SIMATIC S5

Programmable Controllers SIMATIC S5-010W and K

Programming Instructions

Siemens Aktiengesellschaft

Subject to change without prior notice

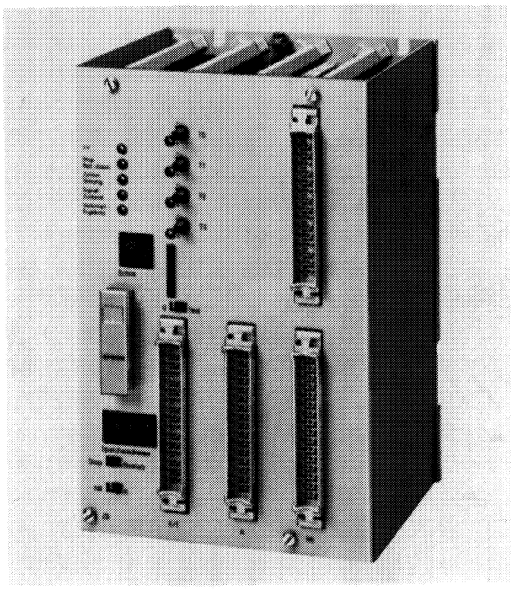
Order No.: GWA 4NEB 807 1071–02
Printed in the Federal Republic of Germany
AG 9830.3 FI 36 en

SIEMENS

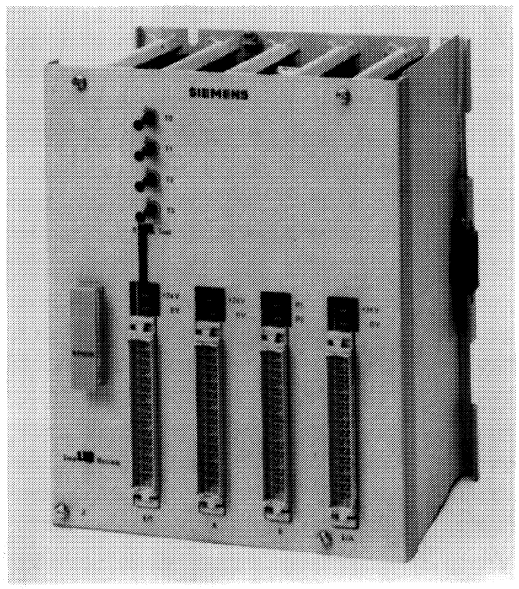
SIMATIC S5-010W and K Programmable Controllers

Programming Instructions

Order No. GWA 4NEB 807 1071-02



S5-010W



S5-010K

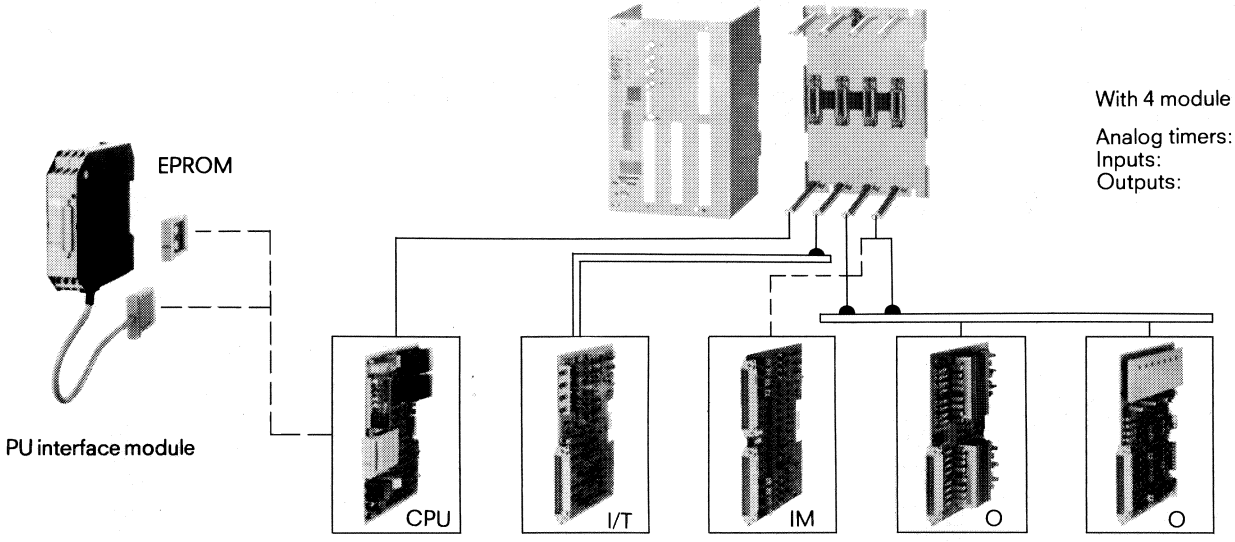
Contents **Page**

1. Introduction

1.1 Construction

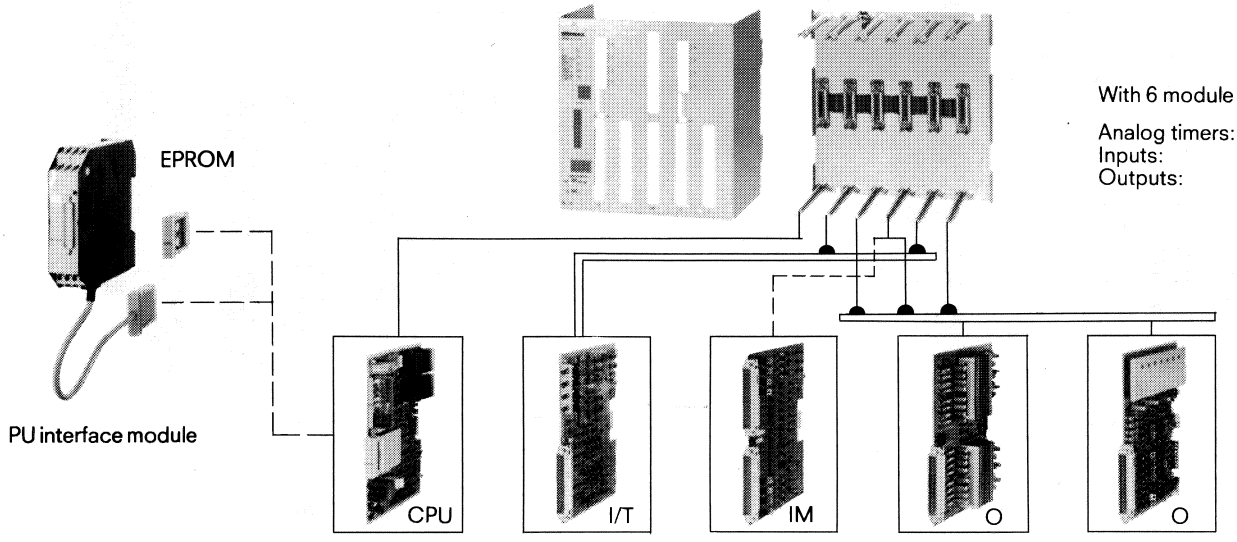
S5-010W, Module configuration

The CPU, the interface module and the input/timer modules are assigned to fixed specific locations. One of the two output modules can be used instead of the interface module. This permits the following maximum configuration:



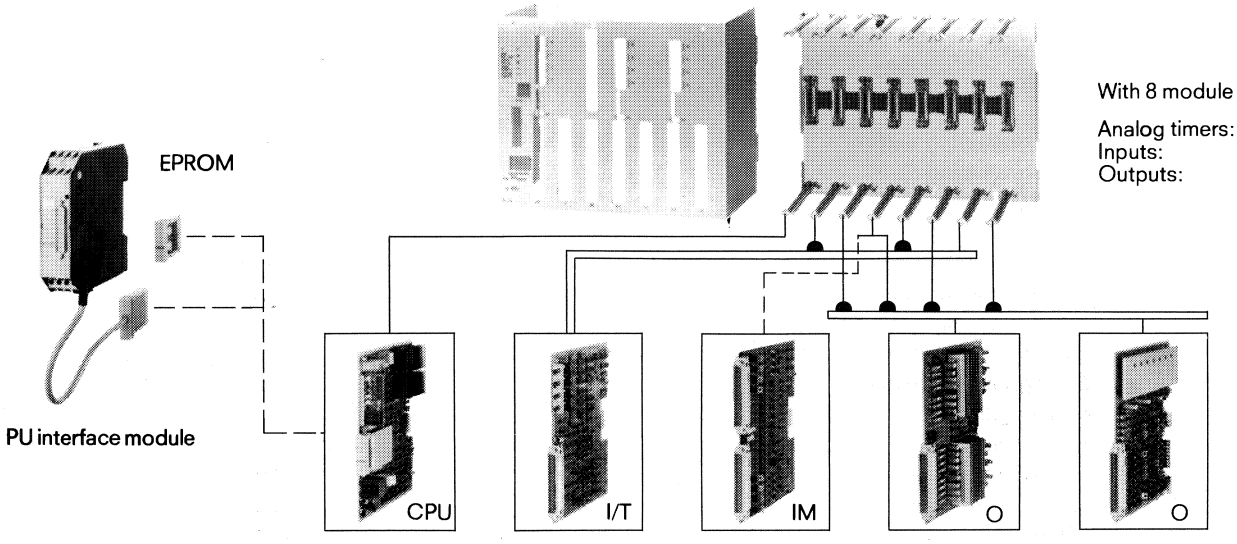
With 4 module locations:

Analog timers:	4
Inputs:	40
Outputs:	56



With 6 module locations:

Analog timers:	8
Inputs:	80
Outputs:	84



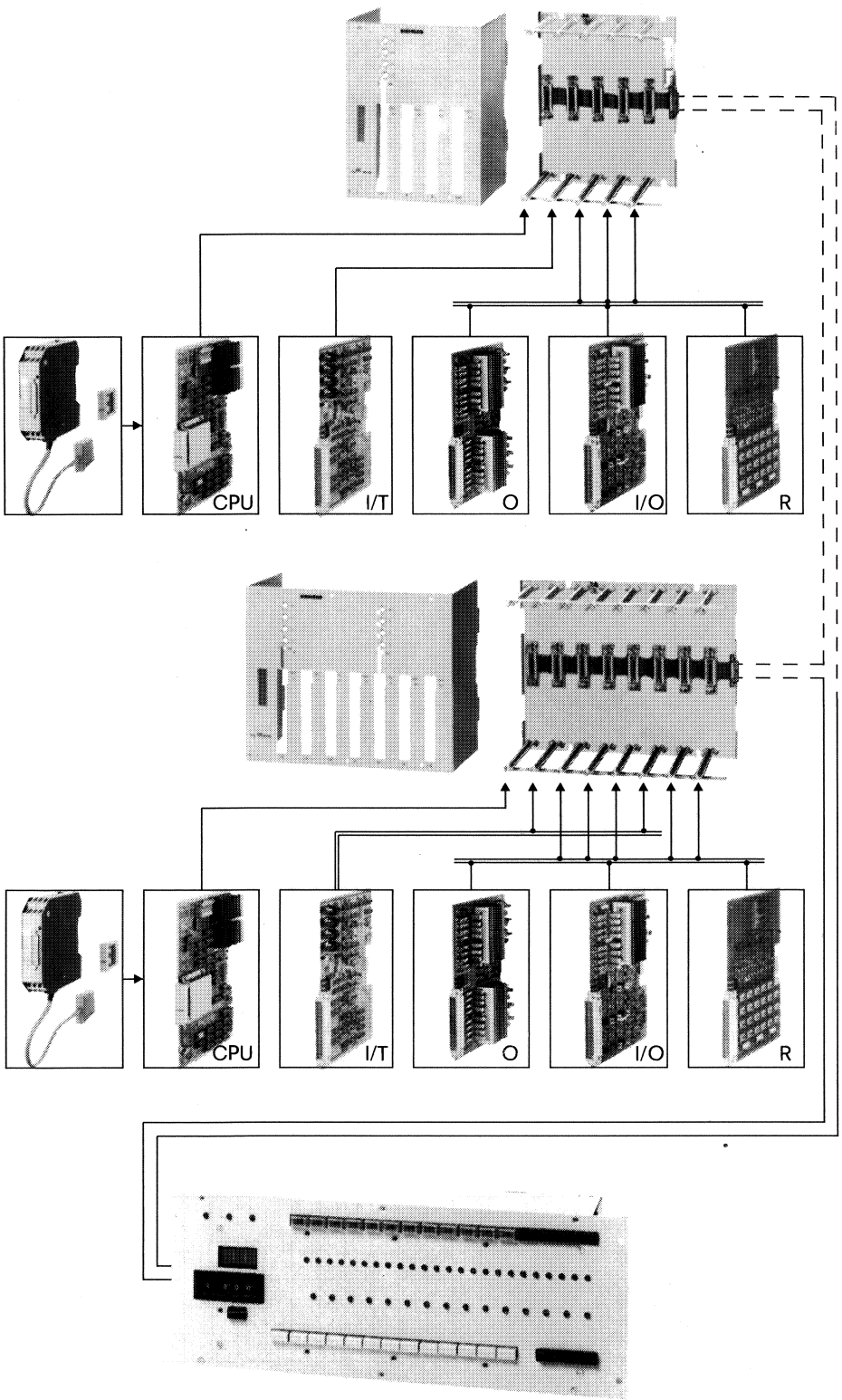
With 8 module locations:

Analog timers:	12
Inputs:	120
Outputs:	112

S5-010K, Module configuration

The CPU and the input/timer modules are assigned to specific fixed locations. The modules can be plugged into the other locations as required.
The following maximum configuration is thus possible:

It should be noted that, for example, a maximum configuration with relay modules restricts the possible number of inputs and solid-state outputs.
If an operator's panel is connected, the maximum configuration can be reduced (cf. Address decoding).



