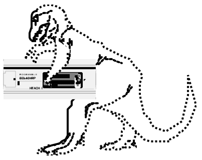
Table of contents:

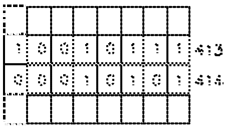
MANUAL Hitachi series EC

How to read this manual



- History, Background (page 1)

A short history and presentation of xxx, Hitachi and PLC in general is described here.



- Symbols, abbreviations, etc. (page 5)

The basic contents of a PLC, the common abbreviations and principles of addressing and the memory areas (e.g. Special memories) are described here.

```
¦                  ¦ ORG     10
+--¦ +---------( )--¦ OR  NOT 11
¦  10  ¦      211   ¦ OUT    211
¦      ¦            ¦
+--¦/+--+           ¦
¦  11               ¦
```

- Logic Program instructions (page 8)

The basic ladder programming is described first. Thereafter Timers, Counters and comparing is described. The chapter ends with mixed program examples.

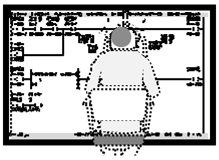
- Special instructions (page 22)

These are first given in a short presentation with a page reference to a more detailed description

```
¦        +----------+ ¦ ORG      20
+--¦ +--¦FUN20   00¦-¦ FUN20     0
¦  20   ¦FUN11  400¦ ¦ FUN11   400
¦        +----------+ ¦
```

- Programming tools and handling in practice (page 71)

Here is a description of the programming units, computer programming, start up and documentation



- Technical details (page 129)

Finally the technical specification, installation, connection and dimensions are presented.

History, background:

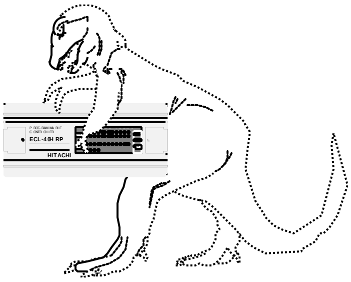
Short history about xxx:

[here you can write a short information and history about your own company].

Short history about Hitachi:

Hitachi Ltd was started in 1910. The original business was based on electro-mechanical products. Today Hitachi is the largest company in Japan manufacturing electronic and electro-mechanical products. It also belongs to the largest companies world wide, all categories.
Today Hitachi is known for a number of products (all the way from manufacture of integrated circuits, consumer electronics to nuclear power generators).
In common for all product ranges is the quality approach, which been Hitachi's priority for many years. The PLC product range from Hitachi is a good example of this.
Thanks to the availability of Hitachi's own integrated circuit development Hitachi is in the front line of PLC development.

Short history about PLC:



"PLC" stands for "Programmable Logic Controller". The PLCs have today almost completely replaced the older generations of control systems. The relay systems belong to this group. The relays were connected in order to form a logic combination between inputs and outputs. When the micro processor was invented this technique was used in products to replace the relays. These products were different from other micro processor solutions as the user programming structure was designed to be similar to the logic relay combinations and the way of running through the program was made such that all logic circuits seem to run simultaneously. To replace the relays in hard physical environment these product also had to be better prepared to withstand noise, vibrations etc.

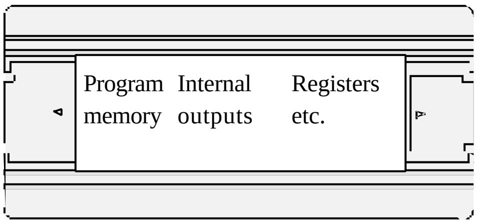
In the beginning these products only took care of logic combinations, as the relay technique. Therefore the word ”Logic” was placed in-between "Programmable" and "Controller".
As the micro processor technique itself offered more possibilities than to handle pure logic it was natural to introduce arithmetic instructions. Many countries decided therefore to delete the word ”Logic” in the name. (this happened in the beginning of the 1980s). The abbreviation ”PC” very soon came into a conflict with another abbreviation. That was ”PC” for personal computer. Therefor most countries returned to ”PLC” even if this abbreviation is not perfect.

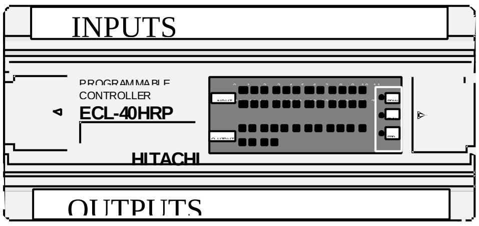
The PLC systems are built around standardised modules. These are manufactured in very large quantities. Often it is an advantage economically to use this technique instead of special designed products even if it is possible to optimise the amount of components in the special solution.
The units are well tested and the failure frequency is low. The documentation is standardised and it can be understood by many people. There are also spare parts available in most countries.

1. Programming

1.1. Addresses, which are used in the program:

1.1.1. Memory, register etc.: symbolic content and explanation





For more information, see appendix page 136

1.1.2. Address map

External Inputs / Outputs

Type	Input no.	Output no.	Note.
EC20	0-11	200-207	
EC40	0-15 20-27	200-215	16-19 not available
EC60	0-15 20-35 40-43	200-215 220-227	16-19 and 36-39 not available 216-219 not available

1.1.2.1. Internal Memories (or internal outputs).

Type	Internal memory no.	Note.
Not retentive **(Without battery backup)**	400-655	
Retentive **(with battery backup)**	700-955	656-699 does not exist.
Special memories	960-991	956-959 does not exist

Here battery backup means memories, which retain their contents at power down (maximum 3 weeks)

Note:
Input 0-2 can be used as inputs to the high speed counter. (see FUN96)
Input 3 can be used as the Interrupt input. (see FUN97)
Input 0-7 have programmable filter times on the inputs. (see FUN97)

1.1.2.2. Timers and Counters

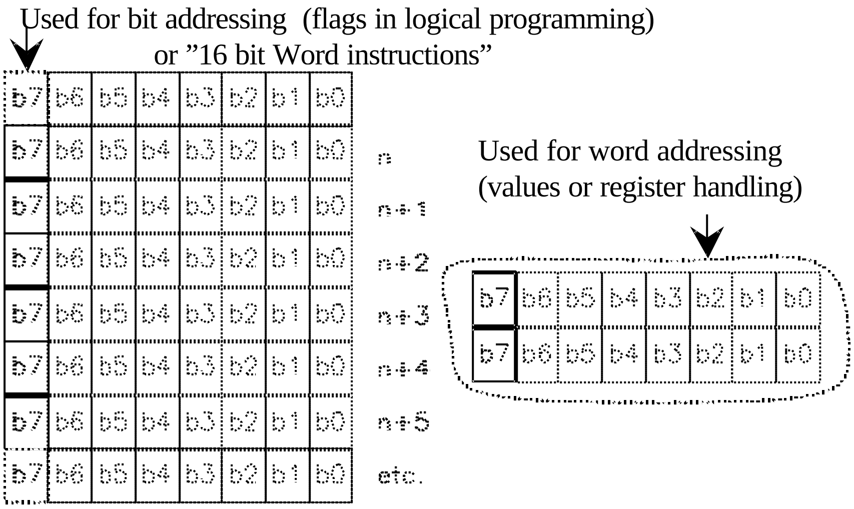
There are a total of 96 Timers and Counters. These share the space in the PLC. **(T/C 0-95)**

(T/C95 can be used as the analogue timer in the ECL-version. For more information see page 126)

1.1.2.3. Bit/Word memories.

All addresses are decimal. Words and bits are mixed in the same area.

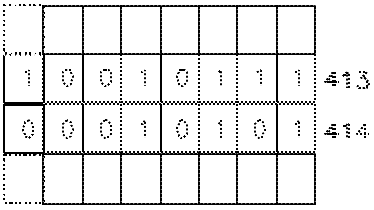
Behind each bit in the memory there are 8 bits available.
When a ”2 Byte Word instruction” is used in the program this address (8 bits) and the next higher address (8 bits) is used.
Together these form a 16 bit word.


Used for bit addressing (flags in logical programming)
or ”16 bit Word instructions”
b7 b6 b5 b4 b3 b2 b1 b0
n
n+1
n+2
n+3
n+4
n+5
etc.
Used for word addressing
(values or register handling)

For more information about the difference between ”2-byte Word instruction” and 16 bit Word instructions, see page 109.

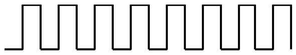
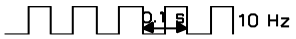
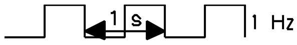
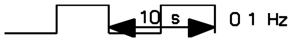
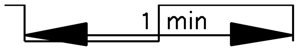
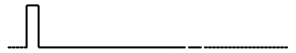
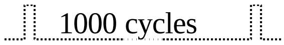
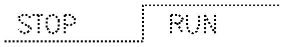
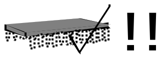
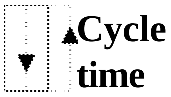
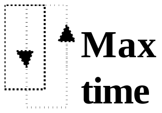
Example:
If 413 is addressed as a bit, e.g. AND 413 the result will be "1" or ”ON” and if 414 is addressed as a bit the result will be "0"

If 413 on the other hand is addressed as a word, e.g. ADD 413 the value 1001 0111 0001 0101 is fetched and added.



A ”**bit**” means a condition ”ON”/”OFF” or "1"/"0"
A ””**word**” means 16 bits in a row, which can represent a value between 0 and 65535 Decimal (0000000000000000-1111111111111111)

1.1.3. Special memories

Bit 960:		Resets all outputs (the output information is not copied to the physical outputs)
Bit 961:		Resets all retentive (battery backup) memories, Counters and Shift registers. (only executed during the first program cycle and it must be programmed in the very beginning of the program). It can e.g. be used together with 967, initial pulse, to make a reset at program start. (See example below)
Bit 962:		Impulse[1] every second cycle
Bit 963:		Repeated Pulse. Frequency 10 Hz (0,1 s period)
Bit 964:		Repeated Pulse. Frequency 1 Hz (1 s period)
Bit 965:		Repeated Pulse. Frequency 0.1 Hz (10 s period)
Bit 966:		Repeated Pulse. 1 minute period
Bit 967:		Initial pulse. A pulse, which occurs only at program start. The length is equal to one program cycle.
Bit 968:		Gives a pulse every 1000 program cycles.
Bit 990:		Always ON
Bit 991:		ON during RUN.
Word 970		Shows a system error as a decimal code in a 2-byte word (more info, see page 79.)
Word 980	**AND 1000 !!**	Shows syntax error as a decimal code in a 2-byte word (more info, see page 76)
Word 982		Shows the execution time (cycle time) in units of 10 ms (2 byte word) (The first program cycle shows 65535.)
Word 984		Shows the maximum execution time (cycle time) in units of 10 ms (2 byte word) (The first program cycle shows 0.)
Bit 972-979 Bit 986-989	-	Not used. 972-977 are not used by the system. The rest are reserved for future purposes.

1 Impulse means a short pulse with the length of one program cycle.

Example:

The start impulse (Bit 967) resets all internal outputs and Shift registers.
All Up-Counters are reset to 0.
Here C50 is reset to 0.

```
¦                                  ¦
+-¦ +----------------( )--¦
¦ 967                 961 ¦
¦                                  ¦
¦                   +------+ ¦
¦                   ¦C50   ¦ ¦
+-¦ +-----------¦      +-¦
¦ 000               ¦preset¦ ¦
¦                   ¦      ¦ ¦
+-¦ +-----------¦400   ¦ ¦
¦ 001               +------+ ¦
```

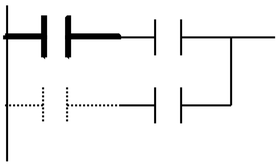
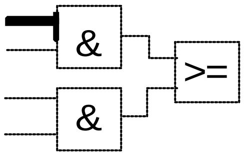
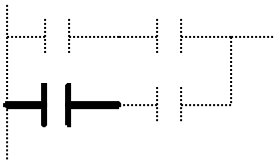
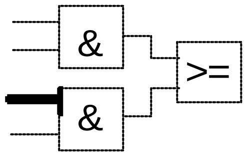
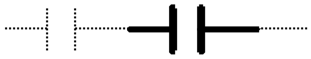
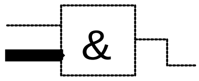
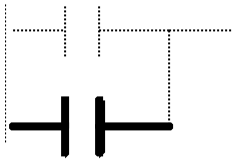
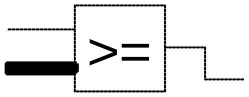
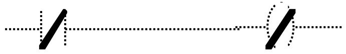
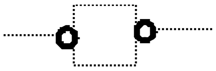
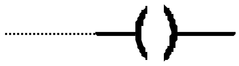
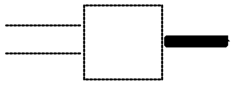
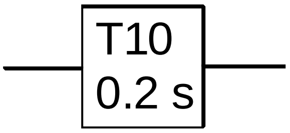
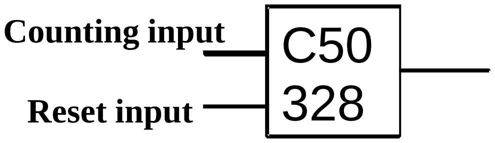
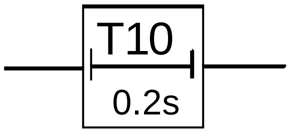
[1] Impulse means here a short pulse with the length of one program cycle. (There is no need the make any edge detection as it last as long as the edge itself.

1.2. Logic Program instructions

A "block" means a combined logical condition ended by e.g. output(s).

1.2.1. Logic program instructions

More info, see page 9.

			Ladder diagram	Logic block diagram
ORG	ORiGin	Origin of a block		
STR	SToRe	Beginning of a branch in a block		
AND	AND	Serial connection of logic contacts.		
OR	OR	Parallel connection of logic contacts		
NOT	NOT	Inverted function of a contact or output.		
OUT	OUT	Output		
T/C	Timer/ Counter	Time delay or (Up-) Counters	 	

1.3. Usage of the Program instructions

1.3.1. Logical combinations

ORG, STR NOT, AND, AND NOT, OR, OR NOT, OUT, OUT NOT

Ladder **Instruction** **Note**

```
+--¦ +--------------------( )--¦
¦   0                    200   ¦
```

```
ORG      0
OUT    200
```

ORG starts a logic block.

```
¦                              ¦
¦                              ¦
+--¦/+--------------------( )--¦
¦   1                    201   ¦
```

```
ORG NOT  1
OUT    201
```

ORG NOT starts a logic block.
OUT gives an output block.
·

```
¦                              ¦
+--¦ +-----¦ +------------( )--¦
¦   2       3            202   ¦
¦                              ¦
¦                              ¦
```

```
ORG      2
AND      3
OUT    202
```

AND
is used for serial connection of closing contacts.

```
¦                              ¦
+--¦ +-----¦/+------------( )--¦
¦   4       5            203   ¦
¦                              ¦
```

```
ORG      4
AND NOT 5
OUT    203
```

AND NOT
is used for serial connection of inverted contacts

```
¦                              ¦
+--¦ +--------------------( )--¦
¦   6   ¦                210   ¦
¦       ¦                      ¦
+--¦ +--+                      ¦
¦   7                          ¦
```

```
ORG      6
OR       7
OUT    210
```

OR
is used for parallel connection of closing contacts.

```
¦                              ¦
+--¦ +--------------------( )--¦
¦   10  ¦                211   ¦
¦       ¦                      ¦
+--¦/+--+                      ¦
¦   11                         ¦
```

```
ORG     10
OR  NOT 11
OUT    211
```

OR NOT
is used for parallel connection of inverted contacts.

```
¦                              ¦
+--¦ +--------------------( )--¦
¦   12              ¦    212   ¦
¦                   ¦          ¦
¦                   +-----( )--¦
                         213
```

```
ORG     12
OUT    212
OUT    213
```

Multiple outputs on the same condition.

```
¦                              ¦
¦                              ¦
+--¦ +--------------------( )--¦
¦   13            ¦      214   ¦
¦                 ¦            ¦
¦                 +-¦ +---( )--¦
¦                    14  215   ¦
```

```
ORG     13
OUT    214
AND     14
OUT    215
```

Multiple outputs on the same condition with additional condition, "14". (Additional condition can only be added to the last output(s))

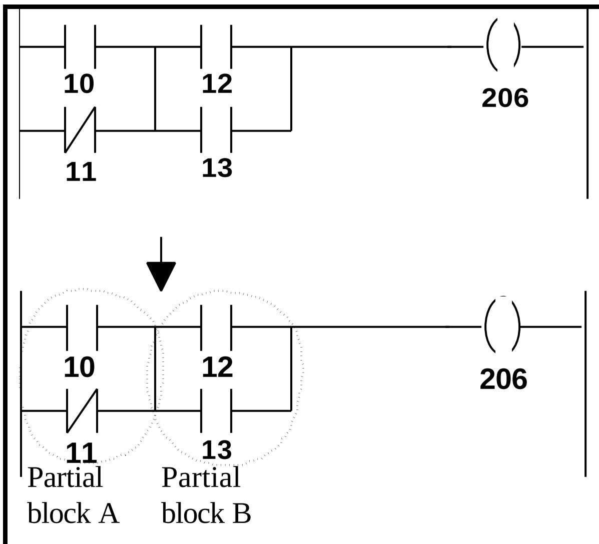
```
¦                              ¦
+--¦ +--------------------(/)--¦
¦   15                   220   ¦
```

```
ORG     15
OUT NOT 220
```

OUT NOT gives an inverted

STR, STR NOT, AND STR, OR STR.

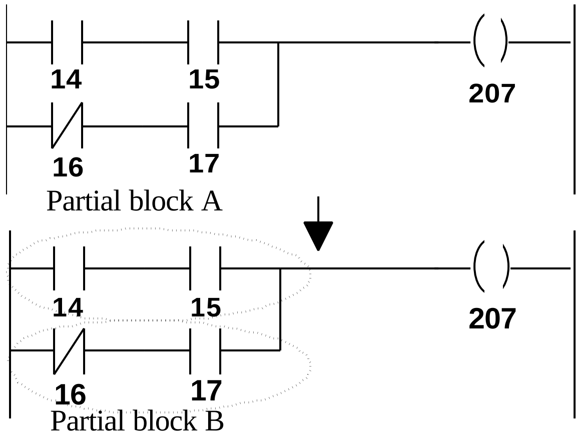
Ladder



This block is broken down through dividing it into two partial blocks (A and B)

Thereafter it is broken down into following instructions:

	Instruction	Note
Start partial block A	ORG 10	Serial connection of two parallel partial blocks with AND STR. (Block A is treated first separately, thereafter block B whereafter they are serial connected with AND STR.)
Start partial block B	OR NOT 11	
	STR 12	
Connection	STR 13	
(serial)	AND STR,	
	OUT 206	
	-	



This block is broken down through dividing it into two partial blocks (A and B)

Thereafter it is broken down into following instructions:

	Instruction	Note
Start partial block A	ORG 14	Parallel connection of two serial blocks with OR STR. (Block A is treated first separately, thereafter block B whereafter they are parallel connected with OR STR.)
	AND 15	
Start partial block B	STR NOT 16	
	AND 17	
Connection	OR STR	
(parallel)	OUT 207	

(”Partial block” means a part of a block which is a natural unit.)

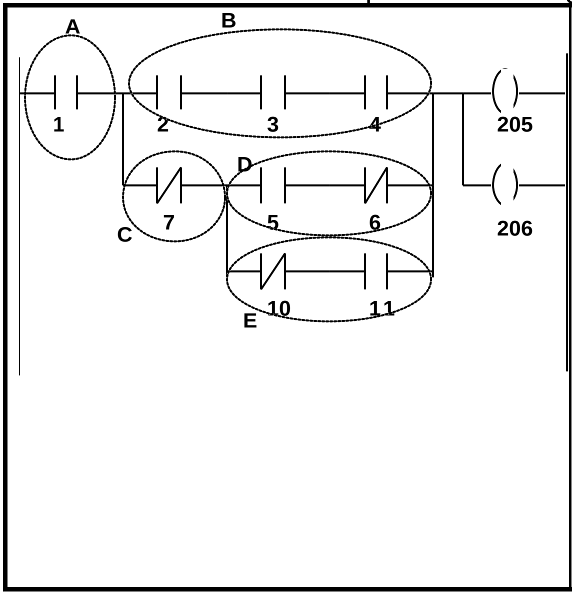
Description:

If parallel or serial partial blocks are connected, these must begin with STR or STR NOT and be ended by OR STR (when it is parallel connected to the previous partial block) or by AND STR (when it is serial connected to the previous partial block)

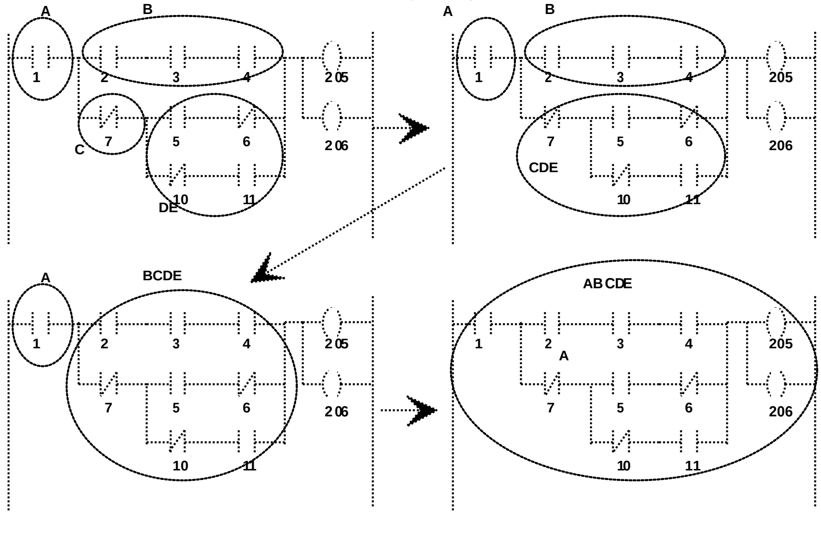
STR or STR NOT must always be followed of a corresponding OR STR or AND STR. These instructions define the start or end of a partial block.
A partial block can be divided into smaller partial blocks. See example page 9.

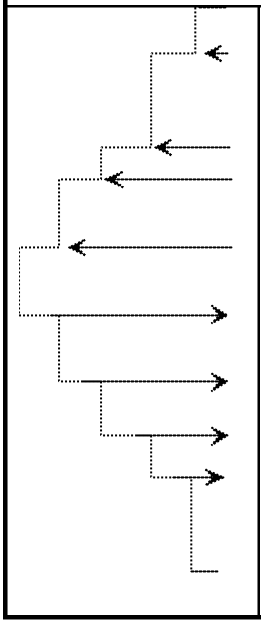
***EXAMPLE** OF A CONVERSION OF A MORE COMPLEX LOGIC BLOCK:*

The instruction is built up in the following way:



This is broken down in following steps:



Levels 5 4 3 2 1	Instruction		Reference in the diagram:
	**ORG**	**1**	partial block A
	STR	**2**	partial block B
	AND	**3**	
	AND	**4**	
	STR NOT	**7**	partial block C
	STR	**5**	partial block D
	AND NOT	**6**	
	STR NOT	**10**	partial block E
	AND	**11**	
	OR STR		parallel connection of D and E to partial block DE
	AND STR		serial connection of C and DE to DE
	OR STR		parallel connection of B and CDE to BCDE
	AND STR		serial connection of A and BCDE to a complete block
	OUT	**205**	The Result is given to the
	OUT NOT	**206**	outputs

The example shows how to divide the partial blocks into the new partial blocks (A and. B). Every time this is done we can say that we have got a new ”level”.
The PLC works internally with a so called ”stack” where the temporary logic result is stored. It is the levels in this stack, which are referred to.)

The example above has 5 levels. As **a maximum 7 such levels** can be handled.

When these levels are connected in instruction code every STR or STR NOT shall correspond to a AND STR or OR STR, which is a good checking rule during program design.
(the exception is Counters, Shift registers and FUN blocks with more than one input, where every extra input (above the first) also corresponds to a STR or STR NOT.

1.3.2. Timers and Counters (T/C)

1.3.2.1. Timers

On delay timers:

Ladder | **Instruction** | **Note**

```
|                        +----+ |
+--| +-------------------|T00 +-|
|   0                    |    | |
|                        |50s | |
|                        +----+ |
|                               |
+--| +--------------------( )--|
|   T00                   201   |
|                               |
```

```
ORG          0
OUT T/C      0   50

ORG T/C      0
OUT        201
```

Condition for a 3-digit timer with the preset 50 s

When 50 s is up the timer contact T00 activates output 251

Conditions:

The timers are defined with a 2-digit address (T00 - T95) and a 3-digit preset (0,1 - 99,9 s or. 1 to 999 s). E.g. T10 45.6 means Timer 10 with preset 45.6 s.

4-digit preset can be programmed on T/C 0-9. You write "." directly after the timer number to separate this from the preset.

e.g. OUT   T/C   2 . 1   2   3   .   4
gives timer 2 with preset 123.4 s

Timers are Up counting. (Starts from 0,0 s. The timer output goes high when the preset is achieved.)

```
             +-----+          +-----------------------------+
Timer        |     |          |                             |
input        |     |          |                             |
-------------+     +----------+                             +----
                                                 +-----------+
Timer                                            |           |
output                                           |           |
-------------------------------------------------+
+---------
             0.....           0.............123.4
```

See also page 38 for example how to handle preset values and counter values.

Off Delay timers:

Use the timer in the following way to get an Off Delay timer.:

Ladder | **Instruction** | **Note**

```
|                        +----+ |
+--|/+-------------------|T00 +-|
|   0                    |    | |
|                        |50s | |
|                        +----+ |
|                               |
+--|/+--------------------( )--|
|   T00                   201   |
|                               |
```

```
ORG NOT          0
OUT T/C          0   50

ORG NOT T/C      0
OUT            201
```

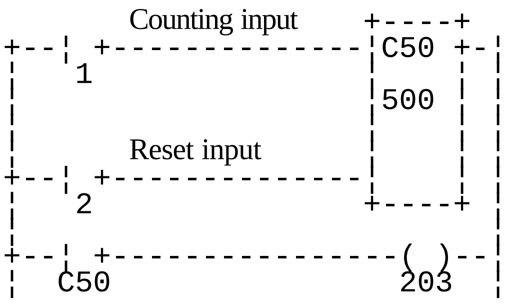
When input 0 goes low the timer starts

When 50 s is up, output 201 turns Off

When input 0 goes High output 201 goes high

1.3.2.2. Counter programming (Up Counters)

Ladder **Instruction** **Note**



```
ORG        1
STR        2
OUT T/C 50  500

ORG T/C 50
OUT      203
```

Input 1 is a counter input and input 2 is a reset input. The preset value is 500.

When 500 pulses are counted, the output contact of the counter goes high.

Explanations:

Counters are programmed with a 2-digit address (C00 - C95) and a 3-digit preset value.
(1 - 999) E.g. C10 456 means Counter 10 with a preset of 456 pulses.

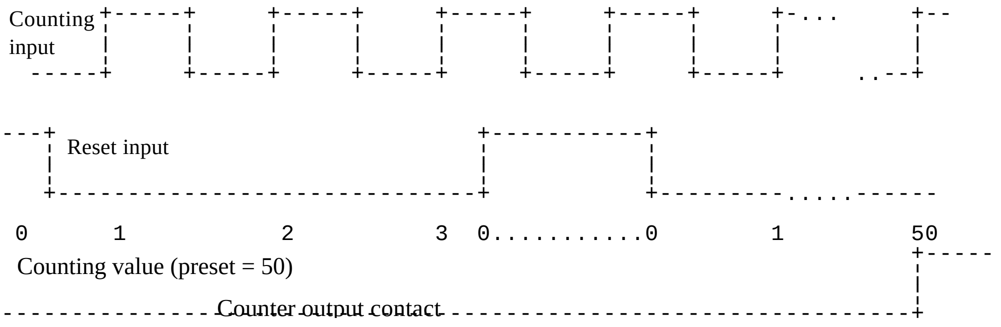
A 4-digit preset value can be programmed on T/C 0-9. You write ”.” directly after the counter number to separate this from the preset value.

e.g. OUT T/C 2 . 1 2 3 4
gives Counter 2 with the preset value 1234

The counters are Up-counting. (Starts from 0, the output contact goes high when the preset value is reached.)

Two inputs must be used; a counting input and a reset input.

The current value is saved at power down or program stop. But if you change the preset value in the program (e.g. with FUN21 C250) this will be reset when the PLC is turn off and on.



.
See also page 29 for an example how to handle the preset- and counting value.

1.3.3. Program example, Logic instructions

Serial connection with self hold:

```
¦                                 ¦
¦                                 ¦ ORG       0     starts the self hold
+--¦ +------¦/+--------------( )--¦ OR      210
¦   0   ¦     1               210  ¦ AND NOT   1     breaks the self hold
¦       ¦                         ¦ OUT     210
+--¦ +-+                          ¦
¦  210                            ¦
¦                                 ¦
```

Serial connection / Parallel connection with partial blocks:

```
¦                                 ¦ ORG       0
+--¦ +----¦/+----¦ +---¦ +----( )-¦ AND NOT   1
¦   0      1  ¦   2     3  ¦ 210  ¦ STR       2
¦             ¦            ¦      ¦ AND       3
¦             +--¦ +-------¦      ¦ OR        4
¦             ¦   4        ¦      ¦ OR       10
¦             ¦            ¦      ¦ AND STR
¦             +--¦ +-------+      ¦ OUT     210
¦                 10              ¦
¦                                 ¦
```

Two parallel connections in series:

```
¦                                 ¦
+--¦ +----¦/+----¦ +---¦ +----( )-¦ ORG       0
¦   0      1  ¦   2     3  ¦ 210  ¦ AND NOT   1
¦             ¦            ¦      ¦ STR       4
+--¦ +----¦ +----¦ +---¦ +-+      ¦ AND       5
¦   4      5      6     7         ¦ OR STR
¦                                 ¦ STR       2
¦                                 ¦ AND       3
¦                                 ¦ STR       6
¦                                 ¦ AND       7
¦                                 ¦ OR  STR
¦                                 ¦ AND STR
¦                                   OUT        210
```

Cascade connection of two timers:

```
¦                                 ¦
¦                                 ¦
¦                                 ¦
+--¦ +-----------------------( )-¦ ORG         0
¦   0                         T00 ¦ OUT T/C  00 50
¦                             50s ¦
¦                                 ¦
¦                                 ¦ ORG T/C  00
+--¦ +-----------------------( )-¦ OUT T/C  01 50
¦  T00                        T01 ¦
¦                             50s ¦
¦                                 ¦
¦                                 ¦ ORG T/C  01
+--¦ +-----------------------( )-¦ OUT      210
¦  T01                        210 ¦
¦                                 ¦
```

Cascade connection of Counters and timers:

```
¦                                  ¦
¦                                  ¦
¦                                  ¦  ORG       0
+--¦ +---¦/+-------------------( )-¦  AND NOT T/C 00
¦   0    T00                  T00  ¦  OUT T/C  00 50
¦                            50 s  ¦
```

Timer output T00 gives an impulse every 50 seconds.

```
¦                         +------+  ¦
¦                         ¦C50   ¦  ¦  ORG      400
+-¦ +---------------------¦      +- ¦  STR        2
¦ T00                     ¦preset¦  ¦  OUT T/C  50 100
¦                         ¦      ¦  ¦
+-¦ +---------------------¦100   ¦  ¦
¦  2                      +------+  ¦
¦                                   ¦
¦                                   ¦
```

The counter counts 100 such intervals.

```
¦                                  ¦
¦                                  ¦  ORG T/C  50
+--¦ +-----------------------( )-  ¦  OUT      210
¦   C50                     210    ¦
¦                                  ¦
```

So output 210 goes high after 5000 s.

Bridge connection:

```
¦                                  ¦
¦                                  ¦
+--¦ +-----¦ +---------------( )-- ¦
¦   0   ¦    1               200   ¦
¦       ¦                          ¦
¦      --- 4                       ¦
¦      ---                         ¦
¦       ¦                          ¦
+--¦ +-----¦ +---------------( )-- ¦
¦   2        3               201   ¦
¦                                  ¦
```

A program block can not be drawn like this !

