducing the number of errors in a computer program. In order to test the code, we need some kind of software or hardware that will execute the PIC micro instructions [5].

DLladder's integrated debugger uses the internal in-circuit debug hardware of the target Flash PIC microcontroller to run and test the application program. When DEBUGGER option is selected in the main window, the application code is programmed into the PIC microcontroller's memory. A small "debug executive" program is loaded into the high area of program memory. Since the debug executive must reside in program memory, the application program must not use this reserved space.

The debug executive runs just like an application in program memory. It uses some locations on the hardware stack and file registers for its temporary variables. Special "in-circuit debug" registers in the target microcontroller are enabled. These allow the debug executive to be activated by the DLadder. The target microcontroller is held in reset by keeping the VPP/MCLR line low. DLadder will raise the VPP/MCLR line to allow the target microcontroller to run, starting from address zero and execute until the program counter reaches the breakpoint address previously stored in the internal debug registers.

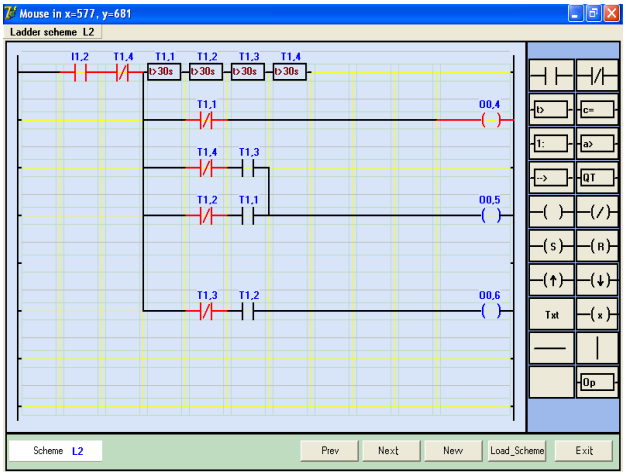

Fig. 9. DLadder debugger window

After the instruction at the breakpoint address is executed, the in-circuit debug mechanism transfers the program counter to the debug executive (much like an interrupt) and the user's application is effectively halted. DLadder IDE communicates with the debug executive via PGC and PGD lines [6, 7], gets the breakpoint status information and sends it back to the DLadder IDE. The DLadder IDE then sends a series of queries to get information about file register contents and the state of the CPU. These queries are ultimately performed by the debug executive.

Based on the described principle it is possible to monitor the status of certain elements of ladder diagram. The program appears on screen with the energized (true) branches highlighted (red color), as shown in Fig. 9, which makes it easy to debug.

The simulator is a software program that runs on the PC to simulate the state of input instructions of the PIC microcontrollers. When SIMULATOR option is selected in the main window, it allows monitoring of ladder diagram and changing (forcing) state of inputs during the program execution.

CONCLUSION

The primary motive for developing DLadder IDE is a desire to facilitate program writing for PIC microcontrollers using the concept of visual programming. Although there are a number of development environments for PIC microcontrollers on the market, not many of them allow writing the programs in ladder logic. DLadder IDE, which is still in the stage of continuous development, is shown to be reliable and easy-to-use. Thus, we believe that DLadder environment can be a useful tool for programming the PIC microcontrollers.

ACKNOWLEDGEMENT

This work was supported by Ministry of Education and Science of the Republic of Serbia under Project TR33037 "Development and Application of Distributed System for Monitoring and Control of Electrical Energy Consumption for Large Consumers".

REFERENCES

[1] W. Bolton, *Programmable Logic Controllers*, Fifth Edition, Newnes, 2009.

[2] M. Verle, *PIC microcontrollers - Programming in C*, Belgrade, Serbia, mikroElektronika, 2009. (in Serbian) http://www.mikroe.com/eng/chapters/view/14/chapter-1-world-of-microcontrollers/#c1v4

[3] Microchip Inc., *PIC16F87XA Datasheet* (10/31/2003) http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010242

[4] V. Milanović, *PC Interfaces*, Lazarevac, Serbia, Elvod Print, 2005. (in Serbian)

[5] M. Verle, *PIC microcontrollers*, Belgrade, Serbia, mikroElektronika, 2007. (in Serbian)

[6] Microchip Inc., *PIC16F87XA Data Sheet 28/40-pin Enhanced FLASH Microcontrollers*, Microchip Technology Inc., 2003. www.datasheetcatalog.org/datasheets2/41/4109312_1.pdf

[7] Microchip Inc., *MPLAB ICD 2 In-Circuit Debugger User's Guide*, Microchip Technology Inc., 2005. http://ww1.microchip.com/downloads/en/devicedoc/51331b.pdf