

## Description

Method for the generation and storage of a user program,  
consisting of instructions, for a programmable con-  
troller, and operating method for a programmable  
5 controller

The present invention relates to a method for the  
generation and storage of a user program, consisting of  
instructions, for a programmable controller, the  
programmable controller, when executing the user program,  
10 determining output signals for an industrial process from  
input signals from the industrial process, as well as to  
an operating method, corresponding thereto, for a  
programmable controller.

Methods of this type are generally known.

15 The object of the present invention is to make  
new methods available.

The object is achieved for the generation method  
and storage method in that the instructions of the user  
program consist in each case of a function code and  
20 function parameters assigned to the function code, in  
that the function parameters refer either to an input  
signal or to another instruction and in that the method  
has the following steps:

- a) during the generation of the user program, for each  
25 output signal, beginning with the instruction to be  
assigned to the respective output signal, the  
instructions to be executed are called up, the  
following procedure being executed:
- b) the function code to be executed is called up and  
30 stored temporarily;
- c) the number of function parameters is established;
- d) the function parameters are called up one after the  
other;

- e) if the function parameter is an input signal, a reference to the input signal is stored temporarily;
  - f) if the function parameter is a new function code, a reference to a new instruction is stored temporarily, a branch is made to this new instruction and the other function code is stored temporarily and the steps c) to f) are repeated for the new instruction;
  - g) the temporarily stored function codes and the temporarily stored function parameters are stored in a nonvolatile memory and, respectively, identified as valid, only after all the instructions of at least the respective output signal have been called up completely.
- 15 In addition, according to the operating method, the instructions of the user program in each case consist of a function code and function parameters assigned to the function code, the function parameters referring either to an input signal or to another instruction. The
- 20 steps according to the invention are as follows:
- a) during the cyclic execution, for each instruction, beginning with the instructions assigned to the output signals, the following procedure is executed:
  - b) the function code assigned to the instruction and the function parameters assigned to this instruction are read;
  - c) for each function parameter a check is made as to whether this refers to another instruction;
  - d) if the function parameter refers to an input signal, the input signal is stored temporarily;
  - e) if the function parameter refers to another instruction, a branch is made to this other instruction and the steps b) to e) are repeated with the other instruction;
  - f) when all the function parameters have been exhausted, the output signal assigned to the instruction is determined.

Further advantages and details emerge from the following description of an exemplary embodiment. In this case:

FIG 1 shows the structure of a programming device and a programmable controller,

FIG 2 shows the user interface of the programming device and

FIG 3 shows a programming example

According to FIG 1, the programming device 1 has a processor 2 which is connected via the bus system 3 to the read-only memory 4 and the read-write memory 5 and to the interfaces 6, 7 and 8. The operating system of the processor 2 is stored in the read-only memory 4, the read-write memory 5 serving as working memory for the programming device 1. Data can be entered into the programming device 1 via the interface 6 by means of the input unit 9, for example a keyboard. Data can likewise be displayed on the display unit 10 by means of the interface 7. By means of the communications interface 8, the programming device is able to communicate with a programmable controller 12 via the communications connection 11 which is drawn with dashed lines.

The programmable controller 12 likewise has a processor 13, which executes a user program. The user program consists of a multiplicity of instructions to be executed and is stored in the electrically eraseable and writeable read-only memory 14. In accordance with the user program to be executed, the processor 13 makes access via the controller 15 and the process interface 16 to an industrial process or to an industrial plant 17, which is not shown in more detail. From there, it reads in input signals, stores them in the read-write memory 18 and then determines output signals by reference to the user program. The output signals are likewise stored in the read-write memory 18 and then output via the controller 15 and the process interface 16 to the

industrial process 17. In addition, the programmable controller 12 has an interface 19 to the programming device. The processor 13, the memories 14, 18, the controller 15 and the communications interface 19 are  
5 likewise connected to one another via a bus 20.

In principle, programming device 1 and programmable controller 12 can be devices which are separate from each other. However, as is shown in FIG 1 by the dashed line and also shown in FIG 2, the programming device 1 is  
10 preferably integrated into the programmable controller 12. However, it is possible in both cases for the programming device 1 to access the read-only memory 14 via the interfaces 8 and 19 and via the communications connection 11. In this arrangement it is possible for a  
15 new user program to be written into the read-only memory 14 and for an existing user program to be read, erased or changed.