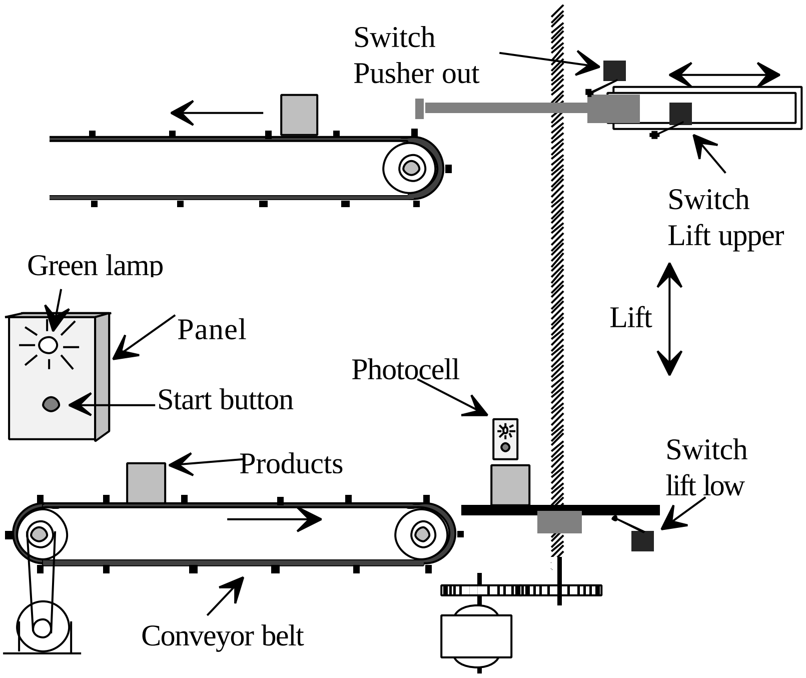
It must be redrawn in the following way:

```
¦                                  ¦
¦                                  ¦  ORG       0
+--¦ +----------¦ +----------( )-- ¦  STR       2
¦   0          ¦  1          200   ¦  AND       4
¦              ¦                   ¦  OR STR
¦              ¦                   ¦  AND       1
+--¦ +--¦ +--+                     ¦  OUT     200
¦   2    4                         ¦
¦                                  ¦
¦                                  ¦  ORG       0
+--¦ +--¦ +-----¦ +----------( )-- ¦  AND       4
¦   0    4     ¦  3          201   ¦  OR        2
¦              ¦                   ¦  AND       3
¦              ¦                   ¦  OUT     201
+--¦ +---------+                   ¦
¦   2                              ¦
```

1.3.4. Application example

Example.

When the machine is energised a green lamp shall be turned ON. It will be ON until the operator pushes the Start button.
Then the lower conveyor shall start and move until the photocell in the end of the conveyor indicates.
Then the lift shall start going up and move until it reaches the top position.
Then the pusher moves until a switch indicates that it is out.
The lift moves down (while the pusher goes back automatically) until a switch indicates that it is down.
The sequence then repeats.



This is a typical sequence program, which can be structured as the sequence below:

```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + START BUTT
¦+---++------------+
¦¦001+¦  CONVEYOR  ¦
¦+---++------------+
¦  + PHOTOCELL
¦+---++-----------+
¦¦002+¦  LIFT UP  ¦
¦+---++-----------+
¦  + LIFT UPPER
¦+---++-------------+
¦¦003+¦  PUSHER     ¦
¦+---++-------------+
¦  + PUSH OUT
¦+---++-----------+
¦¦004+¦  LIFT DOWN¦
¦+---++-----------+
¦  + LIFT LOW
+--+
```

Connect internal outputs, inputs, outputs to the following addresses:

```
000 START BUTT
001 PHOTOCELL
002 LIFT UPPER
003 PUSH OUT
004 LIFT LOW

200 GREEN LAMP
201 CONVEYOR
202 LIFT UP
203 PUSHER
204 LIFT DOWN

400   START STEP
401   STEP1
402   STEP2
403   STEP3
404   STEP4
```

```
¦ ** SEQUENCE PART FOR CONVEYOR AND LIFT MOVEMENT ******
¦
¦START LIFT  START STEP2             STEP1 ¦ 0000 ORG
000
¦BUTT  DOWN  STEP                          ¦ 0001 AND
004
+-¦ +---¦ +---¦ +---¦/+---------------( )- ¦ 0002 AND
400
¦ 000   004   400 ¦ 402               401  ¦ 0003 OR
401
¦                 ¦                        ¦ 0004 AND  NOT
402
¦STEP1            ¦                        ¦ 0005 OUT
401
¦                 ¦                        ¦
+-¦ +-------------+                        ¦
¦ 401                                      ¦
¦                                          ¦
¦PHOTO STEP1 STEP3                   STEP2 ¦ 0006 ORG
001
¦CELL                                      ¦ 0007 AND
401
+-¦ +---¦ +---¦/+---------------------( )- ¦ 0008 OR
402
¦ 001   401 ¦ 403                     402  ¦ 0009 AND  NOT
403
¦           ¦                              ¦ 0010 OUT
402
¦STEP2      ¦                              ¦
¦           ¦                              ¦
+-¦ +-------+                              ¦
¦ 402                                      ¦
¦                                          ¦
¦LIFT  STEP2 STEP4                   STEP3 ¦ 0011 ORG
002
¦UPPER                                     ¦ 0012 AND
402
+-¦ +---¦ +---¦/+---------------------( )- ¦ 0013 OR
403
¦ 002   402 ¦ 404                     403  ¦ 0014 AND  NOT
404
¦           ¦                              ¦ 0015 OUT
403
¦STEP3      ¦                              ¦
¦           ¦                              ¦
+-¦ +-------+                              ¦
¦ 403                                      ¦
¦
¦ ** DELAY BEFORE THE LIFT GOES DOWN AFTER PUSHER IS OUT****
¦                                          ¦
¦PUSH  STEP3                   +---------+ ¦ 0016 ORG
003
¦OUT                           ¦TMR   T000 ¦ 0017 AND
403
+-¦ +---¦ +--------------------¦   3.5   ¦ ¦ 0018 OUT
T000  3.5
¦ 003   403                    ¦DELAY    ¦ ¦
¦                              +---------+ ¦
¦                                          ¦
¦DELAY START                         STEP4 ¦ 0019 ORG
T000
¦      STEP                                ¦ 0020 OR
404
+-¦ +---¦/+---------------------------( )- ¦ 0021 AND  NOT
400
¦ T000¦ 400                           404  ¦ 0022 OUT
404
¦     ¦                                    ¦
¦STEP4¦                                    ¦
¦     ¦                                    ¦
+-¦ +-+                                    ¦
¦ 404                                      ¦
¦                                          ¦
¦LIFT  STEP4 STEP1                   START ¦ 0023 ORG
004
¦DOWN                                STEP  ¦ 0024 AND
404
+-¦ +---¦ +---¦/+---------------------( )- ¦ 0025 OR
400
¦ 004   404 ¦ 401                     400  ¦ 0026 OR
967
¦           ¦                              ¦ 0027 AND  NOT
401
¦START      ¦                              ¦ 0028 OUT
400
¦STEP       ¦                              ¦
+-¦ +-------¦                              ¦
¦ 400       ¦                              ¦
¦           ¦                              ¦
¦INIT       ¦                              ¦
¦PULS       ¦                              ¦
+-¦ +-------+                              ¦
¦ 967                                      ¦
¦ ***    OUTPUT CONTROL   *****            ¦
¦                                          ¦
¦START                               GREEN ¦ 0029 ORG
400
¦STEP                                LAMP  ¦ 0030 OUT
200
```

Divide the program into a **sequence part** and an **output control part**. Use e.g. self hold on the steps.

The start step is initiated at power on. (or special memory 967)

Manual / Auto control and possibility to reset with ”Reset Button”, see page 69.

1.4. Special instructions

Logic special instructions

Number	Name	Explanation	Page
FUN00	**DIF**	Positive edge detection. Gives an impulse when the logical input condition goes from false to true.	26
FUN01	**DFN**	Negative edge detection. Gives an impulse when the logical input condition goes from true to false.	26
FUN02	**IF**	"IF" IF the previous conditions true THEN the following output(s) are effected positive (corresponds to a Set function)	28
FUN03	**IFR**	"IF with Reset" As FUN 02. On top of this a reset pulse (corresponds to a Set-Reset function)	28

Instructions with more that one input

Number	Name	Explanation	Page
FUN40	**UDC**	"Up Down Counter". 16 bits of the internal output area are used (BCD-coded, 4 figures)	29
FUN45	**LATCH**	"LATCH". Set Reset of a internal output or output.	30
FUN47	**SFR**	"ShiFtRegister". Shift register. Shifts the contents of 16 specified bits in the memory from a lower to a higher address.	30

Master Control and Branch Instructions

Number	Name	Explanation	Page
FUN04	**MCS**	"Master Control Set". Defines a master control start of one or more blocks.	32
FUN05	**MCR**	"Master Control Reset". Defines the end of a Master control. Corresponds to a FUN04 (MCS)	32

Jump Instructions

Number	Name	Explanation	Page
FUN06	**JMP**	"JuMP". Jump instruction. Following instructions until FUN 07 (JMP END) are not executed.	33
FUN07	**JMP END**	"JuMP END". End of jump started by FUN 06 (JMP)	33
FUN08	**JMP LAB**	"JuMP LABel". Jump to address. Following instructions until FUN 09 with the corresponding address number are not executed. Multiple jumps with FUN 08 can take place to the same FUN 09.	
FUN09	**LAB END**	"LABel jump END". End of jump started by FUN 08 (JMP LAB)	34

Register handling instructions (store, load)

Number	Name	Explanation	Page
FUN0.	**WLOAD C**	"Word LOAD Constant". Load a 4 digit constant to AR.	35
FUN50	**WLOAD CL**	"Word LOAD Constant Lower byte". Load a constant, 0 - 255, (8 bits) to the least significant part of AR.	35
FUN10	**WLOAD**	"Word LOAD". Load (or copy) 2 addresses (including 8 bits each) in a row from inputs/outputs or internal outputs to 16 bits in AR.	35
FUN20	**WLOAD B**	"Word LoaD Bit". Load (or copy) 16 bits in a row from an address (from inputs, outputs or internal outputs) to AR.	37

FUN21	**WOUT**	Word OUT". Store 2 addresses (including 8 bits each) in a row to inputs/outputs or internal outputs from 16 bits in AR	37
FUN22	**WOUT B**	"Word OUT Bit". Store 16 bits in a row from an address (to inputs, outputs or internal outputs) from AR	38

Arithmetic instructions

Number	Name	Explanation	Page
FUN1.	**ADD C**	4 digit constant is added (BCD-addition) to the register (AR)	41
FUN11	**ADD**	4 digit word is added (BCD-addition) to the register (AR)	41
FUN61	**ADD BIN**	4 digit word is added (Binary addition) to the register (AR)	41
FUN2.	**SUB C**	4 digit constant is subtracted from the register (→ AR)	42
FUN12	**SUB**	4 digit word is subtracted from the register (→ AR)	42
FUN62	**SUB BIN**	4 digit word is subtracted (Binary subtraction) from the register (→AR)	42
FUN3.	**MUL C**	4 digit constant is multiplied by the register (→ AR)	43
FUN13	**MUL**	4 digit word is multiplied by the register (→AR)	43
FUN63	**MUL BIN**	4 digit word is multiplied (Binary multiplication) by the register (→AR)	43
FUN4.	**DIV C**	The register (AR) is divided by a 4 digit constant (→ the register (AR))	45
FUN14	**DIV**	The register (AR) is divided by a 4 digit word (→ the register (AR))	45
FUN64	**DIV BIN**	The register (AR) is divided (Binary division) by a 4 digit word (→ the register (AR))	45

Logic word instructions (masking)

Number	Name	Explanation	Page
FUN5.	**WAND C**	"Word AND Constant" Logic product of AR and a 4 digit constant	47
FUN15	**WAND**	"Word AND" Logic product of AR with a 16-bit word from 2 addresses.	47
FUN6.	**WOR C**	"Word OR Constant" Logic sum of AR and a 16-bit constant.	47
FUN16	**WOR**	"Word OR" Logic sum of AR with a 16-bit word from 2 addresses.	48
FUN85	**WNOT**	"Word Not" Logic inverting of the 16 bits in AR.	48

Compare Instructions

Number	Name	Explanation	Page
FUN7.	**CMP >=C**	"COMPare >= Constant". Compare AR with a constant. If the register (AR) >= the constant the "Carry" bit is set high.	49
FUN17	**CMP >=**	"COMPare >= ". Compare AR with a 16-bit word from 2 addresses. If AR >= the word the "Carry" bit is set high.	49
FUN8.	**CMP = C**	"COMPare = Constant". Compare AR with a constant. If AR = the constant the "Carry" bit is set high.	49
FUN18	**CMP =**	"COMPare = ". Compare AR with a 16-bit word from 2 addresses. If AR = the word the "Carry" bit is set high.	49
FUN9.	**CMP < C**	"COMPare < Constant". Compare AR with a constant. If AR < the constant the "Carry" bit is set high.	49
FUN19	**CMP <**	"COMPare < ". Compare AR with a 16-bit word from 2 addresses. If AR < the word the "Carry" bit is set high.	49

Carry Instruction

Number	Name	Explanation	Page
FUN23	**OUC**	OUt "Carry"" Sets the addressed bit to the status of the "Carry".	49

Converting Instructions (- and Binary conversion)

Number	Name	Explanation	Page
FUN24	BCD	Binary value is converted to a 4 digit BCD value.	52
FUN25	BIN	4 digit BCD value is converted to a binary value.	52

Shift Instructions

Number	Name	Explanation	Page
FUN26	SFR L	Shift of the AR register. 1 bit shift to the left (toward MSB[1])	53
FUN27	SFR R	Shift of the AR register. 1 bit shift to the right (toward LSB[2])	53

Exchange

Number	Name	Explanation	Page
FUN80	SWAP	Changes place on the most and least significant byte of AR.	54
FUN82	XCG	Changes place on the content in AR and ER.	54

Fast update of In- and Outputs

Number	Name	Explanation	Page
FUN91	REFX	Fast update of input.	55
FUN92	REFY	Fast update of output.	55

Interrupt

Number	Name	Explanation	Page
FUN93	INT	Specifies Interrupt depending on the argument.	56
FUN94	RTI	Returns to the jump origin when the interrupt routine is executed.	56

High speed counter Instructions

Number	Name	Explanation	Page
FUN96	HC	Load and store the high speed counter value.	58

Definition of inputs

Number	Name	Explanation	Page
FUN97	MODE	Specifies inputs as High speed counter, sets filter times, and defines interupt inputs etc.	60 (126)

Start- and End instructions etc.

Number	Name	Explanation	Page
FUN98	NOP	No operation (nothing is executed)	63
FUN99	END	"END". Program. The program execution continues on row 0.	63

[1] MSB = Most Significant (highest) Bit,

[2] LSB = Least Significant (Lowest) Bit,

1.4.1. Special instruction (FUN-instructions)

1.4.1.1. Logic special instructions

FUN00 DIF (Positive Edge detection)

```
¦                      +-----------+ ¦ ORG       0
+--¦ +-----------------¦FUN00  410+-¦ FUN 00  410
¦   0                  ¦           ¦ ¦
¦                      ¦           ¦ ¦
¦                      +-----------+ ¦
¦                                    ¦
```

Detection of the positive edge on input 0 gives an ”impulse” (pulse with the length of one program cycle) in the internal output 410.

Pulse diagram:

```
         +-----+    +-----+    +-----+    +-----+
 Input   ¦     ¦    ¦     ¦    ¦     ¦    ¦     ¦
         ¦     ¦    ¦     ¦    ¦     ¦    ¦     ¦
    -----+     +----+     +----+     +----+

Internal ++         ++         ++         ++
output   ¦¦         ¦¦         ¦¦         ¦
    -----++---------++---------++---------+
         ¦¦
      -->¦¦<--
      One program
      cycle
```

This impulse can e.g. be used to change from one step to another in the program or to a special memory at a certain moment. FUN00 and FUN01 are followed by an internal output (not by outputs and input.).

FUN01 DFN (Negative Edge detection)

```
¦                      +-----------+ ¦ ORG       0
+--¦ +-----------------¦FUN01  410+-¦ FUN 01  410
¦   0                  ¦           ¦ ¦
¦                      ¦           ¦ ¦
¦                      +-----------+ ¦
¦                                    ¦
```

Detection of the negative edge on input 0 gives an ”impulse” (pulse with the length of one program cycle) in the internal output 410.

```
         +-----+    +-----+    +-----+    +-----+
         ¦     ¦    ¦     ¦    ¦     ¦    ¦     ¦
 Input   ¦     ¦    ¦     ¦    ¦     ¦    ¦     ¦
    -----+     +----+     +----+     +----+

               ++         ++         ++         ++
 Internal      ¦¦         ¦¦         ¦¦         ¦
 output        ¦¦         ¦¦         ¦¦         ¦
    -----------++---------++---------++---------+
            -->||<--
            One program
            cycle
```

FUN01 is followed by an internal memory. Inputs and outputs can not be used.

FUN02 IF Conditional execution

```
                                  SET
|                                          |
|                  +--------+              |  ORG         10
+--¦ +-------------¦FUN02   +--( )--¦  FUN02
|   10             |        |   420        |  OUT        420
|                  |        |              |
|                  +--------+              |
|                                          |
|                                          |
                                 RESET
|                  +--------+              |  ORG         11
+--¦ +-------------¦FUN02   +--(/)--¦  FUN02
|   11             |        |   420        |  OUT NOT    420
|                  |        |              |
|                  +--------+              |
```

IF input 10 is high then 420 sets high (SET)

IF input 11 is high then 420 sets low. (RESET)

FUN02 creates a conditional execution of the following instruction(s). One or more such outputs/internal outputs can be effected through programming one or more OUT or OUT NOT instructions after the FUN02 instruction

FUN03 IFR (IF Reset)

Creating of Set/Reset function.

```
|                                          |
|            SET   +--------+              |  ORG         12
+--¦ +-------------¦FUN03   +--( )--¦  STR         13
|   12             |        |   421        |  FUN03
|                  |        |              |  OUT        421
|           RESET  |        |              |
+--¦ +-------------|        |              |
|   13             +--------+              |
|                                          |
|                                          |
```

When input 12 is high, 421 is set high.
When input 13 is high, 421 is set low. (reset)

The Reset input has a priority over the Set input.

FUN03 creates as well as FUN02 a conditional execution of the following instruction(s). It has one SET input and one RESET input.

Only OUT-instructions are allowed after FUN03.

1.4.2. Up/Down Counters, Latch, Shift register

FUN40 UDC Up/Down counter

Up and Down Counters, BCD coded Counters

```
|                                         |
|        Up / Down         +----------+   |
+--| |--+------------------|FUN40  400+---|
|    10                    |          |   |
|                          |          |   |
|        Count input       |          |   |
+--| |--+------------------|          |   |
|    11                    |          |   |
|                          |          |   |
|        Reset input       |          |   |
+--| |--+------------------|          |   |
|    12                    +----------+   |
```

```
ORG      10     Up / Down definition
STR      11     Count input
STR      12     Reset input
FUN40   400
```

16 Internal output addresses

```
+--------------------------------+
|       |       |       |        |  <--  increasing order
+--------------------------------+

MSD                            LSD
```

MSD=Most significant digit.
LSD=Least significant digit.

FUN40 is used to program up/down Counters These are 4 digit and they can be placed on any space among the internal outputs (16 in a row). (Outputs can not be used.)

```
--------------------------------------------+
  Up / Down input                           |
                                            |
                                            +-----------

           +-----+     +-----+     +-----+     +-----+     +-----
 Counting  |     |     |     |     |     |     |     |     |
 input     |     |     |     |     |     |     |     |     |
      -----+     +-----+     +-----+     +-----+     +-----+

   ----+                           +---+
       | Reset input               |   |
       |                           |   |
       +---------------------------+   +--------------------------
 Count value
           1          2          3  0...0     1          0
```

If you want the counter value to be retentive, place the counter addresses among the retentive internal outputs. (700-960). When the Up/Down input is high the counter counts up. When the counter passes 9999 it will continue on 0. When the Up/Down input is low the counter counts down. When the counter passes 0 it will continue on 9999.
When the reset input is high, the counter value will be reset to 0.

FUN45 LATCH Set/Reset-function

```
¦                                 ¦
¦                                 ¦
¦        Set input    +----------+ ¦  ORG          0
+--¦ +----------------¦FUN45  400+-¦  STR          1
¦   0                 ¦          ¦ ¦  FUN45      400
¦        Reset input  ¦          ¦ ¦
+--¦ +----------------¦          ¦ ¦
¦   1                 +----------+ ¦
¦                                 ¦
¦                                 ¦
```

```
     +---+       +-+                   +---------+
     ¦   ¦       ¦ ¦                   ¦         ¦
-----+   +-------+ +-------------------+         +------------
Set input
             +--------+                    +--+
             ¦        ¦                    ¦  ¦
-------------+        +--------------------+  +---------------
Reset input
     +-------+                         +---+  +---------------
     ¦       ¦                         ¦   ¦  ¦
-----+       +-------------------------+   +--+
Set/Reset output (Internal output 400)
```

The block above corresponds to following block with self hold function:

```
¦                                 ¦
+--¦ +------¦/+--------------( )--¦
¦   0   ¦     1               400 ¦
¦       ¦                         ¦
+--¦ +--+                         ¦
¦  400                            ¦
```

FUN45 can be used on internal outputs but not on outputs.

FUN47 SFR Shift register 16 bits

(see also shift instructions page 53.)

```
¦            Data input  +----------+ ¦  ORG          0
+--¦ +-------------------¦FUN47  400+-¦  STR          1
¦   0                    ¦          ¦ ¦  STR          2
¦            Shift input ¦          ¦ ¦  FUN47      400
+--¦ +-------------------¦          ¦ ¦
¦   1                    ¦          ¦ ¦
¦                        ¦          ¦ ¦
¦            Reset input ¦          ¦ ¦
+--¦ +-------------------¦          ¦ ¦
¦   2                    +----------+ ¦
```

```
   <---- Shift direction
+-------------------------------+
¦0¦1¦1¦0¦0¦0¦0¦1¦0¦1¦0¦0¦1¦0¦0¦1¦<----
+-------------------------------+

 ¦             ¦               ¦
415           408             400
```

”1” or ”0” from the Data input are shifted in when the Shift input goes from low to high.

After FUN47 the first address in the shift register is given. The shift register then will occupy the next 16 bit upwards. The shift register can be placed anywhere in the memory (but not on outputs). A shift is always from a lower to a higher address.

If you want a shift register of more than 16 positions, you can connect 2 or more registers in cascade.

Example

```
¦                                  ¦
¦                                  ¦
¦           Data input  +----------+ ¦  ORG        431
+--¦ +------------------¦FUN47  432+-¦  STR          1
¦   431                 ¦          ¦ ¦  STR          2
¦           Shift input ¦          ¦ ¦  FUN47      432
+--¦ +------------------¦          ¦ ¦
¦    1                  ¦          ¦ ¦
¦           Reset input ¦          ¦ ¦
+--¦ +------------------¦          ¦ ¦
¦    2                  +----------+ ¦
¦                                    ¦
¦           Data input  +----------+ ¦  ORG        415
+--¦ +------------------¦FUN47  416+-¦  STR          1
¦   415                 ¦          ¦ ¦  STR          2
¦           Shift input ¦          ¦ ¦  FUN47      416
+--¦ +------------------¦          ¦ ¦
¦    1                  ¦          ¦ ¦
¦           Reset input ¦          ¦ ¦
+--¦ +------------------¦          ¦ ¦
¦    2                  +----------+ ¦
¦                                    ¦
¦           Data input  +----------+ ¦  ORG          0
+--¦ +------------------¦FUN47  400+-¦  STR          1
¦    0                  ¦          ¦ ¦  STR          2
¦           Shift input ¦          ¦ ¦  FUN47      400
+--¦ +------------------¦          ¦ ¦
¦    1                  ¦          ¦ ¦
¦           Reset input ¦          ¦ ¦
+--¦ +------------------¦          ¦ ¦
¦    2                  +----------+ ¦

   <---- Shift direction
+--------------------------------+
¦0¦1¦1¦0¦0¦0¦0¦1¦0¦1¦0¦0¦1¦0¦0¦1¦<-+   Gives following shift register
+--------------------------------+  ¦   on 48 bits (400-447)
 ¦             ¦               ¦    ¦   with data input 0,
447           440             432   ¦   shift input 1
+-----------------------------------+   and reset input 2
¦
¦    <---- Shift direction
+--------------------------------+
¦0¦1¦1¦0¦0¦0¦0¦1¦0¦1¦0¦0¦1¦0¦0¦1¦<-+
+--------------------------------+  ¦
 ¦             ¦               ¦    ¦
431           424             416   ¦
+-----------------------------------+
¦
¦    <----Shift direction
+--------------------------------+
¦0¦1¦1¦0¦0¦0¦0¦1¦0¦1¦0¦0¦1¦0¦0¦1¦<---   Data from the Data input are
+--------------------------------+       shifted in to the register.
 ¦             ¦               ¦
415           408             400
```

Copyright Actron AB 1994

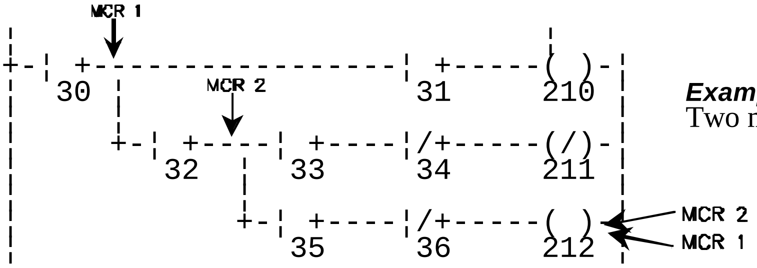
1.4.2.1. Master Control and Branch instructions

FUN04 MCS Master control set

Master Control of one or more following blocks.

FUN05 MCR Master control reset

Defines the end of the Master Control.



```
|                             |
+-| +-----------------| +-----( )-|
|  30 |                31     210 |
|     |                           |
|     +-| +----| +----|/+-----(/)-|
|        32   |  33    34     211 |
|             |                   |
|             +-| +----|/+-----( )-|
|                35    36     212 |
```

Example:
Two master control and two levels.

We change the diagram and use FUN04 and FUN05 for MCS and MCR.

```
|                        +------+ |
+-| +--------------------|FUN04 +-|   ORG      30
|  30                    |      | |   FUN04   MCS1   Master Control level 1
|                        +------+ |
|                                 |   ORG      31
+-| +-------------------------( )-|   OUT     210
|  31                         210 |
|                        +------+ |   ORG      32
+-| +--------------------|FUN04 +-|   FUN04   MCS2   Master Control level 2
|  32                    |      | |
|                        +------+ |
|                                 |   ORG      33
+-| +---|/+-------------------(/)-|   AND NOT  34
|  33    34                   211 |   OUT NOT 211
|                                 |
|                                 |   ORG      35
+-| +---|/+-------------------( )-|   AND NOT  36
|  35    36                   212 |   OUT     212
|                        +------+ |
+------------------------|FUN05 +-|   FUN05   MCR2     End of level 2
|                        |      | |
|                        +------+ |
|                        +------+ |
+------------------------|FUN05 +-|   FUN05   MCR1     End of level 1
|                        |      | |
|                        +------+ |
```

A FUN04 must always correspond to a FUN05.

Up to 4 levels can be programmed.

1.4.2.2. Jump instructions

FUN06 JMP Jump

Beginning of a Jump instruction

FUN07 JMP_END Jump End

End of a Jump instruction

```
¦                        +------+ ¦   ORG     40
+-¦ +--------------------¦FUN06 +-¦-+ FUN06                Jump start
¦  40                    ¦      ¦ ¦ ¦
¦                        +------+ ¦ ¦
¦                                 ¦ ¦ ORG     41
+-¦ +---¦ +-------------------( )-¦ ¦ AND     42
¦  41    42                   265 ¦ ¦ OUT    265
¦                                 ¦ ¦
¦                                 ¦ ¦ ORG     43
+-¦ +---¦ +-------------------( )-¦ ¦ AND     44
¦  43    44                   266 ¦ ¦ OUT    266
¦                                 ¦ ¦
¦                        +------+ ¦ ¦
+------------------------¦FUN07 +-¦ ¦ FUN07                Jump End
¦                        ¦      ¦ ¦<+
¦                        +------+ ¦
```

If the condition before FUN06 is fulfilled, the instructions until FUN07 will not be executed. If a block is passed with a jump instruction, the output status will not be effected independently of the input status.

FUN06 and FUN07 must always be in a pair.

FUN06 and FUN07 can not be programmed between FUN04 (MCS) and FUN05 (MCR).
But FUN04 and FUN05 can be programmed between FUN06 and FUN07.

FUN08 and FUN09 executes the jump faster then FUN06/FUN07.

Jump backwards is not allowed.

FUN08 JMP LAB   Jump to label

Start jump to label instruction

Jumps to the FUN09 with the same argument.

FUN09 END LAB   End Jump to label

End jump to label instruction

```
¦                        +------+ ¦      ORG      40   Jump to label 2
+-¦ +--------------------¦FUN08 +-¦->-+ FUN08     2   starts
¦ ¦40                    ¦2     ¦ ¦   ¦
¦                        +------+ ¦   ¦
¦                                 ¦   ¦ ORG      41
+-¦ +---¦ +-------------------( )-¦   ¦ AND      42
¦ ¦41   ¦42                    265 ¦   ¦ OUT     265
¦                                 ¦   ¦
¦                        +------+ ¦   ¦ ORG      45   Jump to label 2
+-¦ +--------------------¦FUN08 +-¦->-¦ FUN08     2   starts
¦ ¦45                    ¦2     ¦ ¦   ¦
¦                        +------+ ¦   ¦
¦                                 ¦   ¦ ORG      43
+-¦ +---¦ +-------------------( )-¦   ¦ AND      44
¦ ¦43   ¦44                    266 ¦   ¦ OUT     266
¦                                 ¦   ¦
¦                        +------+ ¦   ¦
+------------------------¦FUN09 +-¦   ¦ FUN09     2   End Jump
¦                        ¦2     ¦ ¦<--+                (label 2)
¦                        +------+ ¦
```

If the condition in front of FUN08 is fulfilled, the program jumps to the next FUN09 instruction with the same argument (the same label).

FUN08 and FUN09 have an argument (label) between 0 and 63.

It is not necessary to use FUN08 and FUN09 in a pair.

FUN08 and FUN09 can not be programmed between FUN04 (MCS) and FUN05 (MCR). But FUN04 and FUN05 can be programmed between FUN08 and FUN09.

If a block is passed with a jump instruction, the output status will not be effected independently of the input status.

FUN08 and FUN09 executes the jump faster than FUN06/FUN07.

.

1.4.2.3. Register handling instructions (Load /Store memory)

Concerning the difference between word handling on bits and word handling, see page 109

FUN0. WLOAD C

"Word LOAD Constant".

4 digit constant value AR.

```
FUN0.  6149     gives:
               +-------------------------------+
6149 -->       ¦0¦1¦1¦0¦0¦0¦0¦1¦0¦1¦0¦0¦1¦0¦0¦1¦ AR
               +-------+-------+-------+-------+
                  6       1      4        9
```

FUN50 WLOAD CL

"Word LOAD Constant Lower byte".

Load a constant value between 0 and 255
or 8 bits to the least significant part of AR.

