IEC Structured Text programming
of a small Distributed Control System

Dariusz Rzońca*, Jan Sadolewski*, Andrzej Stec*, Zbigniew Świder*, Bartosz Trybus*, and Leszek Trybus*
*Rzeszow University of Technology.
Faculty of Electrical and Computer Engineering
Wincentego Pola 2, 35-959 Rzeszów, Poland
Email: {drzonca, js, astec, swiderzb, btrybus, ltrybus}@prz-rzeszow.pl

Abstract—A prototype environment called CPDev for programming small distributed control-and-measurement systems in Structured Text (ST) language of IEC 61131-3 standard is presented. The environment consists of a compiler, simulator and configurator of hardware resources, including communications. Programming a mini-DCS (Distributed Control System) from LUMEL Zielona Góra is the first application of CPDev.

I. INTRODUCTION

Domestic control-and-measurement industry manufactures transmitters, actuators, drives, PID (Proportional-Integral-Derivative) and PLC (Programmable Logic) controllers, recorders, etc. Connected into distributed systems, they are used for automation of small and medium scale plants. However, engineering tools used for programming such devices are rather simple and do not correspond to IEC 61131-3 standard [2] (Polish law since 2004).

This paper presents current state of work on engineering environment called CPDev (Control Program Developer) for programming small control-and-measurement devices and distributed mini-systems according to the IEC standard (digits dropped for brevity). First implementation involves instruments from LUMEL Zielona Góra [7]. Initial information on CPDev was presented at the previous IMCSIT conference [5]. Similar environments have been described in [1], [6]

The CPDev environment (called also package) consists of three programs executed by PC and one by the controller. At the PC side we have:
- CPDev compiler of ST language,
- CPSim software simulator,
- CPCon configurator of hardware resources.

The programs exchange data through files in appropriate formats. The CPDev compiler generates an universal code executed by virtual machine VM at the controller side. The machine operates as an interpreter. The code is a list of primitive instructions of the virtual machine language called VMASM assembler. VMASM is not related to any particular processor, however it is close to somewhat extended typical assemblers. In this way, portability of the compiled code for different hardware platforms is provided. On the contrary, other solutions are usually built around the concept of translating IEC language programs into C code [6].

Basic characteristics of VMASM and virtual machine are given in [5]. CPSim simulator also involves the machine (in this case at the PC side).

The CPDev package is developed in C# language of .NET Framework. The virtual machine is written in ANSI C and compiled with appropriate, platform-dependent compilers e.g. avr-gcc in case of the LUMEL’s mini-DCS. Other languages and programming environments are also used in specific cases.

The implementation of CPDev components was supported by lexical diagrams (compiler), object-oriented modelling techniques (programming environment) and coloured Petri nets (communication subsystem), see [5], [9].

II. A PROGRAM IN ST LANGUAGE

Main window of user interface in the CPDev environment is shown in Fig. 1. It consists of three areas:
- tree of project structure, on the left,
- program in ST language, center,
- message list, bottom.

![Fig. 1. User interface in CPDev environment](image)

Tree of the START_STOP project shown in the figure includes Program Organization Unit (POU) with the program PRG_START_STOP, five global variables from START to PUMP, task TSK_START_STOP, and two standard function blocks TON and TOF from IEC_61131 library.

The PRG_START_STOP program seen in the main area is written according to ST language rules. The first part involves
declarations of instances DELAY_ON, DELAY_OFF of the function blocks TON and TOF. Declarations of the global variables (EXTERNAL) are the second part, and four instructions of the program body, the third one. The instructions correspond to FBD (Function Block Diagram) shown in Fig. 2. So one can expect that certain MOTOR is turned on immediately after pressing a button START and a PUMP five seconds later. Pressing STOP or activation of an ALARM sensor triggers similar turn off sequence.

III. GLOBAL VARIABLES AND TASK

Global variables can be declared in CPDev either using individual windows or collectively at a variable list. The list for the START_STOP project is shown in Fig. 3.

The addresses specify directly represented variables [2], [3] and denote relative location in controller memory (keyword AT declares the address in individual window). Here these addresses are called local. Variables without addresses (not used in this project) are located automatically by the compiler.

Window with declaration of the TSK_START_STOP task is shown in Fig. 4. A task can be executed once, cyclically or continuously (triggered immediately after completing, as in small PLCs). There is no limit on the number of programs assigned to a task, however a program can be assigned only once.

Text of the project represented by the tree is kept in an XML text file. Compilation is executed by calling Project->Build from the main menu. Messages appear in the lower area of the interface display (Fig. 1). If there are no mistakes, the compiled project is stored in two files. The first one contains universal executable code in binary format for the virtual machine. The second one contains mnemonic code [5], together with some information for simulator and hardware configurer (variable names, etc.).