According to FIG 2, the programmable controller 12 has screw connections 21 to 24, a keyboard panel 25  
20 and an LCD display 26. The programmable controller 12 can be connected to a 24 volt power supply via the screw connections 21. Via two screw connections 22 in each case, it is possible to output a digital output signal to the industrial process 17. One input signal from the  
25 industrial process 17 can be read in via each of the screw terminals 23. The actuators and sensors, which are connected to the screw terminals 22 and 23, can be supplied with power via the screw terminals 24. The screw terminals 22 to 24 are also an integral part of the  
30 process interface 16.

The LCD display 26 corresponds to the display unit 10 of the programming device 1. In addition, messages from the programmable controller 12 can be displayed via the display 26 during operation. The display 26  
35 therefore also serves as an observation unit. The keyboard panel 25 makes it possible to enter instructions

directly into the programming device 1 and into the programmable controller 12.

As is generally the case, the user program consists of individual instructions, each instruction  
5 respectively consisting of a function code and function parameters assigned to the function code. In operation, the programmable controller executes the following sequence of instructions cyclically:

- It reads in input signals from the industrial process 17.  
10
- It determines output signals for the industrial process by reference to the user program.
- It outputs the output signals to the industrial process 17.

15 This procedure is generally known.

During the generation of the user program, the instructions to be executed are called up one after another by the programming unit 1 from the input unit 9. For this purpose, beginning with the instruction to be  
20 assigned to the first output signal, a query is raised as to which function is intended to be executed. A corresponding query is therefore output on the display unit 10 and the system waits until the user has input the function code via the input unit 9.

25 When the function code has been input, it is stored temporarily by the programming unit 1 in the read-write memory 5. The programming unit then establishes how many function parameters are assigned to this instruction. Here, a differentiation is to be made between the  
30 internal assignment within the programming device 1 and the number of function parameters which are visible to the user. Internally, the number of function parameters is the same for all function codes, equal to four in the predetermined example. Externally, however, it is

always only the number of necessary function parameters which is called up. An example is intended to explain this in more detail:

5 If the code for the OR operation is entered as function code, in actual fact four parameters are called up. These four parameters are then also taken into account later during the execution of the program. However, if, for example, the logic negation is selected as function code, only one parameter, namely the input  
10 signal to be negated, is called up from the user. The other three parameters are filled in automatically by the programming device 1 and not taken into account during the execution of the program.

Following the calling-up and temporary storage of  
15 the function code in the read-write memory 5, the function parameters are also called up and temporarily stored one after another by means of the programming unit 1 working together with the input unit 9 and the display unit 10. Either an input signal or another function code  
20 can be input as function parameter. In this case, when each function parameter is input, it is immediately checked as to whether this is a function code.

If the function parameter is not a function code but an input signal, this is stored temporarily and the  
25 process is continued with the next function parameter. Input signals are in this case to be understood not only as input signals from the industrial process 17 but also signals whose value is known in advance. Examples of such signals are the constant one signal or the constant zero  
30 signal, as well as a clock signal.

If, on the other hand, a new function code is input as function parameter, the function code is not stored temporarily as function parameter but the address of a new instruction. Immediately thereafter, a branch is  
35 made to this new instruction and there the directly previously input function

code is stored temporarily. The number of function parameters is then reestablished and the function parameters are called up and stored temporarily.

5 In the case of these function parameters, too, a check is once more made immediately as to whether they are input signals or function codes. If they are new function codes, once more the address of this new instruction is stored immediately as function parameter and a branch is made to this new instruction. Here, too, the  
10 function code which has just been input is stored once more, it is established how many function parameters are needed and the function parameters are called up and stored temporarily.

This nested calling up of function codes and  
15 function parameters is continued until all the function parameters of the function code last called up and temporarily stored refer to input signals. When all the function parameters refer to input signals, the respective instruction has been completely input. A return is  
20 then made to the preceding instruction and in the case of the latter, if appropriate, the next function parameter is called up and temporarily stored, a branch also being made here, if necessary, to further instructions.