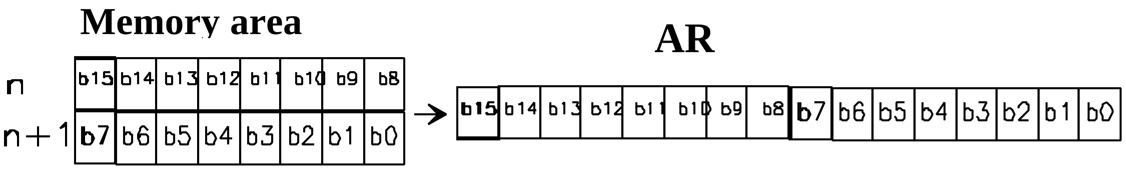
```
FUN50  254    gives:
                +-------------------------------+
 254 -->        ¦x¦x¦x¦x¦x¦x¦x¦x¦1¦1¦1¦1¦1¦1¦1¦0¦ AR
(FE Hex)        +-------+-------+-------+-------+
                                     F       E          (x = no change)
```

The most significant byte in AR is not effected.

FUN10 WLOAD

"Word LOAD".

Load (or copy) 2 addresses in a row from inputs, outputs or internal outputs (containing 8 bits each) to 16 bits in the register AR.


Memory area
AR
n
n+1
b15 b14 b13 b12 b11 b10 b9 b8
b7 b6 b5 b4 b3 b2 b1 b0
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0

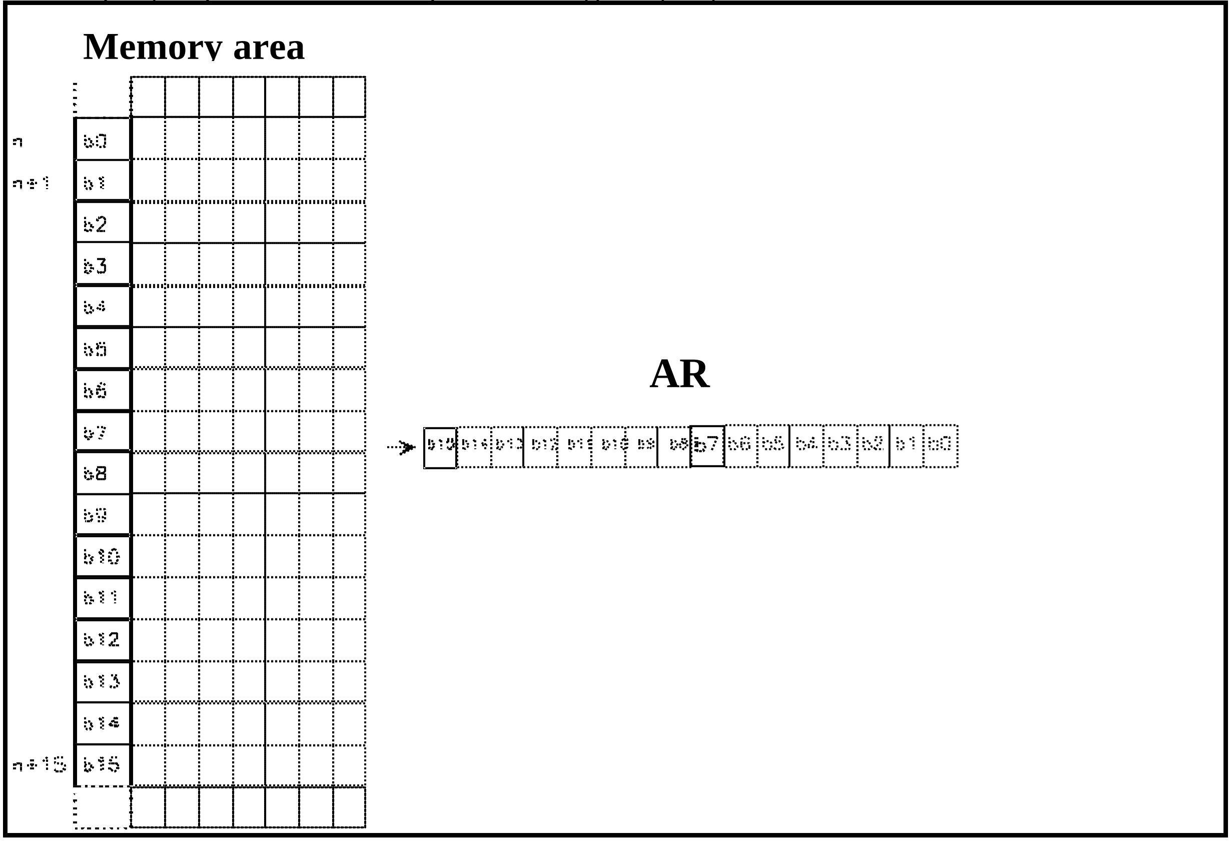
where n is the address in the memory.
FUN 10 400 loads e.g. the 8 bits in memory 400 and the 8 bits in memory 401 to the 16 bits in AR.

This type of word addressing is used to save memory, to reach T/C values and to address some special words.

FUN20 WLOAD B

"Word LoaD Bit".

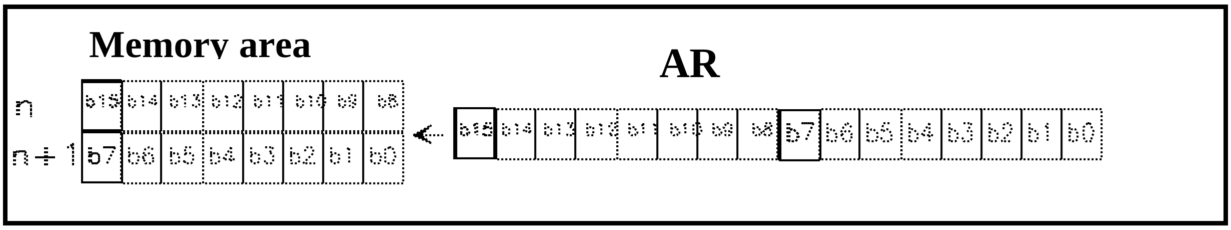
Load (or copy) 16 bits in a row
from inputs, outputs or internal outputs to the register (AR).



FUN21 WOUT

Word OUT".

Load (or copy) in 2 addresses in a row belonging to outputs or internal outputs with 8 bits each from 16 bits in the register (AR).


Memory area
AR
n
n+1
b15 b14 b13 b12 b11 b10 b9 b8
b7 b6 b5 b4 b3 b2 b1 b0
b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0

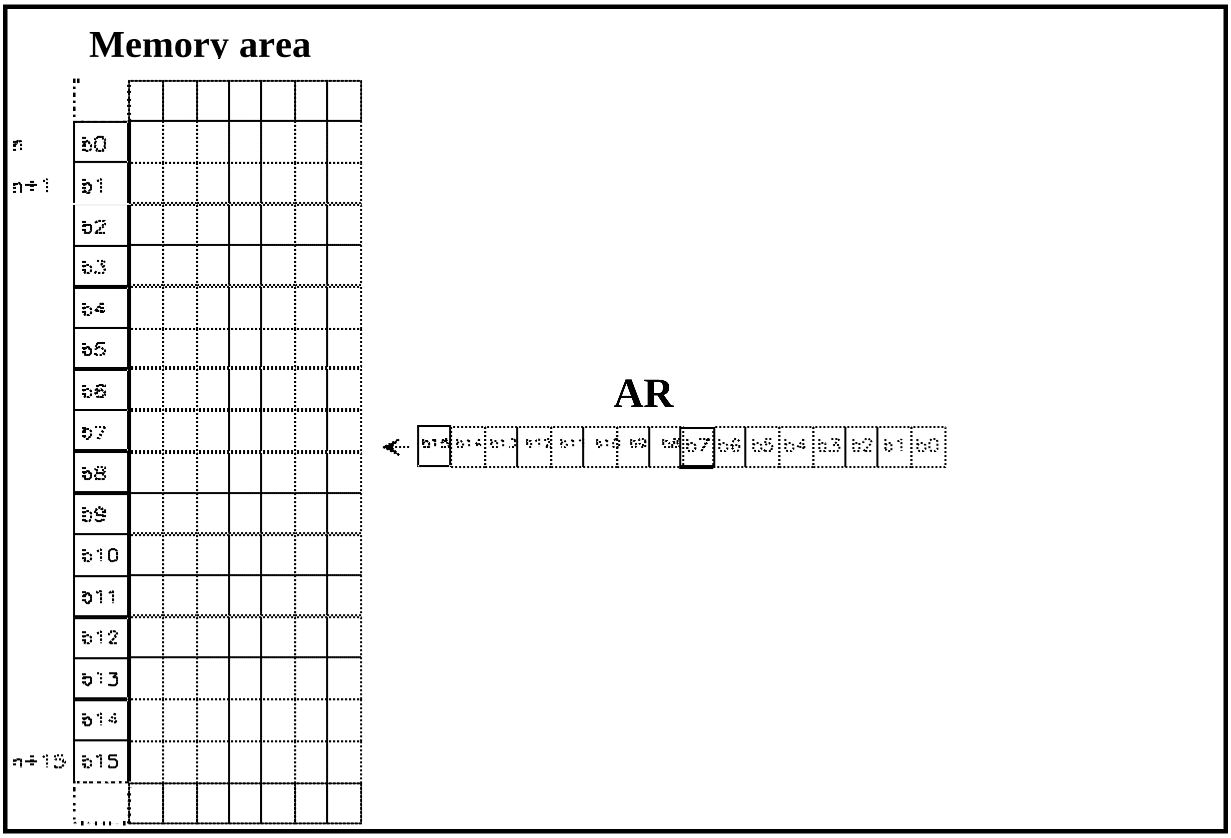
Where n is the address in the memory.
FUN 21 400 stores the 16 bits in AR in the 8 bits in internal output 400 and the 8 bits in 401.

This type of word addressing is used to save memory, to reach T/C values and to address some special words.

FUN22 WOUT B

"Word OUT Bit".

Load (or copy) in 16 bits in a row in outputs or internal outputs from the register (AR).



1.4.2.4. Handling of Timers and Counters

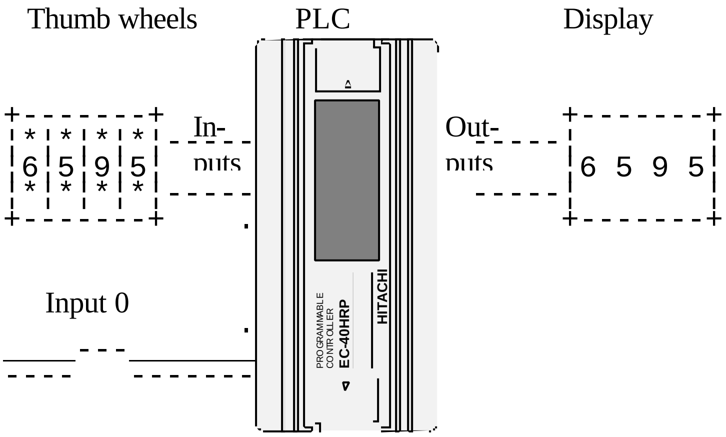
Fetching (loading) the current value of a Timers and Counters is done through using FUN10 and the address of the Timer or Counter + 100. E.g. FUN10 T/C 120 fetches the counter current value T/C 20 to AR.

Fetching (loading) the preset value of a Timers and Counters is done through using FUN10 and the address of the Timer or Counter + 200. E.g. FUN10 T/C 220 fetches the counter preset value T/C 20 to AR

Changing (storing) the current value of a Timers and Counters is done through using FUN21 and the address of the Timer or Counter + 100. E.g. FUN21 T/C 120 stores the value of AR into the current value of T/C 20.

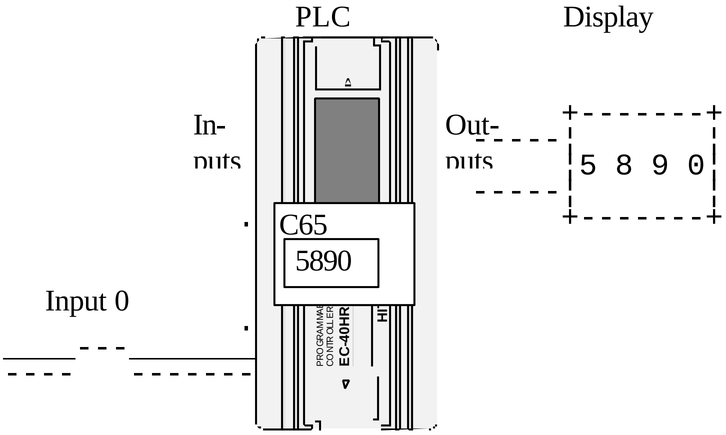
Changing (storing) the preset value of a Timers and Counters is done through using FUN21 and the address of the Timer or Counter + 200. E.g. FUN21 T/C 220 stores the value of AR into the preset value of T/C 20.

Example 1 If input 0 is On:
Read the value of 16 inputs, from no 10 and upwards, and copy these to 16 outputs from no 200 and upwards. (e.g. reading of 4 BCD coded thumb wheels and copying this value to 4 indicator displays.)



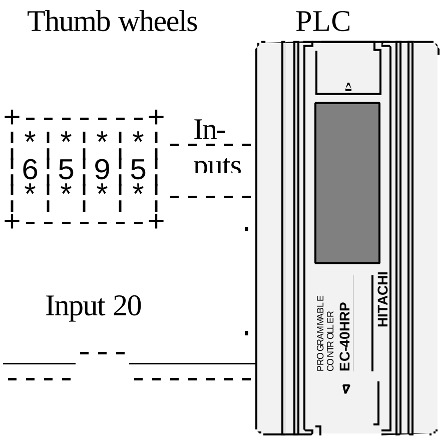
```
|                        +----------+ | ORG       00
+--| +-------------------|FUN20   10+-| FUN20     10
|  00                    |FUN22  200| | FUN22    200
|                        |          | |
|                        +----------+ |
```

Example 2 If input 0 is On:
Read the current value of Counter 65 and copy this to 16 outputs from no 200 and upwards.



```
|                        +-------------+ | ORG       00
+--| +-------------------|FUN10 T/C 165+-| FUN10     T/C 165
|  00                    |FUN22    200 | | FUN22     200
|                        |             | |
|                        +-------------+ |
```

Example 3 If input 20 is On:
Read the value of 16 inputs, from no 0 and upwards, and copy these to the preset value of Timer 210. (the value comes from 4 BCD coded thumb wheels)



```
|
|                        +-------------+ | ORG      20
+--| +-------------------|FUN20      00+-| FUN20    00
|   20                   |FUN21 T/C 210| | FUN21    T/C 210
|                        |             | |
|                        +-------------+ |
```

1.4.2.5. Arithmetic instructions

Concerning BCD code and the difference between BCD and Binary arithmetic: See appendix page 136.

1.4.2.5.1. Addition

FUN11 ADD

BCD Addition, a word is added to AR

FUN11 uses word addressing, that means the value is copied from 2 addresses (memory 400 and 401).

The BCD value in the memories 400,401 is added to the BCD value of the inputs 0-15 and the sum is copied to output 200-215 in BCD format.

```
¦                     +-----------+ ¦ ORG        20
+--¦ +----------------¦FUN20    00¦ ¦ FUN20       0    input (0-15) to AR
¦   20                ¦FUN11   400¦ ¦ FUN11     400    AR + (400,401)→ AR
¦                     ¦FUN22   200¦ ¦ FUN22     220    AR → (200 - 215)
¦                     ¦FUN23   450¦ ¦ FUN23     450    "Carry" → 450
¦                     +-----------+ ¦
```

E.g. 0 2 5 5(Word 400,401) + 0 0 0 3(Input 0-15) = 0 2 5 8(Output 200-215)

If the result is > (greater than) 9999, then the ”Carry ” ”C” will be set high and the value in the register AR will be unchanged.

FUN1. ADD C

BCD addition of a constant, a constant 0-9999 is added to AR

The BCD value in the memories 400,401 is added to the constant "2341" and the sum is copied to output 220-215 in BCD format.

```
¦                     +-----------+ ¦ ORG         0
+--¦ +----------------¦FUN10   400¦ ¦ FUN10     400    word 400,401 to AR
¦   00                ¦FUN1.  2341¦ ¦ FUN1.    2341    AR + "2341" → AR
¦                     ¦FUN22   200¦ ¦ FUN22     200    AR →(200-215)
¦                     ¦FUN23   450¦ ¦ FUN23     450    "Carry" → 450
¦                     +-----------+ ¦
```

If the result is > (greater than) 9999, then the ”Carry ” ”C” will be set high and the value in the register AR will be unchanged.

FUN61 ADD BIN

Binary addition, a word is added to AR

FUN61 uses word addressing. That means that the value is copied from 2 addresses (memory 400 and 401).

The binary value in the memories 400,401 is added to the binary value of the inputs 0-15 and the sum is copied to output 220-215 in binary format.

```
¦                     +-----------+ ¦ ORG        20
+--¦ +----------------¦FUN20    00¦ ¦ FUN20       0    input (0-15) to AR
¦   20                ¦FUN61   400¦ ¦ FUN61     400    AR + (400,401)→ AR
¦                     ¦FUN22   200¦ ¦ FUN22     200    AR →(200 - 215)
¦                     ¦FUN23   450¦ ¦ FUN23     450    "Carry" → 450
¦                     +-----------+ ¦
```

E.g. 0 0 F F(Word 400,401) + 0 0 0 3(Inp. 0-15) = 0 1 0 2(Outp 220-215) (decimal 255 + 3 = 258)

If the result becomes > 65535 the ”Carry” ”C” is set high and AR will contain the part which is < 65535 (That means that if the result is 65536 then the ”Carry” is set high and AR is 1.)

1.4.2.5.2. Subtraction

FUN12 SUB

Subtraction, a word is subtracted from AR

FUN12 uses word addressing, that means the value is copied from 2 addresses (memory 400 and 401).
The BCD value in the memories 400,401 is subtracted from the BCD value of the inputs 0-15 and the difference is copies to output 200-215 in BCD format.

```
¦                    +-----------+ ¦ ORG         0
+--¦ +---------------¦FUN20    00¦ ¦ FUN20       0    input (0-17) to AR
¦   00               ¦FUN12   400¦ ¦ FUN12     400    AR - (400,401) →
AR
¦                    ¦FUN22   200¦ ¦ FUN22     200    AR  →(200-215)
¦                    ¦FUN23   450¦ ¦ FUN23     450    "Carry" → 450
¦                    +-----------+ ¦
```

E.g. 0 2 5 8(Word 400,401) - 0 0 0 3(Inp 0-15) = 0 2 5 5(Outp 200-215)

If the result is < 0 then the ”Carry” ”C” is set high and the previous value in AR is unchanged.

FUN2. SUB C

BCD Subtraction of a constant, a constant 0-9999 is subtracted from AR

The constant "2341" is subtracted from the value in the memories 400,401 and the difference is copied to output 200-215.

```
¦                    +-----------+ ¦ ORG        40
+--¦ +---------------¦FUN10   400¦ ¦ FUN10     400      word 400,401 to AR
¦   20               ¦FUN2.  2341¦ ¦ FUN2.    2341      AR - "2341" → AR
¦                    ¦FUN22   200¦ ¦ FUN22     200      AR  →(200-215)
¦                    ¦FUN23   450¦ ¦ FUN23     450      "Carry" → 450
¦                    +-----------+ ¦
```

If the result is < 0 then the ”Carry” ”C” is set high and the previous value in AR in unchanged.

FUN62 SUB BIN

Binary Subtraction, a word is subtracted from AR

The binary value in the memories 400,401 is subtracted from the binary value of the inputs 0-15 and the difference is copied to the output 200-215 in binary format.

```
¦                    +-----------+ ¦ ORG        20
+--¦ +---------------¦FUN20    00¦ ¦ FUN20       0     input (0-17) to AR
¦   20               ¦FUN12   400¦ ¦ FUN12     400     AR - (400,401) → AR
¦                    ¦FUN22   200¦ ¦ FUN22     200     AR  →(200-215)
¦                    ¦FUN23   450¦ ¦ FUN23     450     "Carry" → 450
¦                    +-----------+ ¦
```

E.g. 0 1 0 2(Word 400,401) - 0 0 0 3(Inp. 0-15) = 0 0 F F(Outp 200-215)
If the result is < 0 then the ”Carry” ”C” is set high and AR will contain the ”two complement” of the result . (That means -1 sets the ”Carry” high and AR contains 65535, -2 will be 65534 with ”Carry” high”)

1.4.2.5.3. Multiplication

FUN13 MUL

BCD Multiplication, a word is multiplied by AR

The BCD value in the memories 400,401 is multiplied by the BCD value of the inputs 0-15 and the result is copied to output 200-215.

```
¦                         +-----------+ ¦ ORG        0
+--¦ +--------------------¦FUN20    00¦ ¦ FUN20      0     input (0-17) to AR
¦    00                   ¦FUN13   400¦ ¦ FUN13    400     AR x (400,401) →
AR
¦                         ¦FUN22   200¦ ¦ FUN22    200     AR  →(200 - 215)
¦                         ¦FUN23   450¦ ¦ FUN23    450     "Carry" → 450
¦                         +-----------+ ¦
```

FUN13 uses word addressing, that means that the value is copied from 2 addresses (memory 400 and 401).
If the result is > (greater than) 9999, then the ”Carry ” ”C” will be set high and the value in the register AR will be unchanged.

FUN3. MUL C

BCD Multiplication with constant, a constant is multiplied by AR

The constant "2341" is multiplied by the value in the memories 400,401 and result is copied to output 200-215.

```
¦                         +-----------+ ¦ ORG       40
+--¦ +--------------------¦FUN10   400¦ ¦ FUN10    400     word 400,401 to AR
¦    00                   ¦FUN3.  2341¦ ¦ FUN3.   2341     AR x "2341" → AR
¦                         ¦FUN22   200¦ ¦ FUN22    200     AR  →200 -215
¦                         ¦FUN23   450¦ ¦ FUN23    450     "Carry" → 450
¦                         +-----------+ ¦
```

If the result is > (greater than) 9999, then the ”Carry ” ”C” will be set high and the value in the register AR will be unchanged.

FUN63 MUL BIN

Binary Multiplication (32 bits multiplication)

A binary multiplication between the content of AR and the value of a 2 byte word is executed.
The product will be stored in AR and ER. The least significant bits will be stored in AR and the most significant in ER..

16 bits binary		16 bits binary		16 most significant bits		16 least significant
AR	*	**2 bits word**	=	**ER**	+	**AR**

If the result is > 65535 then the "Carry" bit is set high.

Example: The binary value in the memories 400,401 is multiplied by the binary value on the inputs 0-15 and the result will be copied to output 200-215 (Least significant part)

```
¦                  +-----------+ ¦ ORG       0
+--¦ +-------------¦FUN20    00¦ ¦ FUN20     0   input (0-17) to AR
¦    00            ¦FUN63   400¦ ¦ FUN63   400   AR * (400,401) → AR and ER.
¦                  ¦FUN22   200¦ ¦ FUN22   200   AR → (200 - 215) Least significant
¦                  ¦FUN82      ¦ ¦ FUN82         Exchange content of AR and ER.
¦                  ¦FUN22   220¦ ¦ FUN22   220   AR → (220 - 227) Most significant
¦                +------------+ ¦
```

1.4.2.5.4. Division

FUN14 DIV

BCD Division

The value of the inputs 0-15 is divided by the value in memory 400,401 and quotient is copied to output 200-215.

```
¦                         +-----------+ ¦ ORG         0
+--¦ +--------------------¦FUN20    00¦ ¦ FUN20       0   input (0-15) to AR
¦    00                   ¦FUN14   400¦ ¦ FUN14     400   AR / (400,401) → AR
¦                         ¦FUN22   200¦ ¦ FUN22     200   AR  →(200 -215)
¦                         ¦FUN23   450¦ ¦ FUN23     450   "Carry" → 450
¦                         +-----------+ ¦
```

FUN14 uses word addressing, that means that the value is copied from 2 addresses (memory 400 and 401).
The rest is ignored. The ”Carry” is set high if there is a division by 0.

FUN4. DIV C

BCD Division by constant (0-9999)

The value of the inputs 0-15 is divided by the value in memory 400,401 and quotient is copied to output 200-215.
The value of the memories 400,401 is divided by the constant "2341" and the quotient is copied to output 200-215.

```
¦                         +-----------+ ¦ ORG        10
+--¦ +--------------------¦FUN10   400¦ ¦ FUN10     400    word 400,401 to AR
¦    10                   ¦FUN4.  2341¦ ¦ FUN4.    2341    AR / "2341" → AR
¦                         ¦FUN22   200¦ ¦ FUN22     200    AR  →200 - 215
¦                         ¦FUN23   450¦ ¦ FUN23     450    "Carry" → 450
¦                         +-----------+ ¦
```

If AR is divided by 0 the Carry ”C” is set high and the previous value (before the FUN4 operation) remains.
The rest is ignored. The ”Carry” is set high if there is a division by 0.

FUN64 DIV BIN

Binary Division with remainder

The content of AR (16 bits binary) is divided by the content in the addressed 2 byte word. The quotient is copied to AR and the remainder to ER.

16 bits binary 16 bits binary 16 bits quotient binary 16 bits remainder
binary

AR / **2 bytes word** = **AR** **ER**

Example: The value of the memories 400,401 is divided binary by the value of the inputs 0-15 and the binary quotient is copied to 200-215 and the remainder to 500-515.

```
¦                     +-----------+ ¦ ORG    20
+--¦ +----------------¦FUN10   400¦ ¦ FUN10 400   400,401 to AR
¦    20               ¦FUN64     0¦ ¦ FUN64   0   AR / (Input 0-15)bin → AR
¦                     ¦FUN22   200¦ ¦ FUN22 200   AR →(200-215) The quotient
¦                     ¦FUN82      ¦ ¦ FUN82       Swap AR and ER.
¦                     ¦FUN22   500¦ ¦ FUN22 500   AR  →(500 -515) The
remainder
```

```
¦                  +------------+ ¦
```

"Carry" "C" is set high if you divide by 0.

1.4.2.6. Logic word instructions (Masking instructions)

FUN5. WAND C

"Word AND Constant" The Logic product of AR and a BCD constant 0-9999.

Each bit in AR is compared with the corresponding bit in the mask. 1 and 1 gives 1 to AR. All other combinations give 0.

0	**1**	**1**	0	0	0	0	1	0	1	0	0	1	0	0	**1**	**AR**

Masked by constant

0	**1**	**1**	1	0	1	0	0	0	0	0	0	0	0	0	**1**	**Contant 7 4 0 1**

Gives ↓ **FUN5. WAND C 7401**

0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**AR**

OBSERVE! As there is a limitation to the figures 0-9 all types of masking can not be performed by this instruction. For these cases, see FUN15 and FUN 55 below.

FUN15 WAND

"Word AND" Logic product of 16 bits in AR and 16 bits in a word from 2 addresses.

0	**1**	**1**	0	0	0	0	1	0	1	0	0	1	0	0	**1**	**AR**

Masked by word

0	**1**	**1**	1	0	1	0	0	0	0	0	0	0	0	0	**1**	**Word 400, 401**

Gives ↓ **FUN15 WAND 400**

0	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**AR**

FUN6. WOR C

”Word OR Constant" Logic sum of AR and a 4 digit BCD constant (0-9999).

0	**1**	**1**	0	0	0	0	1	0	1	0	0	1	0	0	**1**	**AR**

Masked by constant

0	**1**	**1**	1	0	1	0	0	0	0	0	0	0	0	0	**1**	**Contant 7 4 0 1**

Gives ↓ **FUN6. WOR C 7401**

0	**1**	**1**	**1**	0	**1**	0	**1**	0	**1**	0	0	**1**	0	0	**1**	**AR**

OBSERVE! As there is a limitation to the figures 0-9 all types of masking can not be performed by this instruction. For these cases, see FUN16 and FUN 56 below.

FUN16 WOR

"Word OR" Logic sum of 16 bits in AR and 16 bits in a word from 2 addresses.

0	**1**	**1**	0	0	0	0	1	0	1	0	0	1	0	0	**1**	**AR**

Masked by word

0	**1**	**1**	1	0	1	0	0	0	0	0	0	0	0	0	**1**	**Word 400, 401**

↓

Gives

FUN16 WOR 400

0	**1**	**1**	**1**	0	**1**	0	**1**	0	**1**	0	0	**1**	0	0	**1**	**AR**

FUN85 WNOT

"Word NOT" Logic inversion of the 16 bits in AR.

Changes "1" to "0" and "0" to "1" in all bits of AR..

0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	**AR**

Gives

↓

FUN85 WNOT

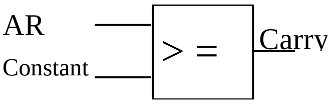
1	0	0	0	1	0	1	1	1	1	1	1	1	1	1	0	**AR**

1.4.2.7. Compare instructions

Compare AR with a constant or a value (word address)

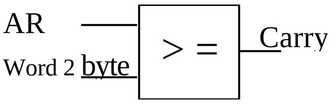
FUN7. CMP >=C

"CoMPare >= Constant". Compare AR with a constant. If AR >= the constant the Carry-bit is set high.



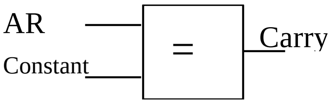
FUN17 CMP >=

"CoMPare >= ". Compare AR with a 16-bit word from 2 addresses. If AR >= the word the Carry-bit is set high.



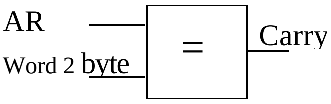
FUN8. CMP = C

"CoMPare = Constant". Compare AR with a constant. If AR = the constant the "Carry" bit is set high.



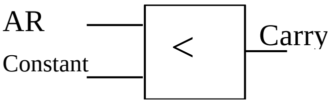
FUN18 CMP =

"CoMPare = ". Compare AR with a 16-bit word from 2 addresses. If AR = the word the "Carry" is set high.



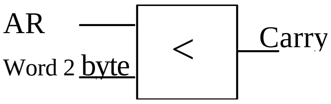
FUN9. CMP < C

"CoMPare < Constant". Compare AR with a constant. If AR < the constant the "Carry" bit is set high.



FUN19 CMP <

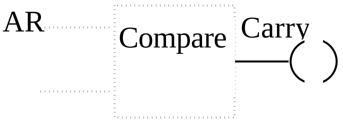
"CoMPare < ". Compare AR with a 16-bit word from 2 addresses. If AR < the word the Carry-bit is set high.



1.4.2.8. The Carry instruction

FUN23 OUC

"OUt "Carry"" Sets the bit, which is addressed, to the same status as the ”Carry”.



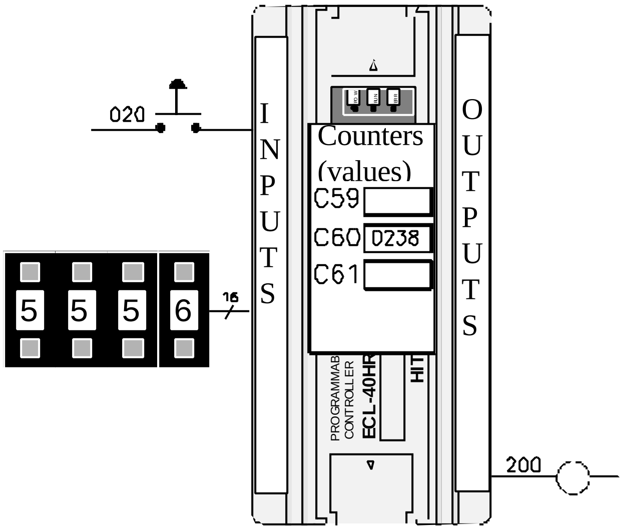
Comparison example.

Example 1

Compare Thumb wheel with Counters.

