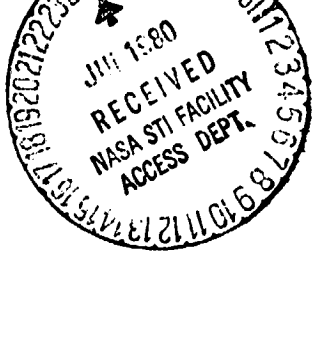
NASA TECHNICAL MEMORANDUM

NASA TM-76243

HIGH LEVEL LANGUAGE FOR MEASUREMENT COMPLEX CONTROL BASED ON THE COMPUTER E-100I

B. V. Zubkov

Translation of "Yazyk vysokogo urovnya dlya upravleniya izmeritel'nym kompleksom na baze EVM E-100I", Academy of Sciences USSR, Institute of Space Research, Moscow, Report Pr-404, pp 1-32, 1978



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
WASHINGTON D.C.
JUNE 1980

A description is given of a high level language designed to control the process of conducting an experiment using the computer "Elektronika-100I." Program examples are given to control the measuring and actuating devices. The procedure of including these programs in the suggested high level language is described.

HIGH LEVEL LANGUAGE FOR MEASUREMENT COMPLEX CONTROL BASED ON THE COMPUTER E-100I

By B. V. Zubkov

Introduction

/3*

Currently there is a fairly large fleet of small computers of the type "Elektronik-100," and Saratov. A considerable part of this fleet is used in scientific research to control the process of conducting an experiment. It is well known that the efficient use of a computer is determined to a very great degree by the program support. Despite this, clearly insufficient attention is now being focused on the program support of small computers.

It would be much easier for the experimenter who desires to automate his experiments if he could write the controlling programs in a natural language, for example, in Russian. Unfortunately as yet there is no computer that could interpret it sufficiently effectively. On the other hand, communication with a machine in a language that is understandable to it requires great efforts on the part of man.

The high level programming languages (HLPL) were developed as a compromise. These languages (FORTRAN, BASIC, FOCAL, and so forth) permit the programmer to concentrate attention precisely on programming, i.e., on the compilation of an

---

*Numbers in margin indicate pagination in original foreign text.

algorithm, and not on a detailed study of the system of commands employed by the computer. However, the mentioned languages were mainly developed to support mathematical calculation, and are not completely convenient for the programming of the /4 control of an experiment in small systems based on domestic small computers of the aforementioned type.

In this work we will be concerned with the HLPL EXPRO-77 (EXPERIMENT PROGRAMMING) developed as a result of a modification of the interpreter "FOCAL-69", and with its possible uses on the small computer "Elektronika-100I" [1]. During the writing of this work it was assumed that the reader is familiar with programming in the machine language of the small computer of the indicated type. These rules are stated fairly completely in [2] and [3].

The goal of the work is to show in specific examples the inclusion in the suggested HLPL of programs to control measuring and actuating devices. Attention should be drawn to the fact, that the use of program interfaces in many cases will permit changes to be avoided in the previously employed and well checked-out programs. This will considerably facilitate the translation into use of the suggested HLPL of the already active measuring complexes. For the experimenter who is automating measurements for the first time, its use will be especially valuable, since outlays of time are considerably reduced for writing and checking out the programs, i.e., it will save him many months for more interesting work.

1. Certain Requirements for the Language Designed to Program the Process of Conducting an Experiment

What interferes with the use of available HLPL for programming an experiment?

As already indicated, they were developed for mathematical calculations. This means that they include as standard many procedures that are often encountered in calculations. For operations of information input-output a comparatively limited set of devices is used both in type and in number. These devices were created in /5 the space of a long time. They have been well checked out. In the majority of cases their control has been placed on the equipment of the devices themselves, and does not require complex control programs.

In an experiment one has to face the fact that diversity and the number of employed devices for information input and output is considerably wider. Often the experimenter does not have the necessary equipment, and the lacking devices are created under laboratory conditions based on universal interfaces. It is clear, that under such conditions control of these devices is mainly placed on the programming resources. As a result, the HLPL must include a large number of diverse and fairly cumbersome programs to control these devices. Besides these differences in the requirements for the HLPL, we will cover certain shortcomings that follow from the features of the HLPL of varying classes.

At present a generally accepted form of writing instructions and commands to control the measuring and actuating devices has not yet been formed, not to mention their inclusion in the HLPL. Further, the use of compilers (FORTRAN, PL/1 and others) requires an excessively large volume of the memory in order store simultaneously the compiler itself, the original program and the object (compiled) program. Otherwise, the process of checking out the program and even making the slightest changes in it is drastically complicated.

The problem-oriented languages (BASIC, FOCAL and others), for their part operate very slowly, which restricts the rate of information collection to several tens

of results per second. The restriction in capacity is also significant; thus, no more than 100 numbers can be contained in the memory of the small computer E-100I (with the use of the HLPL FOCAL-69-8K), including the results of measurements, constants and intermediate results. Publication [4] discusses the problem of setting up a special HLPL to control the equipment. Here we will only state that the development of such a language in a nonspecialized organization is hardly justified today due to the great complexity of this task; in particular, if one considers that the extant languages themselves are far from exhausted. /6

Currently several versions of the languages BASIC and FOCAL are known [5], [6], that make it possible to organize experiment control. However, nothing is said of the elimination of the shortcomings named above. This, naturally, suggests to mind, that they have not been overcome. Here one should again note that we are concerned with the small computer with word length of 12 digits that is not equipped with a disc memory.

The experimental operation of a small control-computer complex (CCC) that was set up in our laboratory [7], made it possible to formulate a number of requirements for the HLPL used for the small computer E-100I. We will name the chief of them.

1. The HLPL must be (in the case of the absence of a disc memory) the interpreter.

2. The HLPL must permit inclusion in it of programs to fulfill a large number of new functions (NF) and commands, written by the user himself (or from the library of standard programs) on an assembler or procedure-oriented language. The procedure of including the NF and commands must be standard.

3. The interpreter of the HLPL must permit the possible use of its large program modules in programs of NF.

4. The quantity of numbers simultaneously stored in the small computer memory must be the greatest possible. In the limit case it is desirable to place these numbers in the entire free memory of a small computer.

