

Description

Synchronization method

5 The present invention relates to a synchronization method for an automation system which is of redundant design and is composed of at least two subsystems, in which the at least two subsystems each have at least one processor, are coupled to one another
10 via communication means, carry out functionally identical sequences of machine commands in order to control a process and are synchronized to one another from time to time via the communication means.

 Synchronization methods for automation systems
15 are generally known. In the case of the method according to EP-A-0 497 147, the subsystems of an automation system which is composed of two subsystems are synchronized at the latest after a time which can be preselected by the user has elapsed. A similar
20 situation applies to the subject matter of EP-A-0 616 274. Here, the synchronization is carried out as a function of the actual elapsed time for a sequence of machine commands. To do this, it is not only necessary to know the elapsed times for the commands in the user
25 program, but also permanently to add up these elapsed times. The known methods are not yet optimal since an unnecessarily long time is wasted on synchronization.

 The object of the present invention is, in consequence, to specify a synchronization method with
30 minimal complexity for an automation system which is composed of at least two subsystems. In this case, the synchronization method is intended to be independent of the presence of specific hardware. The synchronization method according to the invention is at the same time
35 intended to have as little effect as possible on the computation performance of the automation system.

The object is achieved by a method according to the features specified in the characterizing part of patent claims 1 and 2.

5 The synchronization method is based on a counter for each of the subsystems - called a synchronization counter in the following text - which is incremented, that is to say increased by one, by incrementation instructions. Incrementation is a particularly simple task for a processor and, in
10 specific cases, is also carried out within a single clock cycle, and thus particularly quickly.

 Synchronization is carried out at least whenever one of the at least two subsystems - called the alarmed subsystem in the following text - records a
15 special event, for example an interrupt or an alarm. In this case, the alarmed subsystem transmits the current value of its synchronization counter to the at least one other subsystem. This transmission at least informs the at least one other subsystem that a special event
20 has occurred. For its part, the at least one other subsystem transmits the current value of its synchronization counter to every other subsystem. Each of the subsystems is thus informed of the current value of its own synchronization counter as well as the value
25 of the synchronization counter of the at least one other subsystem.

 At the time when the special event occurs, one of the subsystems may be ahead of the at least one other subsystem in carrying out the sequence of machine
30 commands. If this is the case, then it is furthermore possible that the subsystem which is ahead has already carried out a greater number of incrementation instructions, so that the values of the synchronization counters in the subsystems differ.

35 Each subsystem uses the value of its own synchronization counter and the value of the synchronization counter

of the at least one other subsystem to determine the currently greatest value. The value of the synchronization counter which all the subsystems can reach together by continuing to carry out the sequence of machine instructions is the greatest value of the synchronization counter increased - that is to say incremented - by one. The greatest value of the synchronization counters of all the subsystems is thus incremented in each subsystem. This value, which is the same for all subsystems, is called the synchronization counter required value in the following text. As an alternative, this can, of course, also be defined by incrementing the values of the synchronization counters first of all and then determining the synchronization counter required value as the maximum synchronization counter value.

The subsystems interrupt the execution of the sequence of machine commands on reaching the position of that incrementation instruction - called the synchronization position in the following text - at which the values of the respective synchronization counters reach the synchronization counter required value. Since the incrementation instructions occur at mutually corresponding points in the functionally identical sequences of machine commands, the incrementation instructions are in consequence particularly highly suitable for use as a potential synchronization position.

The reaching of such a synchronization position can once again be determined particularly easily by comparing the value of the respective synchronization counter with the synchronization counter required value. The synchronization method is thus based essentially on incrementation and comparison operations, which can be carried out particularly quickly and efficiently by a processor.

Since the two subsystems can reach their synchronization position at different times, each

subsystem uses a synchronization signal to signal to the at least one other subsystem that the synchronization position has been reached.

Each subsystem then waits to receive the synchronization signal from the at least one other subsystem, by which means the latter signals that it has itself reached the synchronization position. If
5 there is more than one other subsystem, each subsystem waits to receive the synchronization signals from all the other subsystems, so that each subsystem does not carry out any further actions until all the subsystems have reached the synchronization position. However,
10 applications are equally feasible in which no synchronization signal is exchanged on reaching the synchronization position.

After reaching the synchronization position and after recording the synchronization signal from the at
15 least one other subsystem, each subsystem has a specific reaction for the special event, and carries it out. This reaction is a specific sequence of machine commands for the special event. After this, each subsystem continues to carry out that sequence of
20 machine commands whose execution was interrupted as a result of the synchronization on reaching the synchronization position and as a result of the reaction to the special event.

The alarmed subsystem transmits the special
25 event to the at least one other subsystem at the earliest immediately after recording the special event, for example in conjunction with the transmission of the current value of the synchronization counter, but at the latest in conjunction with the synchronization
30 signal.

There are also cases in which the exchange and/or waiting for the synchronization signal do not take place for carrying out the specific reaction for the special event. In such a case, the reaction which
35 is specific to the special event is carried out immediately on reaching the synchronization position. Although no time synchronization then occurs, nevertheless, the method according to the invention ensures

that, although the reaction which is specific to the special event is not carried out at the same time, it is, however, carried out at functionally identical positions within the sequence of machine commands for the respective subsystem. This ensures the desired synchronization, since the synchronization positions are arranged at mutually corresponding points in the sequences of functionally identical machine commands.

The synchronization position can be reached in a particularly efficient manner if, as claimed in claim 3, the address of each machine command in the sequence of machine commands can be determined, in particular the address of the machine command currently to be carried out as well as the address of that machine command which corresponds to the synchronization position.

In consequence it is possible that, when carrying out the sequence of machine commands - at least after the alarmed subsystem has recorded the special event - the address of the machine command currently to be carried out is continuously compared with the address of that machine command which corresponds to the synchronization position. If a match between the two addresses to be compared is found, then the synchronization position has been reached. The execution of the sequence of machine commands can in this way be interrupted precisely on reaching the synchronization position.

In the situation where the synchronization position is within a subsequence (which may conditionally or unconditionally be repeated) in the sequence of machine commands, the individual machine commands are processed more than once. The position of an incrementation instruction does not, however, become the synchronization position until the synchronization counter reaches the synchronization counter required value with the execution of the incrementation instruction. Thus, in the case of loops, it is possible for an incrementation instruction

to be processed more than once in the course of the loop, but not to reach the synchronization position until a certain number of processing operations have been carried out. Thus, in this case, it is not
5 sufficient to determine the address of the incrementation instruction. In this case, the synchronization position is not reached until the address of the machine command currently to be carried out matches the address of that machine command which
10 corresponds to the potential synchronization position, and the synchronization counter at the same time matches the synchronization counter required value.