If input 20 is ON:

Read the value of 16 inputs from no. 0 and upwards and compare this to value with the current value of Counter 60. If the value is >= (greater than or equal to) the current counter value, then the output 200 goes high. (Get the value e.g. from 4 thumb wheels)



```
¦                        +----------+ ¦ ORG       20
+--¦ +-------------------¦FUN20   00+-¦ FUN20     00
¦   ¦20                  ¦FUN17 C160¦ ¦ FUN17   C160
¦                        ¦FUN23  200¦ ¦ FUN23    200
¦                        +----------+ ¦
¦                                     ¦
```

Example 2

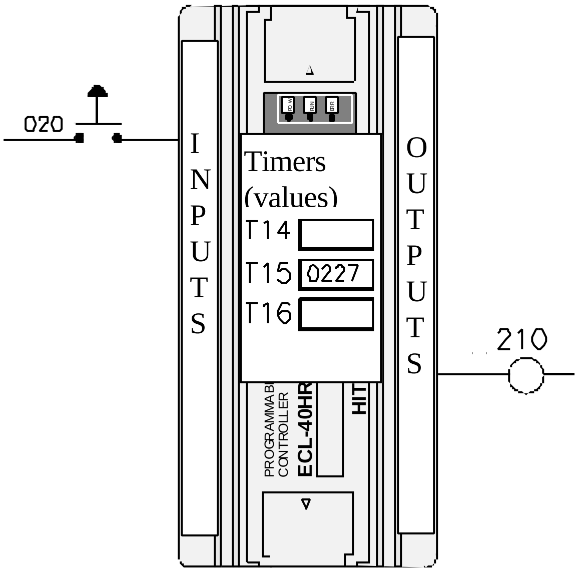
Compare timer with the value of a 2 byte memory

:

If input 20 is high:

Read a value from address no. 400 and 401 and compare this value with the current value of timer 15.

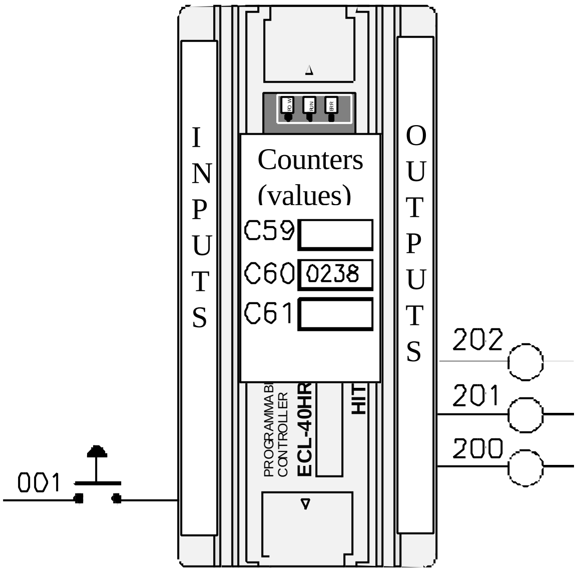
If the timer value is < (less than) the value of the internal outputs the output 210 goes high.



```
¦                        +-------------+ ¦ ORG       20
+--¦ +-------------------¦FUN10 T/C 115+-¦ FUN10 T 115
¦   ¦20                  ¦FUN19     400¦ ¦ FUN19   400
¦                        ¦FUN23     210¦ ¦ FUN23   210
¦                        +-------------+ ¦
¦                                        ¦
```

Example 3

If input 1 is ON:
Compare the counter value of counter 60 with different values of constants. Set different outputs high when these are passed.



```
 |              +-------------+ |  ORG         001
 +--| +-------|FUN10 T/C 160+-|  FUN10 T/C 160      Fetch the counter value
 |   1         |FUN7.      200|  |  FUN7.       200      >= 200 ?
 |             |FUN23      200|  |  FUN23       200     If >=: then output 200 ON
 |             |FUN7.     1450|  |  FUN7.      1450      >= 1450 ?
 |             |FUN23      201|  |  FUN23       201      If >=: then output 201 ON
 |             |FUN7.     5050|  |  FUN7.      5050      >= 5050 ?
 |             |FUN23      202|  |  FUN23       202      If >=: then output 202 ON
 |             +-------------+  |
 |                              |
```

1.4.2.9. Converting instructions (BCD- and Binary conversion etc.)

FUN24 BCD

Binary value is converted to 4 digit BCD value.

FUN25 BIN

4 digit BCD value is converted to a binary value.

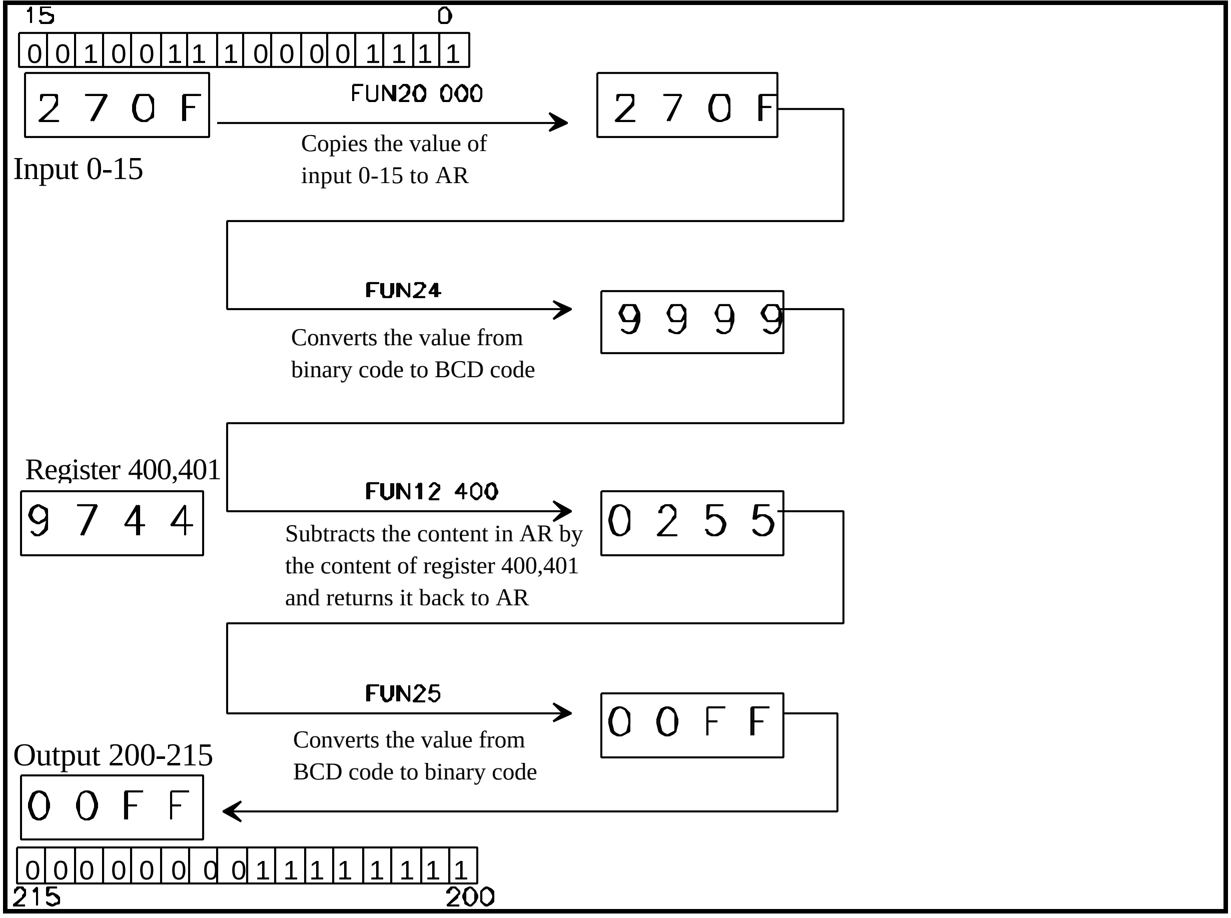
Example:

Fetch a binary value from input 0-15.

Convert the BCD value in order to subtract the content in register 400,401. Thereafter convert the value again to binary and copy this to output 200-215.

```
 ¦                                +-----------+ ¦ ORG     990
 +--¦ +---------------------------¦FUN20    00¦ ¦ FUN20    20     input (20-35) → AR
 ¦   990                          ¦FUN24      ¦ ¦ FUN24           AR binary → BCD
 ¦                                ¦FUN12   400¦ ¦ FUN12   400     AR - (400,401) → AR
 ¦                                ¦FUN25      ¦ ¦ FUN25           AR BCD → binary
 ¦                                ¦FUN22   200¦ ¦ FUN22   200     AR → output (200-215)
 ¦                                +-----------+ ¦
```

If input 0-15 is:	0010 0111 0000 1111 bin	= 2 7 0 F HEX
and register 400,401 is:	1001 0111 0100 0100 bin	= 9 7 4 4 BCD


15
0
0 0 1 0 0 1 1 1 0 0 0 0 1 1 1 1
2 7 0 F
Input 0-15
FUN20 000
Copies the value of input 0-15 to AR
2 7 0 F
FUN24
Converts the value from binary code to BCD code
9 9 9 9
Register 400,401
9 7 4 4
FUN12 400
Subtracts the content in AR by the content of register 400,401 and returns it back to AR
0 2 5 5
FUN25
Converts the value from BCD code to binary code
0 0 F F
Output 200-215
0 0 F F
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
215
200

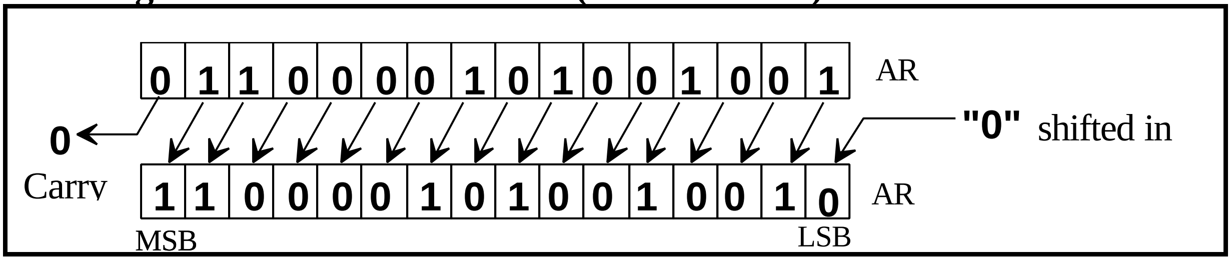
1.4.2.10. Shift instructions (see also FUN47)

(see also FUN47 on page 30)

These instructions do not have an argument.

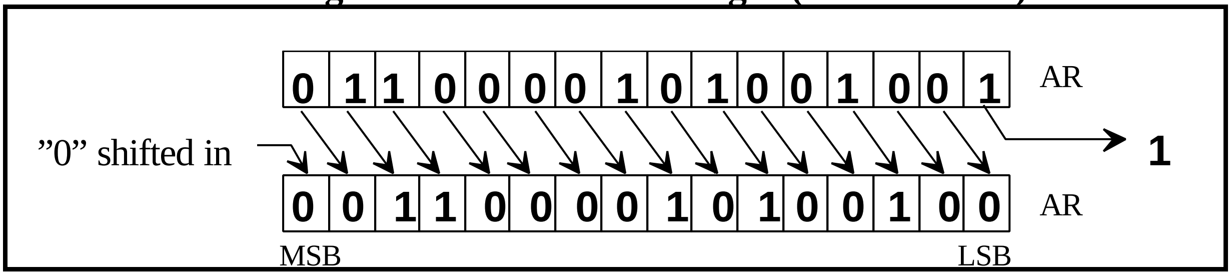
FUN26 SFR L

Shift register. 1 bit shift to the left (towards MSB)



FUN27 SFR R

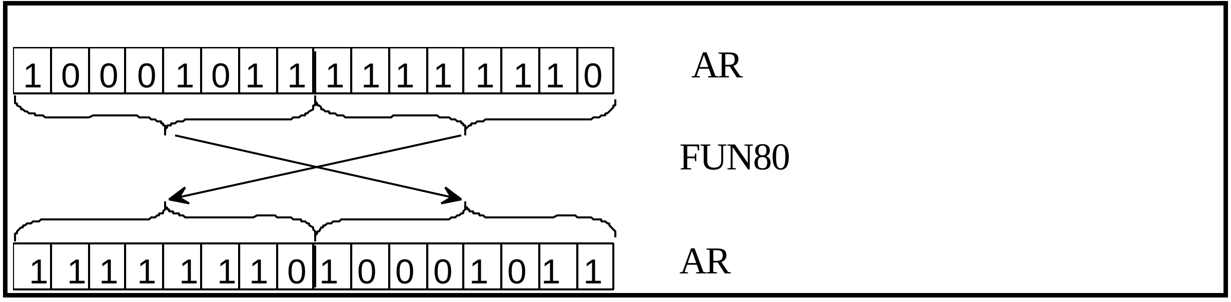
Shift register. 1 bit shift to the right (towards LSB)



1.4.2.11. Exchange instructions

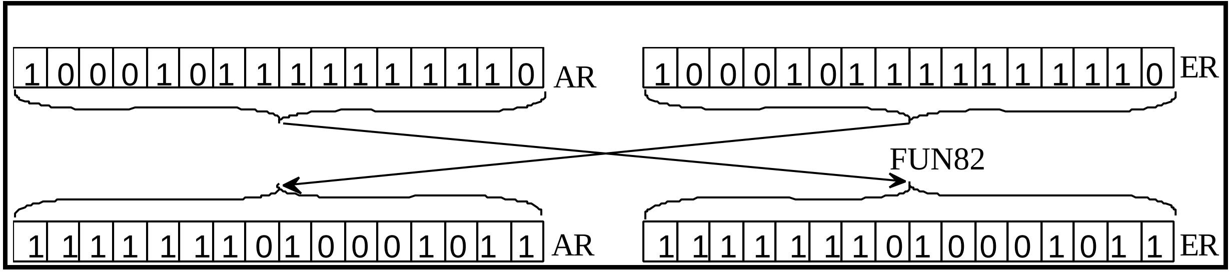
FUN80 SWAP

Exchanges the place of the most and the least significant byte of AR



FUN82 XCG

Exchange the place of the content in AR and ER.



1.4.2.12. Fast update of I/O

FUN91 REFX (Refresh IN)

Fast update of input:

Fetches directly the status of an input without waiting for the I/O-updating in the beginning of an ordinary program cycle. This can be useful when a fast response is needed or when the program cycle is long.

REFX does not use a start condition. (No ORG instruction)

FUN92 REFY (Refresh OUT)

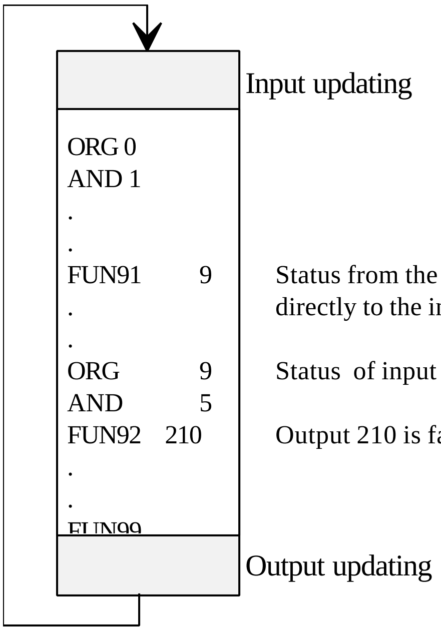
Fast update of output.

Copies the status of an internal output to an output, which is addressed without waiting for the I/O-updating after an ordinary program cycle. This can be useful when a fast response is needed or when the program cycle is long.

REFX and REFY can be used multiple times in a program for the same address to minimise the response time.

Normal program cycle. (See appendix on page 139.)

Example: A program cycle with fast update:



```
ORG 0
AND 1
.
.
FUN91     9     Status from the physical input ”9” is copied
.               directly to the input memory.
.
ORG       9     Status  of input 9 is used in a logic combination for output 210.
AND       5
FUN92   210     Output 210 is fast updated.
.
.
FUN99
```

1.4.2.13. Interrupt

An ”interrupt” means that the program execution is temporarily stopped and an ”interrupt” routine , which is placed after the ordinary program, is executed. After this routine is executed the ordinary program execution continues from where it was interrupted.

FUN93 INT (INTerrupt)

Interrupt

FUN 93 has an argument between 0 and 2, which defines the type of interrupt.:

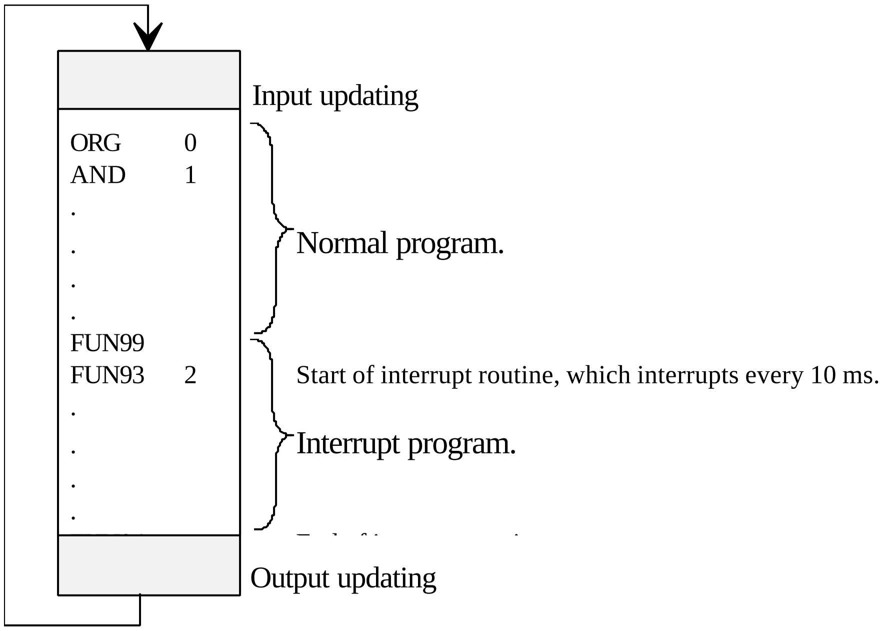
FUN 93 0 means	Interrupt when the High speed counter value = the preset value.
FUN 93 1 means	Interrupt when the interrupt input (Input No. 3) goes high or low. Normally this happens when the input goes high. (positive edge) You can change the interrupt condition to a negative edge through using (FUN 97 0) (see FUN 97)
FUN 93 2 means	Interrupt is activated every 10 ms.

FUN94 RTI (ReTurn from Interrupt)

Return from interrupt:

Specifies end of interrupt routine and performs a return jump. (no argument).

Example: A program cycle with interrupt:



```
ORG       0
AND       1
.
.
.
.
FUN99
FUN93     2
.
.
.
.
```

1.4.2.14. Definition of inputs

FUN97 MODE (Condition)

Specifies inputs like High speed counters, filter times, interrupt inputs etc.

This instruction can be written anywhere in the program. It shall be written without a start condition (no ORG instruction before).
If no FUN97 is written in the program there will be a default mode.

FUN 97 has an argument 0 to 5, which defines type of input specification:

FUN 97 0 means that interrupt from input 3 will occur on a negative edge (positive edge is default.)

```
 ¦                        +---------+ ¦   FUN97  0
 +------------------------¦FUN97  0 +-¦
 ¦                        +---------+ ¦
```

FUN 97 1 means that a High speed counter on input 0-2 will be a one phase type (up counting on input 0 positive edge, down counting on input 1 negative edge and reset of the counter on input 2)

```
 ¦                        +---------+ ¦   FUN97  1
 +------------------------¦FUN97  1 +-¦
 ¦                        +---------+ ¦
```

FUN 97 2 means that a High speed counter on input 0-2 will be a 2-phase type (phase discriminator) (up counting and down counting is defined by the 90 degree phase difference between input 0 and input and reset of the counter value on input 2)

```
 ¦                        +---------+ ¦   FUN97  2
 +------------------------¦FUN97  2 +-¦
 ¦                        +---------+ ¦
```

FUN 97 3 means that the filter time on input 0 - 3 (High speed counter and the interupt input) is changed to 0,25 ms. This filter time can be used if the High speed counter works below 1.5 kHz.

```
 ¦                        +---------+ ¦   FUN97  3
 +------------------------¦FUN97  3 +-¦
 ¦                        +---------+ ¦
```

FUN 97 4 means that the filter time on input 0-7 is changed to 16 ms. This filter time can be used e.g. if there is electrical noise on the inputs which could be detected as a signal if the normal filter of 4 ms is used.

```
 ¦                        +---------+ ¦   FUN97  4
 +------------------------¦FUN97  4 +-¦
 ¦                        +---------+ ¦
```

FUN 97 5 means that the filter time on input 0-7 is changed to 0 ms. This filter time can be used e.g. if the High speed counter works on frequencies above 1.5 kHz.

```
 ¦                        +---------+ ¦   FUN97  5
 +------------------------¦FUN97  5 +-¦
 ¦                        +---------+ ¦
```

More info, see under analogue timer for ECL page 126

If a short filter time only is needed for some inputs, a filter can be implemented by the program on the others as below:

```
 ¦                        +---------+ ¦
 +---¦ +------------------¦T/C   0  +-¦  ORG 3
```

```
¦    3                  ¦0.01 s   ¦ ¦  OUT T/C 0  0.01  .
¦                       +---------+ ¦    .
```

1.4.2.15. High speed counter instruction

FUN96 HC

Loading and storing of the High speed counter values. More info, see page 61.

FUN 96 has an argument between 0 and 2, which defines what shall done.

FUN 96 0 means that the current value is copied to AR and ER.

```
      Registers                   High speed counter
     +-------+                  +---------------+
ER   ¦3¦6¦1¦7¦  <----+          ¦5¦6¦8¦2¦1¦0¦0¦0¦  Preset value  ( 8 digits)
     +-------+       ¦          +---------------+
     +-------+       ¦          +---------------+
AR   ¦2¦9¦6¦0¦  <-----------    ¦3¦6¦1¦7¦2¦9¦6¦0¦  Current value  ( 8 digits)
     +-------+                  +---------------+
```

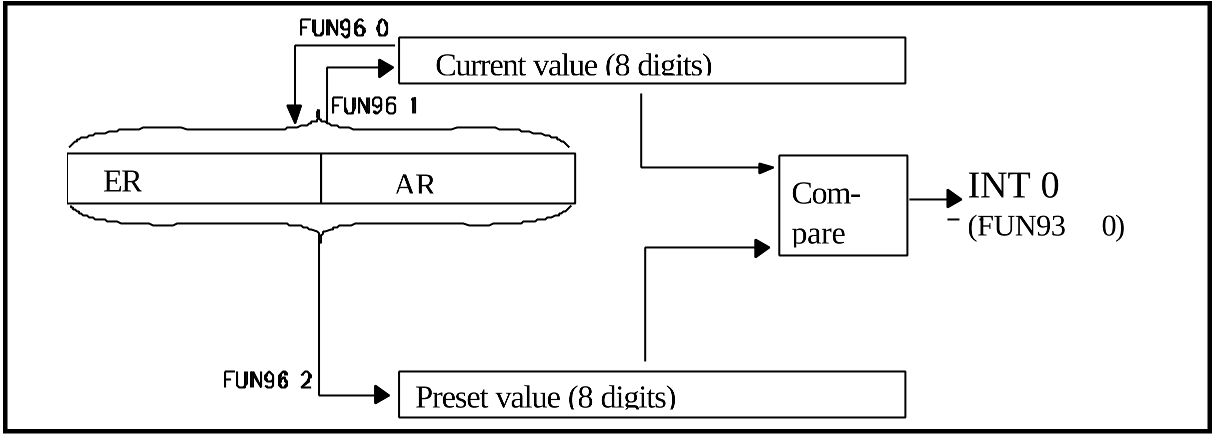
FUN 96 1 means that AR and ER are copied to the current value.

```
         Registers                High speed counter
     +-------+                  +---------------+
ER   ¦3¦6¦1¦7¦   ----+          ¦5¦6¦8¦2¦1¦0¦0¦0¦  Preset value  ( 8 digits)
     +-------+       ¦          +---------------+
     +-------+       ¦          +---------------+
AR   ¦2¦9¦6¦0¦   ----------->   ¦3¦6¦1¦7¦2¦9¦6¦0¦  Current value  ( 8 digits)
     +-------+                  +---------------+
```

FUN 96 2 means that AR and ER are copied to the preset value.

```
         Registers                High speed counter
     +-------+                  +---------------+
ER   ¦5¦6¦8¦2¦   ----------->   ¦5¦6¦8¦2¦1¦0¦0¦0¦  Preset value  ( 8 digits)
     +-------+       ¦          +---------------+
     +-------+       ¦          +---------------+
AR   ¦1¦0¦0¦0¦   ----+          ¦3¦6¦1¦7¦2¦9¦6¦0¦  Current value  ( 8 digits)
     +-------+                  +---------------+
```

In this way the high speed counter is preset and loaded. It could also be an interrupt when the current value is equal to the preset value. (see FUN 93 page 56).
In this interrupt routine you can use the refresh instructions (FUN 91 page 55 and FUN 92 page 55) to give an optimal response on the outputs.



1.4.2.16. High speed counter programming

To program the High speed counter on series EC, use the following instructions:

FUN 97 MODE to define type of counter and filter time.

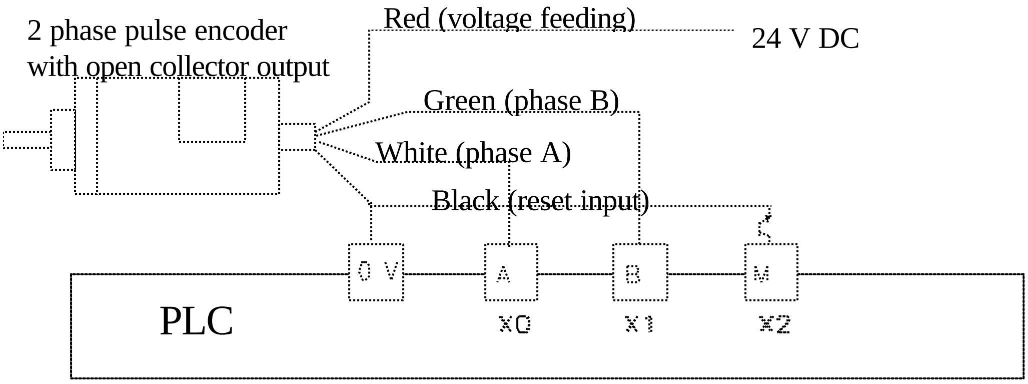
FUN 93 INT to detect when the counter has reached its preset value and fast execute a relevant action in the program.

FUN 96 HC to copy the current value and the preset value between the program and the high speed counter.

See also these instruction on page 60, 56 and page 58.

Example how to use a high speed counter:
Input 0-2 shall be used as a 2 phase high speed counter. A counting frequency of 1.5 kHz is enough. Preset the counter with "105000". When the counter has reached this value a certain program part shall be executed. (the routine which begins with FUN 93), which effects output 210.

```
¦                   +-----------+ ¦
+-------------------¦FUN97     2+-¦FUN97  2      Defines a 2 phase counter
¦                   ¦FUN97     3¦ ¦FUN97  3      Defines the filter time of
¦                   +-----------+ ¦              input 0-3 to 0,25 ms
¦                                 ¦
¦                   +-----------+ ¦
+-¦ +---------------¦FUN0.    10+-¦FUN0. 10      10 → AR
¦                   ¦FUN82      ¦ ¦FUN82         AR ↔ ER (10 → ER)
¦                   ¦FUN0.  5000¦ ¦FUN0.  5000   5000 → AR
¦                   ¦FUN96     2¦ ¦FUN96  2      Transfers to the preset value
¦                   +-----------+ ¦
¦                                 ¦
¦    End of normal program
¦                                 
¦                   +-----------+ ¦
+-------------------¦FUN93     0+-¦FUN93  0      Interupt when the counter =
¦                   +-----------+ ¦              = preset value
¦    Interrupt program            ¦
¦                                 ¦
¦                                 ¦
+--[ logic condition ]-----( )----¦OUT 210       Output 210 is effected by a
¦                                 ¦              condition
¦                                 ¦
¦                   +-----------+ ¦
+-------------------¦FUN94      +-¦FUN94         End of interrupt program.
¦                   +-----------+ ¦
¦                                 ¦
```


2 phase pulse encoder
with open collector output
Red (voltage feeding)
24 V DC
Green (phase B)
White (phase A)
Black (reset input)
0 V
A
B
M
PLC
X0
X1
X2

1.4.2.17. End instructions etc.

FUN99 END

Program End

```
.                           .   .
¦                           .   .
¦                        +-------+ ¦   FUN99
+-----------------------¦FUN99   +-¦
¦                        ¦       ¦ ¦
¦                        +-------+ ¦
¦                                  ¦
```

FUN99 defines the program end. Normally it is not needed to program this instruction as the program is filled with FUN99 when the program is started. E.g. when testing a part of the program FUN99 can be inserted before the test program when it is not used. This part of the program will not be executed then.

It is not necessary to program a FUN99 before a subroutine or an interrupt routine.

FUN98 NOP

No operation

```
.                                  .   .
¦                                  ¦   .
¦                        +-------+ ¦  FUN98
+-----------------------¦FUN98   +-¦  FUN98
¦                        ¦FUN98  ¦ ¦   .
¦                        +-------+ ¦   .
.                                  .
```

FUN98 does not perform anything and it not necessary to use it.

The reason for its existence is that if there is a NOP instruction in a long program it is a little faster to insert an instruction before the NOP. The following ”non-NOP” instruction then keeps its original address.

1.4.3. Program example with special instructions.

Up /down Counters with preset and position comparison:

```
¦                             +----------+ ¦
¦        Up / Down input      ¦          ¦ ¦
+--¦ +------------------------¦FUN40  400+-¦
¦   10                        ¦          ¦ ¦
¦        Count input          ¦          ¦ ¦
+--¦ +------------------------¦          ¦ ¦
¦   11                        ¦          ¦ ¦
¦        Reset input          ¦          ¦ ¦
+--¦ +------------------------¦          ¦ ¦
¦   12                        +----------+ ¦
¦                             +----------+ ¦
+--¦ +------------------------¦FUN00  500+-¦
¦   13                        ¦          ¦ ¦
¦                             ¦          ¦ ¦
¦                             +----------+ ¦
¦                                          ¦
¦                             +----------+ ¦
+--¦ +------------------------¦FUN0. 4000+-¦
¦  500                        ¦FUN22  400¦ ¦
¦                             ¦          ¦ ¦
¦                             +----------+ ¦
¦                                          ¦
¦                                          ¦
¦                             +----------+ ¦
+--¦ +------------------------¦FUN20  400+-¦
¦  990                        ¦FUN8. 2000¦ ¦
¦                             ¦FUN23  200¦ ¦
¦                             +----------+ ¦
¦                                          ¦
```

```
ORG      10
STR      11
STR      12
FUN40   400
```

Up/Down Counters on the addresses 400 - 415

```
ORG      13
FUN00   500
```

Edge detection on the preset input (input 3).

```
ORG      500
FUN0.   4000
FUN22    400
```

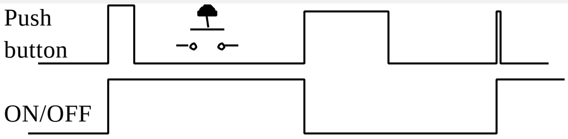
The Counter is preset on the edge of the preset input. The preset value is 4000 in BCD.

```
ORG      990
FUN20    400
FUN8.   2000
FUN23    200
```

The current value is compared to the preset value 2000.
When the preset is = 2000 the output 200 goes high

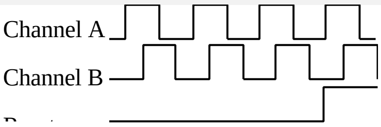
Toggling function:

Example. The momentary push button gives a toggling function, ON and OFF.



```
¦ *** Push button (momentary), which gives a toggling function  ****
¦     every other ON, every other OFF
¦                                                 ¦
¦PB TO                            +---------+ ¦ 0000 ORG             000      PB PB TOGGL
¦GGL                              ¦FUN00 400¦ ¦ 0001 FUN00 DIF       400      PB   EDGE
+-¦ +-----------------------------¦DIF      ¦ ¦
¦ 000                             ¦PB   EDGE  ¦
¦                                 +---------+ ¦
¦                                             ¦
¦ON/   PB                          ON/        ¦ 0002 ORG             401      ON/  OFF
¦OFF   EDGE                        OFF        ¦ 0003 AND  NOT        400      PB   EDGE
+-¦ +---¦/+------------------------( )-       ¦ 0004 STR NOT         401      ON/  OFF
¦ 401   400 ¦                       401       ¦ 0005 AND             400      PB   EDGE
¦           ¦                                 ¦ 0006 OR STR
¦ON/   PB   ¦                                 ¦ 0007 OUT             401      ON/  OFF
¦OFF   EDGE ¦                                 ¦
+-¦/+---¦ +-+                                 ¦
¦ 401   400                                   ¦
¦                                             ¦
```

Pulse counting with normal inputs.