The procedure mentioned above is continued until  
25 all the functions necessary to determine an output signal have been nested inside one another, that is to say all the function parameters of each instruction have been input. Only when this has been carried out is the instruction assigned to the next output signal called up  
30 and the procedure repeated, to the extent necessary. In this way, all the output signals are programmed one after another.

Of course, the input of the user program can be aborted at any time by the user. In this case, however,  
35 no storage of the part-program which has been input and temporarily

stored is carried out. When the programming device 1 is switched off, the data which have been input are lost. Storage in the nonvolatile memory 14 of the user program which has been called up and temporarily stored in the read-write memory 5 is only made possible when the programming process has been fully completed. The programming process or the generation of the user program counts as fully completed

- when an instruction has been input for each output signal, and
- when each function parameter of each function code which has been programmed is occupied.

FIG 3 shows an example of a simple user program. In this case, the continuous lines signify the connections necessary on the basis of the desired linkages. The dashed lines show the additional connections required on the basis of the programming scheme, and the chain-dotted lines show the connections automatically added by the programming device 1. The input of the example program according to FIG 3 takes place as follows:

Firstly, the programming device asks, via display unit 10 and input unit 9, which function code is needed for the determination of the output signal of the first output A1. The user then inputs, via the input unit, the fact that an AND operation 27 is to be carried out. The corresponding function code is stored temporarily in the read-write memory 5 of the programming device 1. Then, one after another, the first, the second and the third function parameters, that is to say three of the four input signals of the AND operation 27, are called up and stored temporarily.

The first and the second input parameters are the input signals E1 and E2. Therefore, during the input of these two function parameters, the system continues immediately with the next function parameter. When the third function parameter is called up,

the user inputs the fact that he desires to undertake a NOT operation 28. As the third function parameter, therefore, the address of the instruction at which the NOT operation 28 will be handled, is temporarily stored.

5           A branch is then made by the programming device 1 to this instruction, and a query is raised as to which the (single) function parameter of the NOT operation 28 is intended to be. Here, the user inputs the input signal E3. The programming device 1 then stores this function  
10           parameter temporarily and supplements the instruction with three further function parameters, which in each case refer to the logic one. These three function parameters are also stored temporarily in the memory 5. This then completes the setting of the parameters for the NOT  
15           operation 28. A return is therefore made to the further setting of parameters for the AND operation 27, and the fourth function parameter is called up and temporarily stored. Since the AND operation 27 according to the  
20           example of FIG 3 only has three true input parameters, the fourth input is connected to the logic one by means of a corresponding input by the user.

          This completes the programming of the output A1, and the process continues with the programming of the output A2.

25           Firstly, the query is raised and, following input by the user, the fact is temporarily stored that the function code for the generation of the output signal A2 is an OR operation 29. The four function parameters of the OR operation 29 are then called up one after another.

30           According to the programming example of FIG 3, the already programmed NOT operation of the input E3 is input as the first function parameter. Since this operation has already been

ready-programmed, no further input in relation to this function parameter is necessary and the process is continued with the second function parameter immediately after the temporary storage.

5           The second and the third function parameters are the (real) input signals E2 and E4, so that the process can continue immediately with the fourth function parameter.

10           According to the programming example of FIG 3, the OR operation 29 also has only three input signals. The fourth function parameter therefore refers to the constant zero.

          Since this also completes the programming of the output A2, the process continues with the output A3.

15           The outputs A3 and A4 are actually not needed in accordance with the given programming example. Therefore, the NOT operation 30 is selected for the output A3, which is linked to the logic one. As a result, the output A3 is connected to constant zero. The same takes place with the  
20           output A4, for which the NOT operation 31 is input. As an alternative, it is also possible to connect the outputs A3 and A4 directly to logic zero.

          The program resulting therefrom is then stored in the memory 14. It is shown in the Table below.

|    |     |      |       |
|----|-----|------|-------|
| 25 | B1: | FC   | (AND) |
|    |     | ADDR | (E1)  |
|    |     | ADDR | (E2)  |
|    |     | ADDR | (B5)  |
|    |     | ADDR | ("1") |
| 30 | B2: | FC   | (OR)  |
|    |     | ADDR | (B5)  |

```

        ADDR (E2)
        ADDR (E4)
        ADDR ("0")
B3:  FC  (NOT)
5      ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
B4:  FC  (NOT)
10     ADDR ("1")
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")
B5:  FC  (NOT)
15     ADDR (E3)
        ADDR ("1")
        ADDR ("1")
        ADDR ("1")

```

On the left in Table 1, in each case the beginning of the individual instructions is represented by B1 to B5. This information, however, serves only for explanation and is not an integral part of the stored program. In the actual program, FC stands for function code, ADDR for address.