If - as described above - the address of each machine command can be determined, then a "break point"
15 can furthermore be linked to the synchronization position, so that the execution of the sequence of machine commands is interrupted exactly at the synchronization position. The processor then directly carries out the check to determine whether the
20 synchronization position - that is to say the address linked to the break point - has been reached. Normally - processors from INTEL, for example i80386, i80486, Pentium etc. - an interrupt is initiated on reaching the address linked to the break point, whose occurrence
25 unambiguously indicates that the address in question has been reached.

If the processor can be operated in a single-step mode, then, as an alternative, it is also possible when executing the sequence of machine commands - at
30 least after the alarmed subsystem has recorded a special event - continually to compare the address of the machine command currently being carried out with the address of the synchronization position. If a match is found, then the synchronization position has been
35 reached, and the processing of the individual commands in the sequence of machine commands is interrupted.

A further efficient possibility for the reaching of the synchronization position consists in that, as claimed in the features of

claim 4, when carrying out the sequence of machine commands - at least after the alarmed subsystem has recorded the special event the value of the synchronization counter is continually compared with the synchronization counter required value, and in that the execution of the sequence of machine commands is interrupted when the value of the synchronization counter reaches the synchronization counter required value. To do this, a first memory point and a second memory point are determined, in which case the value of the synchronization counter is stored in the first memory point, and the synchronization counter required value is stored in the second memory point. In the case of conventional processors, in particular the said processors from INTEL, the processor can be programmed such that the execution of the sequence of machine commands is interrupted as soon as the content of the first memory point matches the content of the second memory point. This specific functionality of the processor has been primarily used in the passed for programs that search for faults - so-called debuggers. It can, of course, also be used for reaching the synchronization position.

The positions of the incrementation instructions within the sequence of machine commands can be defined on the bases of various strategies. If the sequences of machine commands are produced from command sequences in a programming language, it is advantageous to insert the incrementation instructions into the sequences of machine commands as early as during the conversion of the command sequences. However, the scope of the invention also includes the insertion of the incrementation instructions after the production of the sequence of machine commands but before they are carried out, by means of a relocation to them.

If the command sequences are command sequences in a programming language in which associated command sequences are combined in a block, the

incrementation instructions are in each case inserted into the sequence of machine commands at at least one of the boundaries of the block. Examples of such programming languages are all the "structured" programming languages, such as Pascal, PEARL, C/C++, etc. - which all have language means for agreeing a block, in which case, furthermore, specific language means also implicitly define a block.

As also in the prior art according to EP-A-0 497 147 - incrementation instructions can also be inserted in conjunction with all the jump commands, for example conditional and unconditional jump commands. The term jump commands in this case means all commands and thus all instructions from so-called high-level languages which lead to a conditional or unconditional jump at the language level which the processor can understand directly. This is necessary since a jump may result in incrementation instructions being jumped over. In particular, it would be conceivable that, as a result of a jump over a relatively long time period, and possibly even over an unlimited time period, for only machine commands between two incrementation instructions to be carried out, so that no incrementation instructions would be carried out during this time period. In this case, no potential synchronization positions would correspondingly be reached, so that it would not be possible to carry out synchronization. Thus, the incrementation instructions are frequently, but not necessarily, inserted before each such jump command.

When so-called subroutines are called, it is advantageous if the incrementation instructions are inserted within the subroutine, that is to say at the start of the subroutine and at the end of the subroutine. This advantageously makes it considerably easier to insert the incrementation instructions into the sequences of machine commands, since the calling of a subroutine cannot always be unambiguously identified,

although the start and the end of a subroutine can be clearly identified.

CALL and RET instructions can be quoted as examples of this for the assembler programming language, each of which initiates a jump to a subroutine or a jump back from a subroutine.

5 In the case of programmable logic controllers, whose specific programming languages normally provide no language means for block-oriented structuring, the incrementation instructions must always be inserted before jump commands. Particularly in the case of
10 programmable logic controllers, the sequence of machine commands is frequently structured by means of so-called "modules". A module can thus always be regarded as a special form of block. Since the module changes which become necessary during the execution of the sequence
15 of machine commands in a programmable logic controller are carried out with the aid of jump commands, the insertion of incrementation instructions before a jump command also leads to the desired result here. The same also applies in an analogous manner to unstructured
20 programming languages and programming languages that are structured only to a minor extent, such as assembler or BASIC.

 Furthermore, incrementation instructions can also be inserted into the sequence of machine commands
25 after a fixed number (which can, in particular, be predetermined by the user) of individual commands in the sequence of machine commands. A sequence of incrementation instructions predetermined in this way has the advantage that the mean time interval between
30 the occurrence of two incrementation instructions always remains roughly the same. The time period between the occurrence of two incrementation instructions is defined by the sum of the elapsed times for the individual commands to be carried out between
35 the incrementation instructions.

 If the potential synchronization positions which correspond to the incrementation instructions are desired to be at actually equidistant time intervals,

then, when the command sequences in the programming language are being incorporated into the

sequence of machine commands, the elapsed time for each individual command is determined, and the elapsed time for a sequence of individual commands is determined by adding up the elapsed times for the individual commands. On reaching a specific elapsed time (which can, in particular, be predetermined by the user), an incrementation instruction is inserted into the sequence of machine commands. After an incrementation instruction, the addition of the elapsed times for the individual commands starts from zero again. However, for the reasons which have already been mentioned above, it is always necessary to insert incrementation instructions before jump commands. The incrementation instruction before a jump command can also be carried out in addition to the insertions, on the basis of the fixed elapsed time/separation grid. However, alternatively it is possible for the elapsed time or separation counting to start from zero after an incrementation instruction which is inserted before a jump command.

Furthermore, the incrementation instructions can also be inserted into the sequence of machine commands or, respectively, into the sequence of command sequences - called the source train or, respectively, the source sequence in the following text - by investigating the source train or, respectively, the source sequence for specific machine commands or, respectively, specific command sequences, in particular for specific subsequences of the sequences of machine commands or, respectively, specific subsequences of command sequences - called a search train or, respectively, search sequence in the following text.

If the search train or, respectively, the search sequence is identified within the source train or, respectively, the source sequence, it is possible for the incrementation instructions to precede or be attached to the search train or, respectively, to the search sequence. The search train or, respectively, the

search sequence is the respective expression of those elements in the source train or, respectively, the source sequence to which, according to the above statements, an incrementation instruction must be
5 allocated - that is to say at least block boundaries and/or jump commands.

According to one specific implementation, the insertion of the incrementation instruction is carried out with the aid of a specific compiler. The compilation process by the specific compiler differs
5 from that of a standard compiler in that those language means which, for example, define a block or a jump command are identified and are translated into the associated machine commands, and, in addition, into the incrementation instructions.

10 One alternative is a so-called "pre-processor". The pre-processor allows the command sequences to be pre-processed in a programming language. With the assistance of a specific Syntax - which is in principle known by a programmer -, such a pre-processor can be
15 caused to exchange specific language means in a programming language, for example those language means which initiate a block, for a language means which initiates such a block as well as a further language means using which the incrementation instruction is
20 triggered.