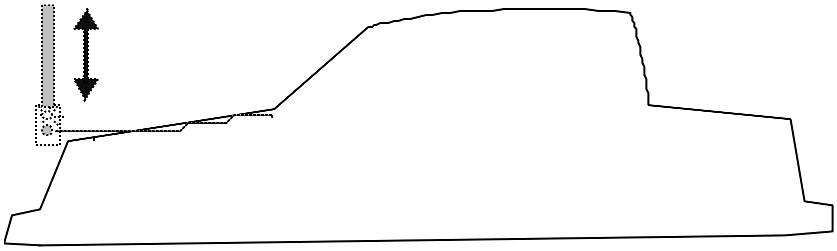
Example. A two phase Pulse encoder connected to normal inputs. (Works on low frequencies (up to approx. 100 Hz)

```
¦                                         ¦
¦CHAN                         +---------+ ¦ 0000 ORG             004      CHAN A
¦A                            ¦FUN00 400¦ ¦ 0001 FUN00 DIF       400      POS. EDGE
+-¦ +-------------------------¦DIF      ¦ ¦ 0002 FUN01 DFN       401      NEG. EDGE
¦ 004 ¦                       ¦POS. EDGE¦ ¦
¦     ¦                       +---------+ ¦
¦     ¦                       +---------+ ¦
¦     ¦                       ¦FUN01 401¦ ¦
¦     +-----------------------¦DFN      ¦ ¦
¦                             ¦NEG. EDGE¦ ¦
¦                             +---------+ ¦
¦                                         ¦
¦POS.  CHAN                               ¦ 0003 ORG             400      POS. EDGE
¦EDGE  B                                  ¦ 0004 AND  NOT        005      KANAL B
+-¦ +---¦/+-+                             ¦ 0005 STR             401      NEG. EDGE
¦ 400   005 ¦                             ¦ 0006 AND             005      KANAL B
¦           ¦                             ¦ 0007 OR STR
¦NEG.  CHAN ¦                 +---------+ ¦ 0008 STR             400      POS. EDGE
¦EDGE  B    ¦                 ¦FUN40 700¦ ¦ 0009 OR              401      NEG. EDGE
+-¦ +---¦ +-------------------¦UDC      ¦ ¦ 0010 STR             006      RESET
¦ 401   005                   ¦COUNTER.0¦ ¦ 0011 FUN40 UDC       700      COUNTER.0
¦                             ¦         ¦ ¦
¦POS.                         ¦         ¦ ¦
¦EDGE                         ¦         ¦ ¦
+-¦ +-------+                 ¦         ¦ ¦
¦ 400       ¦                 ¦         ¦ ¦
¦           ¦                 ¦         ¦ ¦
¦NEG.       ¦                 ¦         ¦ ¦
¦EDGE       ¦                 ¦         ¦ ¦
+-¦ +-------------------------¦         ¦ ¦
¦ 401                         ¦         ¦ ¦
¦                             ¦         ¦ ¦
¦RESET                        ¦         ¦ ¦
¦                             ¦         ¦ ¦
+-¦ +-------------------------¦         ¦ ¦
¦ 006                         ¦         ¦ ¦
¦                             +---------+ ¦
```

Control , with one reference:

Example. A motor shall be controlled up and down using a photo electric switch (photocell) This could e.g. be a car wash machine.

We will use two timers, which delay the start of the motor to avoid fast on and off and reversing of the motor.

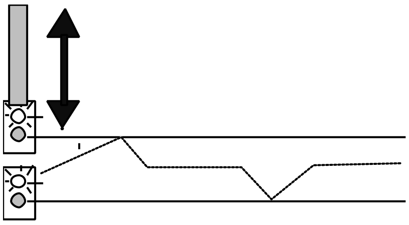


```
¦ *** FOLLOWING CONTROL IS DONE USING A PHOTOELL ****
¦                                         ¦
¦PHOTO                        +---------+ ¦ 0000 ORG             003      PHOTOCELL
¦CELL                         ¦TMR   T000 ¦ 0001 OUT            T000  1.5 DELAY UP
+-¦ +-------------------------¦   1.5   ¦ ¦
¦ 003                         ¦DELAY UP ¦ ¦
¦                             +---------+ ¦
¦                                         ¦
¦PHOTO                        +---------+ ¦ 0002 ORG NOT         003      PHOTOCELL
¦CELL                         ¦TMR   T001 ¦ 0003 OUT            T001  1.5 DELAY DWN
+-¦/+-------------------------¦   1.5   ¦ ¦
¦ 003                         ¦DELAY DWN¦ ¦
¦                             +---------+ ¦
¦                                         ¦
¦DELAY                              UP    ¦ 0004 ORG            T000      DELAY UP
¦ UP                                      ¦ 0005 OUT             200      UP
+-¦ +--------------------------------( )- ¦
¦ T000                               200  ¦
¦                                         ¦
¦DELAY                              DOWN  ¦ 0006 ORG            T001      DELAY DWN
¦ DWN                                     ¦ 0007 OUT             201      DOWN
+-¦ +--------------------------------( )- ¦
```

```
¦ T001                                    201  ¦
¦                                              ¦
```

Control, with two references:

Example. The above control is not perfect as this will cause continuous control up and down even on a flat surface.
If we add another photocell we can achieve better control.
We create a timer for the motor to, in normal cases, stop between the two photocells when the motor is going up and down.



Therefore we make a self hold on the Up and Down movements of the motor, which is initiated by the photocells when the reverse time has expired. The self hold is broken because of the other photocell or timer, which searches for the correct position. It is also possible to add an extra condition of the opposite photocell (which is not necessary here).

```
¦ ***Control between 2 Photocells ******
¦                                           ¦
¦UP    DOWN                    +----------+ ¦ 0000 ORG NOT         200      UP
¦                              ¦TMR   T000  ¦ 0001 AND  NOT        201      DOWN
+-¦/+---¦/+--------------------¦   2      ¦ ¦ 0002 OUT            T000    2 REVERSTIME
¦ 200   201                    ¦REVERSTIME  ¦
¦                              +----------+ ¦
¦                                           ¦
¦PH C  PH C                    +----------+ ¦ 0003 ORG NOT         000      PH C UPPER
¦UPPER LOWER                   ¦TMR   T001  ¦ 0004 AND             001      PH C LOWER
+-¦/+---¦ +--------------------¦   1.5    ¦ ¦ 0005 OUT            T001  1.5 RIGHT POS.
¦ 000   001                    ¦RIGHT POS.  ¦
¦                              +----------+ ¦
¦                                           ¦
¦PH C   REVER PH C  RIGHT DOWN        UP    ¦ 0006 ORG             000      PH C UPPER
¦UPPER  STIME LOWER  POS.                   ¦ 0007 AND            T000      REVERSTIME
+-¦ +---¦ +---¦ +---¦/+---¦/+---------( )-  ¦ 0008 OR              200      UPP
¦ 000   T000¦ 001   T001  201         200   ¦ 0009 AND             001      PH C LOWER
¦           ¦                               ¦ 0010 AND  NOT       T001      RIGHT POS.
¦UP         ¦                               ¦ 0011 AND  NOT        201      DOWN
¦           ¦                               ¦ 0012 OUT             200      UP
+-¦ +-------+                               ¦
¦ 200                                       ¦
¦                                           ¦
¦PH C  REVER PH C  RIGHT  UP         DOWN   ¦ 0013 ORG NOT         001      PH C LOWER
¦LOWER S TID UPPER  POS.                    ¦ 0014 AND            T000      REVERSTIME
+-¦/+---¦ +---¦/+---¦/+---¦/+---------( )-  ¦ 0015 OR              201      DOWN
¦ 001   T000¦ 000   T001  200         201   ¦ 0016 AND  NOT        000      PH C UPPER
¦           ¦                               ¦ 0017 AND  NOT       T001      RIGHT POS.
¦DOWN       ¦                               ¦ 0018 AND  NOT        200      UP
¦           ¦                               ¦ 0019 OUT             201      DOWN
+-¦ +-------+                               ¦
¦ 201                                       ¦
```

1.4.3.1. Sequence control with Auto / Manual and Reset

Example

Sequence control with AUTO / MANUAL and Reset. (See example on page 18). The switch AUTO is added to input 5 and the push buttons PB LIFT UP, PB LIFT DOWN, PB CONVEY and PB PUSH are added to input 5-9 for manual control and push button RESET on input 10.

When the switch AUTO is ON the sequence works as before. When it is OFF it is possible to exercise manual control vial the push buttons. When RESET is effected the sequence is broken and the Start step is activated.

Sequence part: Here MCS (Master Control Set) is used and MSR (Master Control Reset)

```
¦                                          ¦
¦ ** SEQUENCE PART FOR THE CONVEYOR AND THE¦LIFT MOVEMENT ******
¦                                          ¦
¦RESET                         +---------+ ¦ 0000 ORG NOT         010      RESET
¦                              ¦FUN04    ¦ ¦ 0001 FUN04 MCS
+-¦/+--------------------------¦MCS      ¦ ¦
¦ 010                          ¦         ¦ ¦
¦                              +---------+ ¦
¦                                          ¦
¦START LIFT  START STEP2             STEP1 ¦ 0002 ORG             000      STARTBUTT
¦BUTT  DOWN  STEP                          ¦ 0003 AND             004      LIFT LOW
+-¦ +---¦ +---¦ +---¦/+----------------( )-¦ 0004 AND             400      STARTSTEP
¦ 000   004   400 ¦ 402                401 ¦ 0005 OR              401      STEP1
¦                 ¦                        ¦ 0006 AND  NOT        402      STEP2
¦STEP1            ¦                        ¦ 0007 OUT             401      STEP1
¦                 ¦                        ¦
+-¦ +-------------+                        ¦
¦ 401                                      ¦
¦                                          ¦
¦PHOTO STEP1 STEP3                   STEP2 ¦ 0008 ORG             001      PHOTOCELL
¦CELL                                      ¦ 0009 AND             401      STEP1
+-¦ +---¦ +---¦/+----------------------( )-¦ 0010 OR              402      STEP2
¦ 001   401 ¦ 403                      402 ¦ 0011 AND  NOT        403      STEP3
¦           ¦                              ¦ 0012 OUT             402      STEP2
¦STEP2      ¦                              ¦
¦           ¦                              ¦
+-¦ +-------+                              ¦
¦ 402                                      ¦
¦                                          ¦
¦LIFT  STEP2 STEP4                   STEP3 ¦ 0013 ORG             002      LIFT LOW
¦UP                                        ¦ 0014 AND             402      STEP2
+-¦ +---¦ +---¦/+----------------------( )-¦ 0015 OR              403      STEP3
¦ 002   402 ¦ 404                      403 ¦ 0016 AND  NOT        404      STEP4
¦           ¦                              ¦ 0017 OUT             403      STEP3
¦STEP3      ¦                              ¦
¦           ¦                              ¦
+-¦ +-------+                              ¦
¦ 403                                      ¦
¦                                          ¦
¦ ** DELAY BEFORE THE LIFT GOES DOWN AFTER PUSHING ****
¦                                          ¦
¦PUSH  STEP3                   +---------+ ¦ 0018 ORG             003      PUSH OUT
¦OUT                           ¦TMR   T000 ¦ 0019 AND             403      STEP3
+-¦ +---¦ +--------------------¦   3.5   ¦ ¦ 0020 OUT            T000  3.5 DELAY
¦ 003   403                    ¦DELAY    ¦ ¦
¦                              +---------+ ¦
¦                                          ¦
¦DELAY START                         STEP4 ¦ 0021 ORG            T000      DELAY
¦      STEP                                ¦ 0022 OR              404      STEP4
+-¦ +---¦/+----------------------------( )-¦ 0023 AND  NOT        400      STARTSTEP
¦ T000¦ 400                            404 ¦ 0024 OUT             404      STEP4
¦     ¦                                    ¦
¦STEP4¦                                    ¦
¦     ¦                                    ¦
+-¦ +-+                                    ¦
¦ 404                                      ¦
¦                                          ¦
¦                              +---------+ ¦ 0025 FUN05 MCR
¦                              ¦FUN05    ¦ ¦
+------------------------------¦MCR      ¦ ¦
¦                              ¦         ¦ ¦
¦                              +---------+ ¦
¦                                          ¦
¦LIFT  STEP4 STEP1                   START ¦ 0026 ORG             004      LIFT LOW
¦DOWN                                STEP  ¦ 0027 AND             404      STEP4
+-¦ +---¦ +---¦/+----------------------( )-¦ 0028 OR              400      STARTSTEP
¦ 004   404 ¦ 401                      400 ¦ 0029 OR              010      RESET
¦           ¦                              ¦ 0030 OR              967      INIT PULSE
¦START      ¦                              ¦ 0031 AND  NOT        401      STEP1
¦STEP       ¦                              ¦ 0032 OUT             400      STARTSTEP
+-¦ +-------¦                              ¦
¦ 400       ¦                              ¦
¦           ¦                              ¦
¦RESET      ¦                              ¦
¦           ¦                              ¦
+-¦ +-------¦                              ¦
¦ 010       ¦                              ¦
¦           ¦                              ¦
¦INIT       ¦                              ¦
¦PULSE      ¦                              ¦
+-¦ +-------+                              ¦
¦ 967                                      ¦
```

Continuing.

Output control: Here the conditions for Auto and Manual control works in parallel.

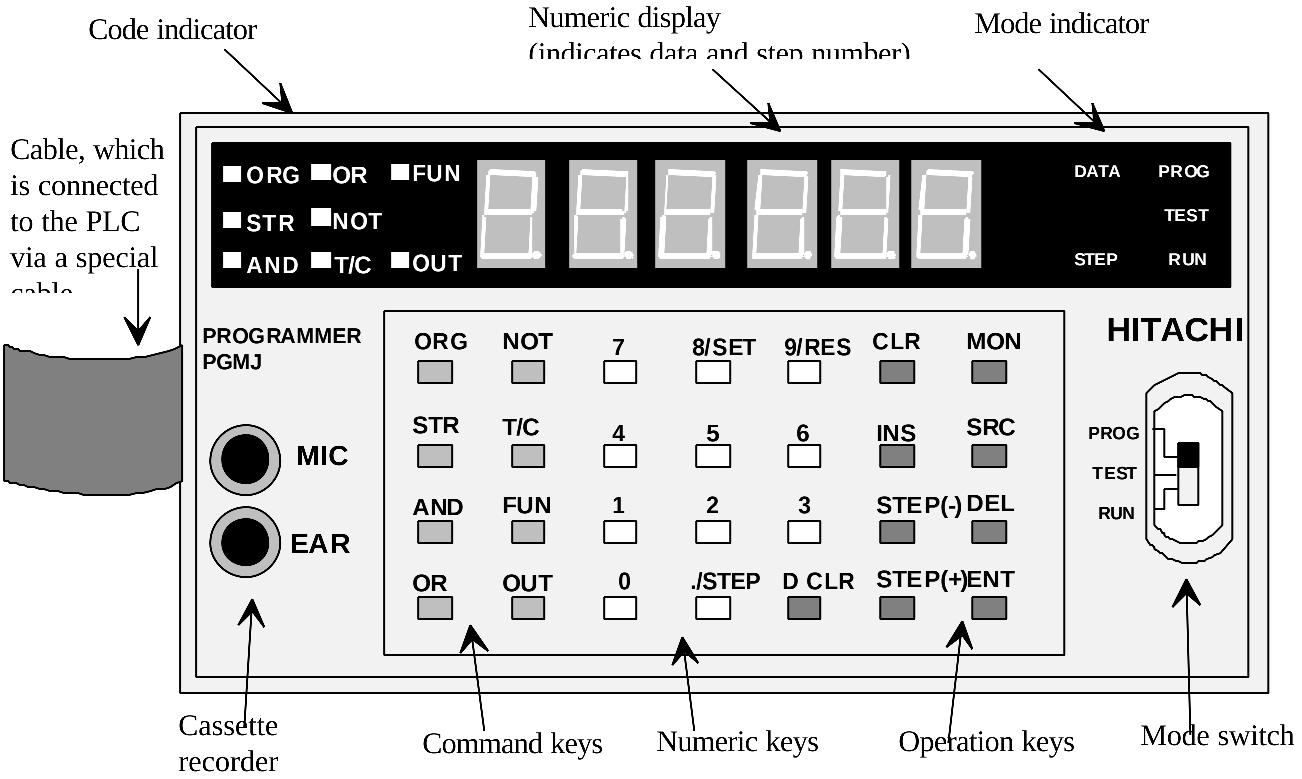
```
¦ ***    OUTPUT CONTROL OF OUTPUTS ****
¦START                               GREEN ¦ 0033 ORG             400      STARTSTEP
¦STEP                                 LAMP ¦ 0034 OUT             200      GREEN LAMP
+-¦ +---------------------------------( )- ¦
¦ 400                                 200  ¦
¦                                          ¦
¦STEP1 AUTO                          CONVE ¦ 0035 ORG             401      STEP1
¦                                    YOER  ¦ 0036 AND             005      AUTO
+-¦ +---¦ +---------------------------( )- ¦ 0037 STR             006      PB CONVEY
¦ 401   005 ¦                         201  ¦ 0038 AND  NOT        005      AUTO
¦           ¦                              ¦ 0039 OR STR
¦PB CO AUTO ¦                              ¦ 0040 OUT             201      CONVEYOR
¦NVEY       ¦                              ¦
+-¦ +---¦/+-+                              ¦
¦ 006   005                                ¦
¦                                          ¦
¦STEP2 AUTO  LIFT                    LIFT  ¦ 0041 ORG             402      STEP2
¦            DOWN                    UP    ¦ 0042 AND             005      AUTO
+-¦ +---¦ +---¦/+---------------------( )- ¦ 0043 STR             007      PB LIFT UP
¦ 402   005 ¦ 204                     202  ¦ 0044 AND  NOT        005      AUTO
¦           ¦                              ¦ 0045 OR STR
¦PB LI AUTO ¦                              ¦ 0046 AND  NOT        204      LIFT DOWN
¦FT UP      ¦                              ¦ 0047 OUT             202      LIFT UP
+-¦ +---¦/+-+                              ¦
¦ 007   005                                ¦
¦                                          ¦
¦STEP3 AUTO                          PUSHE ¦ 0048 ORG             403      STEP3
¦                                    R     ¦ 0049 AND             005      AUTO
+-¦ +---¦ +---------------------------( )- ¦ 0050 STR             008      PB PUSHER
¦ 403   005 ¦                         203  ¦ 0051 AND  NOT        005      AUTO
¦           ¦                              ¦ 0052 OR STR
¦PB PU AUTO ¦                              ¦ 0053 OUT             203      PUSHER
¦SHER       ¦                              ¦
+-¦ +---¦/+-+                              ¦
¦ 008   005                                ¦
¦                                          ¦
¦STEP4 AUTO  LIFT                    LIFT  ¦ 0054 ORG             404      STEP4
¦            UPP                     DOWN  ¦ 0055 AND             005      AUTO
+-¦ +---¦ +---¦/+---------------------( )- ¦ 0056 STR             009      PB LIFTDWN
¦ 404   005 ¦ 202                     204  ¦ 0057 AND  NOT        005      AUTO
¦           ¦                              ¦ 0058 OR STR
¦PB LI AUTO ¦                              ¦ 0059 AND  NOT        202      LIFT UP
¦FTDWN      ¦                              ¦ 0060 OUT             204      LIFT DOWN
+-¦ +---¦/+-+                              ¦
¦ 009   005                                ¦
```

Alternative ways to solve a sequence part of the program above using FUN02

```
¦START LIFT  START +---------+       STEP1 ¦ 0000 ORG             000      START BUTT
¦BUTT  DOWN  STEP  ¦FUN02    ¦             ¦ 0001 AND             004      LIFT LOW
+-¦ +---¦ +---¦ +--¦IF       ¦--------( )- ¦ 0002 AND             400      STARTSTEP
¦ 000   004   400  ¦         ¦¦       401  ¦ 0003 FUN02 IF
¦                  +---------+¦            ¦ 0004 OUT             401      STEP1
¦                             ¦      START ¦ 0005 OUT NOT         400      STARTSTEP
¦                             ¦      STEP  ¦
¦                             +-------(/)- ¦
¦                                     400  ¦
¦                                          ¦
¦PHOTO STEP1 +---------+             STEP2 ¦ 0006 ORG             001      PHOTOCELL
¦CELL        ¦FUN02    ¦                   ¦ 0007 AND             401      STEP1
+-¦ +---¦ +--¦IF       ¦--------------( )- ¦ 0008 FUN02 IF
¦ 001   401  ¦         ¦¦             402  ¦ 0009 OUT             402      STEP2
¦            +---------+¦                  ¦ 0010 OUT NOT         401      STEP1
¦                       ¦            STEP1 ¦
¦                       ¦                  ¦
¦                       +-------------(/)- ¦
¦                                     401  ¦
¦                                          ¦
¦LIFT  STEP2 +---------+             STEP3 ¦ 0011 ORG             002      LIFT LOW
¦UPPER       ¦FUN02    ¦                   ¦ 0012 AND             402      STEP2
+-¦ +---¦ +--¦IF       ¦--------------( )- ¦ 0013 FUN02 IF
¦ 002   402  ¦         ¦¦             403  ¦ 0014 OUT             403      STEP3
¦            +---------+¦                  ¦ 0015 OUT NOT         402      STEP2
¦                       ¦            STEP2 ¦
¦                       ¦                  ¦
¦                       +-------------(/)- ¦
¦                                     402  ¦
¦                                          ¦
¦PUSH  STEP3                   +---------+ ¦ 0016 ORG             003      PUSH OUT
¦OUT                           ¦TMR   T000 ¦ 0017 AND             403      STEP3
+-¦ +---¦ +--------------------¦   3.5   ¦ ¦ 0018 OUT            T000  3.5 DELAY
¦ 003   403                    ¦DELAY    ¦ ¦
¦                              +---------+ ¦
¦                                          ¦
¦DELAY +---------+                   STEP4 ¦ 0019 ORG            T000      DELAY
¦      ¦FUN02    ¦                         ¦ 0020 FUN02 IF
+-¦ +--¦IF       ¦--------------------( )- ¦ 0021 OUT             404      STEP4
¦ T000 ¦         ¦¦                   404  ¦ 0022 OUT NOT         403      STEP3
¦      +---------+¦                        ¦
¦                 ¦                  STEP3 ¦
¦                 ¦                        ¦
¦                 +-------------------(/)- ¦
¦                                     403  ¦
¦                                          ¦
¦LIFT  STEP4 +---------+             START ¦ 0023 ORG             004      LIFT LOW
¦DOWN        ¦FUN02    ¦             STEP  ¦ 0024 AND             404      STEP4
+-¦ +---¦ +--¦IF       ¦--------------( )- ¦ 0025 OR              010      RESET
¦ 004   404 ¦¦         ¦¦             400  ¦ 0026 OR              967      INIT PULSE
¦           ¦+---------+¦                  ¦ 0027 FUN02 IF
¦RESET      ¦           ¦            STEP1 ¦ 0028 OUT             400      STARTSTEP
¦           ¦           ¦                  ¦ 0029 OUT NOT         401      STEP1
+-¦ +-------¦           +-------------(/)- ¦ 0030 OUT NOT         402      STEP2
¦ 010       ¦           ¦             401  ¦ 0031 OUT NOT         403      STEP3
¦           ¦           ¦                  ¦ 0032 OUT NOT         404      STEP4
¦INIT       ¦           ¦            STEP2 ¦
¦PULSE      ¦           ¦                  ¦
+-¦ +-------+           +-------------(/)- ¦
¦ 967                   ¦             402  ¦
¦                       ¦                  ¦
¦                       ¦            STEP3 ¦
¦                       ¦                  ¦
¦                       +-------------(/)- ¦
¦                       ¦             403  ¦
¦                       ¦                  ¦
¦                       ¦            STEP4 ¦
¦                       ¦                  ¦
¦                       +-------------(/)- ¦
¦                                     404  ¦
```

2. Programming tools

2.1.PGMJ, Hand programming unit



2.1.1. Key pad functions

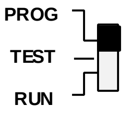
Command keys	**ORG**	Origin	Start of a logic block
	STR	Store	Start of a logic partial block
	AND	And	Logic Serial connection
	OR	Or	Logic Parallel connection
	NOT	Not	Logic inverted function
	T/C	Timer/Counter	Timer or Counter function
	FUN	Function	Special instructions
	OUT	Out	Output instructions
Data- keys	**0 - 9**	Numeric values	Figures 0 to 9
	.		Decimal point
Operations- keys	**CLR**	Clear	Reset of PGMJ status
	D CLR	Data clear	Resetting the current data of the PGMJ
	INS	Insert	Insert of program instruction
	DEL	Delete	Delete of program instruction
	MON	Monitor	Monitor of a memory status during run
	SRC	Search	Search for an address
	ENT	Enter	Memory storage of a program STEP
	STEP	Step	Program STEP choices
	SET	Set	Forced set of outputs / memories
	RES	Reset	Forced reset of outputs / memories
	STEP(+)	Step plus	One step forwards
	STEP(-)	Step minus	One step backwards

2.1.1.1. Possible functions with PGMJ

Programming functions:	Write a program, Delete a program, Read a program
Edit functions:	Change instructions, Insert instructions, Delete instructions
Monitor functions:	Monitor of status on inputs, outputs, internal outputs, timers and Counters
Control functions:	Syntax check, type in control, address control
Cassette recorder function:	Load a program from a tape cassette, store and compare program on cassette
Monitor of an instruction:	Monitor of instructions with LEDs
Monitor of data:	Monitor of data on a 6 digit numeric display
STEP monitor:	Monitor of STEP numbers on a 3 digit numeric display
Monitor of condition:	LED[1] shows PROG (programming mode) TEST and RUN (run condition) exchange between STEP and DATA with the key ./STEP

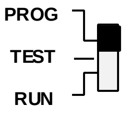
2.1.2. Explanation of the function

2.1.2.1. Total clear of the program

Function	Mode	Status PLC	Note
Total clear		Stop	Always clear before start of a new project. By a total clear, following is deleted: - The Program instructions - Timer/ Counter current and preset values - Content in a shift register - Retentive internal outputs

Type in	Display			Note
	Code monitor	Numeric Display	Mode monitor	
[CLR]			‾ PROG ‾ DATA	
[ENT]		E		
[DEL]		-		Total clear is ready

2.1.2.2. Switching between STEP- and data monitor on the display

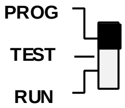
Function	Mode	Status PLC	Note
Switch between Data/ STEP monitor on the display:		Stop	Normally this is shown on the display. Through pressing [./STEP] the step number is shown. To return to DATA, press [./STEP] again.

Type in	Display			Note
	Code	Numeric Display	Mode monitor	
[CLR],[ENT],[DEL]		-	‾ PROG ‾ DATA	Data display

[1] LED = Light emitting diode

[. / STEP]		**0**	¯ PROG ¯ STEP	STEP (step) display
[. / STEP]		**-**	¯ PROG ¯ DATA	Data display

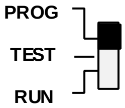
2.1.2.3. Type in

Function	Mode	Status PLC	Note
Type in of a new program		Stop	

```
¦                                 ¦
¦                                 ¦ ORG        0
+--¦ +------¦/+--------------( )--¦ OR       200
¦   0  ¦     1               200  ¦ AND NOT    1
¦      ¦                          ¦ OUT      200
+--¦ +-+                          ¦
¦  200                            ¦
¦                                 ¦
¦                                 ¦
```

	Display			
Type in	**Code**	**Numeric Display**	**Mode monitor**	**Note**
[CLR][ENT][DEL]		**-**	¯ DATA ¯ PROG	When the [ENT] button is pressed the data which are indicted by the diodes and the numeric display are added to the program.
[ORG][0] [ENT]	ORG	**0**		
[OR][2][0][0] [ENT]	OR	**200**		
[AND][NOT][1] [ENT]	AND NOT	**1**		
[OUT][2][0][0] [ENT]	OUT	**200**		

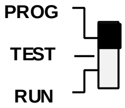
2.1.2.4. Type in of additional program

Function	Mode	Status PLC	Note
Add on programming		Stop	Through pressing [CLR] [.STEP] you will come to the first free line in the program. From this line the additional programming can start.

```
¦                                 ¦
¦                                 ¦ ORG       0
+--¦ +-----¦ +---------------( )--¦ AND       2
¦   0       2               T/C 00¦ OUT T/C   00  9.5s
¦                           9.5s  ¦
¦                                 ¦
¦                                 ¦ ORG T/C   00
+--¦ +-----------------------( )--¦ OUT      201
¦ T/C 00                    201   ¦
¦                                 ¦
```

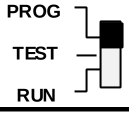
	Display			
Type in	**Code**	**Numeric Display**	**Mode monitor**	**Note**
[CLR][SRC]			¯ PROG ¯ DATA	
[ORG][0] [ENT]	ORG	**0**		
[AND][2] [ENT]	AND	**2**		
[OUT][T/C][0][0][.] [0][9][.][5] [ENT] [ORG][T/C][0] [ENT]	OUT T/C ORG T/C	**0 09.5** **0**		
[OUT][2][0][1] [ENT]	OUT	**201**		

2.1.2.5. Reading

Function	Mode	Status PLC	Note
Reading of the program		Run and Stop	

Program read out	**Type in procedure**
Read from STEP number	`[CLR] [STEP+]..[STEP+] or[STEP-]`
Read from row with specified address	`[CLR] [Address] [SRC]  ......`
Read from row with specified command	`[CLR] [command] [SRC] ......`
Read from row with specified address and command	`[CLR] [command] [address] [SRC] ..`
Read from specified row	`[CLR] [STEP nr] [./STEP] ......`

2.1.2.6. Inserting program steps

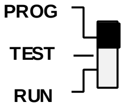
Function	Mode	Status PLC	Note
Insertion of program		Stop	

```
¦                                     ¦
¦                                     ¦  ORG         0
+--¦ +------¦/+---------------( )--¦  OR        200
¦   0    ¦     1         ↑       200  ¦  AND NOT     1
¦        ¦     Insert input 3         ¦  OUT       200
+--¦ +-+                              ¦
¦  200                -¦ +-           ¦
¦                                     ¦
¦                      3              ¦
¦                                     ¦
```

```
¦                                     ¦
¦                                     ¦  ORG         0
+--¦ +------¦/+------¦ +-----( )--¦  OR        200
¦   0    ¦     1      3      200  ¦  AND NOT     1
¦        ¦                            ¦  AND         3
+--¦ +-+                              ¦  OUT       200
¦  200                                ¦
¦                                     ¦
```

Type in	**Display**		
	Command	Numeric display	Mode display
[CLR] [OUT] [2] [0] [0] [SRC]	¯ OUT	2 0 0	¯ DATA
[DCLR]			
[AND] [3]	¯ AND	3	¯ PROG
[INS]	¯ OUT	2 0 0	

2.1.2.7. Deletion of a program step

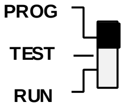
Function	Mode	Status PLC	Note
Deletion of a program step		Stop	

```
¦                                       ¦    ORG       0
+--¦ +------¦/+------¦ +-----( )--¦    OR      200
¦   0   ¦      1         3       200    ¦    AND NOT   1
¦       ¦                 |             ¦    AND       3
+--¦ +-+                  ↓             ¦    OUT     200
¦  200                                  ¦
¦            Input 3 is                 ¦
```

Type in	Display		
	Command	Numeric display	Mode display
[CLR] [AND] [3] [SRC]	¯ AND	3	¯ PROG
[DEL]	¯ OUT	2 0 0	¯ STEP

```
¦                                       ¦    ORG       0
+--¦ +------¦/+--------------( )--¦    OR      200
¦   0   ¦      1                 200    ¦    AND NOT   1
¦       ¦                               ¦    OUT     200
+--¦ +-+                                ¦
¦  200                                  ¦
```

2.1.2.8. Syntax check

Function	Mode	Status PLC	Note
Syntax check of a program		Stop	The display shows the error type. If you press SRC again the next error is shown and the first not used instruction address is shown. The error code is written in word 980.