The program as such consists of blocks of five bytes each. The first byte in each case represents the function code of the instruction, the other four bytes represent the four function parameters. The first four instructions B1 to B4 always define the instructions for the four output signals. Beginning at the fifth instruction B5, instructions are then stored to which access must be made indirectly during the calculation of the output signals.

The execution of the user program takes place as follows:

The instructions B1 to B4 are executed cyclically. During the execution of each of the instructions B1 to B4, the function code assigned to the instruction is read first. The function parameters assigned to this instruction are then read one after the other. During the reading of each function parameter, a check is immediately made as to whether this refers to another instruction. If the function parameter refers to an input signal, the latter is stored temporarily.

If, on the other hand, the function parameter refers to another instruction, a branch is made to this instruction immediately, that is to say before the reading of the other function parameters, and the function code assigned to this instruction and the function parameters assigned to this instruction are read. Here, too, if the function code points to a new instruction, this instruction is executed in turn, before the further function parameters are exhausted. When all the function parameters of an instruction have been exhausted, the output signal assigned to this instruction is determined, if appropriate a jump is made back to the calling instruction, and the determined output signal is used as a base for the further execution of the user program.

Here, too, it is intended to explain the principle in more detail with reference to the program according to the Table.

Firstly, therefore, the function code of the instruction B1 is read. The function parameters are then read, which refer to the inputs E1 and E2. Upon reading the third function parameter, a branch is made to the instruction B5. Its function code and all four function parameters (since all refer to input signals) are read and the output signal assigned to the instruction B5 is determined. A return is then made to the instruction B1 with this output signal and the fourth function parameter is read. The output signal A1 is then determined and stored in the read-write memory 18.

Then, the function code of the instruction B2 is read and the first function parameter of the instruction B2. A branch is then made once more to the instruction B5, whose function code and whose four function parameters are read and the output signal of the instruction B5 is determined. A return is then made to the instruction B2 and the other three function parameters are read. The output signal of the instruction B2 is then determined.

Finally, the function code and the function parameters of the instruction B3 are also read and executed and the output signal A3 thus determined. The same takes place for the instruction B4 and the output signal A4 resulting therefrom.

Following the determination of the last output signal A4, the output signals A1 to A4 thus determined are output to the industrial process 17 and the input signals E1 to E4 are read in again. A new cycle in the determination of the output signals A1 to A4 is then begun.

As an alternative to the creation method described, it is also possible to store the instructions which belong to an output signal immediately in the nonvolatile memory 14, if the respective output signal has been completely programmed. Therefore, for example, the instructions B1 and B5 could be stored immediately, as soon as the output signal A1 has been completely programmed.

Likewise, it is also possible to store each input function code and each input function parameter immediately in the nonvolatile memory 14. In this case, at the beginning of a program creation process, it would be necessary to store the fact that the programming of all the output signals A1 to A4 is invalid in an area of memory of the nonvolatile memory 14 specifically provided for this. For example, four 1-bit memory cells could be set to logic zero, in order to indicate that the four output signals A1 to A4 are not validly programmed.

The four 1-bit memory cells could then be set to logic one when the output signals A1 to A4 have been validly programmed. The as-valid identification can in this case be carried out optionally following the creation of the entire user program or separately for each output signal A1 to A4 following the calling up of all the instructions of the respective output signal A1 to A4.

The instructions also do not have to be stored in the order according to the Table. It is, for example, possible to store all the instructions which are needed for the determination of the output signal A1 here the instructions B1 and B5, one after the another in the memory 14. The instructions B2 to B4 for determining the output signals A2 to A4 would then be stored after these instructions (here, B1 and B5). During the execution of the user program it is necessary in this case to track which is the highest address which is called up for determining the output signal A1. The instruction for determining the output signal A2 then begins at the next highest address. In a similar way, namely temporary storage of the highest address needed, it is then possible to determine at which location the instructions for determining the output signals A3 and A4 are arranged. However, it is entirely up to the person skilled in the art which procedure is given preference in the implementation of the present invention.