PC with 5 module locations

Analog timers:	4
Inputs:	40
Outputs:	48 solid-state 72 relays

(without operator's panel)

PC with 8 module locations

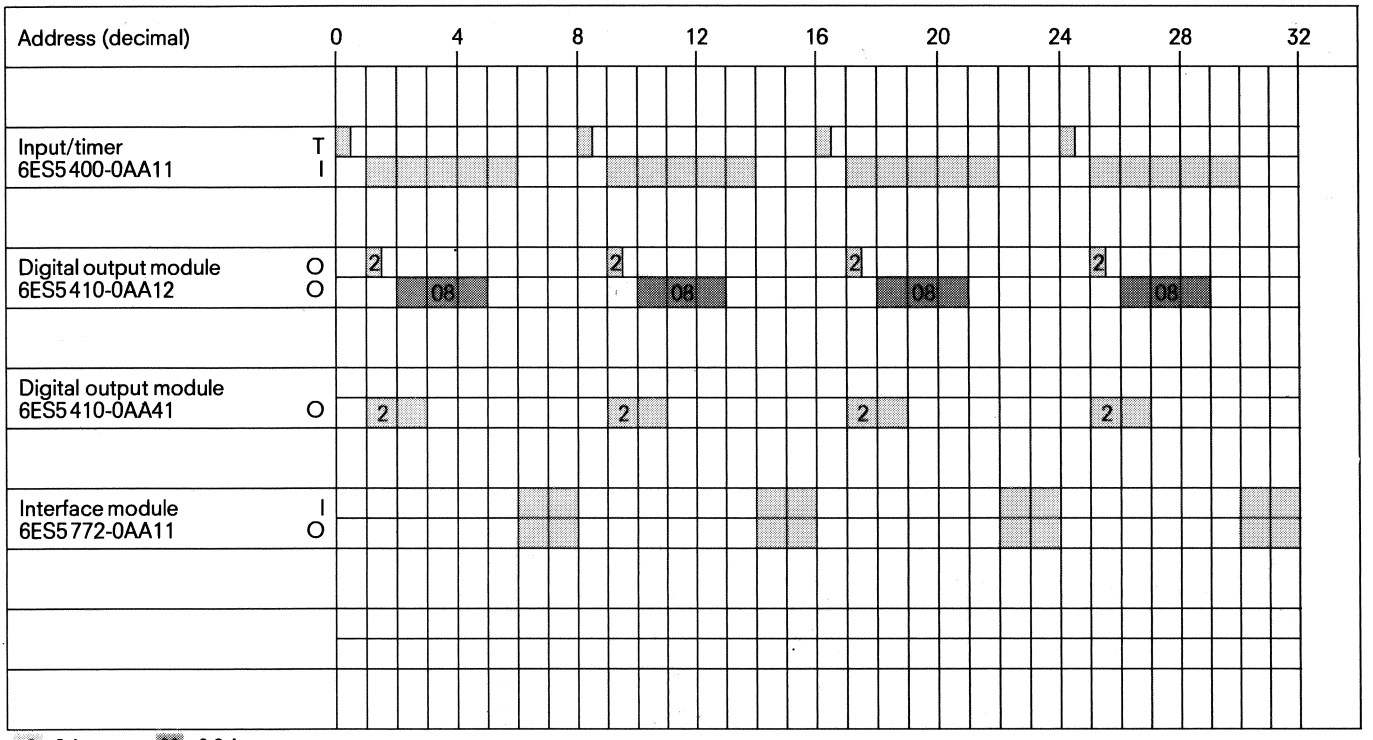
Analog timers:	8
Inputs:	80
Outputs:	80 solid-state 96 relays

(without operator's panel)

Operator's panel

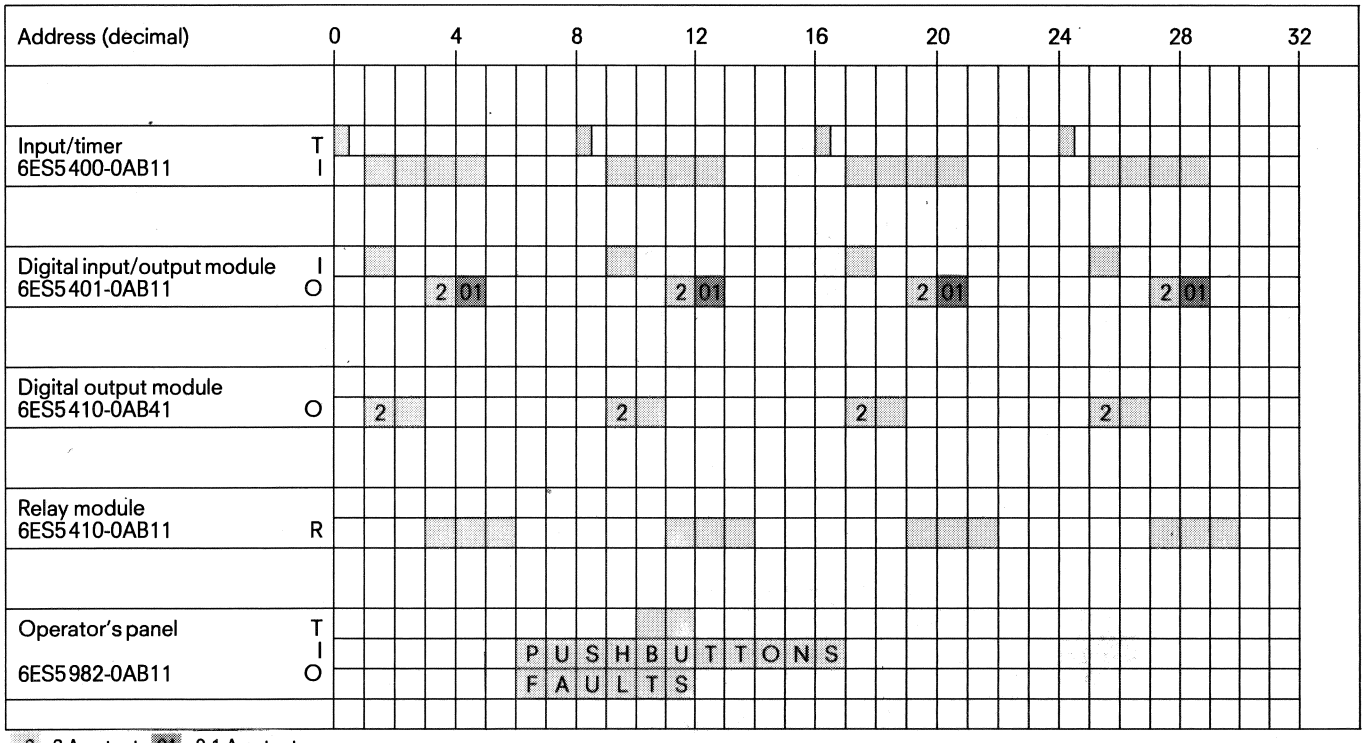
Digital timers:	15
Inputs:	32 pushbuttons
Outputs:	32 fault messages on 7-segment display
Additional display:	24 LEDs (24 V signals)

S5–010W
Peripheral I/O module addressing range (jumper-selectable)



2 2 A output 08 0.8 A output

S5–010K
Peripheral I/O module addressing range (jumper-selectable)



2 2 A output 01 0.1 A output

The desired control functions of the S5-010 programmable controller are determined by the program consisting of a number of individual STEP 5 statements.

The program statements are written consecutively into the locations of the memory from a programming unit.

During operation, the processor scans the memory cyclically, selecting the memory addresses one after the other. The statement read out of the memory location is interpreted and the corresponding operation executed.

When the end of the program is reached, i.e. the BE operation in the last memory location has been executed, the processor starts again from the beginning of the program.

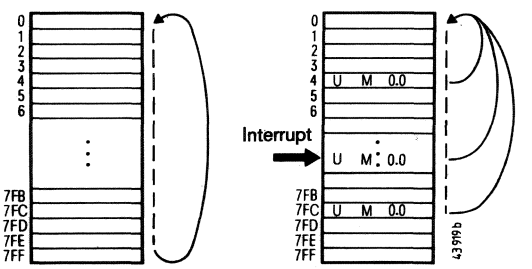
Example of how statements are processed:

```
A I 7.1
A I 5.3
S Q 1.0
```

The statement AI 5.3 causes the signal status of terminal 3 of the input module in location 5 to be scanned. The result of this scanning operation is then ANDed with the result of the previously executed statement.

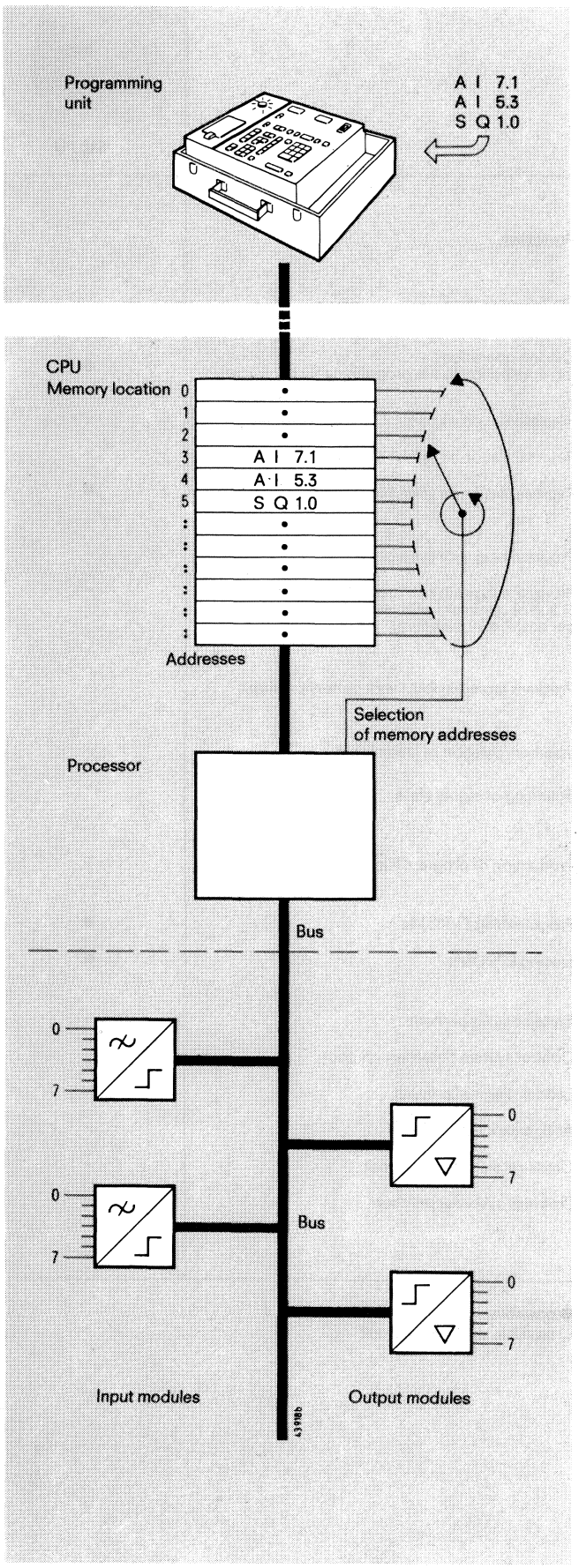
The result of this logical operation is temporarily stored and when an output statement, in this case SQ1.0, occurs, is made available at terminal "0" of the output module in location "1" as an output command.