5. It is necessary to be able to enter files of measurement results in the memory accessible for HLPL, by-passing the HLPL interpreter from the program to fulfill NF.

2. Main Features of the Suggested Language /7

To realize the indicated requirements the only possible path was selected, modification of the available language FOCAL. Publication [5] cites the comparative characteristics of the languages BASIC and FOCAL. Among the advantages of FOCAL the following are indicated: the possibility of more convenient modification of the program and the possibility of using abbreviations in writing the program. One should add here the more convenient form of entering numbers in the floating point block (FPB).

Thus, in the creation of EXPRO-77 the interpreter FOCAL-69 in the version 8K with ten-digit accuracy was selected as the base. In EXPRO-77 the majority of the aforementioned requirements were successfully satisfied. We will name features.

1. The created interpreter mainly made it possible to preserve the grammar of the FOCAL-69 language and the possibility of simple modification of the program.

2. It permits the inclusion of 77 NF, whose object programs are written on the assembler or procedure-oriented language.

3. A standard procedure has been developed for including programs for fulfillment of NF, that can be available in any free fields of the memory.

4. The NF programs, if necessary, can use the FPB of the interpreter EXPRO-77 directly from its memory field.

5. The interpreter makes it possible to simultaneously store in the memory 3,150 numbers in the "floating point" format.

6. Of the programs of NF fulfillment simple entry is permitted of files of experimental information in the memory, that is common to the HLPL, by bypassing the HLPL interpreter. This permits collection (output) of information in the necessary components of the controlling program with a velocity that is restricted only by the small computer characteristics, but not by the HLPL interpreter.

7. The pattern of "interruption" is not used.

In the process of modifying the base language interpreter it is desirable not to make unsubstantiated changes or restrictions, i.e., to preserve if possible, the semantics and syntax of the language, that were verified and sharpened over a long /8 time. The changes made in FOCAL only concern the forms of entering NF, the assignments for the format of number output into print, and the designations of the variables.

The number of variables was expanded by means of strict distribution of the memory. Now they are entered into registers and random designation is not permitted. Here, the memory is used more economically, since there is no longer any need for storing the variable identifier. Such regulations have been preserved for the use of variables in FOCAL, that roughly cover 90 variables. One should not confuse the term introduced here "register" with the term that designates the apparatus register.

The newly introduced variables (registers) are designated R(N), where N can have an integral part from 0 to 3,071. Designation by the variables R0, R1(N), BR(N) and so forth, will be interpreted as the designation accepted in FOCAL; the number of such variables does not exceed 90. The introduced registers are designated

by one letter R, after which the index immediately follows in parentheses.

It was decided to change the arrangement of the printing format "floating point," since arbitrary reduction in the number of printable signs represents certain conveniences. Thus, in the original, the E-format was assigned by the operater T%, in our version it is assigned by the operater T%N, %; here N indicates the number of printable digits after the period. For example, in the case of fulfilling the operater T%2, %, 1 the following will be printed: = . 10E+01.

The potential user, of course is interested in how to include in the suggested HLPL the program for fulfillment of his NF; we will examine below, in examples, the realization of inclusion in the HLPL of different functions.

To enter the NF the following form is used: FR(A, B, ...H); here FR is the name of all the NF; the contents in the parentheses are the function descriptors. The number of these descriptors must not exceed eight. The descriptors are separated by commas. The first descriptor is interpreted as the number of the function (conventional address of the device that fulfills the function). The remaining descriptors are interpreted as the number of the register R(B), that contains the parameter or as the parameter directly. There are no restrictions here. The selection of the method for using the function descriptors is presented to the programmer who compiles the program for fulfillment of the NF.

We recall, that the FOCAL grammar, as a result of fulfilling the function, includes not only the fulfillment of operations (calculated or controlling), but also the numerical results. This number needs: a) to be substituted into the mathematical or logical formula, b) to be assigned to a variable (sent to the register), c) to be open.

This number has a real meaning in the fulfillment of functions of collecting information of calculation. If the controlling functions for output of information are fulfilled, whose primary result is the triggering of the external device, this numerical result reflects the response of the actuating device. If a response is not provided for by the instrument designed, then the numerical result can be an arbitrary number. This arbitrary number can simply be sent to one of the registers, and there it is "rubbed out" by another number. In the examples below the utility of this number will be shown.

3. Brief Description of Certain New Functions and Example of Their Use

We will now describe the main characteristics of certain NF included in the HLPL. FR (1, B) is the function for controlling key 1 (device with conventional address 1). B designates the number of the register R(B) that contains the argument. The positive side of the argument causes the key to be closed, the negative--to be opened, while the equality of the argument to zero--printing of an error. /10 The numerical value of the function is equated to the contents of the register R(B) after fulfillment of the controlling operations of the function. FR (2, B) is the control function of key 2. The rest is unchanged. FR (3, B) is the function to delay fulfillment of the program. The clock has the conventional address 3. R(B) contains the time in seconds. The numerical value of the function is equated to the contents of the register R(B).

Now, by using the described functions, we will write a program for switching the valve (the valve controls key 1) 20 times in 1.5 s.

```
OI.IO  SET R(1)=1.5;SET R(2)=1;SET R(3)=0
OI.20  FOR N=1,20;SET R(2)=-FR(1,2);SET R(3)=R(3)+FR(3,1)
OI.30  T %3.01,R(3),!;Q
```

When line 01.30 is fulfilled, the full time of waiting between valve switching will be printed. If in line 01.10 the assignments are fulfilled in addition: S B=1; S C=2; then line 01.20 can be rewritten in another form, which does not result in changes in the program operation:

```
01.20  F N=1,20;S R(C)=-FR(1,C);S R(3)=R(3)+FR(3,B)
```

In the cited program the numerical value of the functions is efficiently used. The numerical value FR (3, B) is used to compute the full delay time of fulfillment of the program. The numerical value of the function FR (1, 2) is used as follows. With the first passage of the line 01.20 the argument (contents of register R (2)) is positive at the moment the function FR (1, 2) is fulfilled. This causes engagement of the valve. As a result of adoption the sign of the number in the register R(2) becomes negative. With the second pass the negative argument causes disengagement of the valve, and so forth. /11

4. Language Versions of EXPRO-77

We will dwell on the distribution of the operational memory of the small computer E-100I between individual components of the interpreter EXPRO-77. There are three versions: 12K, 16K and 20K. Here the numbers indicate the minimum accessible memory volume that is required only for placement of the interpreter and the variables. The interpreter of the NF and the programs for fulfillment of the NF are not arranged in the indicated fields, but require an additional memory. The memory field from 0 to 7 is distributed for different versions as follows.

