Universal language extension for conducting laboratory experiments on analogue and digital circuit design

George Popov[1*], *Filip* Krastev[2]

[1]TU FCST, Information Technology for the Industry, 8 Kliment Ohridski Blvd, 1000 Sofia, Bulgaria

[2]TU FCST, Information Technology for the Industry, 8 Kliment Ohridski Blvd, 1000 Sofia, Bulgaria

Abstract. The article examines in a specialized language different variants of software provision in a specialized environment for conducting laboratory experiments on analogue and digital circuits. A universal language for ULLE laboratory experiments was developed as a resident driver. When declaring an I / O variable (to connect to lab model), it is transmitted to the driver, which is invoked similarly to the system interrupt dispatcher. This way, each time a call is invoked, the model's output channels and input variables are refreshed. Such a conversion allows the driver to work with programs written in different programming languages.

1 Introduction

Despite the diversity and complexity of the specialized company software of automated systems for conducting laboratory experiments [1,2,3,4], it can be said that there are two basic approaches:

- monolith systems – these are hardware systems whose functionality is available through user menus, as data is entered/displayed in windows or charts are to be taken;
- soft systems – these are systems that use a specialized language for conducting laboratory experiments.

Monolithic systems are menu-oriented and perform a strictly defined function. Their main advantage lies in the rapid convening with the system and the easier conduct of laboratory experiments. On the other hand, they follow a certain model from which they cannot deviate, which deprives them of the ability to adapt to different types of tasks and applications.

Systems that are based on programming language are more "agile", enabling the researcher to set up a specific research program according to the needs of the experiment. Another advantage of the software systems is that to them it is possible to emulate through a program code part of the studied electronic circuit (or hardware research). Examples of this is to set OOB (Negative Feed Back) in source code or the signal from a research output hardware to be read by the software, to get redesigned and passed to the input entrance of another scheme.

The proposed method in the article was carried in the microcomputer system for the conduct of laboratory experiments Micro Lab [5,6]. It consists of a general-purpose PC and a non-intelligent (non-CPU control) laboratory model containing a constant analog-digital periphery, having 4 analogue inputs, 4 analogue outputs and a removable module (Fig. 1) as well, which is different for each laboratory experiment.

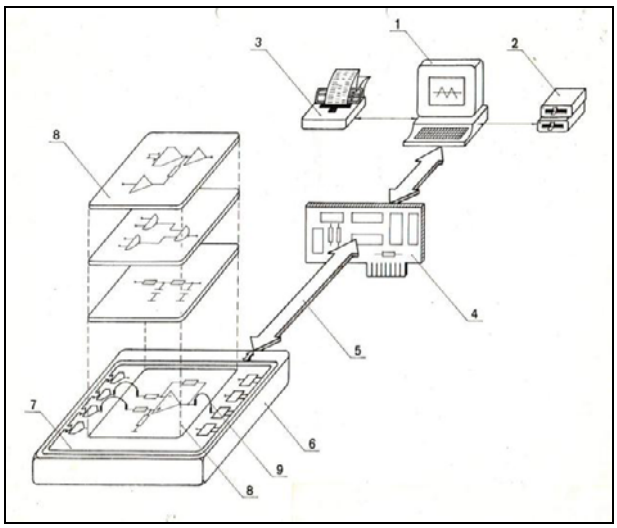

Fig.1. System for laboratory experiments Micro Lab. It can be observed constant (7) and exchangeable modules (8).

The researcher realizes the experimental layout of the laboratory model by performing the necessary connections with commutating(switching) lines (Fig. 2). The study is conducted using the stimulus-response method with the help of a special program written for the specific experiment, that sets values of voltage of output channels and read voltages of input channels.

* Corresponding author: popovg@tu-sofia.bg

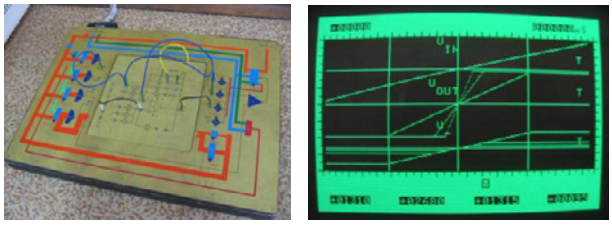

Fig. 2. Associated experimental staging and result output of the experiment. The transmission characteristics at different OA gain values are observed.