The time required for one program run is referred to as the cycle time, and is determined by the number of statements and the time required for processing one statement. The controller of the S5-010W programmable controller requires 20 µs for processing one statement and the S5-010K 12 µs.
For a program containing 1K (1024) statements, the cycle time is approx. 20 ms in the case of the S5-010W PC and 12 ms in the case of the S5-010K PC, ignoring the propagation delays at the inputs. The response time can be shortened if necessary by means of interrupt processing. As soon as the signal state at one of the inputs changes, a group signal is sent to the CPU. The interrupt signal is evaluated with the AF 0.0 statement. Program processing is interrupted and recommences from the beginning of the program (address "0"). By programming the AF 0.0 statement several times within the program the response time can be shortened considerably.

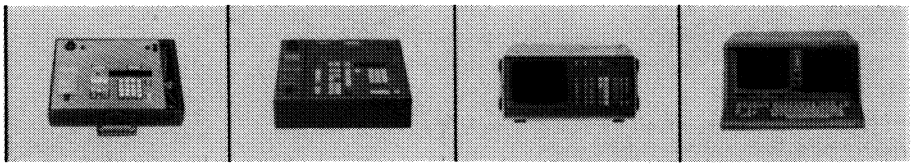


Cyclic program processing

Interrupt-driven processing



2. Programming units



Function	610 PU	630 PU	631 PU	670 PU
On-line operation (with connection to the programmable controller)	–	●		●
Off-line operation (no connection to the programmable controller)	●	●		●
Representation as control system flowchart	–	–		●
Representation as ladder diagram	–	●		●
Representation as statement list	●	●		●
Programming with symbolic addresses	–	–		●
Programming with comments (1 line of comment with max. 32 characters per network or segment)	–	–		●
Program stored in RAM with battery backup	–	●		●
Insertion/deletion of statements	–	●		●
Scanning of signal state	–	●		●
Production of program libraries	–	–		●
Programming EPROMs	●	●		●
Erasing EPROMs	●	●	O	●
Statement list printout	–	O		O
Control system flowchart printout	–	–		O
Ladder diagram printout	–	O		O
Assignment list printout	–	–		O
Cross-reference list printout	–	O		O
Program overview printout	–	–		O

● possible
O possible with optional unit

	PU 610	PU 630	PU 631	PU 670	PU 690
Programming the programmable controllers	S5-010 and S5-110A PCs	S5-010 and S5-110A and S5-130A, 130K PCs	S5-010 and S5-110A and S5-130A and 130K PCs	S5-010 S5-110A to S5-150K PCs	S5-010 S5-110A to S5-150K PCs
Program input	Function keys and numeric keys	Function keys and numeric keys	Function keys and numeric keys	Function keys and alphanumeric keys	Function keys and alphanumeric keys Punched cards via 670 PU
Program output	Display panel Hexadecimal	Display panel Statement list or ladder diagram; absolute parameters	Screen Statement list or ladder diagram; absolute parameters	Screen Statement list, ladder diagram or control system flow-chart; absolute or symbolic parameters	Screen Statement list or control system flow-chart; absolute or symbolic parameters
Documentation	–	PT80/TTY printer	PT80/TTY printer	PT80/TTY printer	Line printer
Programming	off-line	off-line on-line	off-line on-line	off-line on-line	off-line
Link to PC	–	parallel (3 m)	parallel (3 m)	parallel (3 m) for S5-010 S5-110A, 130A, 130K, PCs serial (up to 1000 m) for S5-130W, 150A, 150K, PCs	
Data medium	EPROM	EPROM	EPROM	EPROM Mini-floppy-disk	Cartridge Floppy disk
Special features	EPROM erasing facility	EPROM erasing facility, printer connection	No EPROM erasing facility, printer connection	EPROM erasing facility, printer connection 2 mini-floppy-disk drives	(4 floppy disk drives) Line printer, (connection for MODEM) Connection for 670 PU
Aids for system start-up and maintenance		Displays: Signal state RLO (binary) Forcing:[1] Outputs and flags	Displays: Signal state, RLO (binary and digital) Forcing:[1] Outputs and flags Generation of cross-reference lists	Displays: Signal state, RLO (binary and digital) Forcing:[1] Outputs and flags Generation of cross-reference lists	Generation of cross-reference lists, Assignment lists (sorted), Block documentation, Programm structure

[1]) Not for the SIMATIC S5-110A and S5-010 programmable controllers.

3. Programming

3.1 STEP 5 programming language

STEP 5 programming language

The STEP 5 programming language is an integral part of the SIMATIC S5 programmable controller system.

The operation set of this programming language makes it possible to program automation schemes, ranging from simple binary logic to complex digital processing.

The program can be written in three different methods of representation:
- Ladder diagram (LAD) with contact symbols similar to a schematic circuit diagram
- Statement list (STL) with mnemonic abbreviations
- Control system flowchart (CSF) with function symbols.

The three methods of representation correspond to the DIN draft 19239. The operation set for the SIMATIC S5-010 programmable controller is a subset of the total STEP 5 operation set.

The program of a PC consists of a number of individual statements. The basic component of the statement is the operation, which specifies the function the controller has to perform. In this connection, a distinction is made between the following:

Binary logic operations

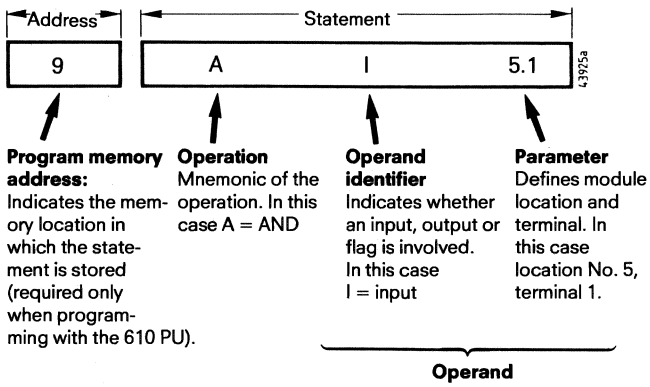
The signal statuses of inputs, outputs and flags are scanned. The result of the scan is ANDed or ORed with the result of a preceding logic operation. The new result is then stored.

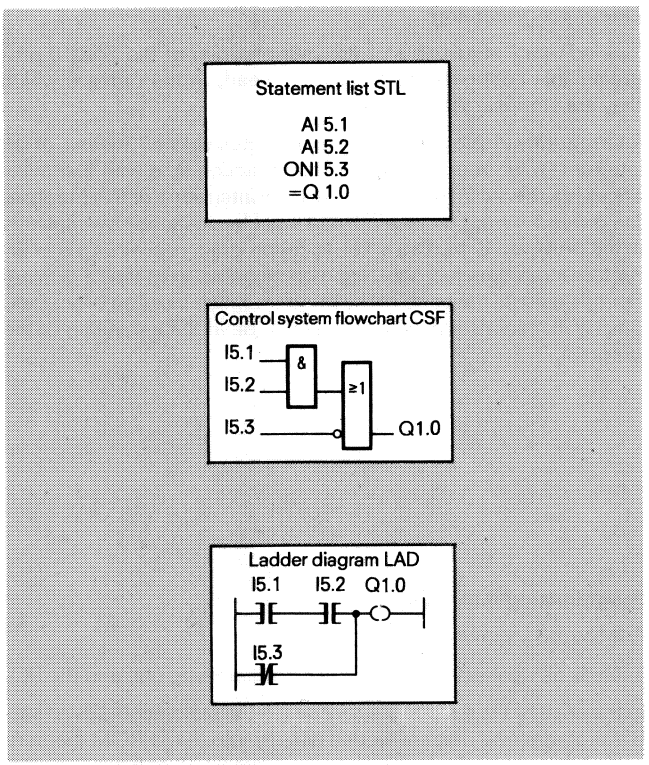
Memory operations

These are executed as a function of the result of previous scanning operations, and include operations with which outputs or flags can be set or reset.

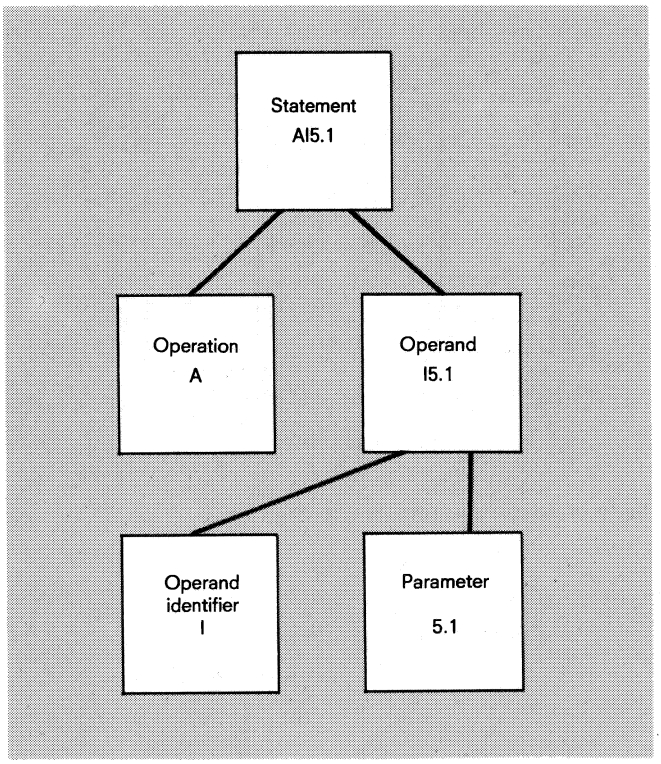
Organisational operations

These serve to influence program execution.





STEP 5 programming methods



Breakdown of a STEP 5 statement

The **STEP 5** programming language is used for writing user programs for programmable controllers of the SIMATIC S5 system. The program can be represented either as a statement list (STL), control system flowchart (CSF), or ladder diagram (LAD).

The **statement list (STL)** describes the automation task by means of mnemonic function designations.

The **control system flowchart (CSF)** is a graphic representation of the automation task, using symbols to DIN 40 700/DIN 40 719.

The **ladder diagram (LAD)** uses graphic circuit diagram symbols (American representation) to represent the automation task.

The types of representation are in keeping with DIN 19 239 (draft).

The type of representation to be used for programming depends on the relevant programming unit and the type of representation selected for that particular programming unit.

The programming unit converts the control system flowchart or ladder diagram into a statement list. In the memory of the programmable controller, the program is stored in MC 5 machine code.

The **statement** is the smallest STEP 5 program component. It comprises the following:

- Operation, i.e. "What is do be done?" and
- the operand, i.e. "What is it to be done with?"

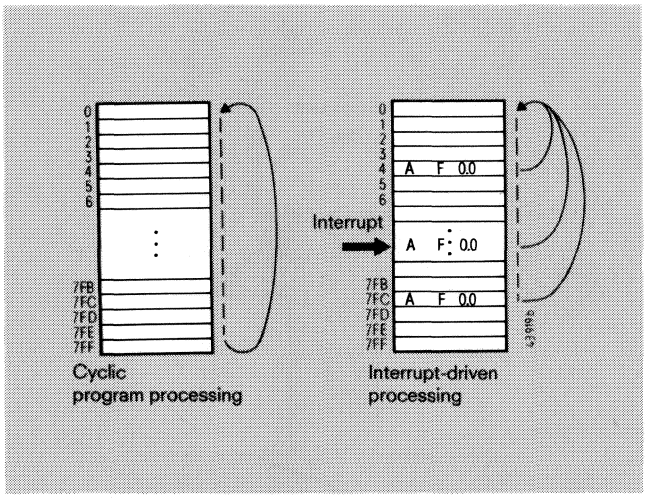
The operand comprises the following:

- Operand identifier (input, output etc.) and
- parameter.

The parameter identifiers the number of the input/output etc. addressed by the statement.

In the case of the 670 programming unit, the operand may include an absolute parameter, e.g. I 5.1, or a symbolic parameter, e.g. I "LS1". Programming is considerably simplified in the latter case, as the actual plant designation is directly used to describe the device connected to the input or output.

A statement takes up one word (2 bytes) in the program memory.

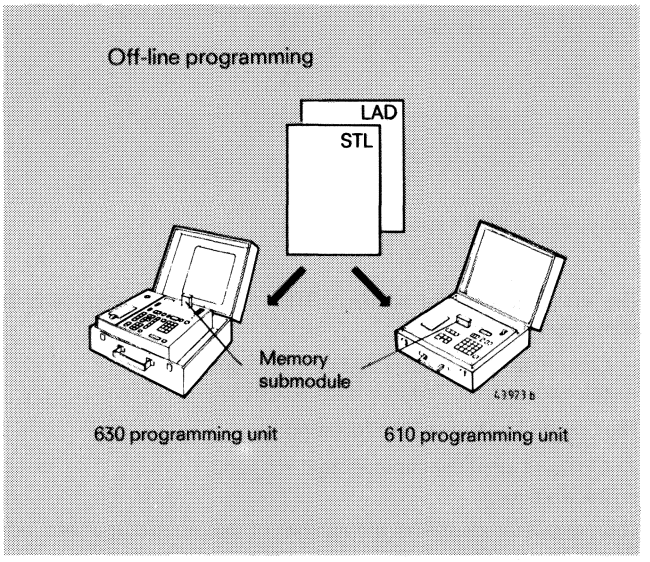


Structure and processing of a linear user program

Linear programming

The individual statements of the user program of the S5-010 programmable controller are processed linearly in the order in which they are stored in the memory.