TABLE.

Components of Interpreter EXPRO-77	Time of EXPRO-77		
	12K	16K	20K
Main interpreter of HLPL	0	0	0
Discharge of interpreter HLPL, 90 variables buffer registers	0	0	0
Programs in HLPL	1	1	4
Registers (0)-(1023)	2	2	2
Registers (1024)-(2047)	-	3	3
Registers (2048)-(3071)	-	-	1
NF interpreter	3	5	5
NF programs	3-7	4-7	5-7

5. Procedure for Inclusion of the NF Programs

To write programs for fulfillment of the NF it is necessary to observe certain rules. Before beginning the NF fulfillment, i.e., before transmitting control to the program fulfilling it, the descriptors of this function entered in the parentheses are computed and entered (1 per register) in the following order: /12

0017--number of NF descriptors,

0020--first descriptor,

0021--second descriptor,

.....

0027--eighth descriptor.

The descriptor that is entered into the corresponding register is the whole part of the descriptor indicated in parentheses in the NF. The descriptor must be positive and not exceed the value 4095. One should recall, that in parentheses the descriptor is entered in a decimal representation, and in the register, naturally, in eight representations.

The beginning address of the program to fulfill the NF is placed in one of the registers of the address table for the NF program. The number of the register

must correspond to the number of the NF. The address table begins with register 0210 and ends with register 0324, i.e., the maximum number of NF equals 77. For example, the address for the beginning of the program to fulfill the function FR (1, B) is placed in register 0211, the address for the beginning of the program to fulfill the function FR (8, ...)--in register 0220 and so forth. The output from the program for fulfillment of the NF is implemented by the unconditional transfer of control indirectly to the 0176 register. All the indicated register numbers correspond to that field of the memory in which the NF interpreter is located.

The program for NF fulfillment, as a rule, is located on a separate punched tape. This punched tape must contain:

1. directly the program for NF fulfillment;
2. the constants necessary for operation;
3. the program address in the table of NF addresses.

6. Use of the Floating Point Block in Programs of New Functions

The floating point block (FPB) of the base interpreter has been considerably modified. Its main purpose consisted of creating that FPB that would permit direct turning to it from the NF programs located in any memory field of the small computer. The need to create such an FPB is exceptionally large size (the FPB occupies roughly 4,000 registers in the octuple presentation). It would not be wise to introduce a second such block into the free memory field for its use in the NF programs. /13

The system of commands for the FPB has been considerably expanded, which will permit a reduction in the volume of NF programs, and will facilitate the use of the FPB in them. The FPB has been altered such that it makes it possible to work

only with the contents of the registers R(N), as well as several buffer registers. The buffer registers are designed to store intermediate computation results; it is impossible to turn to it from the HLPL. The FPB commands can use only indirect addressing. The absolute addresses of the registers R(N) are formed by the service subprogram (according to the number of register), and are generally entered on the zero and current pages of the memory. The absolute addresses of the function registers and certain important constants are entered on the zero page of the current memory field. One should stress once again, that the FPB operates only with operands located in the memory fields of the small computer "Elektronika-100I" from zero to three.

If the constants necessary for calculation are located on the current memory field from 4 to 7, then they should be preliminarily rewritten into buffer registers, by using the service subprogram.

We will become acquainted with the purpose of the FPB commands in an example. Assume that it is necessary to process the measured amount according to the formula

$$(((A1-A2+A3)*A4)/A5)\wedge A6,$$

In this formula A1, A2 and so forth are the operand addresses; (A1), (A2) and so forth are the contents of the addresses, i.e., the operands themselves; (A5) is the measured amount entered into A5 previously without normalization. /14

The computation program will look like

6202	between 6202 and 4407
4407	input into FPB
0435	(A5) → FAR

7000	normalization (FAR)
6435	(FAR)→A
043I	(A1) →FAR
2432	(FAR)-(A2) →FAR
I433	(FAR) + (A3)→FAR
4434	(FAR) x (A4)→FAR
3435	(FAR)/(A5) →FAR
5436	(FAR) ↑ (A6)→ FAR
0000	output from FPB

Here, (A5)→FAR designates the operation of entering the address components A5 into the floating accumulation register (FAR). The previous contents of FAR are erased. The contents for the address A5 are preserved.

(FAR)→A5 is an inverse operation. The previous (A5) is erased, and the (FAR) is preserved.

The input and output from the FPB do not alter the FAR contents; if there is no turning to the FPB, then the (FAR) is preserved unchanged with the use of the service subprograms. In the given example it was assumed, that the registers 0031-0036 contain the address of the corresponding operand A1-A6. The operand address A1, A2 and so forth is the address of the first of four successively arranged registers. The order of the number in the three subsequent mantissas of this number is entered into the first register.

Besides the FPB commands given in the example, there is a set of commands of conditional and unconditional transmission of control in the FPB limits.

7410 unconditional transmission of control,

7440 omission of next register (FAR)=0,

7450 omission of next register if (FAR)$\neq$0,

7500 omission of next register if (FAR)$<$0,

7510 omission of next register if (FAR)$\geq$0,

7540 omission of next register if (FAR)$\leq$0,

7550 omission of next register if (FAR)$>$0.

With the fulfillment of these commands the (FAR) is not altered. The purpose of these commands is to obtain a branching in the course of computations in accordance with the obtained results. We will examine their use in an example

0400/7440

0401/7750

0402/7410

0403/0040

Assume that the moment the command is fulfilled according to the address 0400 (FAR)=0, then register 0401 will be omitted, and unconditional transfer of control will be implemented (since (0402)=7410) for the address 0403+0040=0443. If (FAR)$\neq$0, then the control will be transferred according to the address 0401+7750=0351. It is apparent from the given example, that in the register following after the command for transfer of control, a shift is indicated in the address (where the control will be transferred to) in relation to the address of this register. Such a method of addressing makes it possible to have NF programs that are not "linked" to specific memory registers, and it is easy to transfer them to any free place.