The described system provides a complete functional basis for conducting the experiments described in [7].

2 Description of languages for the conduct of laboratory experiments.

The universal language for laboratory experiments ULLE was established in 1988 at the Technical University of Sofia, developed over the years adequately to the software and hardware standards of the modern computer systems. The main idea of most versions of the language consist in the possibility to declare input/output variables of a type model. Another feature is that these variables are refreshed automatically without engaging the researcher.

2.1. ULLE for PCs Apple II, IBM PC with DOS operating system.

For an Apple II PC, ULLE is an extension of the built-in language BASIC. The idea of realization comes from the built-in language operator & that transmits governance in a specific point (address) where the processing of the additional command from the special user program can be performed. BASIC usability is that the researcher can send special commands in programming and dialog mode, i.e. to stop and continue the process of implementation (interpretation) of the program. In the IBM PC with DOS operating system, a ULLE universal resident driver has been implemented, working with most of the programming languages of its time: BASIC of C and Pascal.

The idea of the realization is the following: after declaring a variable of type model, its address is intercepted by the extension of the language. Two types of actions are being carried out:

- If the variable is of type OUTPUT, its value is constantly fed by the dedicated hardware at the output of the model;
- If the variable is of type INUPT, it is constantly refreshed by the output of the model. In this case, the manufacturer (the lab model) continuously produces data and update it in memory cells, which are really addresses of variables in the language of high level, and the user (the program written in the language for conducting laboratory experiments) read them only when needed.

In order to achieve the above features two conversions have been made. In the earlier version, the driver is called out by an interruption 1C h or 70 h, thus refreshing the corresponding input/output variables [8]. Hence the term “live variable” and “Live Variables BASIC” or “LVBASIC” as a version of ULLE.

The later version imposed use of special hardware that monitors for change in values of the “living” variables and generates the interruption that invokes driver execution. This implementation seems modern, but unfortunately it would complicate greatly the laboratory staging. Examples of code for these versions of ULLE (LVBASIC version).

2.1.1 Declaration of variable

- For the input channel on the laboratory staging
&INP (<CHANNEL>, <VAR>, < VALUE>);

- For the output channel
&OUT (<CHANNEL >, <VAR>, < VALUE>);
where:

CHANNEL – the ID number of the input or output channel in Micro Lab periphery;

VAR – the given name of the “living” variable, which will monitor (and transmit) value in the chosen channel;

VALUE – the actual value in millivolts. If the input voltage on the specified entrance changes with a bigger value than stated, an interruption with VALUE is generated to refresh the variable.

2.1.2 Variable release

Release is done with the command:

& KILL <VAR>

After release, the “live variable” stops tracking the channel by becoming a regular variable and retaining its last value. This can be used to memorize value at a time and is extremely convenient when working with arrays.

2.1.3 Notation of data structures and operations

A living variable can participate in an arithmetic expression similar to an ordinary variable. For example, if UI is the input voltage and U2 is the output and are both defined as “live variables”, the transmission characteristic can be plotted with:

100 DRAW (*Ul*, *U2*): GOTO 100,

and the dependence of the output power with load R on the output voltage is plotted with:

100 DRAW (*U2*, (*U2***U2*)/*R*): GOTO 100

Filling a buffer (array) with data from an input channel is realized as follows:

10 DIM *A* [100]

20 FOR TO 100

30 *&INP* (1, *A*[I], 200)

40 NEXT *I*

Here the process of "come to life" on each new element of the array, the previous one holds its constant value. In case the following line is inserted:

25 NOT STROBE THEN GOTO 25

It can be performed software strobing to data entry through the living variable STROBE, reading at the input channel of the system.

Similarly, the output of a predefined data buffer is implemented in the following way:

10 DIM *A* [100] ; Array to output

20 &OUT (1, *X*) ; Define living variable X for output channel 1

30 FOR *I*=1 TO 100: *X*=*A*[I]: NEXT ; Send the array through the output channel 1

When it is necessary to remember the current value of a "living" variable, for example X, it can be done in several ways:

- release the variable

&KILL *X*

- assign to a different variable (e.g. TEMP):

TEMP= *X*

-define another "live" variable on the channel:

&INP (1, *Y*, 150)

It is possible that a "living" variable X is simultaneously defined as input and output. Then it will output to the output channel the magnitude that is observed on the input.