Interrupt-driven program processing is possible by means of an interrupt (with group signal) if a short reaction time with fine tolerances is to be achieved in response to an interrupt. For this purpose, the group interrupt flag 0.0 (AF 0.0) must be scanned several times in the program. If this flag is set, linear program processing is interrupted and program processing recommences from the beginning. For this reason, the parts of the program which must be processed quickly in response to an interrupt must be at the beginning of the program.



Off-line programming

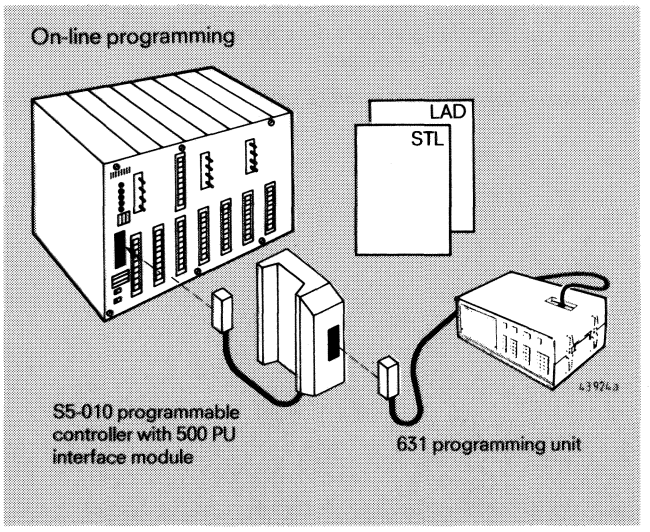
Programming methods

Off-line programming

There is no connection between the programming unit and the PC.

In the case of the 610 programming unit, the statements keyed into the programming unit are written directly into the EPROM memory submodule, which is plugged into the programming unit.

In the case of the 630, 631 and 670 programming units, the statements entered are first written into the RAM contained in the programming unit. The contents of this memory are then transferred to the EPROM submodule plugged into the programming unit.



On-line programming

On-line programming (not with the 610 programming unit)

The programming unit is hooked up to the PC through the 500 programming unit interface module.

The statements entered into the programming unit are stored initially in the RAM incorporated in the programming unit. This program is then processed by the PC, which is connected via the programming unit interface module. The functions of the PC can thus be tested and, if necessary, changed. Furthermore, on-line programming permits the display of signal states and the results of logical operations.

In order to transfer the program to the 910 EPROM submodule, the latter is plugged into the relevant receptacle on the programming unit and the contents of the programming unit memory (RAM) transferred to the 910 memory submodule. In this way, programs can easily be duplicated.

		Operation		Parameter range with programming unit 610	630, 670, 690	Program as Flowchart	Ladder diagram	Statement list
AND logic	Scan for "1" signal status							
	of an input	A	I	0.0...F.7	0.0...15.7	I 1.0 & Q 2.0 / **I 1.1**	I 1.0 **I 1.1** Q 2.0	A I 1.0 **A I 1.1** = Q 2.0
	of an output	A	Q	0.0...F.7	0.0...15.7			
	of a flag	A	F	0.1...3F.7	0.1...63.7			
	Scan for "0" signal status							
	of an input	AN	I	0.0...F.7	0.0...15.7	I 1.0 & Q 2.0 / **I 1.1**	I 1.0 **I 1.1** Q 2.0	A I 1.0 **AN I 1.1** = Q 2.0
	of an output	AN	Q	0.0...F.7	0.0...15.7			
	of a flag	AN	F	0.1...3F.7	0.1...63.7			
OR logic	Scan for "1" signal status							
	of an input	O	I	0.0...F.7	0.0...15.7	I 1.0 ≥1 Q 2.0 / **I 1.1**	I 1.0 Q 2.0 / **I 1.1**	O I 1.0 **O I 1.1** = Q 2.0
	of an output	O	Q	0.0...F.7	0.0...15.7			
	of a flag	O	F	0.1...3F.7	0.1...63.7			
	Scan for "0" signal status							
	of an input	ON	I	0.0...F.7	0.0...15.7	I 1.0 ≥1 Q 2.0 / **I 1.1**	I 1.0 Q 2.0 / **I 1.1**	O I 1.0 **ON I 1.1** = Q 2.0
	of an output	ON	Q	0.0...F.7	0.0...15.7			
	of a flag	ON	F	0.1...3F.7	0.1...63.7			
Setting/resetting operations	A "1" signal appears at the output (or flag) if the logic condition is satisfied; a "0" appears if the condition is not satisfied	= =	Q F	0.0...F.7 0.1...3F.7	0.0...15.7 0.1...63.7	I 1.0 & Q 2.0 / **I 1.1**	I 1.0 **I 1.1** Q 2.0	A I 1.0 A I 1.1 = **Q 2.0**
	The output (or flag) is set to "1" (stored) if the logic condition is satisfied; if the condition is not satisfied, the signal status does not change	S S	O F	0.0...F.7 0.1...3F.7	0.0...15.7 0.1...63.7	Q 2.0 / I 1.0 & S / **I 1.1** / E 1.2 R Q	I 1.0 **I 1.1** Q 2.0 / E 1.2 A 2.0	A I 1.0 A I 1.1 **S Q 2.0**
	The output (or flag) is set to "0" (stored) if the logic condition is satisfied; if the condition is not satisfied, the signal status does not change	R R	Q F	0.0...F.7 0.1...3F.7	0.0...15.7 0.1...63.7	Q 2.0 / I 1.0 & S / **I 1.1** / E 1.2 R Q	I 1.0 **I 1.1** Q 2.0 / E 1.2 A 2.0	A I 1.2 **R Q 2.0**
Interrupt scan		A	F	0.0	0.0	On a signal change from "0"→"1" or "1"→"0" on the input module with group signal. Automatic jump to the beginning of the program.		

		Operation	Parameter range with programming unit		Remarks
			610	630, 631, 670	
Organisational operations	No operation	NOP 0	–	–	No operations are carried out. This operation is used to overwrite the contents of a memory location.
	No operation	NOP 1	–	–	No operations are carried out. This operation is used to keep a memory location free for patching or expanding programs.
	Block end	BE	–	–	End of program Jump to beginning of program.
	Conditional end of block	BEC	–	–	End of program, depending on the result of the logic operation. If result is "1", jump to beginning of program; if "0", no effect, but RLO set to "1".

Note:

The 670 PU inserts screen statements into the user program. This has the following effect:
The maximum number of statements (depending on the memory submodule used) is reduced approximately by the number of rungs programmed.
Programs in 4K EPROMS loaded by other PUs can only be read out by the 670 PU if free space is available for the rungs programmed (i.e. 4K statements minus the number of rungs programmed).

Timer functions

There are no timer processing operations in the operation set of the S5-010 programmable controller. A timer module is therefore started with output operations and scanned with input operations.

Starting a timer:

SQ	If the result of the logic operation (RLO) is "1", the timer is started. Before the timer function can be started again, the timer must be reset with RQ.
=Q	If the result of the logic operation (RLO) is "1", the timer is started. Before the timer function can be started again, the operation =Q must be processed at least once with RLO = "0".

Scanning a timer:

A I, O I	Scan results in "1" if the timer is running
AN I, ON I	Scan results in "1" if the timer is not running or has already run.

Scanning the signal state at the input of a timer:

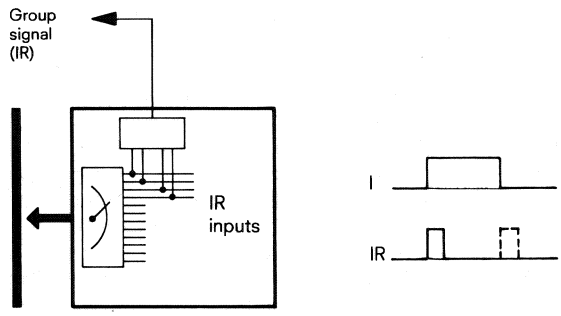
A Q, OQ	Scan: timer started (output flag set)
AN Q, ON Q	Scan: timer reset or not started (output flag not set)

Example:
A Q
AN I
Meaning: Timer has been started and has run down

With the timer module, the time is set roughly by sliding switches on the frontplate of the module and is finely adjusted with a potentiometer, also on the frontplate. Fine setting is also possible by means of an external potentiometer (not for the S5-010K). If the CPU is in stop status, the timers can be activated and set with the "TEST" switch (very short times via the outputs X5. . .X8, trigger signal X9).

For the programming of on/off delays, clock generators and times, see the programming examples.

Interrupt processing



Input module with group signal inputs

A STEP 5 statement is processed by the S5-010W programmable controller in 20 µs. If the program is 1K statements long, therefore, the cycle time is 20 ms. With an input delay of 6 ms, this results in a maximum response time of 26 ms, which is more than adequate for the normal applications of this controller. If this response time is too long, it can be considerably shortened by using an input module with group signal inputs. The S5-010K's cycle time for 1K statements is only 12 ms. The response time for a program of this length is 18 ms. Response times can be shortened in this case, too, by means of interrupt inputs.
As soon as the signal state at one of these inputs changes from "0" to "1" (or from "1" to "0"), the module sends a group signal (JR) to the CPU and sets the group signal flip-flop. This flip-flop is scanned with the AF 0.0 statement. If it is "1", cyclic processing is interrupted, recontinued at the beginning of the program and the group signal flip-flop is automatically reset. If the interrupt is to be processed immediately, the interrupt service routines must be located at the beginning of the program. In this case, the interrupt inputs are scanned and the corresponding response initiated.

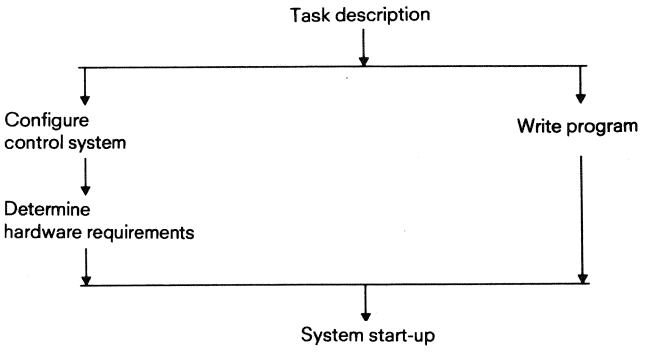
Response time

The response time can be shortened by repeatedly scanning the group signal flip-flop with the AF 0.0 statement during the entire program sequence. The shorter the interval between the individual scans, the shorter the response time.
If the AF 0.0 statement is programmed in every hundredth memory location, the resulting maximum response time is 8 ms (100 statements x 20 µs + input delay of 6 ms). The response time is kept constant by inserting the AF 0.0 statement in the program at regular intervals.

Retentive flags (relay equivalents) (S5-010W only)

The current state of a program sequence is stored in the CPU in the form of flags and output flags (RAM).
For a **cold restart** the switch on the CPU must be set to "NR". All flags referenced in the program with set statements are loaded with "0".
If this switch is set to "R", the last state of all retentive flags used is stored if the program is interrupted (power failure, moving mode switch from RUN to STOP to RUN) and the flags are unaffected on restart. All non-retentive flags and output flags used are reset.
Note: – Since, in the event of a power failure, the input signals usually disappear before the CPU is switched off, the flag image can be distorted. It is helpful to avoid signals which are active low (ANI, ONI). The flag locations which are to be retentive may only be addressed with the SF and RF statements.
If the "STOP" LED lights up on power recovery, the backup battery voltage is too low and the states of the retentive flags have not been stored. By moving the mode switch from RUN to STOP and back to RUN, the controller is ready for operation (cold restart).
The backup battery must be replaced.

Design recommendations



Task definition

Determine the tasks to be handled by the PC
Compile a ladder diagram or statement list
Compile a list of sensors and actuators

Hardware requirements

Select modules

Input/timer modules
Number depends on number of sensors

Output modules
Number depends on number of actuators

Select the size of the memory submodule: estimate length of program; allow approx. 15 memory locations per input and output.

Program

Compile a statement list (STL) or ladder diagram (LAD) for programming with the 610, 630, 631 or 670 programming unit.
The preprinted forms for statement lists (STL), ladder diagrams (LAD) or control system flowcharts (CSF) appended to these programming instructions are a valuable aid and can be copied (DIN A4). Preprinted forms in DIN A3 format are available with the following Order Nos.:

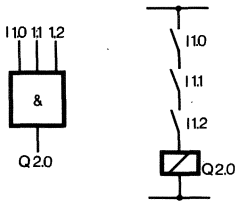
Statement list	Paper	(3) E 88310-V244-L92
Ladder diagram	Foil	S 6360
Ladder diagram	Paper	S 6361
Control system flowchart	Foil	S 6362
Control system flowchart	Paper	S 6363

It is advisable to arrange and list the inputs, outputs and flags defined in the program, using one of the appended forms (which may also be copied).