Two commands have been introduced that make it possible to turn from the FPB to the subprogram written in the system of FPB commands.

7560 turning to the subprogram,

7570 return from the subprogram.

The address of the subprogram that we have turned to, is indicated in the registers following the command to turn. One should focus attention on the fact that here the absolute address is indicated, and not the shift, as in the commands for transfer of control. The subprogram must end with a command to return, i.e., /16 command 7570. Within the subprograms written in the system of FPB commands, repeated turnings are not permitted to any subprogram that is written with the use of the FPB command; however outlet from the FPB and subsequent input into it is permitted. The use of the FPB subprogram makes it possible to compute the basic functions whose argument is the (FAR).

The operations on the FAR contents also fulfill the following FPB commands.

7420 inversion of (FAR), i.e., (FAR)→FAR

7430 deletion of fractional part of (FAR)

codes of FPB commands

7460

7470

7520

7530

reserved for purposes of future use.

In conclusion of this paragraph we would like to stress, that one should not confuse FPB commands with the commands of the central processor. Despite a certain purely external similarity (especially for the commands for control transfer), these commands operate with a floating accumulation register, and not with the apparatus accumulation register.

7. Service Subprograms for Use in NF Programs

In addition to the NF interpreter service subprograms (s/p) were written. We will give the main ones.

1. 4564--rewrites the file of numbers that occupies the successive registers in the current field, into successive buffer registers. In registers that follow after the command to turn to the s/p, it is indicated respectively: absolute address for the beginning file on the current field, absolute address of first buffer register, /17 number of registers (length of file). Each number of the file occupies four successive registers. During input into the s/p (AR) is any one, and during the outlet from s/p (AR)=0; (FAR) is maintained unchanged.

2. 4565--rewrites the file made of successive buffer registers into registers of the current field. After command to turn the following are indicated: absolute address of first buffer register of file, absolute address of register on current field (register for beginning of file), number of registers (length of file).

3. 4566--causes printing of information on an error; after this control is transferred to the EXPRO-77 interpreter for input of introductions by the programmer. The number of the report on the error is determined by the address of the register that contains command 4566. During input into the s/p (AR) is any one.

4. 4567--forms according to the number of the register R(N) the absolute address of the register. During input into the s/p (AR) is equal to the number of the register, during output from the s/p AR contains the absolute address. (FAR) is maintained unchanged.

5. 4571--verifies the number of NF. If their number equals 1, control is transferred to the next register. Otherwise, a report is printed on an error "number of descriptors does not correspond to the required", and control is transferred to the programmer. During input into the s/p (AR) is any one, during output from the s/p

(AR) equals 0, while FAR is maintained unchanged.

6. 4572 is the same as 4571. Continuation of the NF program is possible with the indicated number of descriptors of the NF equal to 2.

7. 4573 is the same as 4571. Continuation of fulfillment of the NF program is possible when the indicated number of NF descriptors equals 3.

8. 4574 forms in the FAR a whole number (fixed point) from the previous (FAR); after this the result is rewritten in register 0044-0047 of the current memory field.

0044-0027, /18

0045--sign of the number and significant digits of the number,

0046--least significant digits of the number,

0047-0000.

If the whole number cannot be entered into the FAR (the number is large), then a report is printed on an error, and control is transferred to the programmer. During input into the s/p (AR) is any one, during output (AR)=0, and (FAR) equals the previous value.

9. 5576--this command implements output from the NF program after its fulfillment. At first, a number must be entered into the FAR that will also be the numerical result for the fulfillment of the new function.

8. Example of NF Program that Realizes Key Control

The algorithm for the program is given in Figure 1, and the listing in Figure 2. In selecting the program it is necessary to remember, that in register 0021 the second descriptor of the function is written, which for the given function is the number of the register that contains the function argument.