Type in		**Display**			**Notes**
		command	numeric display	mode display	
[CLR]	No error		3 0 0	¯ PROG	Shows first free address
[SRC]	Error		1 1 5 E	¯ STEP	Shows an error on row 115

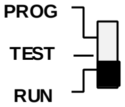
Error .	Display PGMJ	Display PGMJ-R/2	Explanation
0	Tom	Tom	No error
1	E	E	An invalid combination of instructions
2	E	E	Invalid structure of the program or the subroutine
3	E	E	More than one INT instruction has got the same number
4	E	E	Invalid usage of the instructions FUN06 and FUN07
5	E	E	Invalid usage of the instructions FUN08 and FUN09
6	⊔	uE	Too few STR instructions before the specified instruction
7	⊓	oE	Too many STR instructions before the specified instruction
8	⊔	uE	MCR level is too low (less MCR than MCS)
9	⊓	oE	MCR level is too high (more MCR than MCS)
10	E	E	IF or IFR are doubled or OUT T/C not permitted after IF and IFR
11	E	E	Invalid argument (address, constant or corresponding too high or too low)
12	E	E	Not permitted to duplicate an OUT instruction
13	E	dE	Output double used. The program runs anyway
20	A	fE	Undefined instruction or invalid instruction, internal error
30	E	E	The user program failed the sum check

To read the error code, type decimal monitor of the word 980:

CLR 9 8 0 MON MON

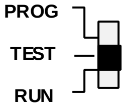
The error code is kept even if the power supply is turned off.

2.1.2.9. Monitor of the status

Function	Mode	Status PLC	Note
Monitor		RUN	Word monitor (2 bit word): Repeat MON. (see below)

Type of monitor	Type in	Display			Notes
		Command	Numeric display	Mode display	
External or internal input /output	[CLR] [address no] [MON] [STEP+] or [STEP-]		address no. address no.		Monitors the status Monitors the status of the new address
2 byte word	[MON] [MON] [MON]		16383 3FFF address no.	¯ DATA	Decimal monitor Hexadecimal monitor Status again
External or internal input /output	[CLR] [OUT]] [No.] [MON] [STEP+]	¯ OUT ¯ OUT	4 0 1 • 4 0 2	¯ RUN	Memory 401 ON Memory 402 OFF
Timer / Counter contact	[CLR] [T/C]] [No.] [MON] [STEP-]	¯ T/C ¯ T/C	1 0 • 7	(¯ TEST)	Timer 10 ON Timer 7 OFF
Timer / Counter output	[CLR] [OUT] [T/C] [no.] [MON] [STEP+]	¯ OUT ¯ T/C ¯ OUT ¯ T/C	6 0 0 0 5 6 1 0 1 0		Counter 60 current value Counter 61 current value

2.1.2.10. Changing the preset of Timers or Counters during run.

Function	Mode	Status PLC	Note
Changing Timer or Counter preset during RUN		RUN	

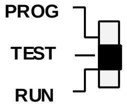
```
¦                                       ¦
+--¦ +----------------------(T00)--¦
¦   0                              ¦
¦                        150 s    ¦
                           ↑
                         11.4 s
```

The old preset of 150 s shall be changed to 11.4 s on timer 00

Type in	Display		
	Command	Numeric display	Mode display
[CLR] [OUT] [T/C] [0] [0] [SRC]	¯ OUT ¯ T/C	0 0 1 5 0	Search for Timer 00
[1] [1] [.] [4] [ENT]	¯ OUT	0 0 1 1 . 4	Write the new preset

	‾ T/C		

2.1.2.11. Forced set / reset of special memories,T/C and output.

Function	**Mode**	**Status PLC**	**Note**
Forced Set and Reset of internal outputs, T/C and outputs.		RUN	

Type in	**Display**		
	Command	Numeric display	Mode display
[CLR] [OUT] [T/C] [0] [0] [MON]	‾ OUT ‾ T/C	0 0 1 0 8	Monitors Timer 00
[SET]	‾ OUT ‾ T/C	0 0 1 0 8	Forced set
[CLR]		0 0 1 1 . 4	Stops forced set

2.1.3. System error codes:

Error codes decimal	**Explanation.**
10	Invalid interrupt has occurred
11	Invalid level of the stack pointer
12	Contradiction of the logic has occurred
13	Incorrect interrupt has occurred
14	NMI interrupt has occurred
20	Data has not been written correctly to the user program memory.
21	Check sum error has occurred in the system ROM memory.
22	System RAM read/write check error
30	Undefined PLC instruction has been used.
31	PLC stack pointer is invalid.
32	Check sum error has occurred in the user memory during run.
40	Receive signal has caused "overflow" in the buffer.
41	Transmitted signal has overflowed the buffer (Link port)

2.1.4. EC series self check procedure by programming unit.

The EC has a self check possibility to permit diagons on the programming unit and the PLC. When the self check is started **the program written in the PLC is deleted**.

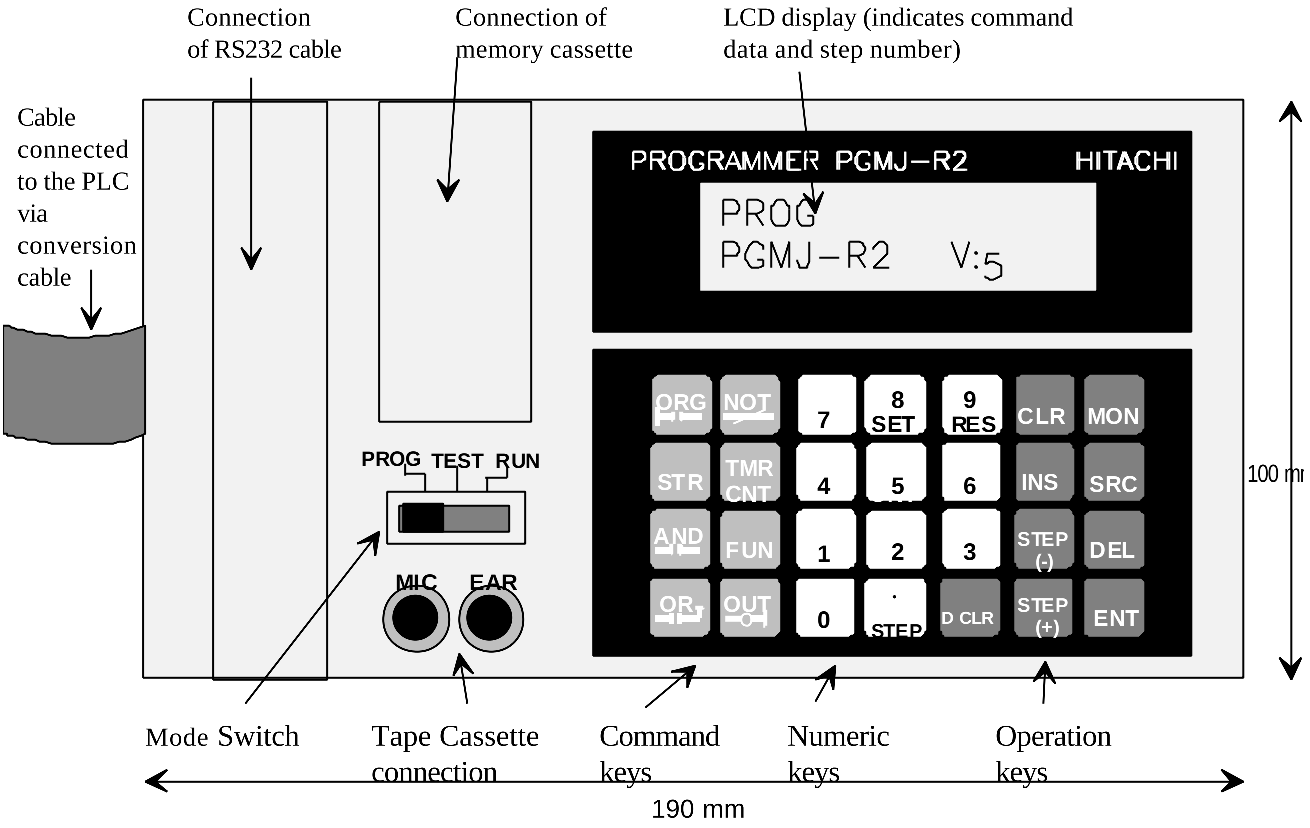
Key in procedure for self check	**Lamp status**	**During self check**	**Check complete**	**Error detected**
[CLR], [SET], [SET], [ENT]				
[FUN], [9], [7], [8], [MON]	Power lamp	ON	ON	ON
[MON]				
[CLR]	Run Lamp	ON	OFF	OFF
[MON]				
[CLR]	Error Lamp	ON	OFF	ON
[MON]				

If the ERR lamp is on, the error code can be checked. To read the error code, do decimal monitor on the word 970:

CLR 9 7 0 MON MON

The error code remains even after a power down.

2.2.PGMJ-R, Hand programming unit with communication



Programming with PGMJ-R(2):

Display after power on:

```
+---------------------------------+
|                                 |
| PROG                            |
|                                 |
| PGMJ-R2       V:5               |
|                                 |
+---------------------------------+
```

The number 5 is the version number. To print out all functions in series EC version 5 of PGMJ-R is required. There is also a newer type of PGMJ-R, which is called PGMJ-R2. The difference is that all monitor features are not supported.
The type in procedure is almost equal to PGMJ but the display is different.

(See also the description of PGMJ, page 71)

2.2.1. Key pad functions on PGMJ-R (See also under 2.1 PGMJ)

Type		Type in	Display	Notes
Total delete		[CLR] [ENT] [DEL]	PROG STEP ¦ ¦ ¦ ¦ PROG STEP ¦ ¦ ¦ ¦ E PROG STEP 0 0 0 0	
Command code example		[ORG] [NOT] [T/C] [4] [ENT]	PROG STEP 0 0 0 0 ORG NOT T/C 0 4	Gives the instructions ORG NOT T/C 4 OUT T/C 10 950
		[ORG] [T/C] [1] [0] [9] [5] [0] [ENT]	PROG STEP 0 0 0 1 OUT T/C 1 0 9 5 0	Decimal monitor Hexadecimal monitor Status again
Syntax check	No error	[CLR] [SRC]	PROG*STEP 0 3 5 6	
	Double use of output		PROG*STEP 0 1 2 3 dE OUT 2 5 0	Error code dE
	Stack under flow		PROG*STEP 0 2 3 3 dE FUN 4 5 5 5 0	Error code uE too many STR (NOT)
	Stack over flow		PROG*STEP 0 3 5 0 oE FUN 4 5 5 6 0	Error code oE, too many AND (OR) STR
			PROG*STEP 0 1 2 0 fE	Error code fE
Monitor	(See also under PGMJ)	[OUT] [5] [0] [MON]	RUN STEP ¦ ¦ ¦ ¦ OUT 2 5 0 ¦	Output 250 high
		[CLR] [STEP+]	RUN STEP 0 0 0 1 OUT 2 0 0 ¦	Internal output 200 high
Type in error		[ORG] [9] [9] [9] [ENT]	PROG STEP 0 0 0 0 E ORG 9 9 9	Address 999 does not exist
Check sum error		-	RUN STEP ¦ ¦ ¦ ¦ 51E	Internal program check
Undefined command error		-	RUN STEP 4 - E	

2.2.2. Type in for PROM programming, cassette tape recorder and printer.

Type	Type in	Display	Notes
Shift to communication mode	[CLR] [SET] [SET] [ENT]	PROG R - - - 0 ROM MODE	PROG, TEST mode STOP
Cassette recorder mode	[FUN] [1]	PROG C - - - 1 CMT MODE	Record and play off tape recorder
PROM programming function (2)	[FUN] [2]	PROG R - - - 2 ROM MODE	ROM programming function when the memory cassette is inserted in PGMJ-R(2)
Forced output	[FUN] [3]	TEST O - - - 3 FORCED OUT	Is done in TEST mode
Printer interface function (1)	[FUN] [4]	PROG P - - - 4 PRINT OUT	Print out for series J-16 and series E
Printer interface function (2)	[FUN] [5]	PROG R - - - 5 PRINT OUT	Print out for the CPU CPM-E
Printer interface function (3)	[FUN] [5]	PROG R - - - 6 PRINT OUT	Print out for series EC, EB and EM II CPUs
Return from communication mode	[CLR] [RES] [RES] [ENT]		PROG, TEST mode STOP

2.2.2.1. Cassette tape recorder communication for PGMJ-R(2)

Type	Type in	Display	Notes
Cassette recorder mode	[CLR] [SET] [SET] [ENT] [FUN] [1]	PROG R - - - 0 ROM MODE	Cassette tape mode is specified
Store (record)	[OUT] [0] [ENT]	PROG C - - - 1 CMT MODE	CPU → Cassette recorder
PROM programming function (2)	[STR] [ENT]	PROG R - - - 2 ROM MODE	Cassette recorder → CPU
Forced output	[AND] [0] [ENT]	TEST O - - - 3 FORCED OUT	CPU ⇔ Cassette recorder
Error display	Type in error	C - - - E	Press CLR and try again
	Transfer error	C 6 2 E OUT	
	Verification error	C 7 - E NOT	
	Format error	C 8 - E NOT	
Return from recorder mode	[CLR] [RES] [RES] [ENT]		

Cable for cassette recorder

```
     OUT O-------------------O MIC
      IN O-------------------O EAR
    PGMJ-R2             Tape recorder
```

2.2.2.2. ROM programming function (FUN 2) (Memory cassette in PGMJ-R2)

The memory cassette is copied and programmed with this function when the memory cassette is mounted in PGMJ-R(2).

EPROM and EEPROM can be handled by PGMJ-R2

Type in procedure for ROM function 2:

Type		Type in	Display	Notes
Copying		[CLR] [SET] [SET] [ENT] [FUN] [2]	PROG R - - - 2 ROM MODE	Cassette tape mode is specified
Store (record)		[OUT] [0] [x] [ENT]	PROG R - - P OUT00 2 ROM MODE	CPU → Memory cassette
Load (play back)		[STR] [ENT]	PROG R - - P STR00 2 ROM MODE	Memory cassette → CPU
Verify		[AND] [0] [ENT]	PROG R - - P AND00 2 ROM MODE	CPU ⇔ Memory cassette
Empty check		[NOT] [ENT]	PROG R - - P NOT 2 ROM MODE	Check if the EPROM is empty
Error display	Type in error		R - - - E	Change memory cassette
	Transfer error		R 6 2 E OUT	
	Verification error		R 7 - E NOT	
	Empty check error		R 6 1 E NOT	
Return from ROM function		[CLR] [RES] [RES] [ENT]		

[x] : x = 0 means 1 k EEPROM in memory cassette [MPM-1E] [MPC-1E]
x = 1 means 2 k EEPROM in memory cassette[MPM-2E] [MPC-2E]
x = 2 means 2 k EPROM in memory cassette[MPM-2R] [MPC-2R]
(x = 4 means 4 k EEPROM in memory cassette[MPM-2E] [MPC-2E])
(x = 6 means 4 k EPROM in memory cassette[MPM-2R] [MPC-2R])

4 K EPROM and EEPROM can only be handled by PGMJ-R2 (as long as the programs are less than 2 K PGMJ-R can be used).

2.2.2.3. Type in procedure for force outputs (FUN 3)

Type in	Display	Notes
[CLR] [SET] [SET] [ENT] [FUN] [3]	PROG O - - - 3 FORCED OUT	Forced output mode
[CLR] [OUT] [2] [5] [0] [SET]	PROG O - - - OUT 3 FORCED OUT 250 ¦	Output 250 ON
[CLR] [OUT] [2] [5] [5] [SET]	PROG O - - - OUT 3 FORCED OUT 205 ¦	Output 255 ON
[RES]	PROG O - - - OUT 3 FORCED OUT 250	Output 255 OFF
[CLR] [OUT] [2] [5] [0] [RES]	PROG O - - - OUT 3 FORCED OUT 250	Output 250 OFF
[CLR] [RES] [RES] [ENT]		Return from forced mode

2.2.2.4. Type in procedure for printer communication (FUN 4, FUN 5)

Type	Type in	Display	Notes
Communication	[CLR] [SET] [SET] [ENT]	PROG R - - - 0 ROM MODE	Communication mode is specified
Printer interface mode 1	[FUN] [4]	PROG P - - - 1 PRINT OUT	For series J-16, Series E
Printer interface mode 2	[FUN] [5]	PROG R - - - 2 PRINT OUT	For CPU CPM-E
Printer interface mode 3	**[FUN] [6]**	**PROG O - - - 3 PRINT OUT**	**Series EC,** series EB, CPUs CPM-E2 and CPM-E3
Error display	Type in error	C - - - E	Press CLR and try again
	Transfer error	C 6 2 E OUT	
	Verification error	C 7 - E NOT	
	Format error	C 8 - E NOT	
Return from recorder mode	[CLR] [RES] [RES] [ENT]		

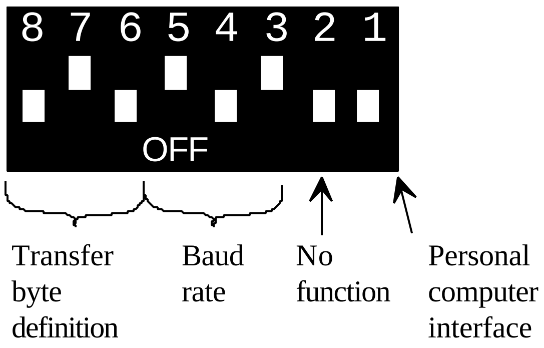
Type in format: [OUT] [0] [format for print out list no.]

Printer cable:	PGMJ-R/ 2	3	______________	3	Printer
		7	______________	7	
		11	______________	11	

List format	Gives			
0	Header	Code list	Ladder diagram	Cross reference
1		Code list	Ladder diagram	Cross reference
2			Ladder diagram	
3				Cross reference

2.2.3. Setting the dip switches of PGMJ-R/2

The transfer rate, definition of the transfer bytes and switch between personal computer interface and hand programmer.

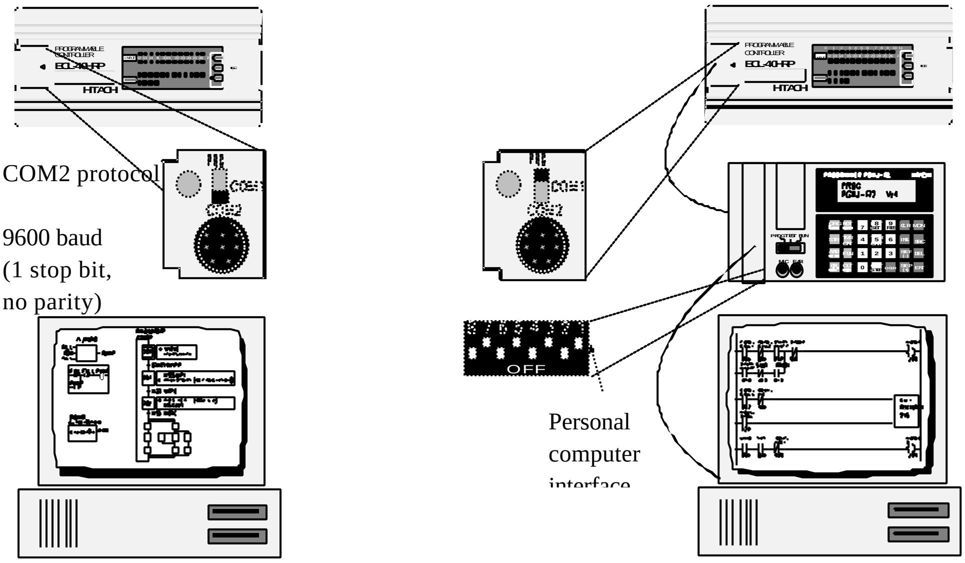


SWITCH NO.			Baud	Note.
5	4	3	rate K	
ON	ON	ON	38.4	
ON	ON	OFF	19.2	
ON	OFF	ON	9.6	Actsip
ON	OFF	OFF	4.8	FACTORY SETTINGS
OFF	ON	ON	2.4	
OFF	ON	OFF	1.2	
OFF	OFF	ON	0,6	
OFF	OFF	OFF	0,3	

SWITCH NO.			Start	Data	Parity	Stop	Notes
8	7	6	bits	bits	bit	bits	
ON	ON	ON	1	7	1(even)	2	
ON	ON	OFF	1	7	1(odd)	2	
ON	OFF	ON	1	7	1(even)	1	
ON	OFF	OFF	1	7	1(odd)	1	
OFF	ON	ON	1	8	-	2	
OFF	ON	OFF	1	8	-	1	FACTORY SETTINGS. Actsip.
OFF	OFF	ON	1	8	1(even)	1	
OFF	OFF	OFF	1	8	1(odd)	1	

2.3. Communication protocol

There are two ways to communicate:
The COM2 protocol (MODE2) is a very flexible protocol, which always is recommended. It offers fast monitor.
The COM1-protocol (MODE1) is done via PGMJ-R2 and normally there is no reason to use this protocol.
To use the COM2 protocol the switch on the front side shall be in position COM2 whereafter the power to the PLC is turned ON.
If the communication goes through a PGMJ-R/2 either to a computer or the key functions of the PGMJ-R/2 are used, the switch shall be in position PGR before the power is turned On. The power must be turned Off and On to enable a change of the protocol switch.
OBSERVE, if you want the control system to be in RUN mode, the switch must be in PRG position!



COM2 protocol (normally used) COM1 protocol (normally not used)

EC series has only one baud rate, 9600 baud. For more information, see separate description of the protocol.

2.3.1. E-COMM Pre prepared communication routines

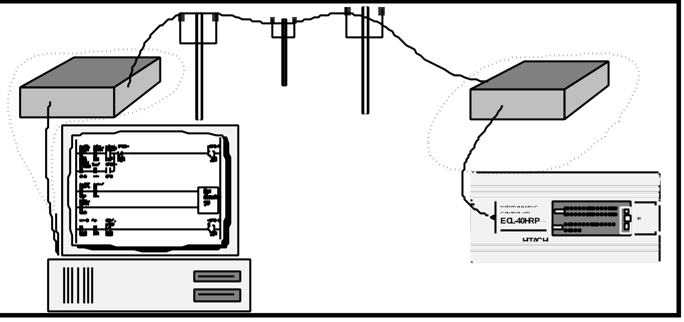
In applications where you want the PLC to communicate with a PC from your own developed software there is a package with communications routines for the COM2 protocol, which simplifies your development.

Example.
- Control and reporting to a supervision system.
- Direct control of the PLC from a special software.
- Connection to "intelligent" sensors, E.g. vision system

Product facts
Written in Microsoft C 6.0
Source code available
For IBM PC compatible computers
GreenLeaf comm library is required.

2.3.2. Modem communication.

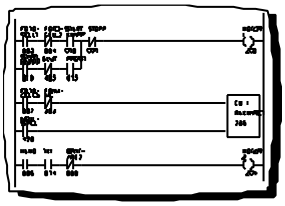
To simplify start up, service and maintenance there is a modem package available:



2.4.Programming-, start up- documentation tools.

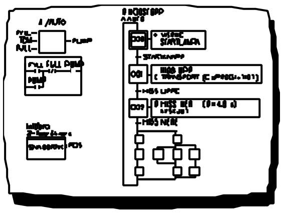
Actsip -Mini

Ladder and instruction code programming for series E, EM,EB and **EC**.



ActGraph

GrafCet programming for series E, EB, EM and **EC**



Act -Mini is a **programming, start up, trouble shooting and documentation tool** for Hitachi series E, EB. EM and EC. Actsip-Mini makes it easy to create a new project or modify an existing one. To simplify the programming you can alternate between ladder and instruction code programming.
Directly after a modification you can check the function and monitor the status.

With ActGraph you can program directly in **Grafcet**. The diagram, which is very easy to understand for almost everybody, is drawn directly on the screen. The PLC code is automatically generated and transferred to the PLC. Through the monitor functions directly in the graphs, the start up and trouble shooting becomes very effective.

All software is based on a windows style technique. **Many years of PLC experience** have been implemented in the user interface of the software.

The software is translated to many languages and the **help** system is complete and easy to use.
It is very easy to type in **comments**, which will remain with the project.

There are many practical features in the software. The ”**automatic allocation**” is one example of this. This means that the program always suggest free addresses for new memories, input/outputs etc., which creates comfortable programming and reduces mistakes like double use of memories etc.

A flexible way of saving program parts that could be used repeatedly is built in. This is called ”**macro** handling”

Many different documentation lists can be created, which gives a complete documentation according to the customers wishes.

For Hitachi’s larger PLC systems, Series-H, there is **Actsip and ActGraph software** available, which out of the users point of view looks the same. Therefore it is easy to use all sizes of the Hitachi PLC´s without learning new systems every time.

For more information, see special manuals and the tutorials in the next chapters.

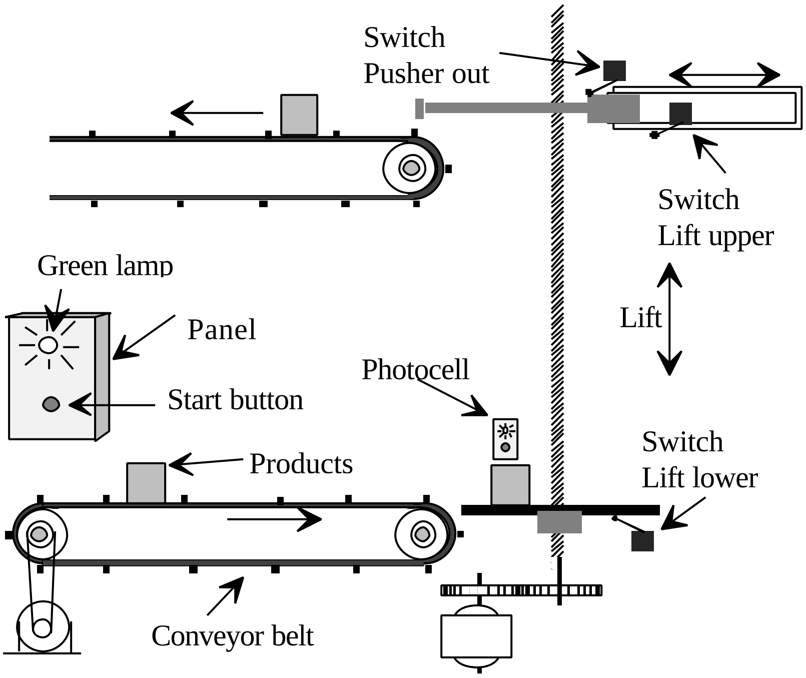
2.5. Programming with Actsip-Mini.

The purpose is to show how to get started and build an application from the start with Actsip-Mini.

For more detailed description, see Manual for **Actsip-E.**

Example.

When the machine is energised a green lamp shall be turned ON. It will be ON until the operator pushes the Start button. Then the lower conveyor shall start and move until the photocell in the end of the conveyor indicates.
Then the lift shall start going up and move until it reaches the top position.
Then the pusher moves until a switch indicates that it is out.
The lift moves down (while the pusher goes back automatically) until a switch indicates that it is down.
The sequence then repeats.



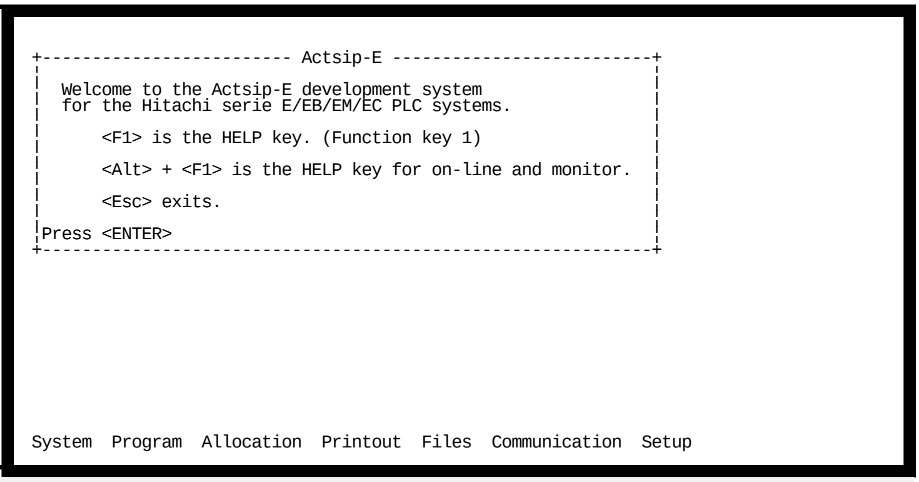
Start of programming:

Start from DOS with the command < E > <Enter>.

Then a welcome window opens.
Note that F1 gives help and that <Alt>+ F1 gives help for ON-line.

E

ENTER



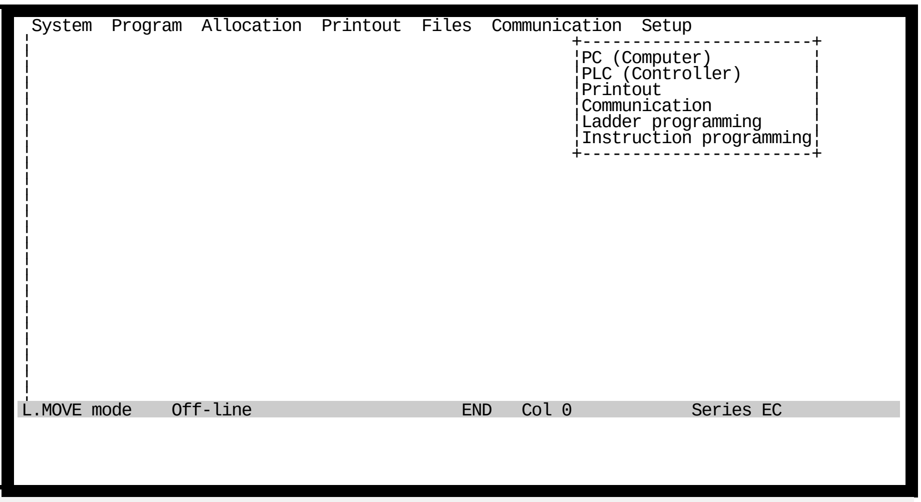
```
+------------------------- Actsip-E --------------------------+
|                                                              |
|  Welcome to the Actsip-E development system                  |
|  for the Hitachi serie E/EB/EM/EC PLC systems.               |
|                                                              |
|      <F1> is the HELP key. (Function key 1)                  |
|                                                              |
|      <Alt> + <F1> is the HELP key for on-line and monitor.   |
|                                                              |
|      <Esc> exits.                                            |
|                                                              |
|Press <ENTER>                                                 |
+--------------------------------------------------------------+



System  Program  Allocation  Printout  Files  Communication  Setup
```

Setup:

Press <Esc> and an empty drawing screen opens.

Before we start to draw the ladder diagram we must do the setup of the computer and the PLC, which is needed.
Press <Esc> and the main menu opens. Move the cursor with the <right arrow> to "Setup".