The programming examples can be run through for checking with all controllers provided they have a CPU, an input/timer module (coding jumpers not inserted) and an output module (coding jumpers not inserted).

Binary logic

AND logic

Original	STEP 5 representation		
	Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 A I 1.1 A I 1.2 = Q 2.0	I I IE 1.0 E 1.1 E 1.2 A 2.0 I +---] [---+---] [---+---] [---+---( )--I I I	E 1.0 ---! & ! E 1.1 ---! ! E 1.2 ---! !-- A 2.0 !___!

A "1" signal appears at output Q2.0 when all the inputs have "1" signals simultaneously.
A "0" signal appears at output Q2.0 if at least one of the inputs has "0" signal.
The number and the sequence of the scans are irrelevant.

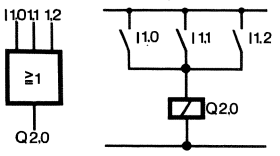
E ≙ INPUT
A ≙ OUTPUT

5. Programming examples

5.1 Binary logic

Binary logic (continued)

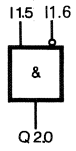
OR logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	O I 1.0 O I 1.1 O I 1.2 = Q 2.0	I IE 1.0 A 2.0 +---] [---+---------+---( )--I I I I IE 1.1 I +---] [---+ I I I I IE 1.2 I +---] [--- + I I I	E 1.0 ---I>=1I E 1.1 ---I I E 1.2 ---I I-- A 2.0 I___I

A "1" signal appears at output Q2.0 if at least one of the inputs has a "1" signal.
A "0" signal appears at ouput Q2.0 if all the inputs have "0" signals simultaneously.
The number and the sequence of the scans are irrelevant.

E ≙ INPUT
A ≙ OUTPUT

Scanning for "0" signal status

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.5 AN I 1.6 = Q 2.0	I IE 1.5 E 1.6 A 2.0 +---] [---+---]/[---+---------+---( )--I I I I I	E 1.5 ---I & I E 1.6 --0I I-- A 2.0 I___I

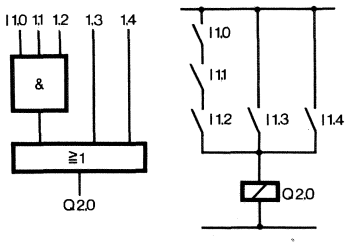
A "1" signal appears at output Q2.0 only when input I1.5 has "1" signal (NO contact operated) and input I1.6 has "0" signal (NC contact not operated).

E ≙ INPUT
A ≙ OUTPUT

Binary logic (continued)

AND-before-OR logic

Original



STEP 5 representation

Statement list STL

```
A  I  1.0
A  I  1.1
A  I  1.2
O  I  1.3
O  I  1.4
=  Q  2.0
```

Ladder diagram LAD

```
I                                                  I
IE  1.0    E  1.1    E  1.2    A  2.0              I
+---] [---+---] [---+---] [---+---( )--I
I                                    I             I
IE  1.3                              I             I
+---] [---+----------+----------+                  I
I                                    I             I
IE  1.4                              I             I
+---] [---+----------+----------+                  I
I                                                  I
```

Control system flowchart CSF

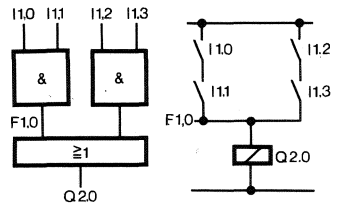
```
E  1.0   ---! & !
E  1.1   ---!   !
F  1.2   ---!   !------!>=1!
             !   !     !   !
             !___!     !   !
                 E  1.3 ---!   !
                 E  1.4 ---!   !-- A  2.0
                           !___!
```

A "1" signal appears at output Q2.0 when either the output of the AND gate is "1" or one of the inputs of the OR gate has a "1" signal. The AND logic must be programmed before the OR logic.

E ≙ INPUT
A ≙ OUTPUT

AND-before-OR gate

Original



STEP 5 representation

Statement list STL

```
A  I  1.0
A  I  1.1
=  F  1.0
A  I  1.2
A  I  1.3
O  F  1.0
=  Q  2.0
```

Ladder diagram LAD

```
I                                                  I
IE  1.0    E  1.1                M  1.0            I
+---] [---+---] [---+----------+---( )--I
I                                                  I

I                                                  I
IE  1.2    E  1.3                A  2.0            I
+---] [---+---] [---+----------+---( )--I
I                    I                             I
IM  1.0              I                             I
+---] [---+----------+                             I
I                                                  I
```

Control system flowchart CSF

```
E  1.0   ---! & !
E  1.1   ---!   !-- M  1.0
             !___!

E  1.2   ---! & !
E  1.3   ---!   !------!>=1!
             !___!     !   !
         M  1.0  ---!   !-- A  2.0
                    !___!
```

A "1" signal appears at output Q2.0 when the output of the least one of the AND gates is "1".
All AND logic operations, except the last one, must be buffered.

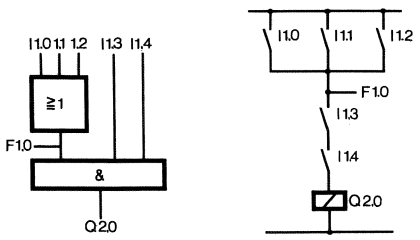
E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

5. Programming examples

5.1 Binary logic

Binary logic (continued)

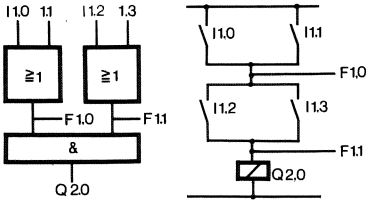
OR-before-AND logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	O I 1.0 O I 1.1 O I 1.2 = F 1.0 A I 1.3 A I 1.4 A F 1.0 = Q 2.0	I IE 1.0 M 1.0 I +---] [---+---------+---( )--I I I I IE 1.1 I I +---] [---+ I I I I IE 1.2 I I +---] [---+ I I I IM 1.0 E 1.3 E 1.4 A 2.0 I +---] [---+---] [---+---] [---+---( )--I I I I I	E 1.0 ---!>=1! E 1.1 ---! ! E 1.2 ---! !-- M 1.0 !___! E 1.3 ---! & ! E 1.4 ---! ! M 1.0 ---! !-- A 2.0 !___!

Flags must always be set in connection with OR-before-AND operations!

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

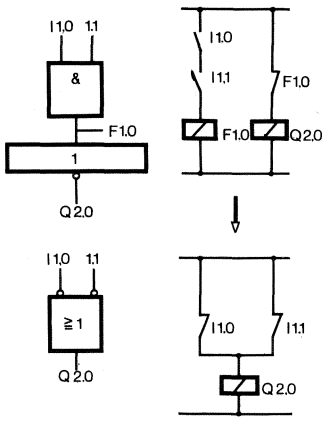
OR-before-AND logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	O I 1.0 O I 1.1 = F 1.0 O I 1.2 O I 1.3 = F 1.1 A F 1.0 A F 1.1 = Q 2.0	I IE 1.0 M 1.0 I +---] [---+---------+---( )--I I I I IE 1.1 I I +---] [---+ I I I I I IE 1.2 M 1.1 I +---] [---+---------+---( )--I I I I IE 1.3 I I +---] [---+ I I I I I IM 1.0 M 1.1 A 2.0 I +---] [---+---] [---+---------+---( )--I I I I I	E 1.0 ---!>=1! E 1.1 ---! !-- M 1.0 !___! E 1.2 ---!>=1! E 1.3 ---! !-- M 1.1 !___! M 1.0 ---! & ! M 1.1 ---! !-- A 2.0 !___!

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

Binary logic (continued)

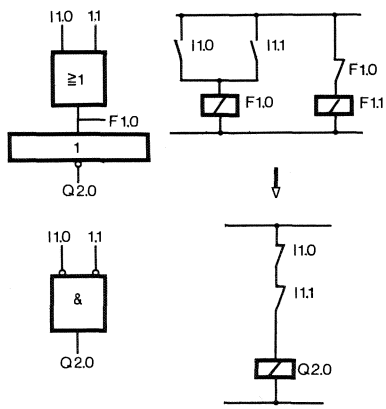
NAND logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 A I 1.0 = F 1.0 AN F 1.0 = Q 2.0 ON I 1.0 ON I 1.1 = Q 2.0	I IE 1.0 E 1.1 M 1.0 I +---] [---+---] [---+---------+---( )--I I I I IM 1.0 A 2.0 I +---]/[---+---------+---( )--I I I ↓ I IE 1.0 A 2.0 I +---]/[---+---------+---( )--I I I I IE 1.1 I I +---]/[---+ I I I	E 1.0 ---! & ! E 1.1 ---! !-- M 1.0 !___! M 1.0 --0! & !-- A 2.0 !___! ↓ E 1.0 --0!>=1! E 1.1 --0! !-- A 2.0 !___!

A "0" signal only appears at output Q2.0 if all inputs have a "1" signal.

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

NOR logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	O I 1.0 O I 1.1 = F 1.0 AN F 1.0 = Q 2.0 AN I 1.0 AN I 1.1 = Q 2.0	I I IE 1.0 M 1.0 +---] [---+---------+---( )--I I I IE 1.1 I I +---] [---+ I I I I I IM 1.0 A 2.0 +---]/[---+---------+---( )--I I I ↓ I I IE 1.0 E 1.1 A 2.0 +---]/[---+---]/[---+---------+---( )--I I I I I	E 1.0 ---!>=1! E 1.1 ---! !-- M 1.0 !___! M 1.0 --0! & !-- A 2.0 !___! E 1.0 --0! & ! E 1.1 --0! !-- A 2.0 !___!

A "0" signal appears at output Q2.0 as soon as at least one input has a "1" signal.

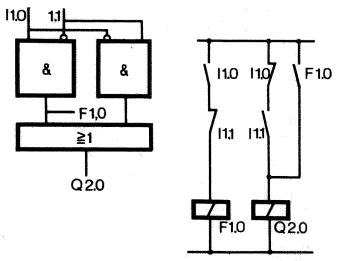
E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

5. Programming examples

5.1 Binary logic

Binary logic (continued)

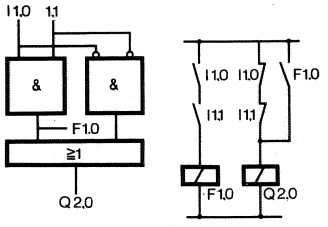
Exclusive OR logic

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 A I 1.1 = F 1.0 AN I 1.0 AN I 1.1 O F 1.0 = Q 2.0	I IE 1.0 E 1.1 M 1.0 +---] [---+---]/[---+---------+---( )--I I I IE 1.0 E 1.1 A 2.0 +---]/[---+---] [---+---------+---( )--I I I I IM 1.0 I I +---] [---+---------+ I I I	E 1.0 ---! & ! E 1.1 --O! !-- M 1.0 !___! E 1.0 --O! & ! E 1.1 ---! !-----!>=1! ! ___! ! ! M 1.0 ---! !-- A 2.0 !___!

Output Q2.0 has a "1" signal if both inputs have **different** signals.

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

Exclusive NOR logic

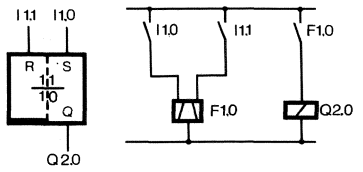
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 AN I 1.1 = F 1.0 AN I 1.0 A I 1.1 O F 1.0 = Q 2.0	I IE 1.0 E 1.1 M 1.0 +---] [---+---] [---+---------+---( )--I I I IE 1.0 E 1.1 A 2.0 +---]/[---+---]/[---+---------+---( )--I I I I IM 1.0 I I +---] [---+---------+ :BE I I	E 1.0 ---! & ! E 1.1 ---! !-- M 1.0 !___! E 1.0 --O! & ! E 1.1 --O! !-----!>=1! ! ___! ! ! M 1.0 ---! !-- A 2.0 !___!

Output Q2.0 has a "1" signal if both inputs have the **same** signals.