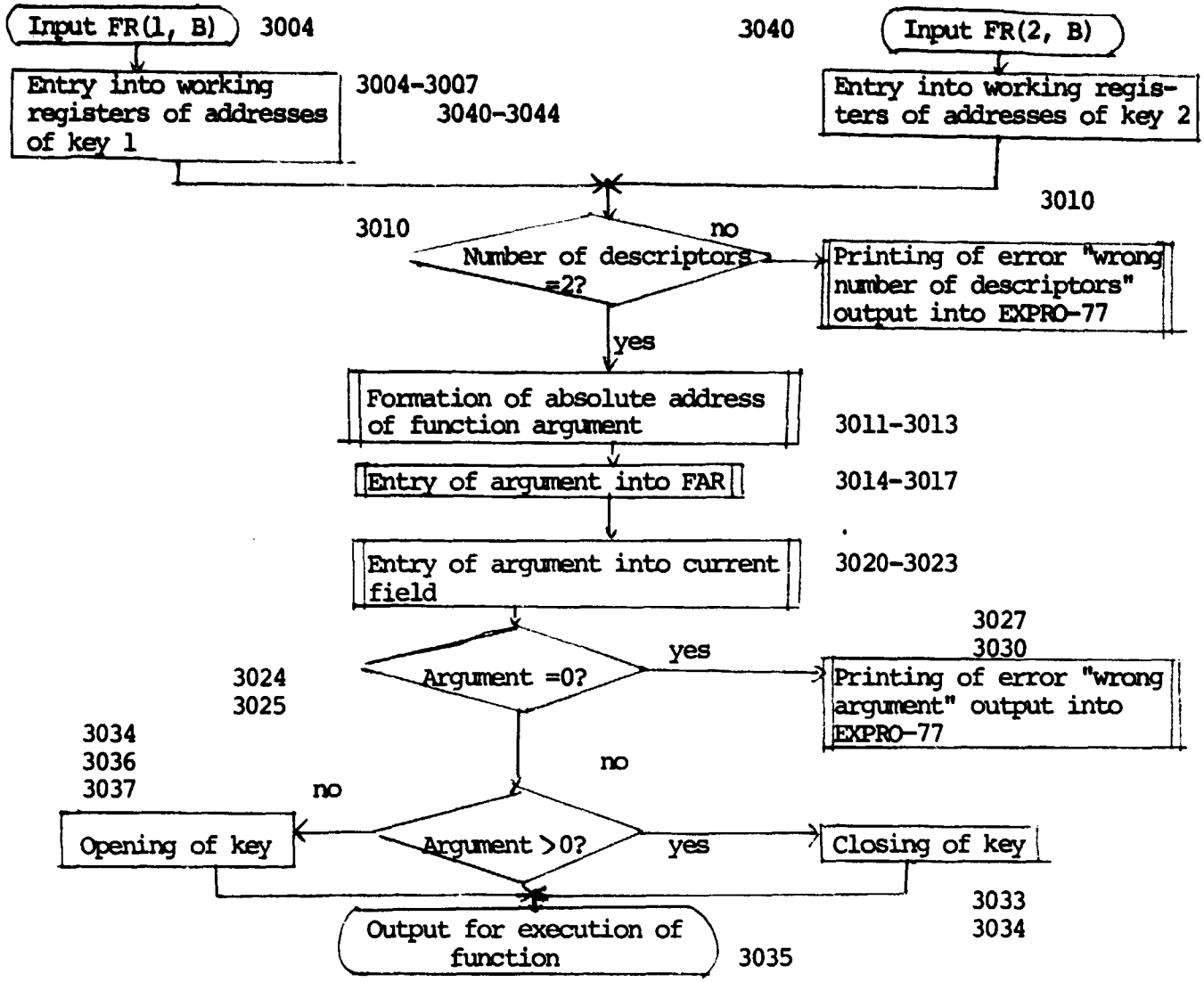


Figure 1. Algorithm for Program to Fulfill Functions FR (1, B) and FR (2, B).

Since the programs of the functions FR (1, B) and FR (2, B) are distinguished only by addresses of the controllable keys, then initially (for each function) the addresses of the corresponding key are written into the working registers, after which the general part of the program is fulfilled. The beginning address of the program FR (1, B) (address 3004) is entered into the corresponding register of the NF address table, i.e., into register 0211. The beginning address of the program FR (2, B) (address 3040) is entered into register 0212. A description of the employed subprogram has been given in section 7. The interface for connection between the key and the small computer is constructed such that in order to open or close the key it is necessary to enter into the AR the address (code) of the operation, after which the command to turn to the external device 6071 is fulfilled. In register 3016, by command 0430 the value of the function argument is written into FAR, which is not changed until the end of the program. With this value (FAR) the /19 output from the program of the NF occurs, i.e., it is also the numerical value of the function.

9. Example of Program that Realizes Lag in the Fulfillment of the Input Program

The algorithm for the program is given in Figure 3, the listing in Figure 4. For the given function it is assumed that the second descriptor is the address of the function argument. For the given program it is necessary to translate the number of argument seconds into the number of cycles of turning to the computer memory. Here certain details given in section 8 have been omitted. The difference from the previous example is determination of the equality of the function argument to zero. In the previous example the argument at the moment of verification was presented in the format of a floating point, therefore it was sufficient to verify the contents of register 0045. In the given example, at the moment of verification the argument

```
3000 /0013    address of key 1 closing
3001 /0000    address of key 1 closing
3002 /0014    address of key 2 closing
3003 /0015    address of key 2 opening
3004 /1200    beginning address of program FR (1, B)
3005 /3022
3006 /1201
3007 /3023
3010 /4572    number of descriptors of function =2?
3011 /1021    number of register→AR
3012 /4567    absolute address of register→AR
3013 /3030
3014 /6202    input into
3015 /4407    FPB
3016 /0430    argument→FAR
3017 /0000    output from FPB
3020 /4565    re-entry of registers from field 0 to current field
3021 /0044    address of register on field 0
3022 /0044    address of register on current field
3023 /0001    number of registers
3024 /1045    sign of number and significant digits→AR
3025 /7440    number equal to 0?
3026 /5231    no
3027 /6203    yes
3030 /4566    printing of report on error
3031 /7700    number =0?
3032 /5236    no
3033 /1022    yes. address of opening of key→AR
3034 /6071    execution
3035 /5576    output for execution of program
3036 /1023    address of key closing→ AR
3037 /5234    transfer to execution
3040 /1202    beginning address of program FR (2,B)
3041 /3022
3042 /1203
3043 /3023
3044 /5210    transfer to fulfillment of general part of program
```

Figure 2. Listing of Program that Fulfills the Function FR(1,B) and FR (2,B)