ESC, ESC →



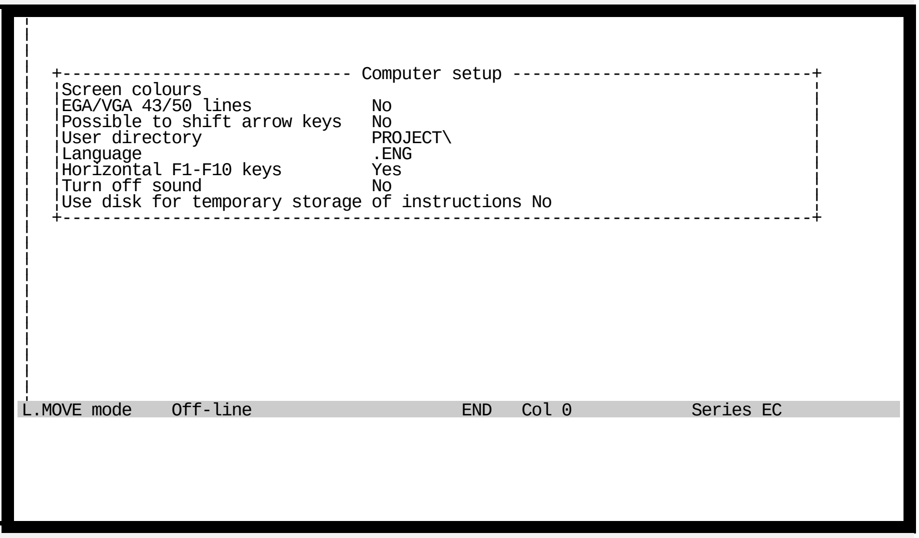
```
 System  Program  Allocation  Printout  Files  Communication  Setup
                                                         +-----------------------+
                                                         |PC (Computer)          |
                                                         |PLC (Controller)       |
                                                         |Printout               |
                                                         |Communication          |
                                                         |Ladder programming     |
                                                         |Instruction programming|
                                                         +-----------------------+

L.MOVE mode    Off-line                    END   Col 0           Series EC
```

Move with the <down arrow> to the choice for PC and press <Enter>.

E.g. if you have a laptop, you do not normally have arrow keys that are possible to shift.

Go down with the <down arrow> and press <Enter> for the information you want to change.

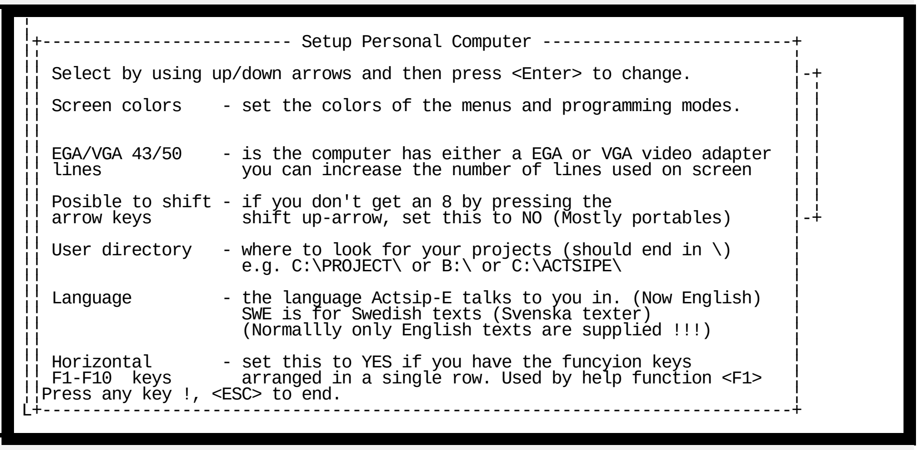
↓, ENTER ↓, ENTER



```
  +----------------------------- Computer setup -----------------------------+
  |Screen colours                                                            |
  |EGA/VGA 43/50 lines            No                                         |
  |Possible to shift arrow keys   No                                         |
  |User directory                 PROJECT\                                   |
  |Language                       .ENG                                       |
  |Horizontal F1-F10 keys         Yes                                        |
  |Turn off sound                 No                                         |
  |Use disk for temporary storage of instructions No                         |
  +--------------------------------------------------------------------------+

L.MOVE mode    Off-line                    END   Col 0           Series EC
```

If you are not sure , press F1 for help.

F1

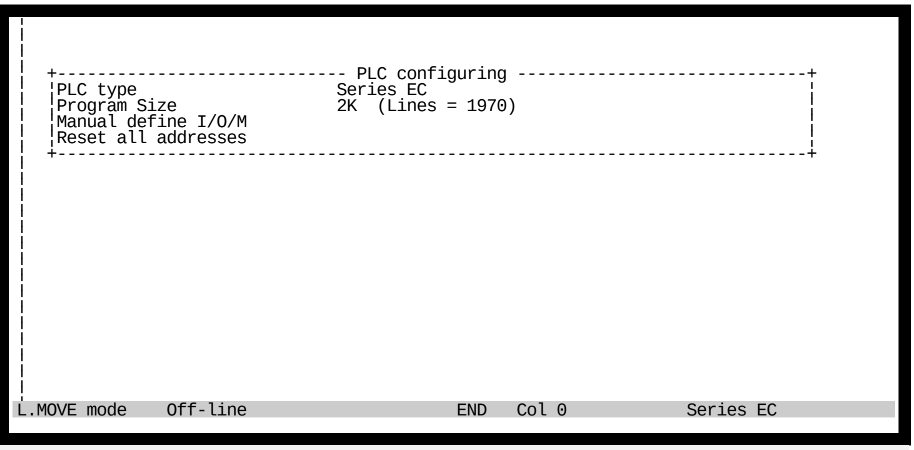


```
 +------------------------ Setup Personal Computer ------------------------+
 | Select by using up/down arrows and then press <Enter> to change.        |-+
 | Screen colors    - set the colors of the menus and programming modes.   | |
 |                                                                         | |
 | EGA/VGA 43/50    - is the computer has either a EGA or VGA video adapter| |
 | lines              you can increase the number of lines used on screen  | |
 | Posible to shift - if you don't get an 8 by pressing the                | |
 | arrow keys         shift up-arrow, set this to NO (Mostly portables)    |-+
 | User directory   - where to look for your projects (should end in \)    |
 |                    e.g. C:\PROJECT\ or B:\ or C:\ACTSIPE\               |
 | Language         - the language Actsip-E talks to you in. (Now English) |
 |                    SWE is for Swedish texts (Svenska texter)            |
 |                    (Normallly only English texts are supplied !!!)      |
 | Horizontal       - set this to YES if you have the funcyion keys        |
 | F1-F10  keys       arranged in a single row. Used by help function <F1> |
 |Press any key !, <ESC> to end.                                           |
L+-------------------------------------------------------------------------+
```

Press <Esc> and make the setup . Press <Esc> again to return to the drawing screen.

Press <Esc> again to do the PLC- setup:

Move with the <down arrow> to PLC and press <Enter>

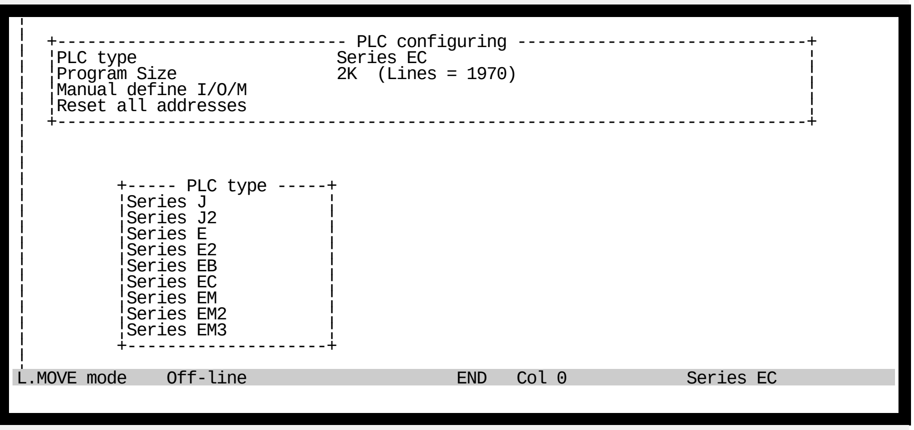


```
 +--------------------------- PLC configuring ----------------------------+
 |PLC type                     Series EC                                  |
 |Program Size                 2K  (Lines = 1970)                         |
 |Manual define I/O/M                                                     |
 |Reset all addresses                                                     |
 +------------------------------------------------------------------------+

L.MOVE mode    Off-line                    END   Col 0            Series EC
```

Choose PLC type. We are using series EC. Press <Enter> and move with the <down arrow> to Series EC.

Press <Enter> and reply Yes on the question about Mode 2 protocol.

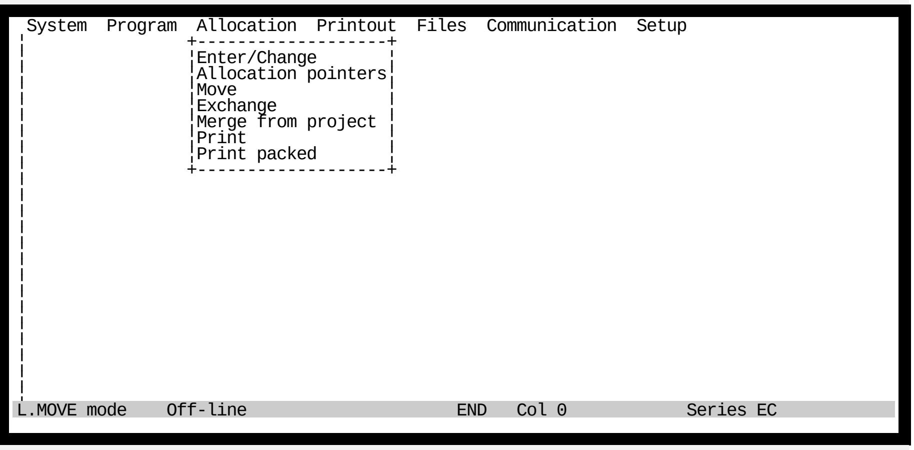
Press <Esc>



```
 +--------------------------- PLC configuring ----------------------------+
 |PLC type                     Series EC                                  |
 |Program Size                 2K  (Lines = 1970)                         |
 |Manual define I/O/M                                                     |
 |Reset all addresses                                                     |
 +------------------------------------------------------------------------+

        +----- PLC type -----+
        |Series J            |
        |Series J2           |
        |Series E            |
        |Series E2           |
        |Series EB           |
        |Series EC           |
        |Series EM           |
        |Series EM2          |
        |Series EM3          |
        +--------------------+
L.MOVE mode    Off-line                    END   Col 0            Series EC
```

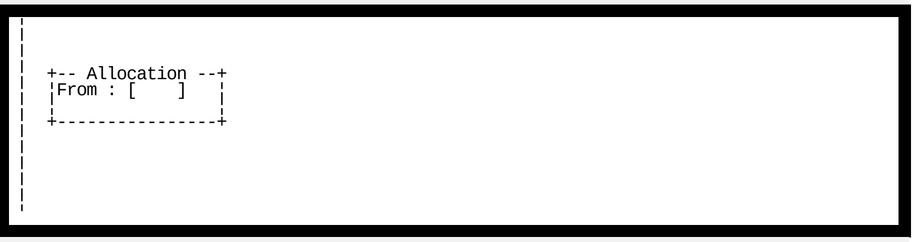
Allocate comments to the addresses:

If we already have connected the Inputs and Outputs to the PLC, we should start to allocate these to the right addresses.
Press <Esc> and go to "Allocation". Go to "Enter/Change" and Press <Enter>.



```
 System  Program  Allocation  Printout  Files  Communication  Setup
                 +-------------------+
                 |Enter/Change       |
                 |Allocation pointers|
                 |Move               |
                 |Exchange           |
                 |Merge from project |
                 |Print              |
                 |Print packed       |
                 +-------------------+

L.MOVE mode    Off-line                    END   Col 0            Series EC
```

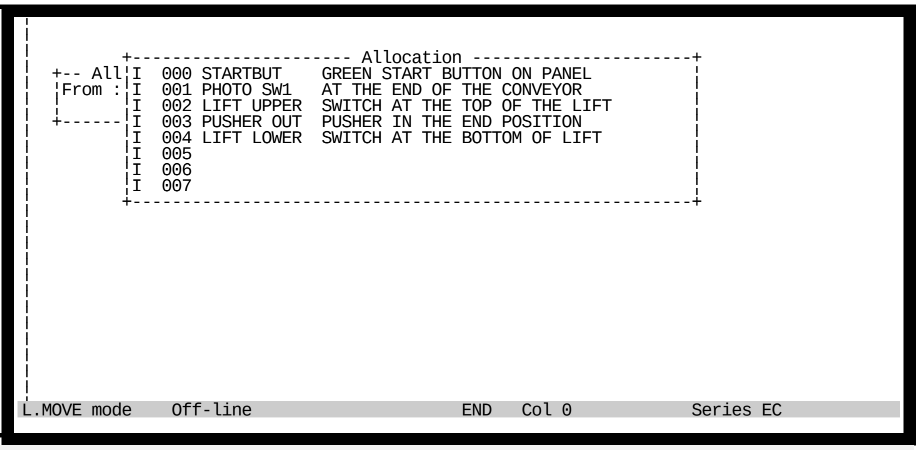
Type the first address to be allocated. We start with input 0.



```
 +-- Allocation --+
 |From : [    ]   |
 |                |
 +----------------+
```

Type the "Short comments" (max. 10 characters) to the left and long comments (max. 30 characters) to the right. You can switch between the columns with the <Tab>-key. Escape with <Esc>

ESC, ESC



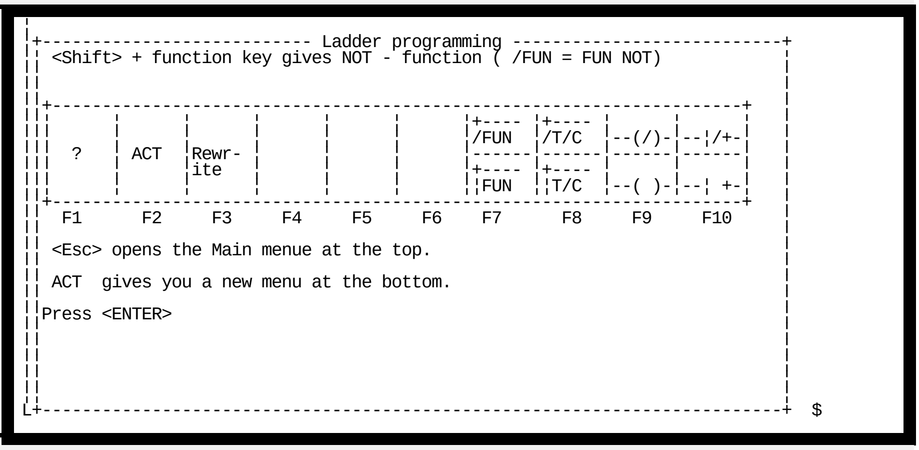
```
|
|        +--------------------- Allocation ----------------------+
|  +-- All¦I  000 STARTBUT    GREEN START BUTTON ON PANEL         ¦
|  ¦From :¦I  001 PHOTO SW1   AT THE END OF THE CONVEYOR          ¦
|  ¦      ¦I  002 LIFT UPPER  SWITCH AT THE TOP OF THE LIFT       ¦
|  +------¦I  003 PUSHER OUT  PUSHER IN THE END POSITION          ¦
|         ¦I  004 LIFT LOWER  SWITCH AT THE BOTTOM OF LIFT        ¦
|         ¦I  005                                                  ¦
|         ¦I  006                                                  ¦
|         ¦I  007                                                  ¦
|        +--------------------------------------------------------+
|
L.MOVE mode    Off-line                     END   Col 0             Series EC
```

Draw Ladder diagram:

We are now ready to start drawing our ladder diagram. To see what drawing commands that are available, press F1. You can here see .e.g. the "closing contact" on F10.

F1

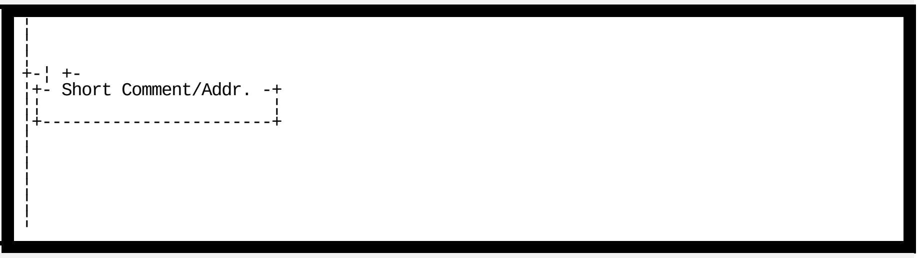
ESC



```
|+--------------------------- Ladder programming ---------------------------+
|| <Shift> + function key gives NOT - function ( /FUN = FUN NOT)            |
||                                                                          |
||+-------------------------------------------------------------------+    |
|||      |      |      |      |      |      |+---- |+---- |      |      |    |
|||  ?   | ACT  |Rewr- |      |      |      |/FUN  |/T/C  |--(/)-|--¦/+-|    |
|||      |      |ite   |      |      |      |+---- |+---- |      |      |    |
|||      |      |      |      |      |      |¦FUN  |¦T/C  |--( )-|--¦ +-|    |
||+-------------------------------------------------------------------+    |
||  F1     F2     F3     F4     F5     F6     F7      F8     F9     F10     |
||                                                                          |
|| <Esc> opens the Main menue at the top.                                   |
||                                                                          |
|| ACT  gives you a new menu at the bottom.                                 |
||                                                                          |
||Press <ENTER>                                                             |
||                                                                          |
L+--------------------------------------------------------------------------+  $
```

Start the sequence program with step 1. Make a closing contact.

F10

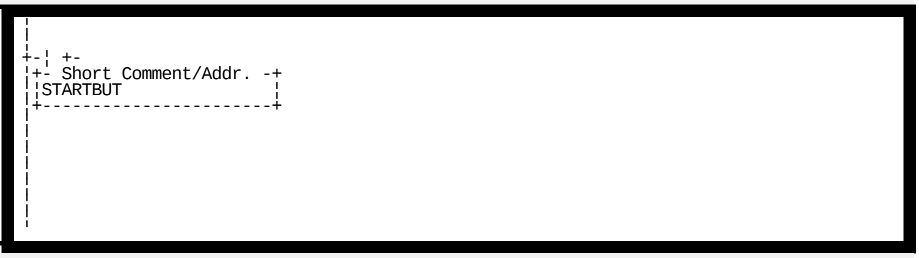


```
|
+-¦ +-
|+- Short Comment/Addr. -+
||                       ¦
|+-----------------------+
|
|
```

Type the name of the input.

START BUT

ENTER

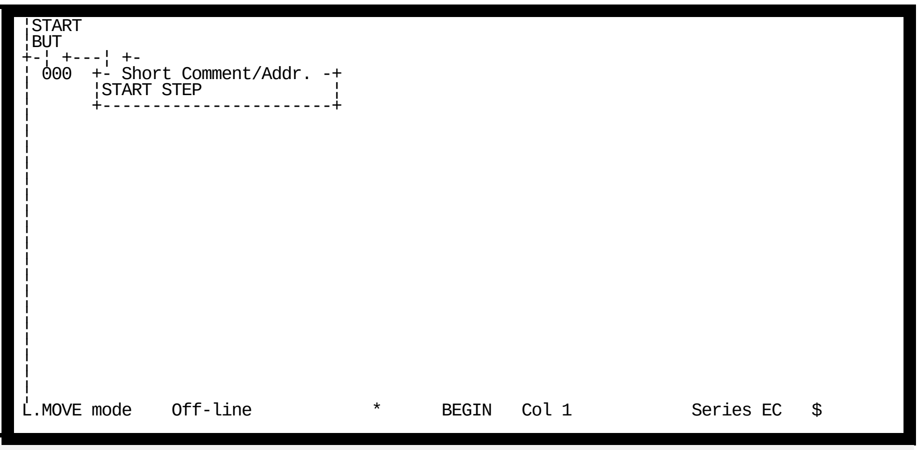


```
|
+-¦ +-
|+- Short Comment/Addr. -+
|¦STARTBUT               ¦
|+-----------------------+
|
|
```

As STARTBUT already is defined the address 000, is automatically written under the contact.
Continue with the next contact (serial connection), which is the address of the previous step.

F10 START STEP

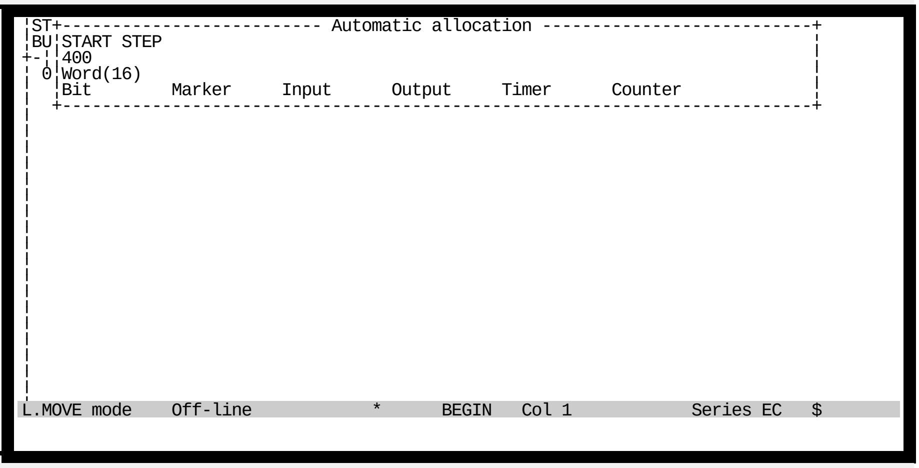
ENTER



```
¦START
¦BUT
+-¦ +---¦ +-
¦ 000  +- Short Comment/Addr. -+
¦      ¦START STEP             ¦
¦      +-----------------------+
¦
¦
¦
L.MOVE mode    Off-line          *      BEGIN   Col 1          Series EC   $
```

As START STEP is not defined before, the **"Automatic Allocation"** will turn up. Here the first free address is suggested . Normally you can accept the address which is suggested. Check if the type (Input, Output, etc.) is correct. Press <Enter>

ENTER



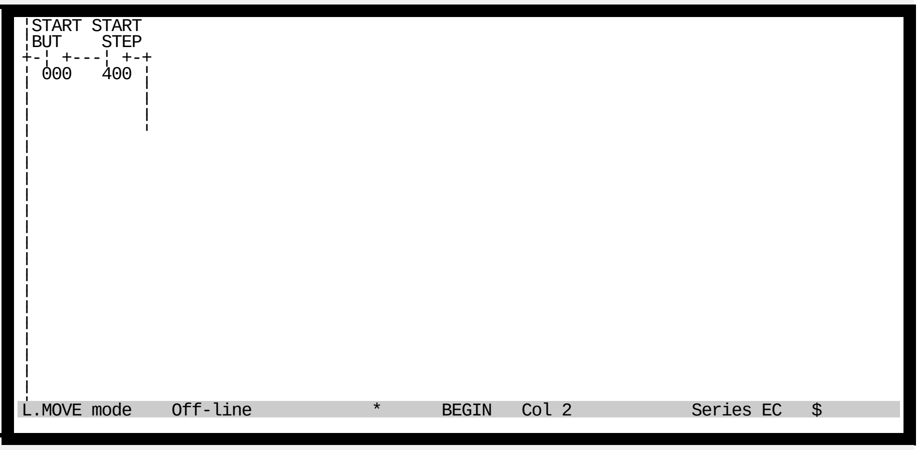
```
¦ST+------------------------- Automatic allocation --------------------------+
¦BU¦START STEP                                                                ¦
+-¦¦400                                                                       ¦
¦ 0¦Word(16)                                                                  ¦
¦  ¦Bit        Marker     Input      Output     Timer      Counter            ¦
¦  +--------------------------------------------------------------------------+
¦
¦
¦
L.MOVE mode    Off-line          *      BEGIN   Col 1          Series EC   $
```

As the step shall have a self hold function we must make a parallel connection.
Draw a line down and through pressing <Shift> <down arrow> (or if the PC does not have shift function on the arrow keys press <Alt><down arrow>)

Shift⇧ + ↓

or

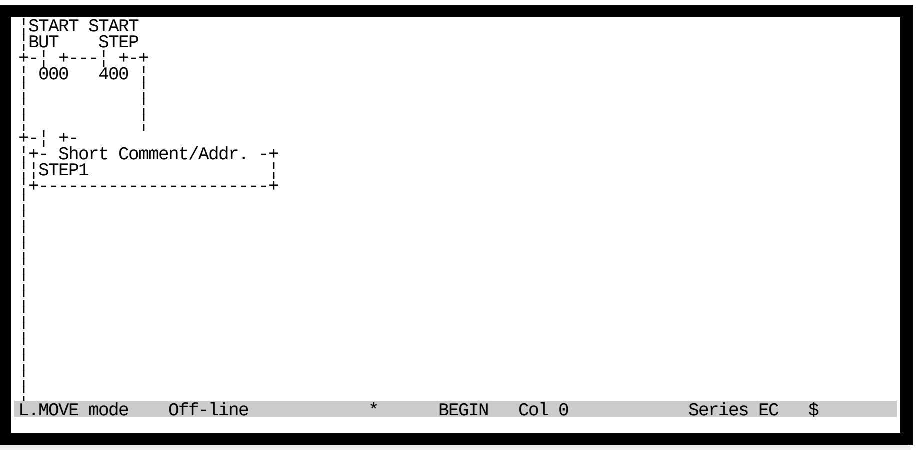
Alt + ↓



```
¦START START
¦BUT    STEP
+-¦ +---¦ +-+
¦ 000   400 ¦
¦           ¦
¦           ¦
¦
¦
¦
L.MOVE mode    Off-line          *      BEGIN   Col 2          Series EC   $
```

Continue to the left with <left arrow> and Press F10 to create the self hold. Type STEP1 and Press <Enter>. Accept the address and Press <Enter>.

←,
←
F10
STEP1
ENTER
ENTER

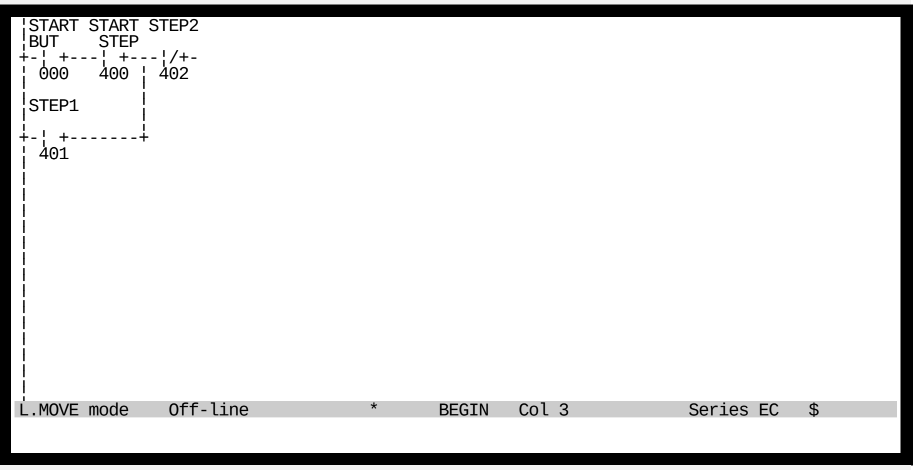


```
¦START START
¦BUT    STEP
+-¦ +---¦ +-+
¦ 000   400 ¦
¦           ¦
¦           ¦
+-¦ +-
¦+- Short Comment/Addr. -+
¦¦STEP1                  ¦
¦+-----------------------+
¦
¦
¦
¦
¦
¦
¦
¦
L.MOVE mode    Off-line            *       BEGIN    Col 0              Series EC    $
```

Draw a line to the right with <Shift><right arrow>, and go up with <up arrow> .
We shall now make a serial connection with the breaking step STEP2.

Press <Shift>+F10.
Type STEP2.

→,
↑
Shift⇧
+F10
STEP
2
ENTER

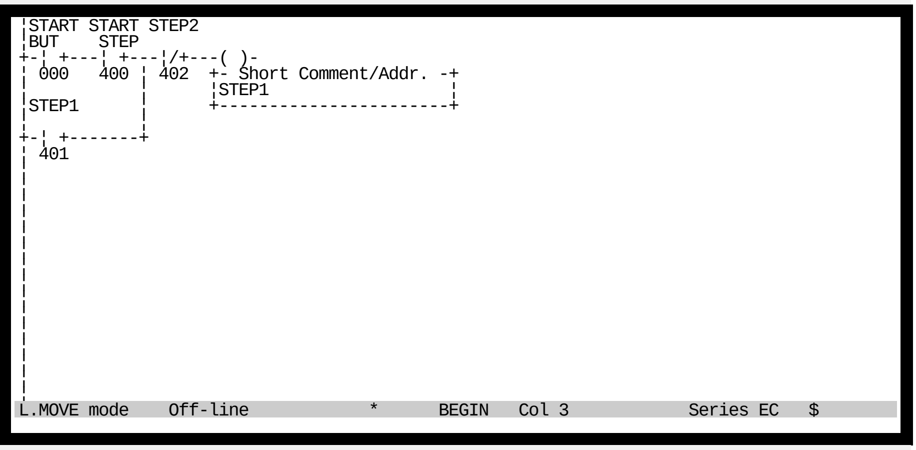


```
¦START START STEP2
¦BUT    STEP
+-¦ +---¦ +---¦/+-
¦ 000   400 ¦ 402
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
¦
¦
¦
¦
¦
¦
¦
L.MOVE mode    Off-line            *       BEGIN    Col 3              Series EC    $
```

Finally we create the output, which is STEP1.

Press F9 and type STEP1.

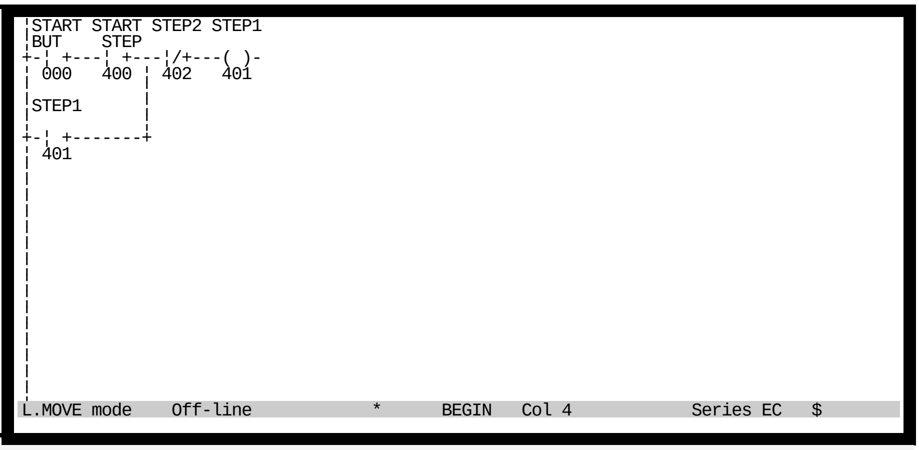
F9
STEG
1
ENTER



```
¦START START STEP2
¦BUT    STEP
+-¦ +---¦ +---¦/+---( )-
¦ 000   400 ¦ 402  +- Short Comment/Addr. -+
¦           ¦      ¦STEP1                  ¦
¦STEP1      ¦      +-----------------------+
¦           ¦
+-¦ +-------+
¦ 401
¦
¦
¦
¦
¦
¦
¦
¦
L.MOVE mode    Off-line            *       BEGIN    Col 3              Series EC    $
```

Until now we have only drawn the circuit. To turn it into real code in the program, **Press <Insert>**

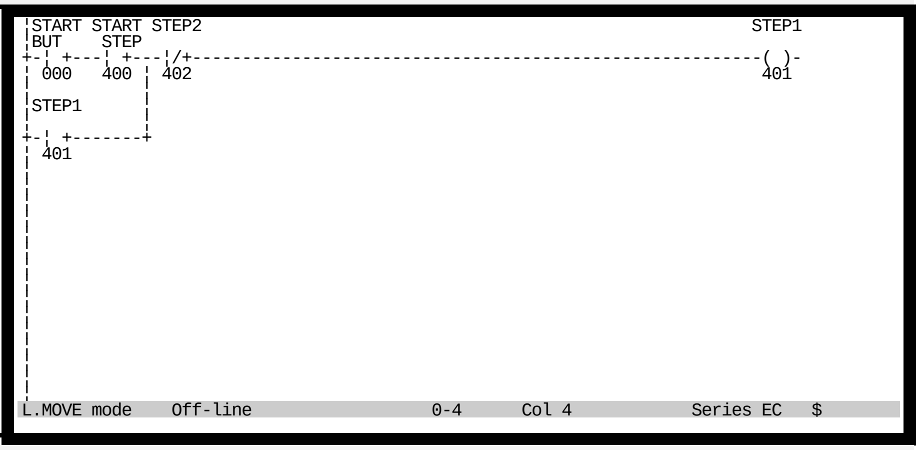
Insert



```
¦START START STEP2 STEP1
¦BUT    STEP
+-¦ +---¦ +---¦/+---( )-
¦ 000   400 ¦ 402   401
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
L.MOVE mode    Off-line            *      BEGIN   Col 4            Series EC   $
```

The first block is now ready and we start the second. Go with the arrow keys to the left under the finished block.