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

Setting/resetting functions

RS flip-flop for stored signal output

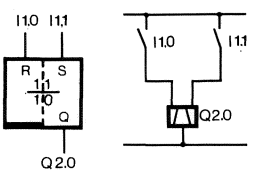
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 S Q 2.0 A I 1.1 R Q 2.0	I IE 1.0 M 1.0 I +---] [---+---+---+---(S)--I I I IE 1.1 M 1.0 I +---] [---+---+---+---(R)--I I I IM 1.0 A 2.0 I +---] [---+---+---+---( )--I	M 1.0 E 1.0 --!S ! E 1.1 --!R Q!- A 2.0

The flip-flop is set when a "1" signal is applied to input I1.0. If the signal at input I1.0 changes to "0", the status remains unchanged, i.e. the signal is stored.
The flip-flop is reset when a "1" signal is applied to input I1.1.
If the signal at input I1.1 changes to "0", this status is still retained.

The last program scanning operation (in this case A I1.1) is effective during the processing of the remaining program if a set signal (input I1.0) and a reset signal (input I1.1) are simultaneously applied.
Setting and resetting have no effect on outputs in programs which are up to 20 statements long (peripheral delay). With longer programs, the output is clocked in accordance with the setting/resetting time relationship.
A prerequisite for correct execution with simultaneous setting and resetting conditions is a program at least 100 statements in length.

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

RS flip-flop with flags

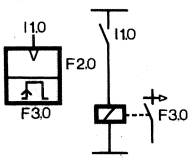
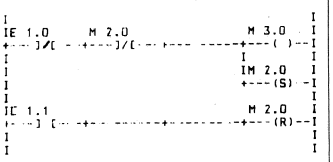
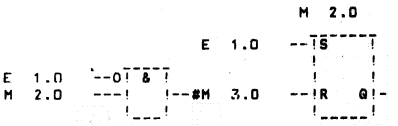
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 S F 1.0 A I 1.1 R F 1.0 A F 1.0 = Q 2.0	I IE 1.0 A 2.0 I +---] [---+---+---+---(S)--I I I IE 1.1 A 2.0 I +---] [---+---+---+---(R)--I I I	A 2.0 E 1.0 --!S ! E 1.1 --!R Q!-

The flip-flop is set when a "1" signal is applied to input I1.0. If the signal at input I1.0 changes to "0", the status remains unchanged, i.e. the signal is stored.
The flip-flop is reset when a "1" signal is applied to reset input I1.1.
If the signal at input I1.1 changes to "0", this status is retained. If a set signal (input I1.0) and a reset signal (input I1.1) are applied simultaneously, the reset signal dominates.

E ≙ INPUT
A ≙ OUTPUT

Setting/resetting functions (continued)

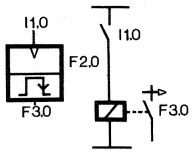
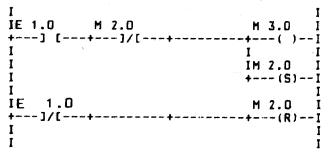
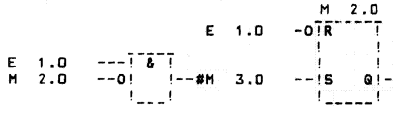
Pulse edge evaluation ⊼

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 AN F 2.0 = F 3.0 S F 2.0 AN I 1.0 R F 2.0		

If input I1.0 has a "0" signal, pulse edge flag F2.0 is always reset. If the input signal changes from "0" to "1", flag F3.0 is set **once per cycle.** In the next cycle, the conditions for the logic operation A I1.0, ANF2.0 are no longer satisfied.

E ≙ INPUT
M ≙ FLAG

Pulse edge evaluation ⊽

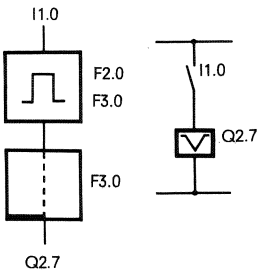
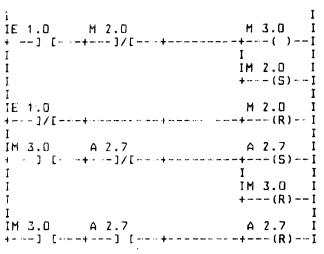
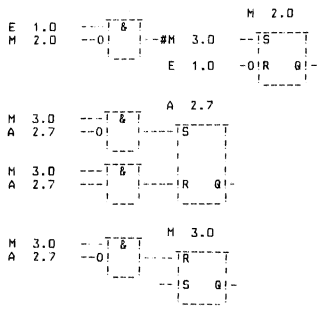
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	AN I 1.0 A F 2.0 = F 3.0 R F 2.0 A I 1.0 S F 2.0		

Processing is analogous to the evaluation of the positive-going edge.

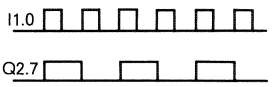
E ≙ INPUT
M ≙ FLAG

Setting/resetting functions (continued)

Binary scaler (with positive-going edge)

Original	STEP 5 representation		
	Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 AN F 2.0 = F 3.0 S F 2.0 AN I 1.0 R F 2.0 A F 3.0 AN Q 2.7 S Q 2.7 R F 3.0 A F 3.0 A Q 2.7 R Q 2.7		

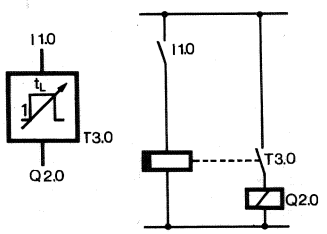
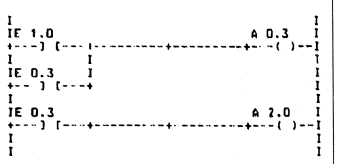
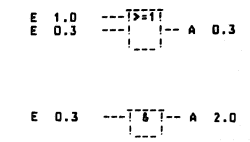
If output Q2.7 is set after recognition of the pulse edge, pulse flag F2.0 must be reset at once to prevent immediate resetting of the output.



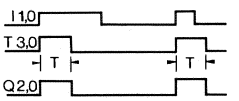
E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

Timer functions

Pulse (contracting and stretching a pulse)

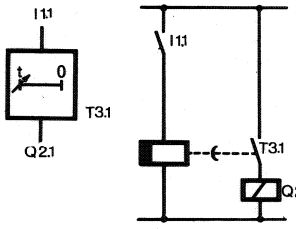
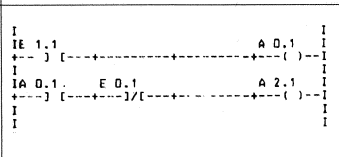
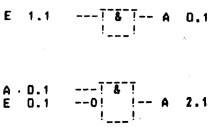
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	O I 1.0 O I 0.3 = Q 0.3 A I 0.3 = Q 2.0		

If RLO = 1, the timer is started by set statements.
If RLO = 0, it is reset.
The scanning operations AI and OI produce "1" signals as long as the timer is running.
Returning of the timer output to the input (OR) results in pulse stretching

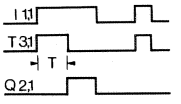


E ≙ INPUT
A ≙ OUTPUT

"On" delay

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.1 = Q 0.1 A Q 0.1 AN I 0.1 = Q 2.1		

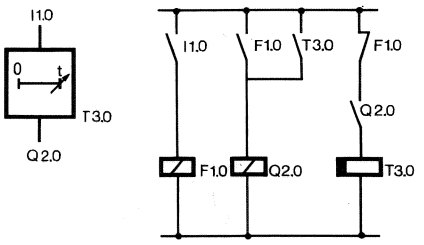
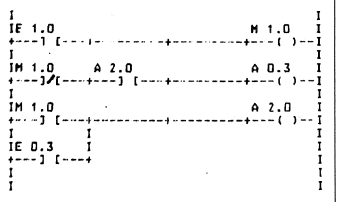
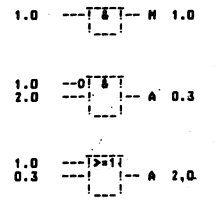
The timer is started, and also reset, via input I1.1.
Output Q2.1 is set after the time has elapsed, as long as input I1.1 still has a "1" signal.



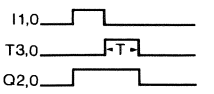
E ≙ INPUT
A ≙ OUTPUT

Timer functions (continued)

"Off" delay

Original	STEP 5 representation		
	Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	A I 1.0 = F 1.0 AN F 1.0 A Q 2.0 = Q 0.3[1)] O F 1.0 O I 0.3[2)] = Q 2.0		

In the case of the "off" delay, timer 3.0 only starts when input 1.0 changes to "0" (An I1.0 scanning operation).
The output is set when input 1.0 has a "1" signal or when T3.0 is running.
If input 1.0 has a "1" signal again before time 3.0 has elapsed, the timer is reset.
It only starts again (with the full duration) when input 1.0 has a "0" signal.

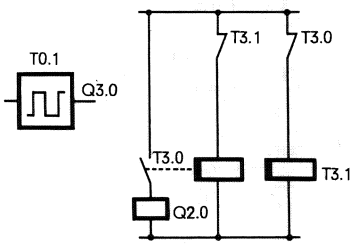
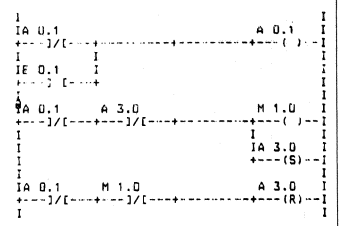
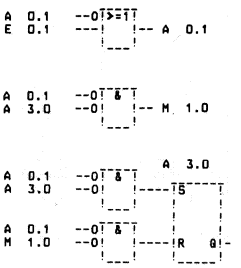


E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

1) Start time
2) Scan time

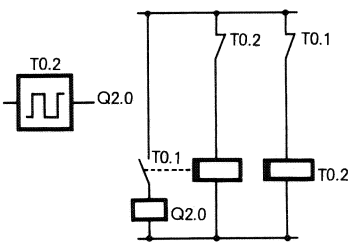
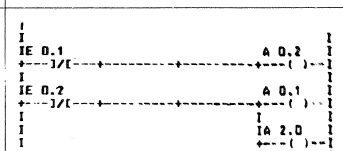
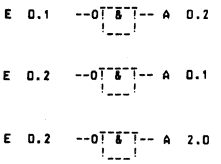
Timer functions (continued)

Clock generator with one timer

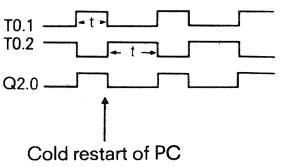
Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	ON Q 0.1 O I 0.1 = Q 0.1 AN Q 0.1 AN Q 3.0 = F 1.0 S Q 3.0 AN Q 0.1 AN F 1.0 R Q 3.0		

E ≙ INPUT
A ≙ OUTPUT
M ≙ FLAG

Clock generator with two timers - free running

Original	STEP 5 representation: Statement list STL	Ladder diagram LAD	Control system flowchart CSF
	AN I 0.1 = Q 0.2 AN I 0.2 = Q 0.1 = Q 2.0		

By setting timers T3.0 and T3.1, the pulse duration and interval between pulses (mark/space ratio) can be changed as required.

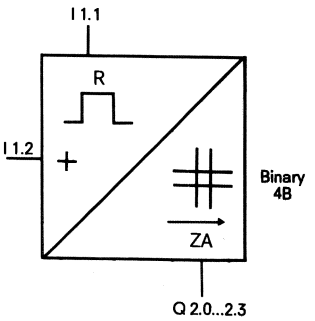


E ≙ INPUT
A ≙ OUTPUT

STEP 5 Statement list

Binary up-counter with pulse edge evaluation

Program memory Address	Operation	Operand Ident.	Operand Parameter	
0	A	I	1.1	Pulse edge evaluation
1	AN	F	1.0	
2	=	F	1.7	
3	A	F	1.7	
4	S	F	1.0	
5	AN	I	1.1	
6	R	F	1.0	
7	A	I	1.2	Reset counter
8	R	Q	2.0	
9	R	Q	2.1	
A	R	Q	2.2	
B	R	Q	2.3	
C	R	F	1.7	
D	A	F	1.7	Bit 0
E	AN	Q	2.0	
F	S	Q	2.0	
10	R	F	1.7	Reset pulse flag
1	A	F	1.7	
2	A	Q	2.0	
3	R	Q	2.0	
4	A	F	1.7	Bit 1
5	AN	Q	2.1	
6	S	Q	2.1	
7	R	F	1.7	Reset edge flag
8	A	F	1.7	
9	A	Q	2.1	
A	R	Q	2.1	
B	A	F	1.7	Bit 2
C	AN	Q	2.2	
D	S	Q	2.2	
E	R	F	1.7	Reset edge flag
F	A	F	1.7	
20	A	Q	2.2	
1	R	Q	2.2	
2	A	F	1.7	Bit 3
3	AN	Q	2.3	
4	S	Q	2.3	
5	R	F	1.7	Reset edge flag
6	A	F	1.7	
7	A	Q	2.3	
8	R	Q	2.3	
9			.	
A			.	
B			.	
C			.	
D			.	
E			.	
F			.	