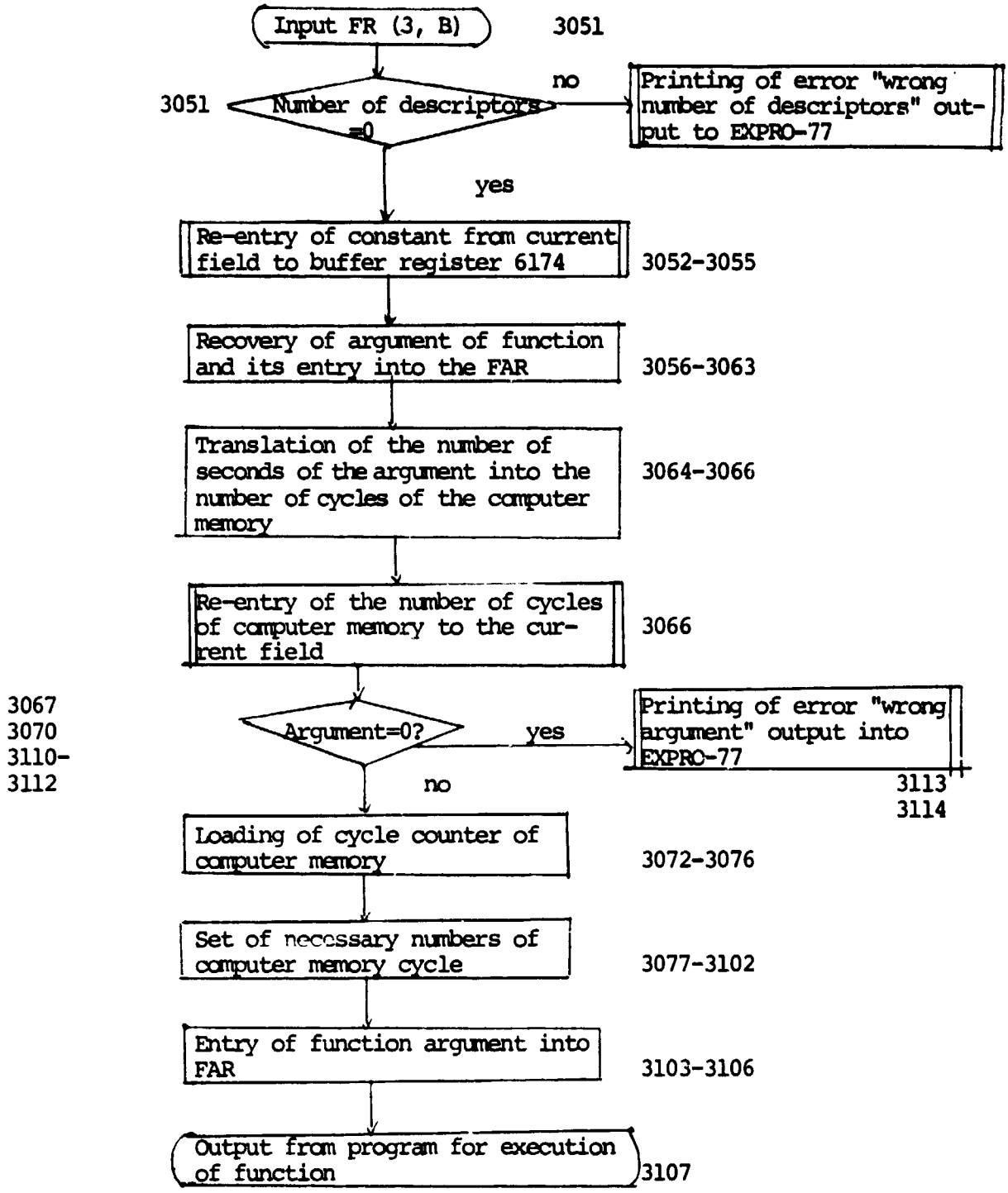


Figure 3. Algorithm of Program that Realizes a Delay in the Fulfillment of the Program

```
3051 /4572      number of descriptors of function equals 2?
3052 /4564      yes; rewrite constants from current field into buffer
3053 /3045
                address of buffer register
3054 /6174      number of registers
3055 /0001
3056 /1021
3057 /4567
3060 /3030      argument of function→FAR
3061 /6202      (FAR) x constant→FAR
3062 /4407
                translation into format of fixed point
3063 /0430
3064 /4654      argument≤0?
                yes
3065 /0000      loading of time delay counter
3066 /4574
                counting of assigned number of cycles
3067 /1045      computer memory
3070 /7550
3071 /5310
3072 /7040
3073 /3045
3074 /1046
3075 /7040      entry of function argument into FAR
                output after fulfillment of program
3076 /3046      printing of error "argument=0"
3077 /2046
3100 /5277
3101 /2045
3102 /5277
3103 /6202
3104 /4407
3105 /0430
3106 /0000
3107 /5576
3110 /1046
3111 /7640
3112 /5272
3113 /6203
3114 /4566
```

Figure 4. Listing of Program that Realizes Lag in Fulfillment of Program

is presented in the format of a fixed point, therefore it is necessary to verify the content of both registers 0045 and 0046.

In order for the numerical value of the function to be equal to the function argument, before output from the program the value of the initial argument (number of seconds) is sent to the FAR. For this the number of cycles in the computer memory was in the FAR.

The address for the beginning of the program (3051) is entered into register 0213 of the NF address table.

10. Example of NF Program Realizing Engine Control

In order to move the sensor in our experimental unit we use a drive by direct current engine [7]. The program of engine control is comparatively complicated (it occupies roughly one page of memory); it was written in the very beginning of the work on automation, when a suitable HLPL was not available. However, the use of a simple program interface makes it possible to use this program without the slightest changes in it. /20

The algorithm for the program interface is given in Figure 5, the listing in Figure 6.

The subprogram of engine control uses (AR) as the code of the deflection angle, (ER)--as the code of rotation direction. If (ER)=0, rotation is clockwise; if (ER)=1, the rotation is counterclockwise.

Engine control from the HLPL is realized by the operater FR (4, B). Here 4 is the conditional address of the device, B is the address of the function argument.