However, it is desirable for the insertion of the incrementation instructions into the sequence of machine commands to be carried out in a manner that is transparent to the user. The instructions to the pre-processor described above are identifiable to the user
25 and can thus be manipulated. However, manipulation could possibly lead to errors in the synchronization and/or to errors in the resulting program itself. A further advantageous alternative for the insertion of
30 incrementation instructions into the sequence of command sequences is therefore to use, instead of the pre-processor, a specific pre-processor which, independently of any other exchanges in accordance with instructions, converts those command sequences to which
35 (according to the above statement) an incrementation instruction can be assigned into specifically these command sequences as well as the incrementation instruction.

Using the said alternatives for insertion of the incrementation instructions, which cannot be influenced significantly by the user, it is also possible for the user himself to insert the incrementation instructions into the sequence of machine commands or into the sequence of command sequences.

Further advantages and details result from the following description of an exemplary embodiment. In the figures:

- 10 FIG 1 shows a block diagram of an automation system,
- FIG 2 shows the principle of the synchronization method,
- FIG 3 shows a flowchart of the synchronization method,
- 15 FIG 4 shows the principle of the insertion method,
- FIG 5 and FIG 6 show a flowchart of the insertion method.

20 According to FIG 1, the redundant automation system 1 comprises at least two subsystems 2, 2', which can be connected to a technical process, which is not illustrated, in order to control and manage said process. Each subsystem 2, 2' has a processor 3, a data
25 memory unit 4, a command sequence memory unit 5 as well as at least one communication means 6. In this case, the data memory unit 4 and the command sequence memory unit 5 are not necessarily separate units in the redundant automation system 1.

30 The at least two subsystems 2, 2' communicate with one another and with the technical process, which is not illustrated, via the communication means 6 and via a bus 7.

The data memory unit 4 is used to stored at
35 least the value of the synchronization counter SZ, SZ' as well as the synchronization counter required value SZZ. The command sequence memory unit 5 is used to store at least the sequence of machine commands MP -. The sequence of machine commands MP is at least

functionally identical to the two subsystems 2, 2'. In order to

control the technical process, which is not illustrated, the processor 3 of the subsystems 2, 2' carries out the sequence of machine commands MP from the command sequence memory unit 5.

5 FIG 2 shows the sequences of the machine commands MP for the at least two subsystems 2, 2'. The sequence of machine commands MP on the left-hand side is in this case intended to be the sequence of machine commands MP of the subsystem 2, and the sequence of
10 machine commands MP on the right-hand side is intended to correspond to the sequence of machine commands MP for the subsystem 2'.

 According to FIG 2, incrementation instructions IA are inserted into the sequences of machine commands
15 MP. With the aid of the incrementation instructions IA, the synchronization counters SZ, SZ' of the respective subsystem 2, 2' are incremented, that is to say increased by 1. The positions denoted by the arrow 10, 10' within the sequence of machine commands MP each
20 denote that machine command which is currently being carried out at the time when the special event 11 occurs. As can furthermore be seen from FIG 2, at least one of the subsystems 2, 2' records the special event 11.

25 According to FIG 2, at this time, the subsystems 2, 2' are carrying out different machine commands in the sequence of machine commands MP. Dealing with the special event 11 at this time could, under some circumstances, lead to inconsistent data in
30 the subsystems 2, 2' of the automation system 1. It is therefore necessary to synchronize the subsystems 2, 2' before dealing with the special event 11. The synchronization counter SZ for the subsystem 2 has the value n at the time when the special event 11 occurs.
35 The synchronization counter SZ' for the subsystem 2' is already at the value n+1 at the time when the special event occurs.

For synchronization, that subsystem 2, 2' which is ahead of the at least one other subsystem 2, 2' in the execution

of the sequence of machine commands MP must interrupt the execution of the sequence of machine commands MP until the other subsystem 2, 2' has also reached the corresponding position within the functionally identical sequences of machine commands MP.

The incrementation instructions IA are in each case directly corresponding positions within the sequence of machine commands MP. In the state illustrated in FIG 2, the synchronization counter SZ' has the value $n+1$. The synchronization counter SZ has the value n . If the execution of the sequence of machine commands MP in the subsystem 2' were stopped in the current state, while the execution of the sequence of machine commands MP in the subsystem 2 was continued, until the synchronization counter SZ had also reached the value $n+1$, synchronization would nevertheless not be achieved.

The next incrementation instruction IA which both subsystems 2, 2' can reach at the same time is thus that incrementation instruction IA for which both the synchronization counter SZ and the synchronization counter SZ' assume the value $n+2$. The value of the synchronization counter SZ, SZ' which both subsystems 2, 2' can reach together is called the synchronization counter required value SZZ. The synchronization position SP in this case corresponds to the position of that incrementation instruction IA within the sequences of machine commands MP for which the value of the synchronization counter SZ, SZ' reaches the synchronization counter required value SZZ.

After recording the special event 11, the alarmed subsystem 2 according to FIG 3 sends the value of its synchronization counter SZ as well as the special event 11 to the at least one other subsystem 2' in order to determine the synchronization counter required value SZZ. - This is the earliest possible time for transmitting the special event 11. The special event must be transmitted at the latest in conjunction with the

transmission of the synchronization signal SS. - In this way, the at least one other subsystem 2' is informed that a special event 11 has occurred and, for its part, sends the current value of its synchronization counter SZ', which the alarmed subsystem 2 receives.

Each subsystem 2, 2' thus knows the value of its own synchronization counter SZ as well as the value of the synchronization counter SZ' of the at least one other subsystem 2, 2'. The synchronization counter required value SZZ is obtained from the maximum value of the two incremented synchronization counters SZ, SZ'. Alternatively, it is, of course, also possible to determine the maximum value first of all, and to determine the synchronization counter required value SZZ by incrementing this maximum value.

After exchanging the synchronization counter SZ, SZ', the two subsystems 2, 2' continue to carry out the sequence of machine commands MP. As soon as one of the subsystems 2, 2' has reached the synchronization position SP while carrying out the sequence of machine commands MP, it interrupts the execution of the sequence of machine commands MP. The fact that the synchronization position SP has been reached can be identified either by comparing the address of the command currently being carried out in the sequence of machine commands MP with the address of the synchronization position SP or else - according to FIG 3 - by monitoring the value of the synchronization counter SZ, SZ' until it reaches the synchronization counter required value SZZ.

Each subsystem 2, 2' which reaches the synchronization position SP signals to the at least one other subsystem 2, 2', by transmitting a synchronization signal SS, that it has reached the synchronization position SP. The alarmed subsystem 2, 2' transmits to the at least one other subsystem 2, 2' the special event 11 at the earliest immediately after recording the special event 11 and at the latest

in conjunction with the transmission of the synchronization signal SS.