SIEMENS AKTIENGESELLSCHAFT

SIMATIC S5 Programmable Controller System

Page of

5.4 Complex functions

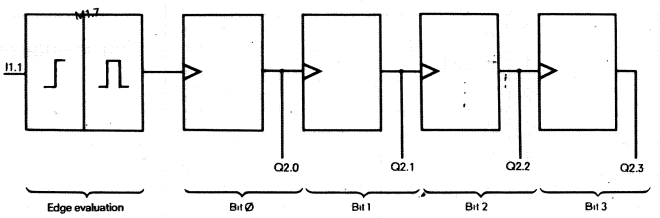
Binary counter with pulse edge evaluation

When using binary scalers with pulse edge evaluation, edge evaluation need only be programmed once. The individual bits of the counter are then programmed. This results in an up-counter counting with the rising edge of the counter input. If, when programming, the sequence of setting and resetting the output is reversed (resetting the pulse flag is now synchronous with the resetting of the output), this results in a down-counter. If counting is to take place on the trailing edge of the input pulse, evaluation of the trailing signal edge is necessary.

Programming a down-counter is as follows for bit 0:

Structure of a 4-bit binary counter,

using binary scalers with pulse edge evaluation



Program memory Address	STEP 5 Statement list Operation	Operand Ident.	Parameter	Binary down-counter
0	A	F	1.7	
1	A	Q	2.0	
2	R	Q	2.0	Reset output
			.	
3	R	F	1.7	Reset pulse flag
			.	
4	A	F	1.7	
5	A N	Q	2.0	
6	S	Q	2.0	Set output
7			.	

The same procedure is used for programming the other bits.

Resetting a counter

The counter just programmed is to be reset with input I1.2. Then, after edge evaluation of input I1.2, a scan is made to see whether it is "1". If this is the case, the outputs of the counter are reset. At the same time, the pulse flag is reset so that counting is not possible as long as the reset signal is applied. Resetting is thus static.

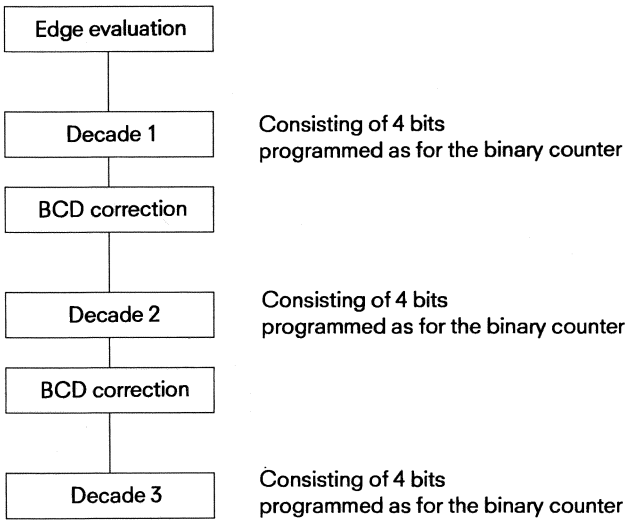
Program memory Address	STEP 5 Statement list Operation	Operand Ident.	Parameter	Reset counter
0	A	I	1.2	Reset?
1	R	Q	2.0	
2	R	Q	2.1	Reset counter
3	R	Q	2.2	
4	R	Q	2.3	
			.	
5	R	F	1.7	Reset pulse flag
6			.	
7			.	

BCD counter

A BCD counter has the same structure as a binary counter, but with a BCD correction after every 4 bits.

Structure of a BCD counter with 3 decades

using binary scalers with pulse edge evaluation

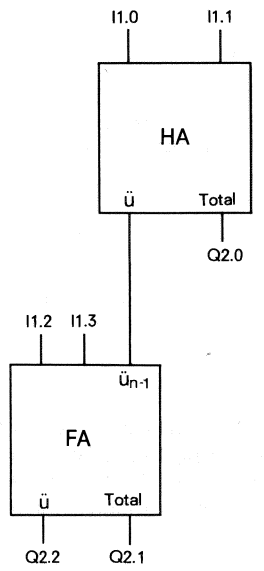


BCD correction means that, on the tenth pulse of each decade, i.e. when the value "10" has been reached in the decade immediately before, the value of the previous decade is corrected to "0". At the same time, the pulse flag is set once more. This functions as a "carry" to the next decade.

Program memory	STEP 5 Statement list			BCD correction for "ones"
		Operand		
Address	Operation	Ident.	Parameter	
0	A	Q	2.1	Scan word "10"
1	A	Q	2.3	
			.	
2	R	Q	2.1	Reset to "0"
3	R	Q	2.3	
			.	
4	=	F	1.7	Set carry (pulse flag)
5			.	

5.4 Complex functions

Program memory Address	STEP 5 Statement list Operation	Operand Ident.	Parameter	Adder for 2 bits, binary
0	A	I	1.0	Total 1st bit
1	A N	I	1.1	
2	=	F	1.0	
3	A N	I	1.0	
4	A	I	1.1	
5	O	F	1.0	
6	=	Q	2.0	
7	A	I	1.0	
8	A	I	1.1	
9	=	F	1.1	Carry 1st bit
			.	
			.	
A	A	I	1.2	Total 2nd bit
B	A N	I	1.3	
C	A N	F	1.1	
D	=	F	2.0	
E	A N	I	1.2	
F	A	I	1.3	
10	A N	F	1.1	
1	=	F	2.1	
2	A	I	1.2	
3	A	I	1.3	
4	A	F	1.1	
5	=	F	2.2	
6	A N	I	1.2	
7	A N	I	1.3	
8	A	F	1.1	
9	O	F	2.0	
A	O	F	2.1	
B	O	F	2.2	
C	=	Q	2.1	
F			.	
D	A	I	1.2	Carry 2nd bit
E	A	F	1.1	
F	=	F	2.3	
20	A	I	1.3	
1	A	F	1.1	
2	=	F	2.4	
3	A	I	1.2	
4	A	I	1.3	
5	O	F	2.3	
6	O	F	2.4	
7	=	Q	2.2	
8			.	
9			.	
A			.	
B			.	
C			.	



HA Half-adder
FA Full-adder
C Carry

SIEMENS AKTIENGESELLSCHAFT

SIMATIC S5 Programmable Controller System

Page of

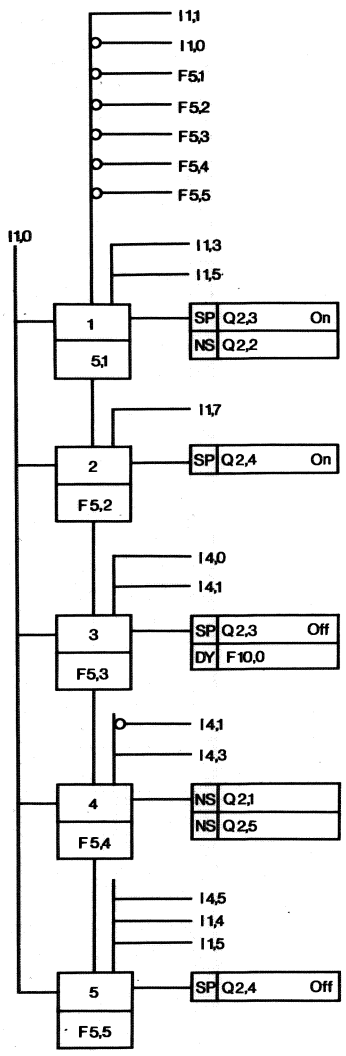
Program memory Address	STEP 5 Statement list Operation	Operand Ident.	Operand Parameter	Sequence control
0	A	I	1.0	Reset sequence cascade
1	R	F	5.1	Reset step flag
2	R	F	5.2	
3	R	F	5.3	
4	R	F	5.4	
5	R	F	5.5	
6	R	Q	2.1	Reset outputs
7	R	Q	2.2	
8	R	Q	2.3	
9	R	Q	2.4	
A	R	Q	2.5	
B	A	I	1.1	Start sequence cascade, if there is no reset and if no step has been set.
C	A N	I	1.0	
D	A N	F	5.1	
E	A N	F	5.2	
F	A N	F	5.3	
10	A N	F	5.4	
1	A N	F	5.5	
2	A	I	1.3	Conditions for step 1
3	A	I	1.5	
4	S	F	5.1	Step flag
5	S	Q	2.3	..Outputs
6	S	Q	2.2	
7	A	F	5.1	Enable step 2
8	A	I	1.7	Condition
9	S	F	5.2	Step flags
A	R	F	5.1	
B	S	Q	2.4	Outputs
C	R	Q	2.2	
D	A	F	5.2	Enable step 3
E	A	I	2.0	Conditions
F	A	I	2.1	
20	S	F	5.3	Step flags
1	R	F	5.2	
2	R	Q	2.3	Output
3	=	F	100.0	
4	A	F	5.3	Enable step 4
5	A N	I	2.4	Conditions
6	A	I	2.3	
7	S	F	5.4	Step flags
8	R	F	5.3	
9	S	Q	2.1	Outputs
A	S	Q	2.5	
B	A	F	5.4	Enable step 5
C	A	I	2.5	Conditions
D	A	I	1.4	
E	A	I	1.5	
F	S	F	5.5	Step flag

Continued on next page

5.4 Complex functions

Program memory Address	STEP 5 Statement list Operation	Operand Ident.	Parameter
0	R	F	5.4
1	R	Q	2.4
2	R	Q	2.1
3	R	Q	2.5

Sequence control (continued)



Outputs

In contrast to logic control, only one part of the program at a time (a sequence step) is enabled for processing in the case of sequence control. Only when the conditions for this step have been fulfilled does processing continue with the next step.

The individual steps of a sequence cascade are identified by auxiliary flags (step flags). A set step flag means that the step concerned has just issued its control commands and the conditions for the next step are enabled. If the conditions of the next step are fulfilled, the step flag is reset and the step flag for the next step is set.

Representation of a sequence cascade

The nature of the output control commands is identified by the SP, NS or DY prefix in the relevant command box of the sequence cascade.

SP "latching"

The output is to be set and retain the "1" state through several sequence steps until it is reset. For this reason, "On" (SQ) or "Off" (RQ) must also be specified.

NS "non-latching"

The output is to have the "1" state if the requirements of the sequence step are fulfilled. It is to resume the "0" state again when progression is made to the next sequence step.

This function cannot be programmed with the = Q since, as soon as a sequence step has been set, the conditions for this step are no longer fulfilled. The output would only remain set for the duration of one cycle. For this reason, SQ is programmed and, in the next step, RQ.

DY "dynamic"

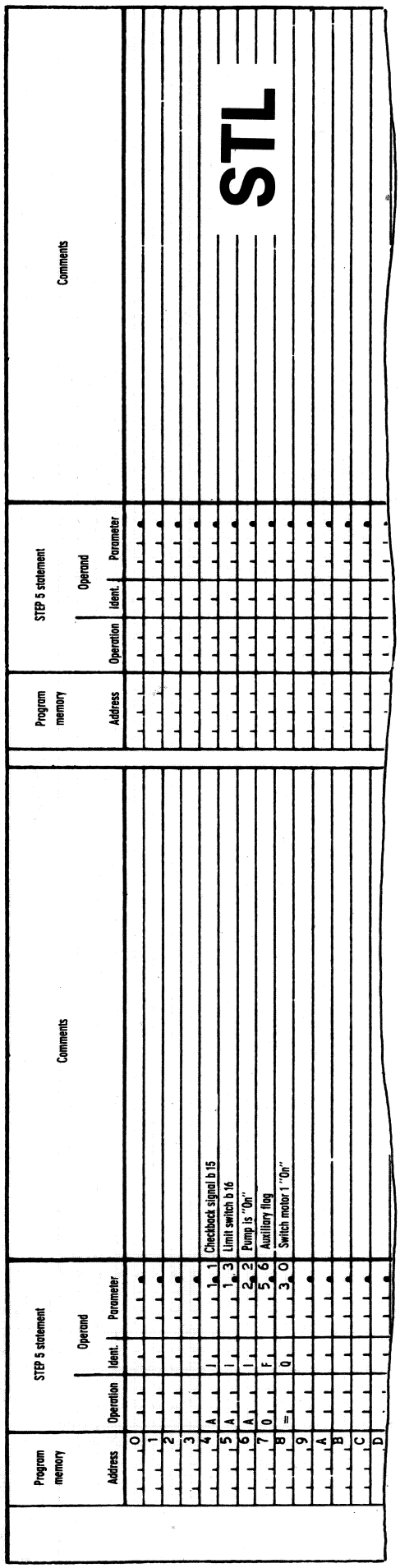
The output is only set as a pulse. A timer must be used to generate external pulses. For internal pulses, it is sufficient if the "1" state exists only for one cycle. For this reason, = is programmed (see above).