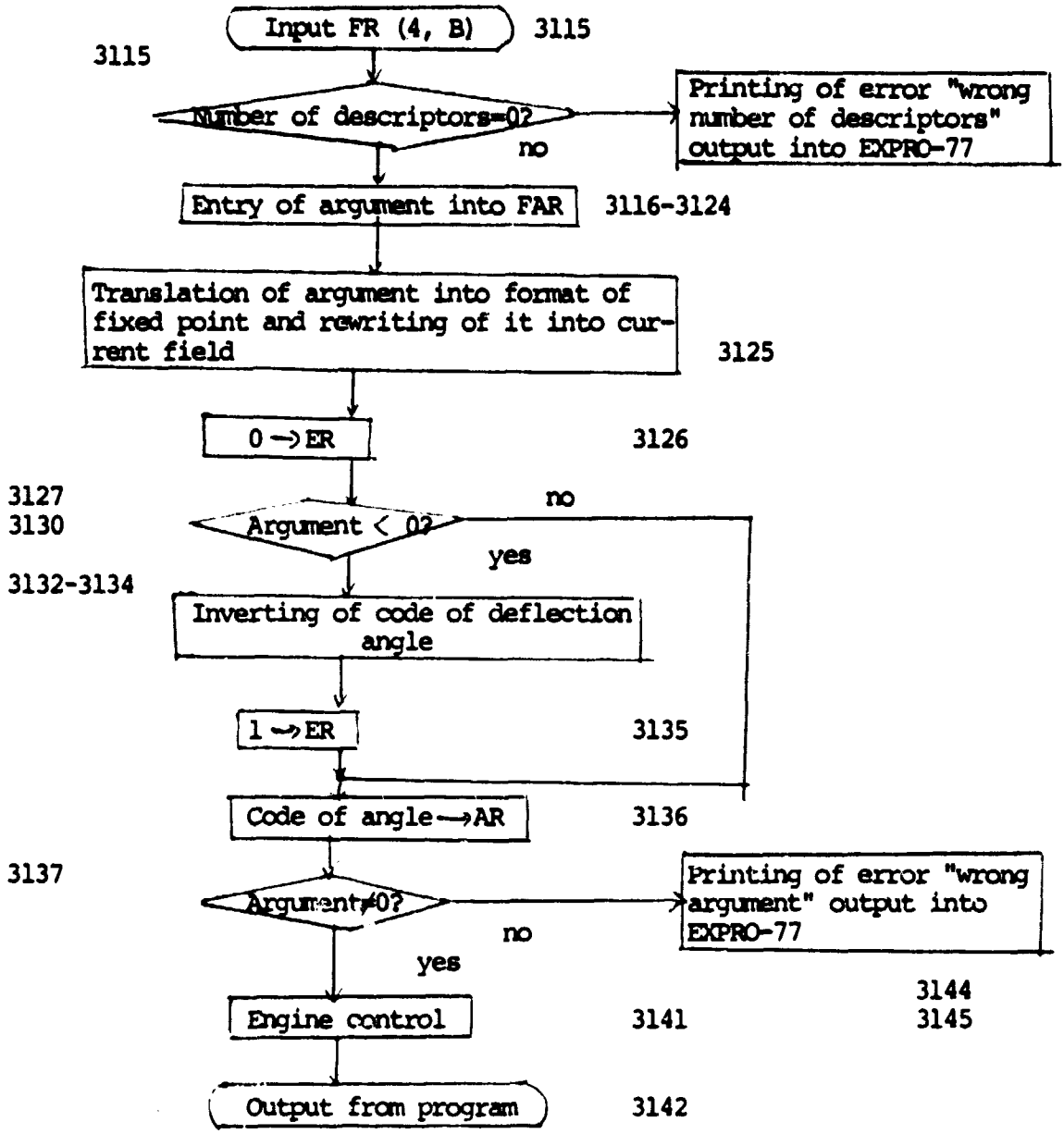


Figure 5. Algorithm of Program Interface for Realization of Function to Control Engine

```
3115 /4572      number of descriptors=2?
3116 /1021
3117 /4567      formation of argument address
3120 /3030
3121 /6202
3122 /4407
3123 /0430      access of argument
3124 /0000
3125 /4574      translation (FAR) into format of fixed point
3126 /7100      0→ER
3127 /1045
3130 /7700      argument=0?
3131 /5336      no
3132 /1046      yes
3133 /7041      inverting of argument
3134 /3046
3135 /7120      1→ER
3136 /1046      argument  AR
3137 /7450      argument≠0?
3140 /5344      no
3141 /4743      yes; turning to subprogram of engine control
3142 /5576      yes; output
3143 /2014      address of subprogram
3144 /6203      printing of error "wrong argument"
3145 /4566
```

Figure 6. Listing of Program Interface for Engine Control Function

The sign of the argument determines the rotation direction of the engine; "+"--clockwise, "-"--counterclockwise. The whole part of the argument equals the number of deflection pitches (one pitch corresponds to 10°). During output from the program of NF, after its fulfillment, (FAR)=whole part of the argument, it will also be the numerical value of the function. The beginning address of the program (3115) is entered into register 0214 of the NF address table.

11. Examples of Program that Realizes Measurement of Counter Intensity

To measure the number of impulses received during an assigned time, a program is used that is turned to from the HLPL by the operator FR (6, B). Here 6 is the conventional address of the device, and B is the argument address.

The number of seconds of measurement time is entered into register R(B). The subprogram for counter control that was previously written, with the help of the program interface was linked to the HLPL. This subprogram uses as an input parameter the code for the number of impulses of the time counter (24 binary digits), that is located in the registers 0045 and 0046 in the format of "fixed point". The measurement results (24 binary digits) in the same format is written in the same /21 registers.

The algorithm of the program interface is given in Figure 7, the listing in Figure 8.

The function of the program interface consists of preparation of the argument, its translation into the required format, and entry according to the required address. After turning to the subprogram for control of the counter the program interface rewrites the measurement result (number of impulses) into FAR. This result will also be the numerical value of the function. The address for the beginning of the