IV. FUNCTIONS AND LIBRARIES

The CPDev compiler provides most of standard functions defined in IEC standard. Six groups of them followed by examples are listed below:

- type conversions: INT_TO_REAL, TIME_TO_DINT, TRUNC,
- numerical functions: ADD, SUB, MUL, DIV, SQRT, ABS, LN,
- Boolean and bit shift functions: AND, OR, NOT, SHL, ROR,
- selection and comparison functions: SEL, MAX, LIMIT, MUX, EQ, LT,
- character string functions: LEN, LEFT, CONCAT, INSERT,
- functions of time data types: ADD, SUB, MUL, DIV (IEC uses the same names as for numerical functions).

Selector SEL, limiter LIMIT and multiplexer MUX from selection and comparison group are particularly useful. Variables of any numerical type, i.e. INT, DINT, UINT and REAL (called ANY_NUM in IEC [2], [3]) are arguments in most of relevant functions.

Typical program in ST language is a list of function block calls, where inputs to successive blocks are outputs from previous ones (see Fig. 2). So far the CPDev package provides three libraries:

- IEC_61131 standard library,
- Basic_blocks library with simple blocks supplementing the standard,
- Complex_blocks library for continuous regulation and sequential control.
Table I lists blocks from the first and second libraries. Blocks such as PID controller, servo positioner, multi-step sequencer, dosing block, etc., belong to the third library.

The user can develop functions, function blocks and programs, and store them in his libraries. Tables of single-size are available only.

V. CPSIM SIMULATOR

The compiled project may be verified by simulation before downloading into the controller. The CPSim simulator can be used in two ways:

- before configuration of hardware resources (simulation of the algorithm),
- after configuration of the resources (simulation of the whole system).

The first way involves logic layer of the CPDev environment. PC computer operates as a virtual machine executing universal code. The second way requires configuration of hardware resources, so it is application dependent. The CPCon configurer generates hardware allocation map (see below) that assigns local addresses to physical ones and specifies conversion of ST data formats into formats accepted by hardware. The objective is to bring simulation close to the hardware level, so CPSim uses both the code and the map. Simulation window of the START_STOP project is shown in Fig. 5. The two faceplates on the left present values of three inputs and two outputs (TRUE is marked). The user can select faceplates, arrange them on the screen and assign variables. Simulated values can be set both in group and in individual faceplates.

So far the window of Fig. 5 is used for simulation only. In future it will be also employed for on-line tests (commissioning).

VI. CPCON CONFIGURER AND MINI-DCS

The CPCon configurer defines hardware resources for particular application. The example considered here involves mini-DCS with SMC programmable controller, I/O modules of SM series and eventually other devices from LUMEL Zielona Góra [7]. Modbus RTU protocol is employed [4] on both sides of the SMC.

Fig. 6 shows test realization of the system with SMC controller (on the left), SM5 binary input module (middle), and SM4 binary output module (on the right). The console with pushbuttons and LEDs (below) is used for testing. The PC runs first the CPDev package and a SCADA (Supervisory Control And Data Acquisition) system later. PC and SMC are connected via USB channel configured as a virtual serial port.

The CPCon configurer functions are as follows:

- configuration of a communication between SMC and SM I/O modules,
addresses beginning from 0000. In SMC these variables have consecutive since BOOL occupies single byte in CPDev, the interface with the code So all variables can be read in a single Modbus transaction inputs in consecutive 16-bit registers beginning from 4003.

START_STOP pushbuttons in the console (Fig. 6) which, in case of the SM4 binary input module (remote). SM5 is connected to SM5 binary input module (remote). SM5 is connected to communication with SM4 binary output module. SM4 controls the console LEDs. Two consecutive variables, MOTOR and PUMP, the first one with the local address 0008, are sent to SM4 by single message with the code FC16 to remote addresses beginning from 4205 (write group of registers). This time 8→16 bit conversion is needed.

VII. CONCLUSIONS

CPDev environment for programming industrial controllers and other control-and-measurement devices according to IEC 61131-3 standard has been presented. The environment consists of ST language compiler, project simulator, and configurator of hardware resources, including communications. The user can program his own function blocks and create libraries. Mini-DCS control-and-measurement system form LUMEL is the first application of the package.

Programs written in the future in other IEC languages, first of all in FBD, will also be compiled to the VMASM code and executed by the virtual machine. Appropriate compilers are under development. XML format for data exchange between languages has already been defined by PLCOpen [1], [8].

ACKNOWLEDGMENT

Support from the MNiSzW R02 058 03 grant is gratefully acknowledged.

REFERENCES


Fig. 7. Communication configuration of the START_STOP project.