As soon as it has reached the synchronization SP and has received the synchronization signal SS from the at least one other subsystem 2, 2', each subsystem 2, 2' determines a specific reaction for the special event 11, and carries out this reaction. After carrying out the specific reaction for the special event 11, the subsystems 2, 2' continue to carry out the sequence of machine commands MP after the synchronization position SP.

As has already been mentioned in the introduction, there are also cases in which the exchange and the waiting for the synchronization signal SS are not necessary in order to carry out the specific reaction for the special event 11. In such a case, the reaction which is specific to the special event is carried out immediately on reaching the synchronization position SP. It should thus be expressly mentioned that, in the flowchart according to FIG 3, the blocks between the question "SZ<SZZ?" and the execution of the "reaction which is specific for the special event" are optional. The optional section is omitted whenever there is no requirement for exact time synchronization and, instead of this, it is sufficient for the reaction which is specific to the special event admittedly not to be carried out at the same time, but nevertheless at functionally identical positions in the sequences of machine commands MP.

According to an alternative refinement of the synchronization method according to the invention, decrementation instructions IA are inserted into the sequences of machine commands instead of the incrementation instructions IA. In such a case, it is possible - in exactly the same way as the synchronization counters SZ, SZ' were previously in each case incremented by 1 starting from 0 - for the synchronization counters SZ, SZ' in each case to be decremented starting from a start value, for example

32,767. The synchronization method is in this case carried out in a completely analogous manner, by evaluating this

synchronization counter SZ, SZ', which is now decremented.

FIG 4 shows the method for inserting the incrementation instructions IA using the example of command sequences in a programming language. Particularly for structured programming languages, the program sequence has been represented in a so-called flowchart. This flowchart shows, in particular, the block boundaries and the structures which thus result. A simple flowchart is illustrated schematically on the left-hand side. The flowchart comprises the two blocks B1 and B2, the block B1 representing, for example, the so-called main program while the block B2, in contrast, shows an example of a so-called subroutine.

The running of the programs starts in the main program B1. Both the main program B1 and the subroutine B2 each form a separate block. They are therefore embedded in block commands. The block command at the start of the main program B1 and of the subroutine B2 respectively initiates the block, and the block command at the end of the main program B1 and of the subroutine B2 ends the block. Further blocks may also occur within the instructions in a block, for example for instructions to be repeated more than once, such as those in the case of loops. The subroutine B2 is called from the inner block in the main program B1.

On the right-hand side, FIG 4 shows the flowchart which results with the incrementation instructions IA inserted. Each block command which initiates a block is in this case transformed into a block command which initiates a block command and transforms an incrementation instruction IA; each block command which ends a block is in this case transformed into an incrementation instruction IA and a block command which ends a block. The sequence of block command and incrementation instruction IA is not limited to the sequence illustrated in FIG 4,

and may also vary. The instruction which is used in the inner block of the main program B1 to call the subroutine B2, is also transformed into an incrementation instruction IA and an instruction by means of which the subroutine B2 is called.

FIGS 5 and 6 each show a flowchart of the insertion method.

FIG 5 illustrates the insertion method as claimed in claim 6, graphically. The value of the command counter BZ is first of all initialized, that is to say set to zero. The machine command currently being carried out in the sequence of machine commands MP is then determined. Only if the machine command is a jump command is an incrementation instruction IA inserted in the sequence of machine commands MP. Only if the command counter BZ exceeds a command counter threshold value BZS is an incrementation instruction IA inserted into the sequence of machine commands MP and the command counter BZ then reset to zero. The determined machine command is then inserted into the sequence of machine commands MP. Following the insertion of the machine command, the command counter BZ is incremented, that is to say increased by 1, since the number of machine commands in the sequence of machine commands MP has now been increased by 1, and only 1, machine command. The processing is continued by determining the next machine command, until all the machine commands have been processed.

An elapsed time counter or an elapsed time counter threshold value can also be used instead of the command counter BZ or the command counter threshold value BZS according to FIG 5. The elapsed time counter is in each case increased by the elapsed time for the machine command that has been inserted. The elapsed time counter threshold value determines the elapsed time between two incrementation instructions IA.

FIG 6 shows the insertion method as claimed in claim 7, which relates to a sequence of machine commands MP and to a sequence of command sequences in a programming language, whose interpretation controls a process, in particular a technical process. For the insertion of incrementation instructions IA, the sequence of machine commands MP and the sequence of command sequences are investigated for the occurrence of the search train or, respectively, of the search sequence. Only if the search train or, respectively, the search sequence has been found within the sequence of machine commands or, respectively, the sequence of command sequences is an incrementation instruction inserted into them. The insertion method is continued with the investigation of the sequence of machine commands or the sequence of command sequences until they have been completely investigated.

Finally, a comment should be made on the meaning of the expression "functionally identical". Functionally identical sequences of machine commands occur, for example, as a result of compilation of instructions in different high-level languages, but which describe one and the same task. Functionally identical sequences of machine commands also occur as a result of compilation of inscriptions in a high-level language, but which have been compiled by different compilers for this high-level language. Functionally identical sequences of machine commands also occur as the result of compilation of instructions from one and the same high-level language, which describe the same task and which have been processed by one and the same compiler but with different parameters for this compiler, for example different optimization levels.



Patent Claims

1. A synchronization method for an automation system (1) which is of redundant design and is composed of at least two subsystems (2, 2'), in which the at least two subsystems (2, 2') each have at least one processor (3), are coupled to one another via communication means (6), carry out functionally identical sequences of machine commands (MP) in order to control a process and are synchronized to one another from time to time via the communication means, characterized
- in that each of the subsystems (2, 2') has a synchronization counter (SZ, SZ'),
 - in that the sequences of functionally identical machine commands (MP) have incrementation and decrementation instructions (IA), respectively, at mutually corresponding points, by means of which instructions the synchronization counter (SZ, SZ') is incremented or decremented, respectively,
 - in that synchronization is carried out at least whenever one of the subsystems (2, 2'), the alarmed subsystem (2, 2'), records a special event (11),
 - in which case this synchronization comprises the following steps:
 - each subsystem (2, 2')
 - a) transmits the current value of its synchronization counter (SZ, SZ') to the other subsystem (2, 2'),
 - b) uses the incremented or decremented value of its own synchronization counter (SZ, SZ') and the incremented or decremented value of the synchronization counter (SZ, SZ') from the other subsystem (2, 2') to determine the currently largest or smallest value, the synchronization counter required value (SZZ),
 - c) interrupts the execution of the sequence of machine commands (MP) on reaching a position within the sequence of machine commands (MP) - called the synchronization position (SP) in the



following text - at which the values of the
respective synchronization counters (SZ, SZ')
reach the synchronization counter required value
(SZZ),