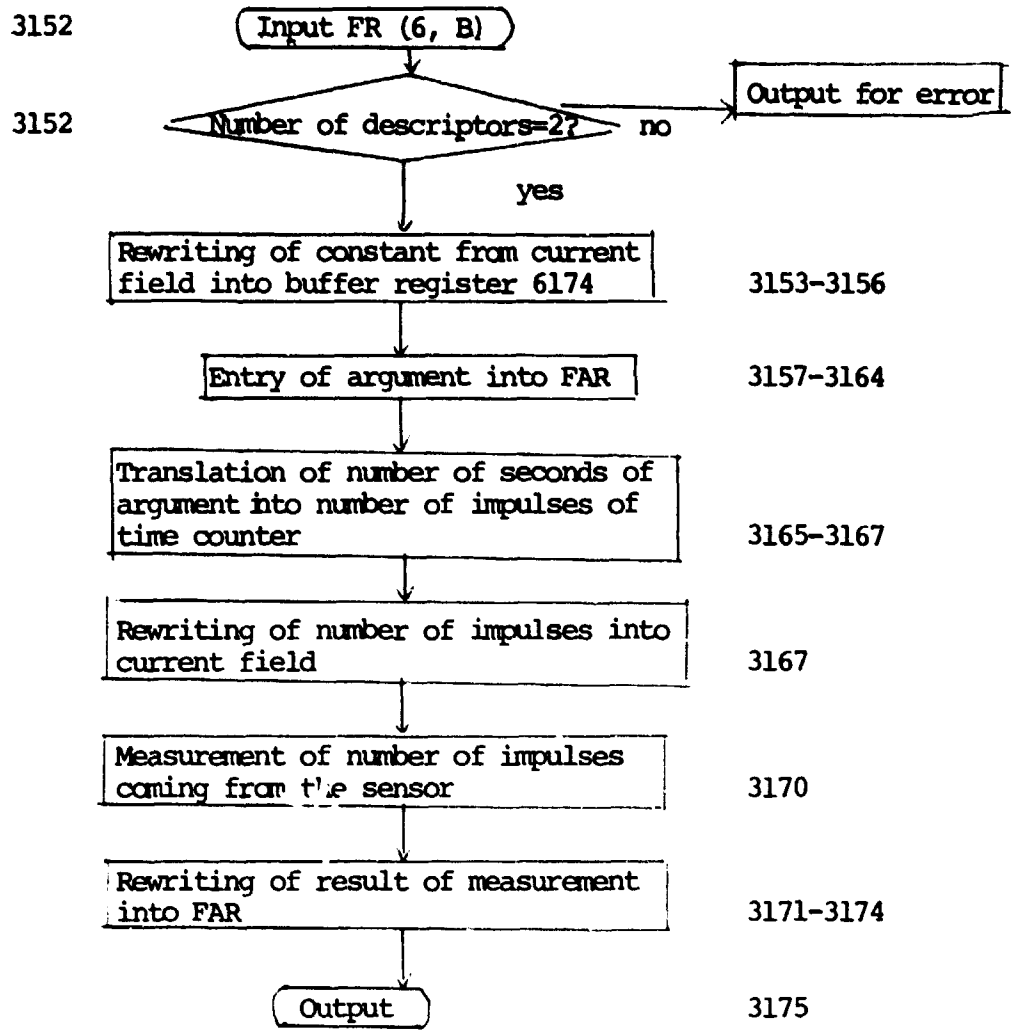


Figure 7. Algorithm of Program Interface for Function "Measurement of Intensity"

```
3146 /0017
3147 /2426
3150 /1505
3151 /0754
3152 /4572        number of descriptors=2?
3153 /4564        rewriting of translation constant
3154 /3146
3155 /6174
3156 /0001
3157 /1021
3160 /4567        formation of argument address
3161 /3030
3162 /6202
3163 /4407
3164 /0430        access of assigned argument
3165 /4654        translation of argument into number of impulses
3166 /0000
3167 /4574        obtaining of whole part of translated argument
3170 /4776        turning to subprogram of control of counter
3171 /4564        rewriting of measurement result into FAR
3172 /0044
3173 /0044
3174 /0001
3175 /5576        output
3176 /2200        address of subprogram for counter control
```

Figure 8. Listing of Program Interface for Function "Measurement of Intensity"

program (3152) is entered into register 0216 of the NF address table.

12. Additional List of NF Included in HLPL

Besides the NF described in sections 8-11, there are three more.

1. FR(5)--numerical result of function equals the code established on the key register of the small computer E-100I at the moment of function fulfillment.

2. FR(7, X, Y, N, T)--by means of this function the graphic information is displayed. Here 7--conventional address of display; X--address of first number of file X; Y--address of first number of file Y; N--number of points of file (size of file); T--time of lighting up of graph (in seconds). The numerical value for the function equals the maximum number of the file Y.

3. FR(9, N, Y)--the function is used to measure the distribution of intensity density in the cross section of the particle flux. Here N is the number of measurement pitches (one pitch corresponds to 10° of the deflection angle of the engine). The result of fulfilling the function is filling of the N registers, beginning with R(Y), number of impulses is accepted as 20 ms, at the corresponding pitch. The numerical value of the function equals the number of pitches, i.e., N. /22

Conclusion

In conclusion we note that despite the convenience of using the proposed language, it requires further perfection. Of great importance is the inclusion into the language of operaters of a turning to the storage devices on magnetic tape, which will significantly increase the power of the controlling complex. One also feels the need for the operater to work with several devices of input/output of alphabet-digital and graphic information, which also requires the introduction of additional operaters to the HLPL. The need for such devices follows from the fact that the

interaction of the experimenter with the unit, that does not require documenting, is implemented considerably more conveniently without display. Documenting of the initial condition of the results of the experiment, for its part, requires the use of teletype and a graph plotter. Currently we document graphic information by a technique suggested in publication [8], or by photography from the graphic display screen.

References /31

1. Akushskiy, I. Ya.; and Troyanovskiy, V. M. Programmirovaniye na "Elektronike-100" dlya zadach ASU TP ["Programming on 'Elektronik-100' for Automatic Control system Tasks TP"], Moscow, Mir, 1978.

2. Flores, A. Organizatsiya vychislitel'nykh mashin ["Organization of Computers"], Moscow, Mir, 1972.

3. Souchek, B. MINI-EVM v sistemakh obrabotki informatsii ["Mini-Computer in the Systems of Information Processing"], Moscow, Mir, 1976.

4. Giglavyy, A. V.; et al. "Certain Aspects of Developing Languages for Controlling Production Processes," Upravlyayushchiye sistemy i mashiny, No 5, 1977.

5. Khrapchenko, T. S. "Comparative Characteristics of Dialogue Languages BASIC and FOCAL," Moscow, Trudy IEUM, No 55, 1976.

6. Angelov, A. Kh.; and Dubovik, L. V. "Control of Moduli KAMAK from the Computer TRA-1001/1 with the Help of the FOCAL Programming Language," Pribory i tekhnika eksperimenta, No 5, 1977.

7. Zubkov, B. V.; and Kalinin, A. P. Izmeritel'no-upravlyayushchiy kompleks dlya avtomatizatsii issledovaniy atomnykh stolknoveniy ["Measuring-Controlling Complex for Automation of Study on Atomic Collisions"], preprint of the Institute of Space Research of the USSR Academy of Sciences, Pr- 366, 1977.

8. Zubkov, B. V.; and Khromov, V. N. Sopryazheniye MINI-EVM "Elektronika-100I" i NR-9821A ["Joining of Mini-Computer 'Elektronika-100I' and NR-9821A"] preprint of the Institute of Space Research of the USSR Academy of Sciences, Pr-312, 1976.

Table of Contents