## Patent Claims

1. Method for the generation and storage of a user program, consisting of instructions (B1 to B5), for a programmable controller,
- 5 - the programmable controller, when executing the user program, determining output signals (A1 to A4) for an industrial process (17) from input signals (E1 to E4) from the industrial process (17),
- 10 - the instructions (B1 to B5) of the user program consisting in each case of a function code and function parameters assigned to the function code,
- the function parameters referring either to an input signal (E1 to E4) or to another instruction (B5),
- having the following steps:
- 15 a) during the generation of the user program, for each output signal, beginning with the instruction (B1 to B4) to be assigned to the respective output signal (A1 to A4), the instructions (B1 to B5) to be executed are called up, the following procedure
- 20 being executed:
- b) the function code to be executed is called up and stored temporarily;
- c) the number of function parameters is established;
- d) the function parameters are called up one after the
- 25 other;
- e) if the function parameter is an input signal (E1 to E4), a reference to the input signal (E1 to E4) is stored temporarily;
- f) if the function parameter is a new function code, a
- 30 reference to a new instruction (B5) is stored temporarily, a branch is made to this new instruction (B5) and the other function code is stored temporarily and the steps c) to f) are repeated for the new instruction (B5);
- 35 g) the temporarily stored function codes and the temporarily stored function parameters are stored in a



nonvolatile memory (14) and, respectively, identified as valid, only after all the instructions (B1 to B5) of at least the respective output signal (A1 to A4) have been called up completely.

- 5 2. Operating method for a programmable controller,  
- the programmable controller cyclically executing a  
user programme consisting of instructions (B1 to B5)  
in order to determine output signals (A1 to A4) for  
an industrial process (17) from input signals (E1 to  
10 E4) from the industrial process (17),  
- the instructions (B1 to B5) of the user program in  
each case consisting of a function code and function  
parameters assigned to the function code,  
- the function parameters referring either to an input  
15 signal (E1 to E4) or to another instruction (B5),  
having the following steps:
- a) during the cyclic execution, for each instruction  
(B1 to B5), beginning with the instructions (B1 to  
20 B4) assigned to the output signals (A1 to A4), the  
following procedure is executed:
- b) the function code assigned to the instruction (B1 to  
B5) and the function parameters assigned to this  
instruction (B1 to B5) are read;
- c) for each function parameter a check is made as to  
25 whether this refers to another instruction (B5);
- d) if the function parameter refers to an input signal  
(E1 to E4), the input signal (E1 to E4) is stored  
temporarily;
- e) if the function parameter refers to another instruc-  
30 tion (B5), a branch is made to this other instruc-  
tion (B5) and the steps b) to e) are repeated with  
the other instruction (B5);



f) when all the function parameters have been exhausted, the output signal (A1 to A4) assigned to the instruction (B1 to B5) is determined.

3. Programming unit for implementing the method according to Claim 1, having

- means (2, 4, 5, 6, 7, 9, 10) for calling up and temporarily storing function codes of instructions to be executed;
- means (2, 4) for establishing the number of function parameters of the instructions to be executed;
- means (2, 4, 5, 6, 7, 9, 10) for calling up and temporarily storing the function parameters;
- means (2, 4) for checking the function parameters as to whether these are a new function code;
- means (2, 4) for branching to a new instruction if the function parameter is a new function code;
- means (2, 4) for preventing the storing of the temporarily stored function codes and the temporarily stored parameters in a nonvolatile memory (14) or, respectively, for preventing the as-valid identification if the function parameters of all instructions to be executed of an output signal (A1 to A4) have not yet all been stored temporarily.

4. Programmable controller for implementing the method according to Claim 2, having

- means (13) for reading function codes and function parameters assigned to instructions;
- means (13) for checking the function parameters as to whether these refer to an input signal or to another instruction;
- means (18) for storing the input signal temporarily if the function parameter refers to the input signal;
- means (13) for branching to the other instruction if the function parameter refers to the other instruction;



- means (13) for returning to a calling instruction;
- means (13) for determining the output signal assigned to the instruction when all the function parameters assigned to the latter have been exhausted.

5

5. Programmable controller according to Claim 4, which is combined with a programming unit according to Claim 3.