- d) determines a specific reaction for the special event (11), carries out this reaction and then continue to carry out the sequence of machine commands (MP) after the synchronization position (SP),
- 5 (SP),
- in which the alarmed subsystem (2, 2') transmits the special event (11) to the other subsystem (2, 2') at the earliest immediately after recording the special event (11) and at the latest in conjunction with the
- 10 determination of the specific reaction for the special event (11).
2. A synchronization method for an automation system (1) which is of redundant design and is composed of at least two subsystems (2, 2'), in which the at
- 15 least two subsystems (2, 2') each have at least one processor (3), are coupled to one another via communication means (6), carry out functionally identical sequences of machine commands (MP) in order to control a process and are synchronized to one
- 20 another from time to time via the communication means, characterized
- in that each of the subsystems (2, 2') has a synchronization counter (SZ, SZ'),
- in that the sequences of functionally identical
- 25 machine commands (MP) have incrementation and decrementation instructions (IA), respectively, at mutually corresponding points, by means of which instructions the synchronization counter (SZ, SZ') is incremented or decremented, respectively,
- 30 - in that synchronization is carried out at least whenever one of the subsystems (2, 2'), the alarmed subsystem (2, 2'), records a special event (11),
- in which case this synchronization comprises the following steps:
- 35 each subsystem (2, 2')
- a) transmits the current value of its synchronization counter (SZ, SZ') to the other subsystem (2, 2' b)
- uses the incremented or decremented value of its own



synchronization counter (SZ, SZ') and the incremented or decremented value of the



to determine the currently largest or smallest value, the synchronization counter required value (SZZ),

- 5 c) interrupts the execution of the sequence of machine commands (MP) on reaching a position within the sequence of machine commands (MP) - called the synchronization position (SP) in the following text - at which the values of the respective synchronization counters (SZ, SZ') reach the synchronization counter required value (SZZ),
- 10 d) uses a synchronization signal (SS) to signal to the other subsystem (2, 2') that the synchronization position (SP) has been reached,
- 15 e) after reaching the synchronization position (SP) and after recording the synchronization signal (SS) from the other subsystem (2, 2'), determines a specific reaction for the special event (11), carries out this reaction and then continues to
- 20 carry out the sequence of machine commands (MP) after the synchronization position (SP),
- in which the alarmed subsystem (2, 2') transmits the special event (11) to the other subsystem (2, 2') at the earliest immediately after recording the special
- 25 event (11) and at the latest in conjunction with the synchronization signal (SS).
3. The synchronization method as claimed in claim 1 or 2, characterized
- in that each machine command within the sequence of
- 30 machine commands (MP) is allocated a unique address by virtue of its position within the sequence of machine commands (MP),
- in that the address of each machine command in the sequence of machine commands (MP) can be determined,
- 35 in particular the address of the machine command currently being carried out as well as the address of that machine command which corresponds to the synchronization position (SP),



- in that, when carrying out the sequence of machine commands (MP) - at least after the alarmed subsystem

(2, 2') has recorded the special event (11) - the
5 address of the machine command currently being carried
out



is continued compared with the address of that machine command which corresponds to the synchronization position (SP), and

- in that the execution of the sequence of machine commands (MP) is interrupted precisely on reaching the synchronization position (SP).

4. The synchronization method as claimed in claim 1 or 2, characterized

- in that, when carrying out the sequence of machine commands (MP) - at least after the alarmed subsystem (2,2') has recorded the special event (11) - the value of the synchronization counter (SZ, SZ') is continually compared with the synchronization counter required value (SZZ), and
- in that the execution of the sequence of machine commands (MP) is interrupted when the value of the synchronization counter (SZ, SZ') reaches the synchronization counter required value (SZZ).

5. The synchronization method as claimed in claim 1, 2, 3 or 4, characterized

- in that the incrementation instructions (IA) are in each case carried out at least
- after a fixed number of individual commands in the sequence of machine commands (MP) or - by assessing the elapsed time for a number of machine commands - at specific, essentially equidistant delay-time intervals as well as
- in conjunction with each jump command.

6. The synchronization method as claimed in one of preceding claims, characterized

- in that the incrementation instructions (IA) are in each case inserted into the sequence of machine commands (MP)
- after a fixed number of individual commands in the sequence of machine commands (MP) or - by assessing the elapsed time for a number of machine commands - at specific, essentially equidistant delay-time intervals as well as
- in conjunction with each jump command.



7. The synchronization method as claimed in one of the preceding claims, characterized

- in that the sequence of machine commands (MP) is generated from a sequence of command sequences in a programming language,
- in that the sequence of machine commands (MP) and, respectively, the sequence of command sequences are investigated for specific machine commands and, respectively, specific command sequences, in particular for specific subsequences of the sequences of machine commands (MP) and, respectively, for specific subsequences of the command sequences - called a search train or, respectively, search sequence in the following text,
- in that the occurrence of the search train or, respectively, the search sequence is identified, and
- in that the incrementation instructions (IA) are placed before or attached to the search train or, respectively, the search sequence.

8. The synchronization method as claimed in claim 7, characterized

- in that associated command sequences are in each case combined in a block, and
- in that one of the search trains or, respectively, one of the search sequences is at least one of the boundaries of the block and/or a jump command.

9. The synchronization method as claimed in claim 8, characterized

- in that the incrementation instructions (IA) are inserted into the sequence of machine commands (MP) during generation of said sequence of machine commands (MP),
- in which the sequence of machine commands (MP) is generated in a manner known per se with the aid of a special compiler,
- in which the special compiler in each case generates the incrementation instructions (IA) and inserts them into the sequences of machine commands (MP) on



identification of the search train or of the search
sequence.



10. The synchronization method as claimed in claim 8 or 9, characterized
- in that the incrementation instructions (IA) are inserted into the sequence of machine commands (MP) before and/or after generation of said sequence of machine commands (MP),
 - in that the sequence of machine commands (MP) is generated in a manner known per se with the aid of a standard compiler,
 - 10 - in that the incrementation instructions (IA) are inserted into the sequence of machine commands (MP) or, respectively, into the sequence of command sequences in each case before and/or after generation, on identification of the search train or,
 - 15 respectively, of the search sequence.

Abstract

Synchronization method

Synchronization method for an automation system (1) which is of redundant design and is composed off at least two subsystems (2, 2') which carry out functionally identical sequences of machine commands (MP) in order to control a process, which sequences of machine commands (MP) have incrementation instructions (IA), for incrementing a synchronization counter (SZ, SZ'), at mutually corresponding points. The synchronization is carried out, using the value of the synchronization counter (SZ, SZ'), at least whenever one subsystem (2, 2') records a special event (11). To this end, a synchronization counter required value (SZZ) is determined, so that the subsystems (2, 2') interrupt the execution of the sequence of machine commands (MP) on reaching a synchronization position (SP) at which the values of the respective synchronization counters (SZ, SZ') reach the synchronization counter required value (SZZ).