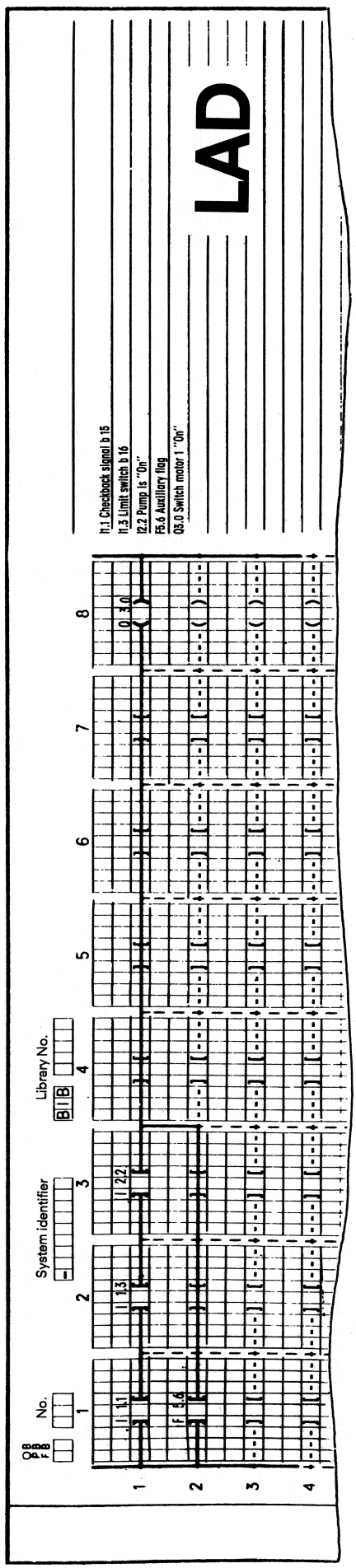
In this example, the sequence cascade is initialised with input I1.0, i.e. all step flags are reset. Only when all step flags have been reset and input I1.0 is "0" can the cascade be enabled with input I1.1. If the conditions for the first step have been met (I1.3 and I1.5 are "1"), the first step is set. The outputs (Q2.3 and Q2.2) are then set and the next step is enabled.

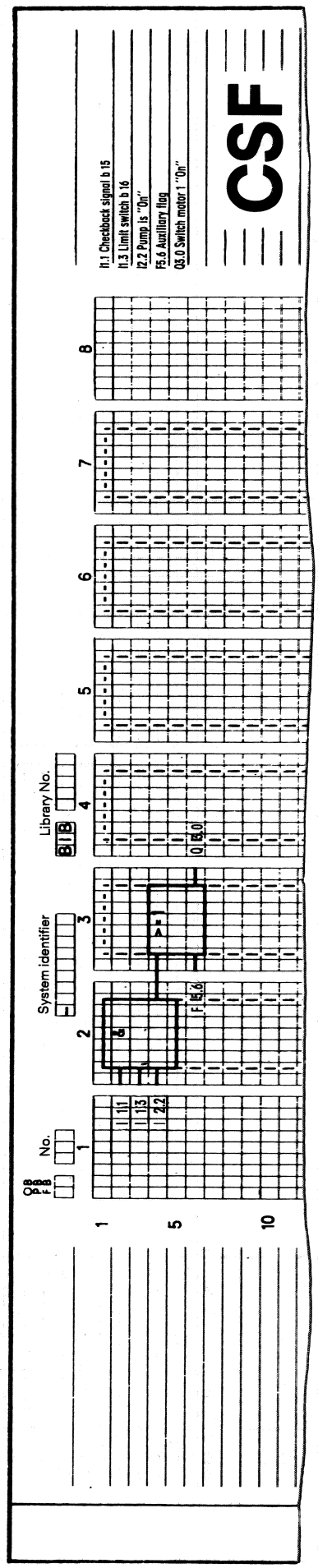
A sequence step therefore consists of

> scanning the preceding step flag ("enable"),
> the conditions of this step,
> setting the step flag and resetting the preceding step flag and
> setting and resetting the outputs to be processed.

Examples of STL, LAD and CSF



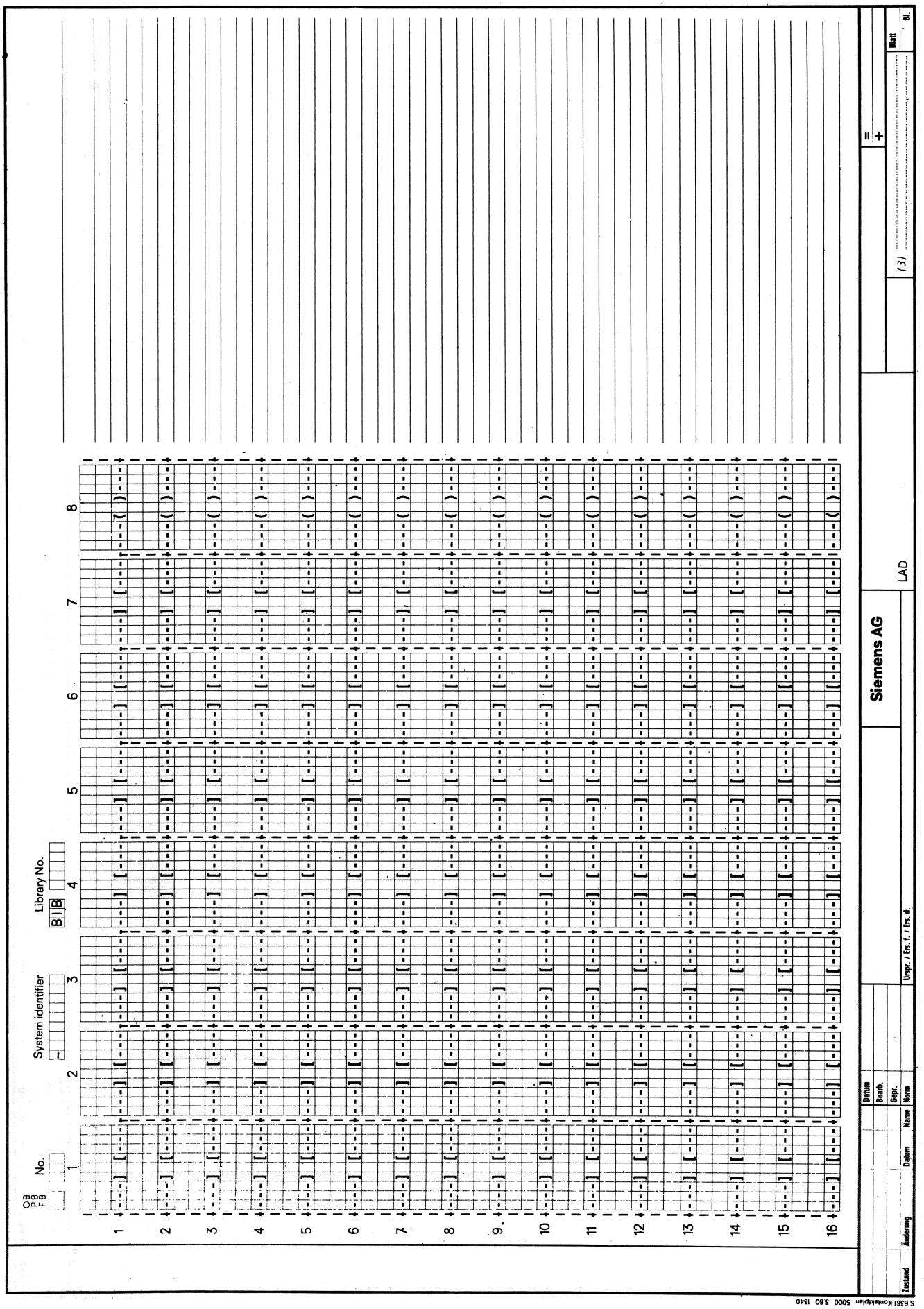




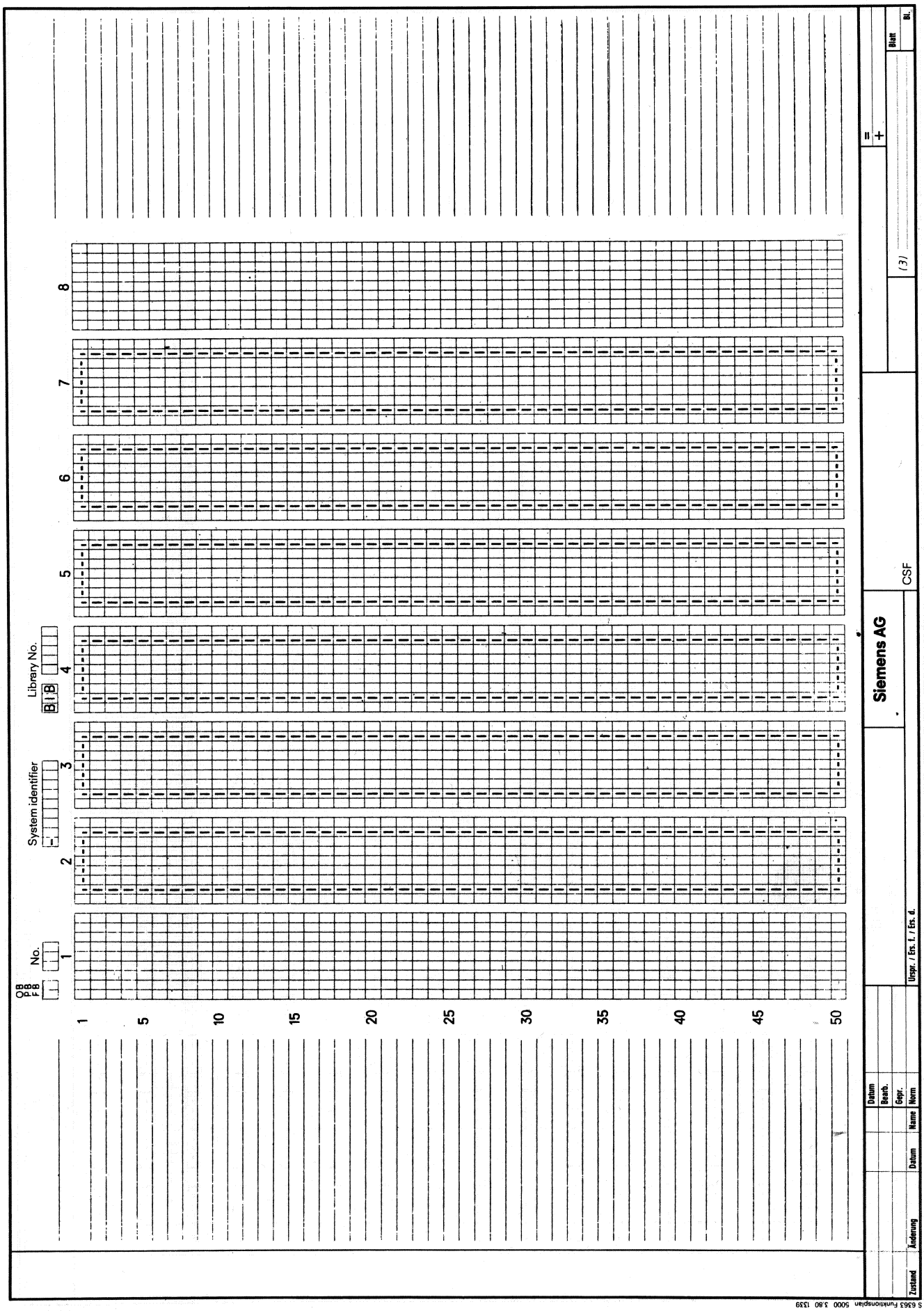
6. STL form

Program memory	STEP 5 statement			Comments	Program memory	STEP 5 statement			Comments
		Operand					Operand		
Address	Operation	Ident.	Parameter		Address	Operation	Ident.	Parameter	
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									
0									
1									
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									

			Datum		Siemens AG			=
			Bearb.					+
			Gepr.				(3)	Blatt
Zustand	Änderung	Datum Name	Norm	Urspr. / Ers. f. / Ers. d.		Anweisungsliste / Statement list		Bl.


OB
PB
FB
No.
System identifier
Library No.
B I B
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Siemens AG
LAD
Datum
Bearb.
Gepr.
Norm
Name
Datum
Änderung
Zustand
Ursgr. / Ers. f. / Ers. d.
Blatt
Bl.
(3)
S 6361 Kontaktplan 5000 3.80 1340

6. CSF form


OB
PB
FB
No.
System identifier
Library No.
B B
1
2
3
4
5
6
7
8
1
5
10
15
20
25
30
35
40
45
50
Siemens AG
CSF
Urspr. / Ers. f. / Ers. d.
Datum
Bearb.
Gepr.
Norm
Name
Datum
Änderung
Zustand
(3)
Blatt
Bl.
S 6363 Funktionsplan 5000 3.80 1339