LVBasic makes it easy to describe an experimental set-up and to implement linked data structures. The following example illustrates the implementation of a large-scale amplifier (through a real operational amplifier of the removable module) and a feedback with a coefficient of transmission specified by the constant FB. The entrance of the inverting scale amplifier is connected to an output channel 1 and its exit to input channel 2.

10 *&OUT* (1, O_1) ; Voltage output 1 will be equal to the value of the variable IN_1

20 *INP* (2, IN_2) ; The variable IN_2 tracks the value of channel 2.

30 *INP* (1, *C*) ; The variable C tracks the value of channel 1.

40 *&O_1* = IN_2**FB*;

50 PRINT IN_2 ; Print the current result.

60 IF *C* < 10000 THEN 40 ; While C is less than 10000 mV, the cycle 40-60 is running.

2.2. ULLE for PC with Windows OS

In order to make full use of the increased capabilities of the computer systems, the following ULLE implementations were made.

- Realization based on multithreading environment, developed in TU-Sofia. The environment represents the core which supports time share arrangement and co-op mode as well as synchronization and communication of user level threads. New features have been added to the kernel resident module API, supporting the entrances and exits of the model. The essential advantage of this realization is the better structure of the code, which allows for the base of this ULLE version to build a newly designed hard system for the conduct of laboratory experiments.
- Implementation of Open MP. The underlying disadvantage of its own multithreading environment is that it represents itself as a process to Windows and cannot efficiently use the resources of multicore platforms. For this purpose, the last modification of ULLE was made.

3. Conclusion

System for laboratory exercises MicroLab is used to conduct laboratory experiments on analogue and digital circuitry disciplines for more than 30 years and has gained considerable experience. For the sake of brevity of the exhibition, not all the features of ULLE are given. Software provisioning was changing with different computer generations, following two main concepts: MicroLab Integrated (Monolithic) Environment and Universal Language for Laboratory Experiments ULLE. The conclusions drawn are:

- To enhance the experiment and put it in the forefront, the tools used should not be too complicated and distracting the learner;
- To add a part of the laboratory experiment in cloud [9] and make option for distance learning;
- There's been a comparative survey among students in analogue circuitry, as most of them prefer the language extension with live variables instead of a library with corresponding functions;
- Using of BASIC is convenient, because it has automatic declaration of variables, provides opportunities for direct work with the interpreter, i.e. the ability to directly set/read out values from the model, including interrupting and continuing a program as well as inserting a code during the execution.

Acknowledgment

This work is supported by the Project № BG05M2OP001-2.009-0033-C01

References

1. H. Austerlitz, AP, *Data Acquisition Techniques Using PC*, San Diego, California, (1991)

2. Nuno Sousa, Gustavo R. Alves, and Manuel G. Gericota, *An Integrated Reusable Remote Laboratory to Complement Electronics Teaching*, IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES, VOL. 3, NO. 3, JULY-SEPTEMBER 2010

3. V.J. Harward et al., "*The iLab Shared Architecture: A Web Services Infrastructure to Build Communities of Internet Accessible Laboratories," Proc. IEEE,* vol. 96, no. 6, pp. 931-950, June 2008

4. D. Magin and S. Kanapathipillai, *"Engineering Students' Understanding of the Role of Experimentation"*, European Journal of Engineering Education, 2000, Vol. 25, no. 4, pp. 351-358

5. Mechkov K.,, *Computer Based Technical Laboratory for Conduction of Laboratory. Experiments, Announcements,* VMEI, 44th, Book 9 (1989)

6. Zhekov Z., K. Mechkov. Modular System for Automation of Practical Laboratory Experiment., 3rd Nat. Ass.: Syst. Auto. Eng. Labour and Sci. Res. SAITNI-89, B.G. Albena, 10 (1989)

7. https://wiki.analog.com/university/courses/electronics/labs

8. Popov G., *"Parallel Data Exchange Through Living Variables"*, SAITNI-93, Albena, 1989 (bulg. version)

9. Hakima Mostefaoui,.Abdelhalim Benachenhou and Abderrahmane Adda Benattia, *"Design of a low cost remote electronic laboratory suitable for low bandwidth connection"* , Computer Applications in Engineering Education Volume 25, Issue 3, pages 480–488, May 2017