FIG 2

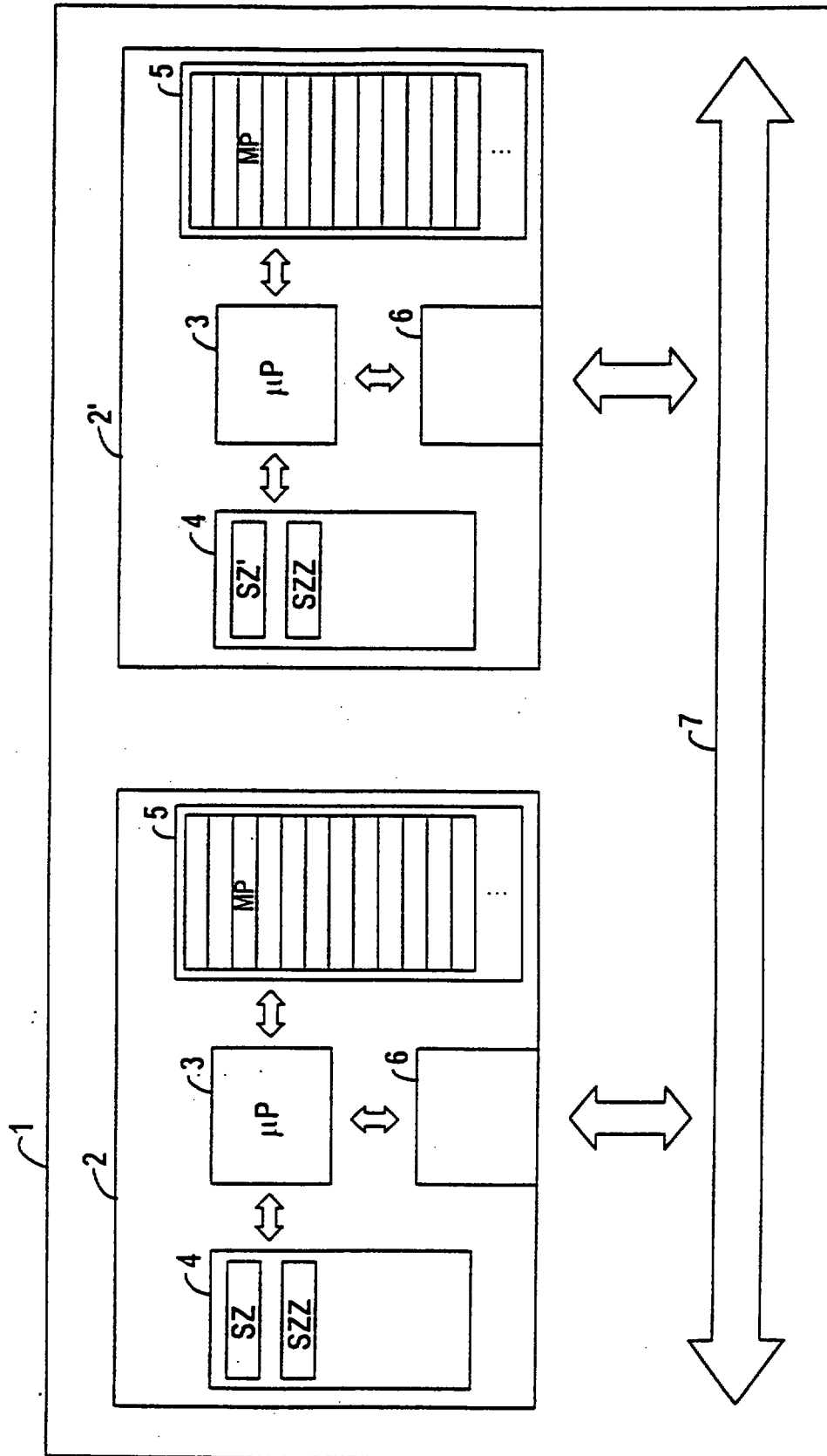


FIG 1

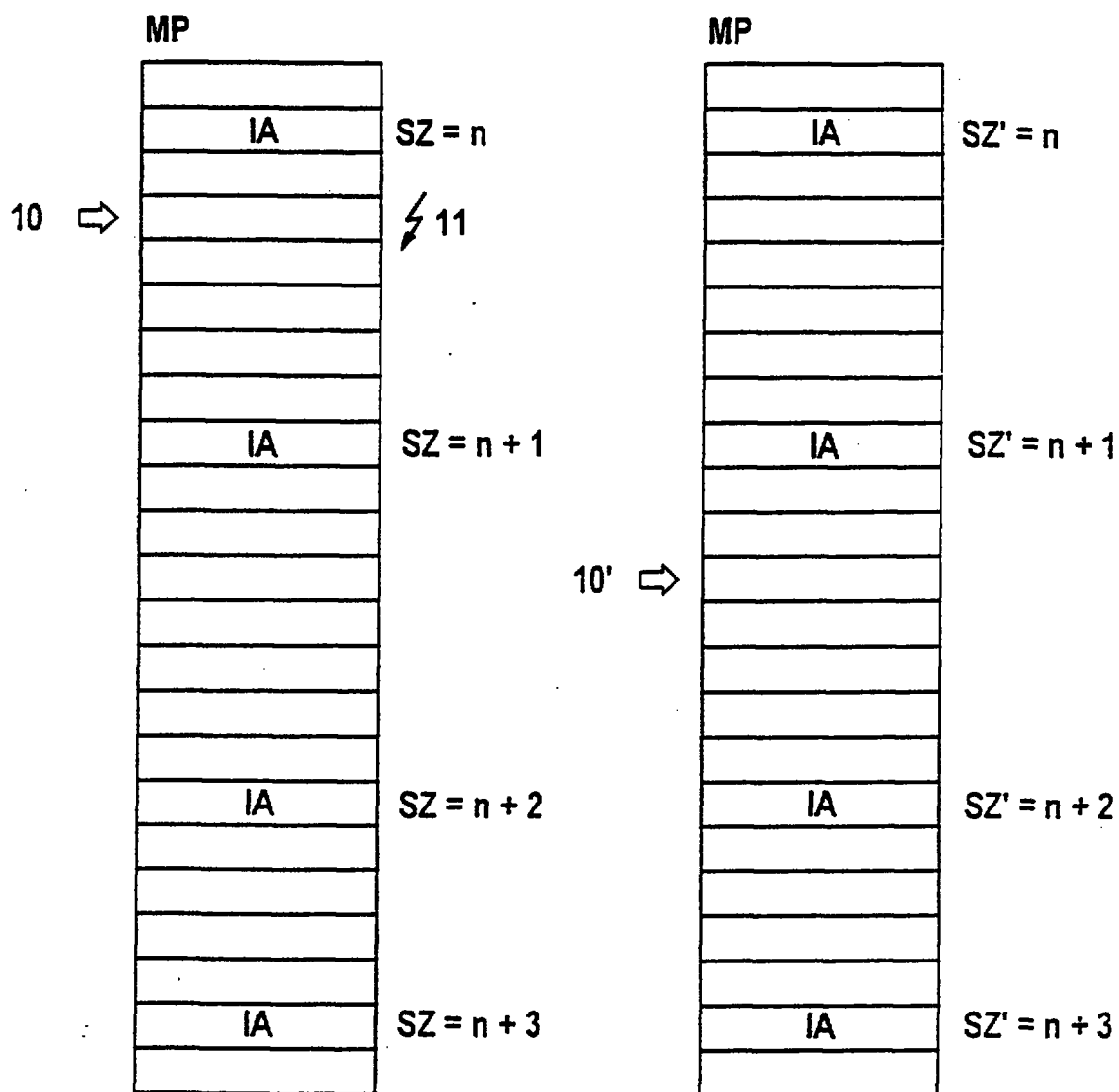


FIG 2

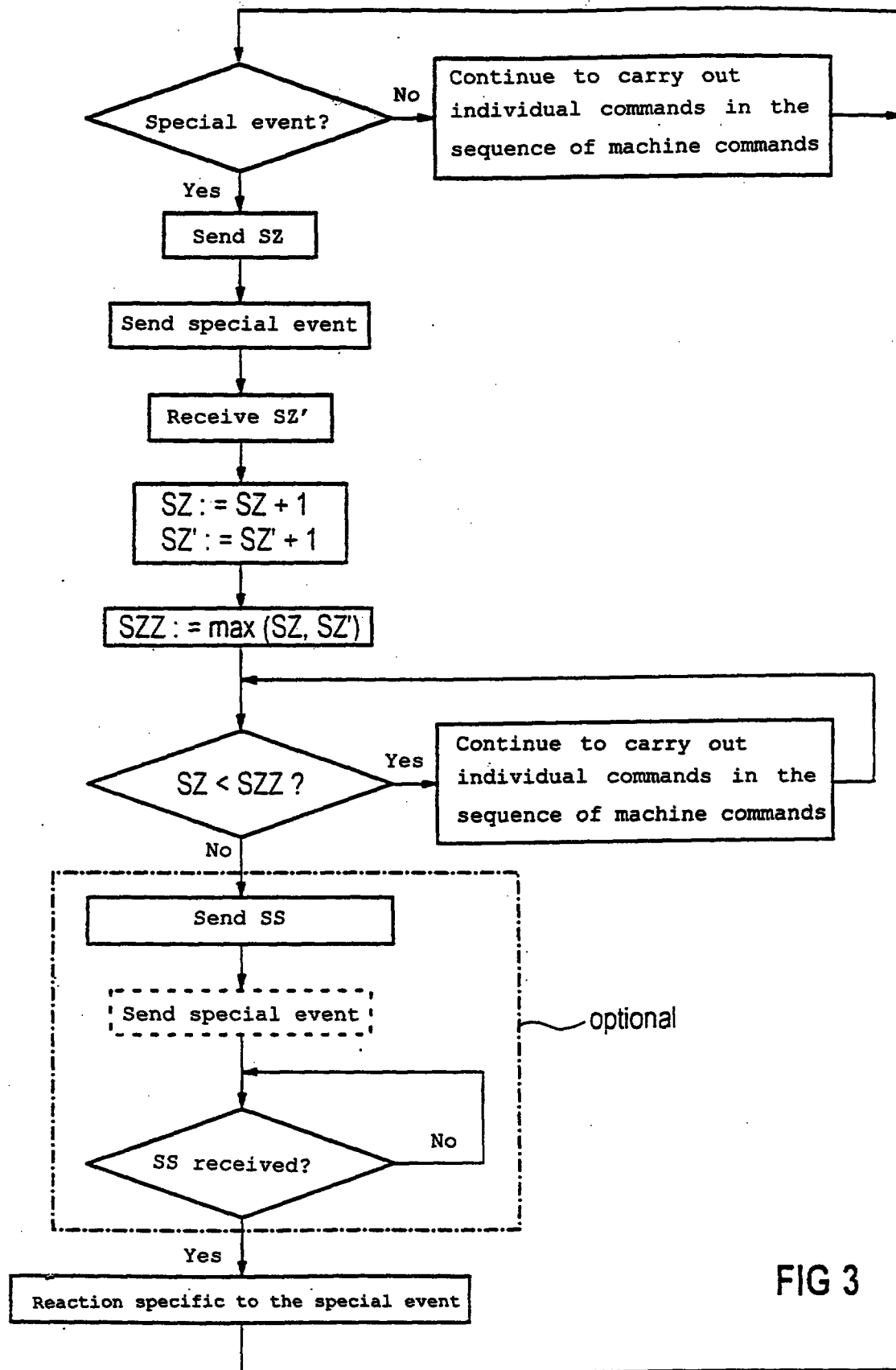


FIG 3

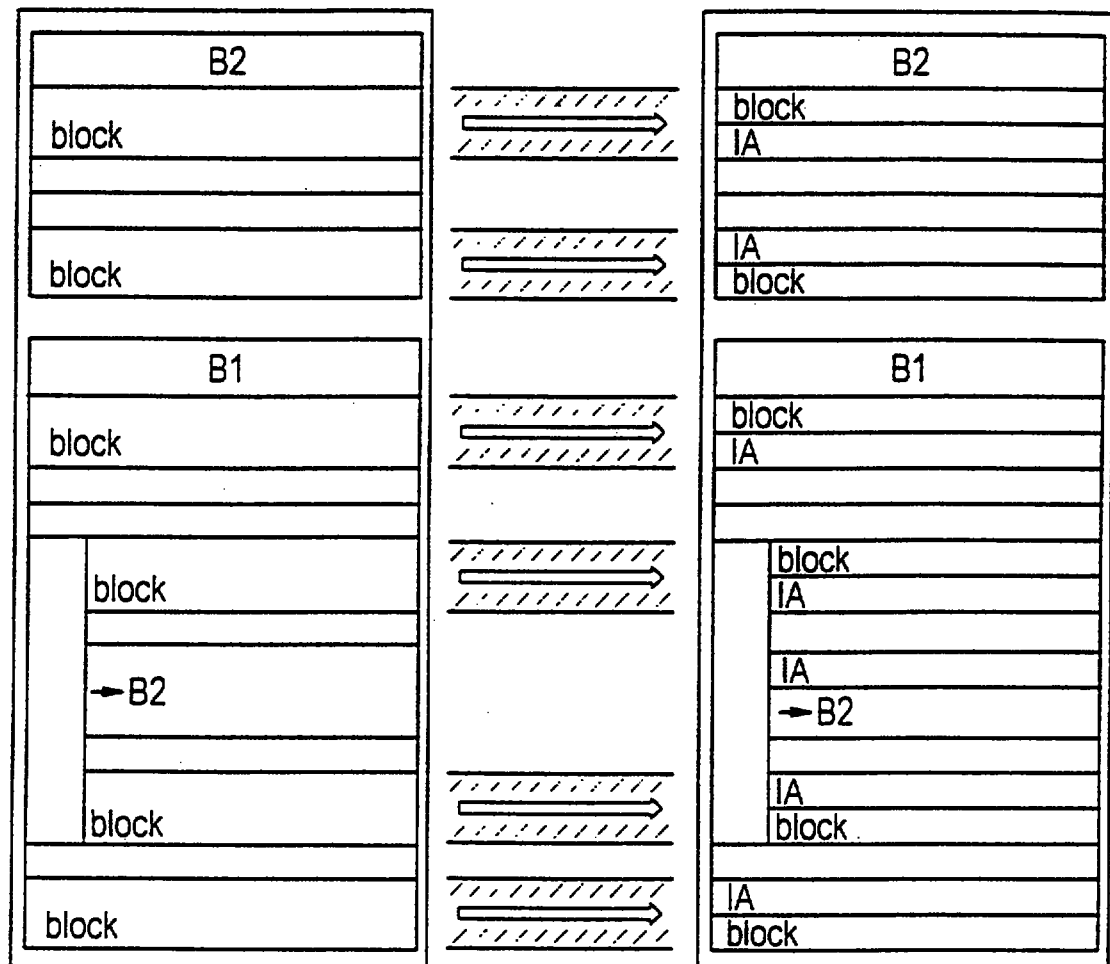


FIG 4

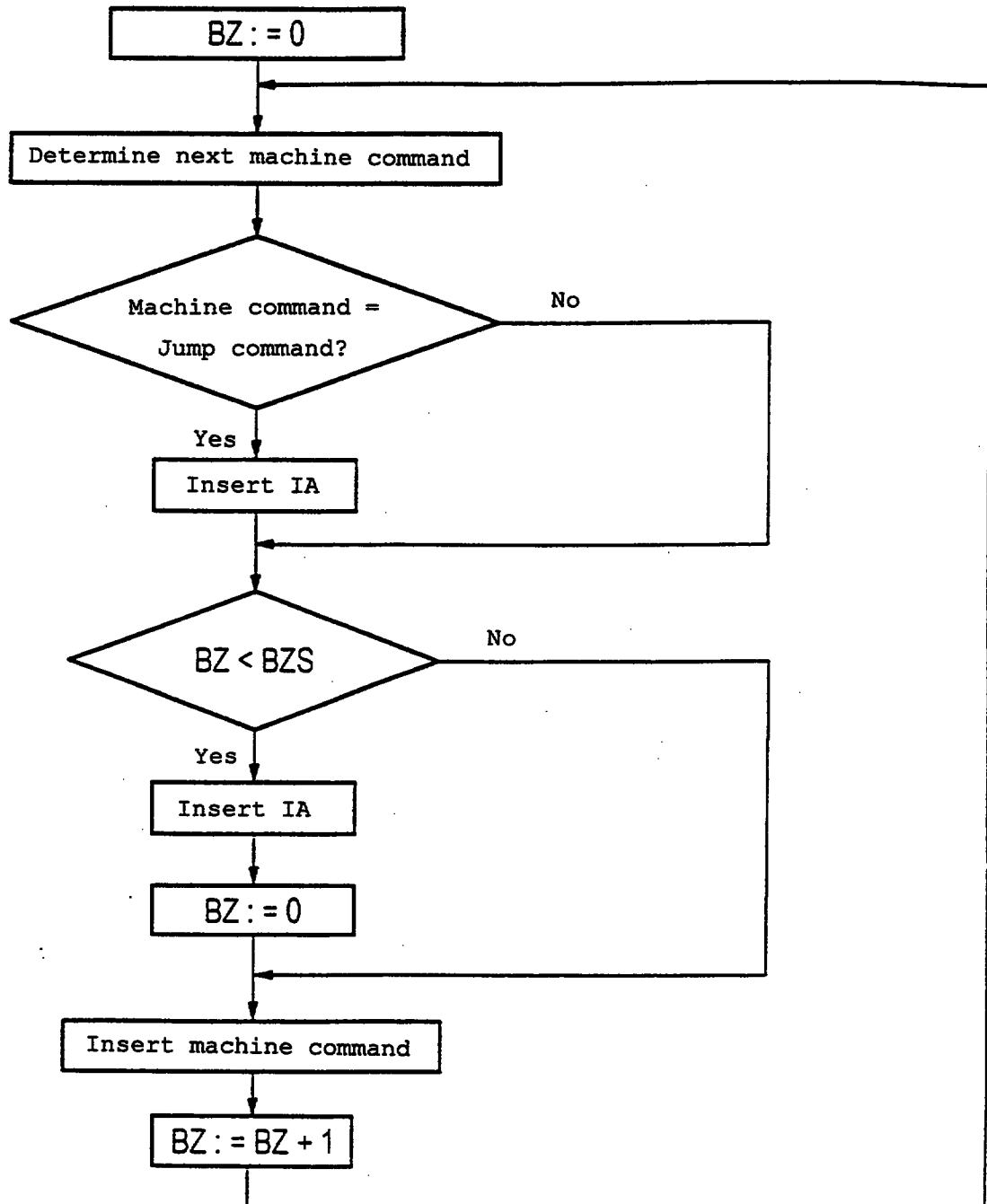


FIG 5

6/6

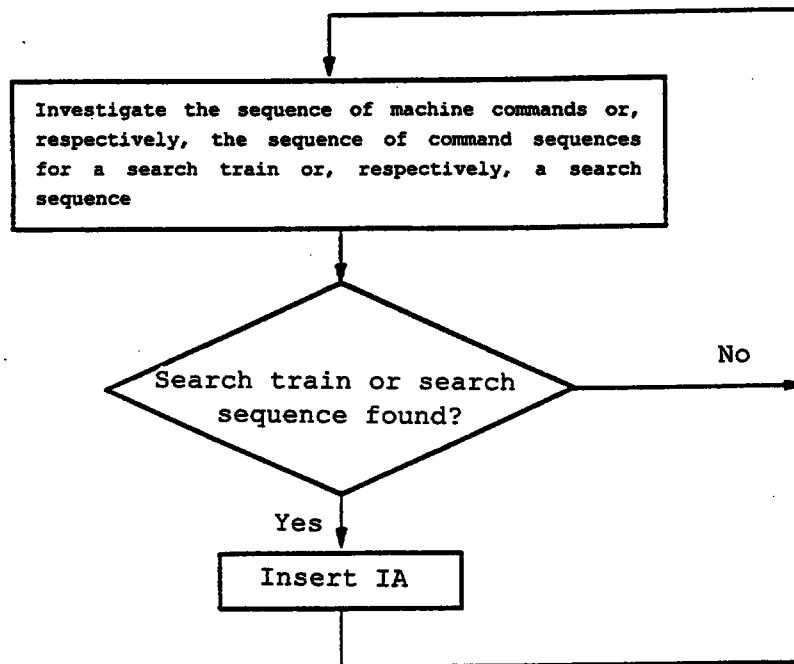


FIG 6

0-8812030