↓ ↓ ← ←

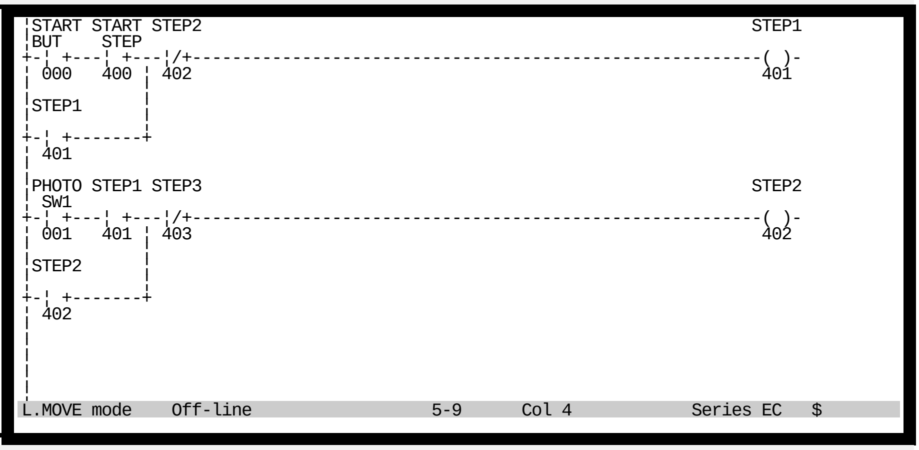


```
¦START START STEP2                                                       STEP1
¦BUT    STEP
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 000   400 ¦ 402                                                         401
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
¦
L.MOVE mode    Off-line                  0-4      Col 4            Series EC   $
```

Create block no 2 in the same way.

F10

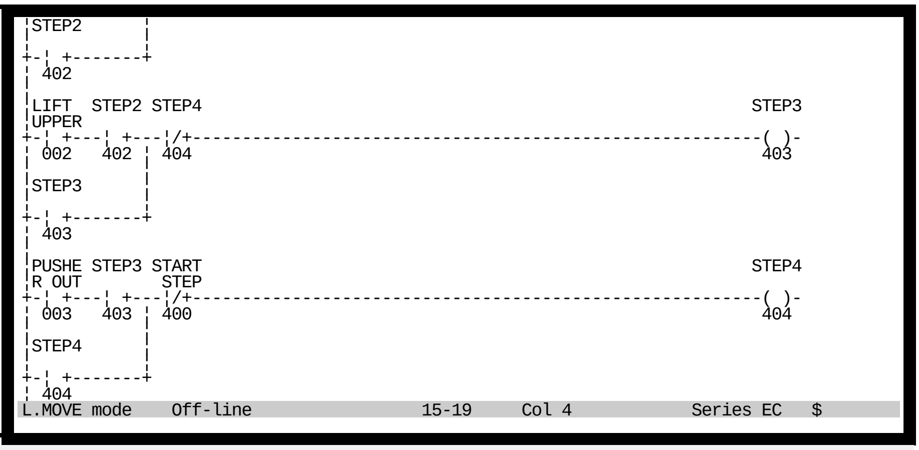
etc.



```
¦START START STEP2                                                       STEP1
¦BUT    STEP
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 000   400 ¦ 402                                                         401
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
¦PHOTO STEP1 STEP3                                                       STEP2
¦ SW1
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 001   401 ¦ 403                                                         402
¦           ¦
¦STEP2      ¦
¦           ¦
+-¦ +-------+
¦ 402
¦
¦
¦
¦
L.MOVE mode    Off-line                  5-9      Col 4            Series EC   $
```

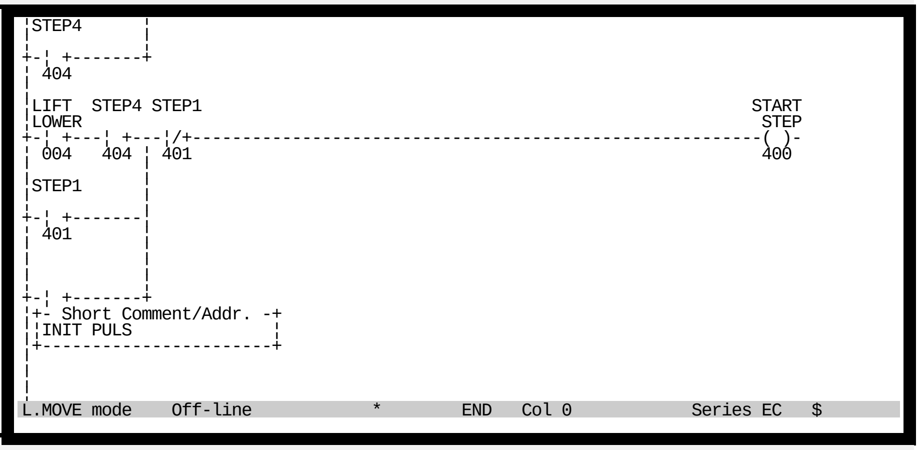
Create block no 3 and 4 in the same way.

At the bottom line you can see the line number of the block that is treated at the moment.



```
¦STEP2      ¦
¦           ¦
+-¦ +-------+
¦ 402
¦
¦LIFT  STEP2 STEP4                                                        STEP3
¦UPPER
+-¦ +---¦ +---¦/+----------------------------------------------------------( )-
¦ 002   402 ¦ 404                                                          403
¦           ¦
¦STEP3      ¦
¦           ¦
+-¦ +-------+
¦ 403
¦
¦PUSHE STEP3 START                                                        STEP4
¦R OUT        STEP
+-¦ +---¦ +---¦/+----------------------------------------------------------( )-
¦ 003   403 ¦ 400                                                          404
¦           ¦
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
L.MOVE mode    Off-line                 15-19     Col 4            Series EC   $
```

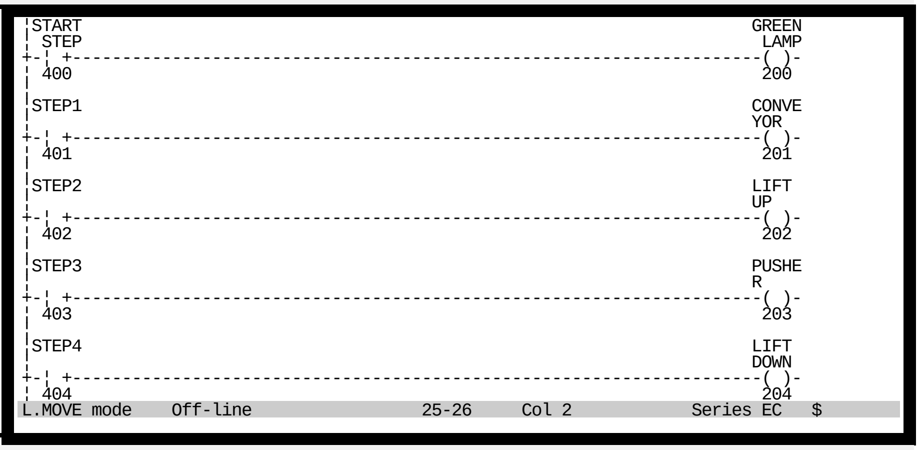
End the sequence part with the start step.
We must here also make a parallel connection to the Init marker (967) so the program starts at power on.



```
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
¦
¦LIFT  STEP4 STEP1                                                        START
¦LOWER                                                                     STEP
+-¦ +---¦ +---¦/+----------------------------------------------------------( )-
¦ 004   404 ¦ 401                                                          400
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------¦
¦ 401       ¦
¦           ¦
¦           ¦
+-¦ +-------+
¦+- Short Comment/Addr. -+
¦¦INIT PULS              ¦
¦+-----------------------+
¦
¦
¦
L.MOVE mode    Off-line            *        END   Col 0            Series EC   $
```

We are now ready with the sequence part.

We must also create the output part of the program.



```
¦START                                                                    GREEN
¦ STEP                                                                     LAMP
+-¦ +----------------------------------------------------------------------( )-
¦ 400                                                                      200
¦
¦STEP1                                                                    CONVE
¦                                                                         YOR
+-¦ +----------------------------------------------------------------------( )-
¦ 401                                                                      201
¦
¦STEP2                                                                    LIFT
¦                                                                         UP
+-¦ +----------------------------------------------------------------------( )-
¦ 402                                                                      202
¦
¦STEP3                                                                    PUSHE
¦                                                                         R
+-¦ +----------------------------------------------------------------------( )-
¦ 403                                                                      203
¦
¦STEP4                                                                    LIFT
¦                                                                         DOWN
+-¦ +----------------------------------------------------------------------( )-
¦ 404                                                                      204
L.MOVE mode    Off-line                 25-26     Col 2            Series EC   $
```

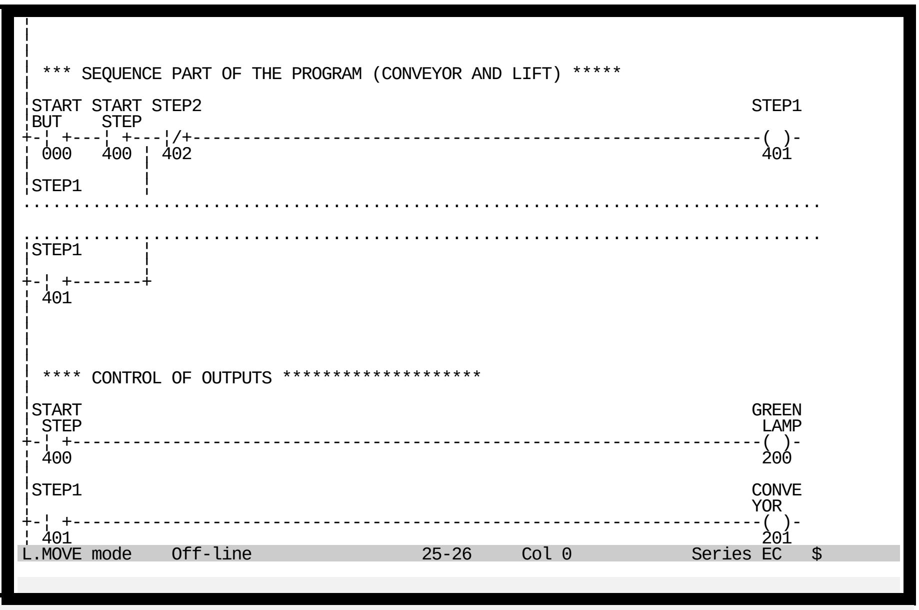
Write Comments:

It is recommended to write as many comments as possible for your own use but also to make the program easy to read for other people.
Type the line comments through moving to the block and press <Enter>. Type the comment and Press <Esc>

ENTER

Com ment

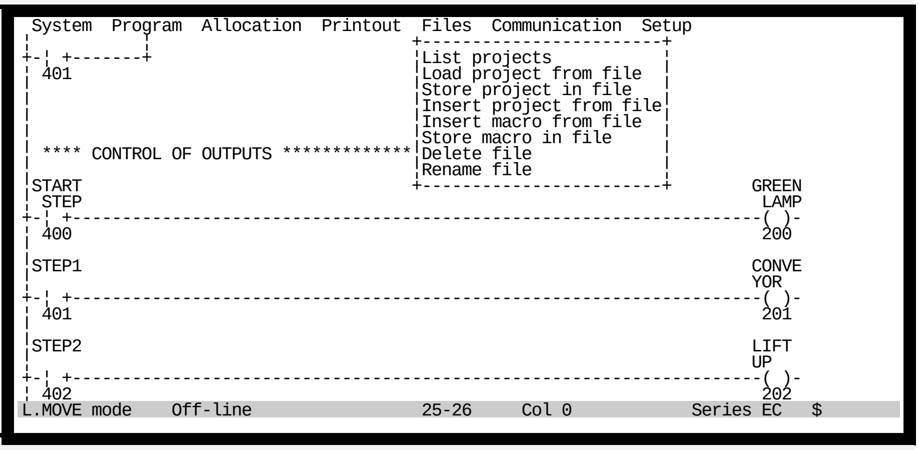
ESC



```
|
|
| *** SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
|
|START START STEP2                                                   STEP1
|BUT    STEP
+-| +---| +---|/+----------------------------------------------------( )-
| 000   400 | 402                                                    401
|           |
|STEP1      |
.................................................................................
.................................................................................
|STEP1      |
|           |
+-| +-------+
| 401
|
|
|
| **** CONTROL OF OUTPUTS ********************
|
|START                                                               GREEN
| STEP                                                                LAMP
+-| +----------------------------------------------------------------( )-
| 400                                                                 200
|
|STEP1                                                               CONVE
|                                                                    YOR
+-| +----------------------------------------------------------------( )-
| 401                                                                 201
L.MOVE mode    Off-line                 25-26     Col 0           Series EC   $
```

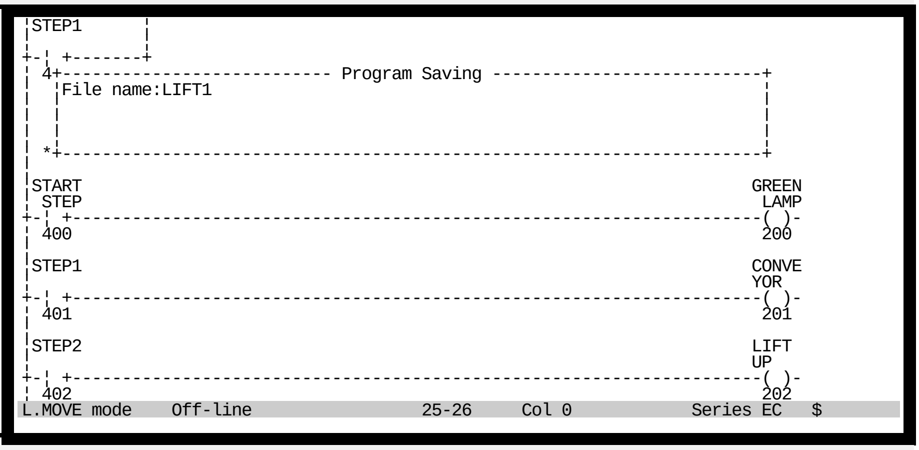
Save the program:

It is important to save the program repeatedly. Save sometimes with different version numbers, so you can go back to an earlier version if needed.
Press <Esc> and go to "Files". Go down and to and choose "Store project in file".



```
 System  Program  Allocation  Printout  Files  Communication  Setup
|           |                         +-------------------------+
+-| +-------+                         |List projects            |
| 401                                 |Load project from file   |
|                                     |Store project in file    |
|                                     |Insert project from file |
|                                     |Insert macro from file   |
|                                     |Store macro in file      |
| **** CONTROL OF OUTPUTS ************|Delete file              |
|                                     |Rename file              |
|START                                +-------------------------+    GREEN
| STEP                                                                LAMP
+-| +----------------------------------------------------------------( )-
| 400                                                                 200
|
|STEP1                                                               CONVE
|                                                                    YOR
+-| +----------------------------------------------------------------( )-
| 401                                                                 201
|
|STEP2                                                               LIFT
|                                                                    UP
+-| +----------------------------------------------------------------( )-
| 402                                                                 202
L.MOVE mode    Off-line                 25-26     Col 0           Series EC   $
```

Name the project and Press <Enter>

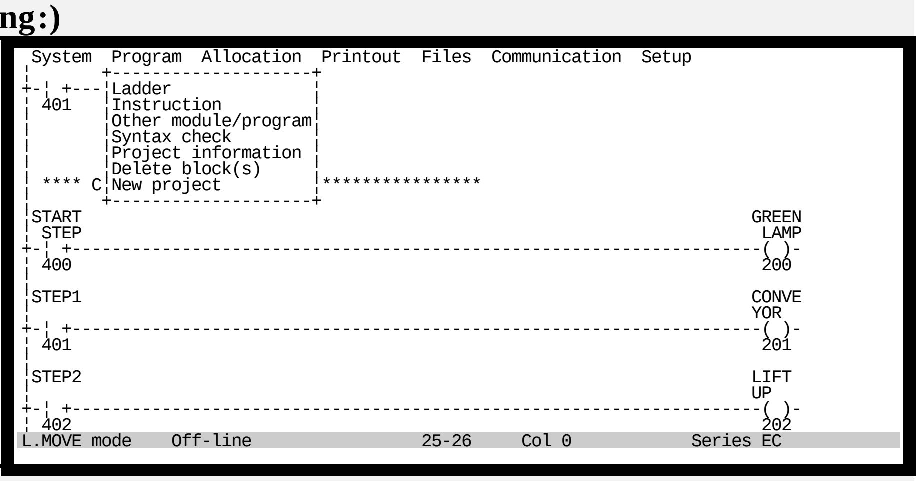


```
|STEP1      |
|           |
+-| +-------+
| 4+-------------------------- Program Saving ---------------------------+
|  |File name:LIFT1                                                      |
|  |                                                                     |
|  |                                                                     |
|  |                                                                     |
| *+---------------------------------------------------------------------+
|
|START                                                               GREEN
| STEP                                                                LAMP
+-| +----------------------------------------------------------------( )-
| 400                                                                 200
|
|STEP1                                                               CONVE
|                                                                    YOR
+-| +----------------------------------------------------------------( )-
| 401                                                                 201
|
|STEP2                                                               LIFT
|                                                                    UP
+-| +----------------------------------------------------------------( )-
| 402                                                                 202
L.MOVE mode    Off-line                 25-26     Col 0           Series EC   $
```

(Instruction code programming:)

Press <Esc> and go to the "Program" menu.

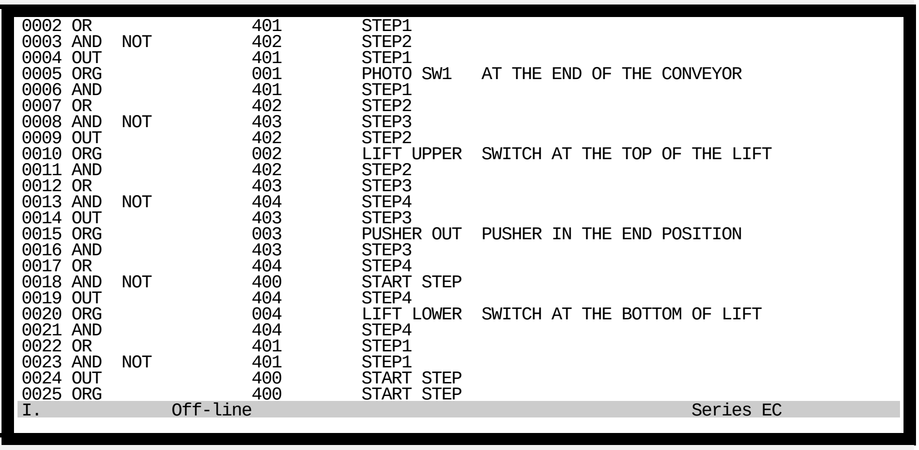
Here you can (if you want) go to "Instruction", where you can see and edit the same program in instruction code.



```
 System  Program  Allocation  Printout  Files  Communication  Setup
 ¦       +---------------------+
+-¦ +---¦Ladder               ¦
¦ 401   ¦Instruction          ¦
¦       ¦Other module/program ¦
¦       ¦Syntax check         ¦
¦       ¦Project information  ¦
¦       ¦Delete block(s)      ¦
¦ **** C¦New project          ¦****************
¦       +---------------------+
¦START                                                                  GREEN
¦ STEP                                                                   LAMP
+-¦ +-------------------------------------------------------------------( )-
¦ 400                                                                    200
¦
¦STEP1                                                                  CONVE
¦                                                                       YOR
+-¦ +-------------------------------------------------------------------( )-
¦ 401                                                                    201
¦
¦STEP2                                                                  LIFT
¦                                                                       UP
+-¦ +-------------------------------------------------------------------( )-
¦ 402                                                                    202
L.MOVE mode    Off-line                 25-26     Col 0           Series EC
```

You can program directly in ladder. Press F1 and see what soft keys are used for this purpose.
E.g.. ORG is F10.

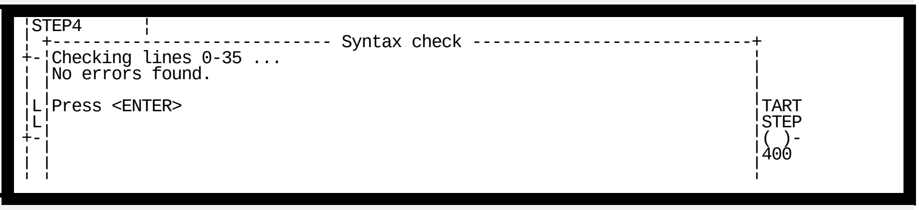
It is also possible to go the other direction back to ladder after editing in instruction code.



```
0002 OR                  401        STEP1
0003 AND  NOT            402        STEP2
0004 OUT                 401        STEP1
0005 ORG                 001        PHOTO SW1   AT THE END OF THE CONVEYOR
0006 AND                 401        STEP1
0007 OR                  402        STEP2
0008 AND  NOT            403        STEP3
0009 OUT                 402        STEP2
0010 ORG                 002        LIFT UPPER  SWITCH AT THE TOP OF THE LIFT
0011 AND                 402        STEP2
0012 OR                  403        STEP3
0013 AND  NOT            404        STEP4
0014 OUT                 403        STEP3
0015 ORG                 003        PUSHER OUT  PUSHER IN THE END POSITION
0016 AND                 403        STEP3
0017 OR                  404        STEP4
0018 AND  NOT            400        START STEP
0019 OUT                 404        STEP4
0020 ORG                 004        LIFT LOWER  SWITCH AT THE BOTTOM OF LIFT
0021 AND                 404        STEP4
0022 OR                  401        STEP1
0023 AND  NOT            401        STEP1
0024 OUT                 400        START STEP
0025 ORG                 400        START STEP
I.             Off-line                                   Series EC
```

Checking the program:

You can make a syntax check of the program in this menu.



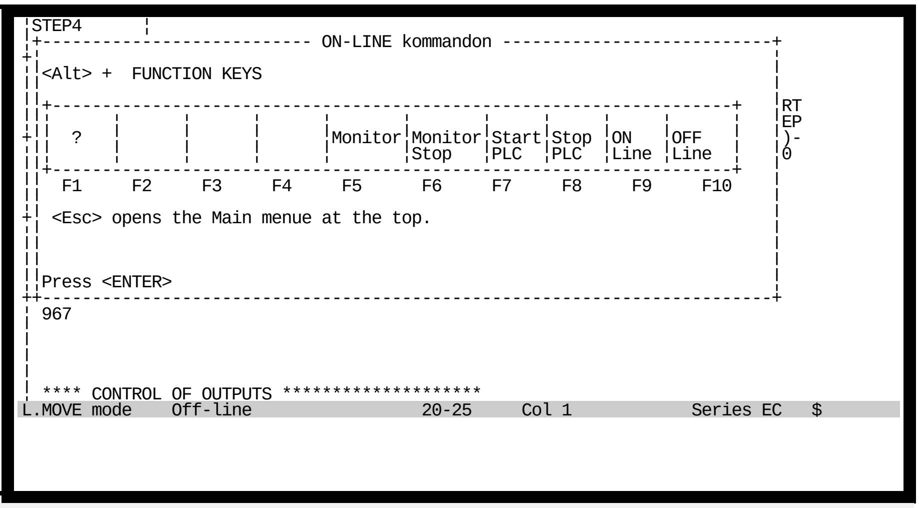
```
¦STEP4       ¦
¦ +--------------------------- Syntax check ----------------------------+
+-¦Checking lines 0-35 ...                                              ¦
¦ ¦No errors found.                                                     ¦
¦ ¦                                                                     ¦
¦L¦Press <ENTER>                                                        ¦TART
¦L¦                                                                     ¦STEP
+-¦                                                                     ¦( )-
¦ ¦                                                                     ¦400
¦ ¦                                                                     ¦
```

Go ON-line

As our program now is complete but so far not tested against the PLC and the machine, we shall now go ON-line. **Press <Alt>+F1 to get ON-line help.** We can see that "Monitor" is F5. This means that if you press **<Alt>+F5 you will go all the way** , ON-line, Start PLC, Monitor.

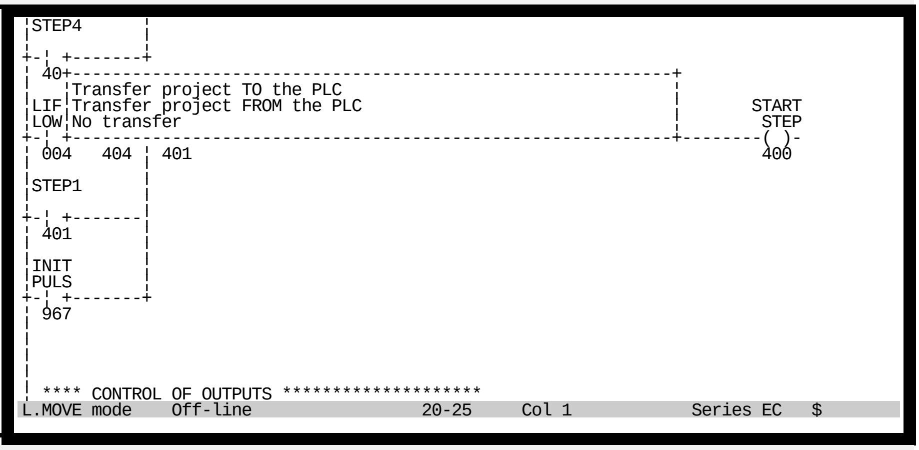
Alt

F5



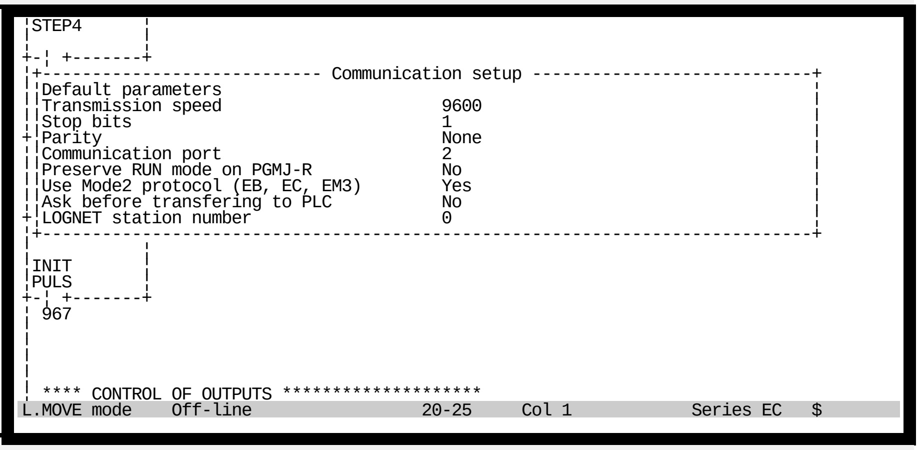
```
¦STEP4       ¦
+-------------------------- ON-LINE kommandon --------------------------+
¦¦<Alt> +  FUNCTION KEYS                                                 ¦
¦¦                                                                       ¦
¦¦+----------------------------------------------------------------+    ¦RT
¦¦¦      ¦       ¦      ¦      ¦Monitor¦Monitor¦Start¦Stop ¦ON   ¦OFF  ¦ ¦EP
+¦¦  ?   ¦       ¦      ¦      ¦       ¦Stop   ¦PLC  ¦PLC  ¦Line ¦Line ¦ ¦)-
¦¦+----------------------------------------------------------------+    ¦0
¦¦  F1     F2     F3     F4     F5      F6     F7    F8    F9    F10     ¦
+¦ <Esc> opens the Main menue at the top.                               ¦
¦¦                                                                       ¦
¦¦                                                                       ¦
¦¦Press <ENTER>                                                          ¦
++-----------------------------------------------------------------------+
¦ 967
¦
¦
¦
¦ **** CONTROL OF OUTPUTS ********************
L.MOVE mode    Off-line                 20-25     Col 1            Series EC   $
```

Reply Yes on the question "Transfer project to the PLC".



```
¦STEP4       ¦
¦            ¦
+-¦ +-------+
¦ 40+--------------------------------------------------------+
¦   ¦Transfer project TO the PLC                              ¦
¦LIF¦Transfer project FROM the PLC                            ¦        START
¦LOW¦No transfer                                              ¦         STEP
+-¦ +---------------------------------------------------------+--------( )-
¦ 004   404 ¦ 401                                                       400
¦            ¦
¦STEP1       ¦
¦            ¦
+-¦ +-------¦
¦ 401        ¦
¦            ¦
¦INIT        ¦
¦PULS        ¦
+-¦ +-------+
¦ 967
¦
¦
¦
¦ **** CONTROL OF OUTPUTS ********************
L.MOVE mode    Off-line                 20-25     Col 1            Series EC   $
```

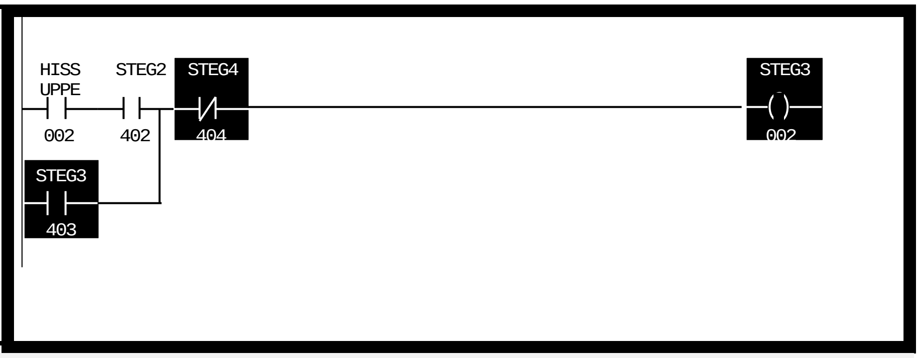
If you do not succeed the first time, go to "Communication Setup" and check e.g. if you have the right serial port connected.



```
¦STEP4       ¦
¦            ¦
+-¦ +-------+
¦+-------------------------- Communication setup ---------------------------+
¦¦Default parameters                                                        ¦
¦¦Transmission speed                        9600                            ¦
¦¦Stop bits                                 1                               ¦
+¦Parity                                    None                            ¦
¦¦Communication port                        2                               ¦
¦¦Preserve RUN mode on PGMJ-R               No                              ¦
¦¦Use Mode2 protocol (EB, EC, EM3)          Yes                             ¦
¦¦Ask before transfering to PLC             No                              ¦
+¦LOGNET station number                     0                               ¦
¦+--------------------------------------------------------------------------+
¦            ¦
¦INIT        ¦
¦PULS        ¦
+-¦ +-------+
¦ 967
¦
¦
¦
¦ **** CONTROL OF OUTPUTS ********************
L.MOVE mode    Off-line                 20-25     Col 1            Series EC   $
```

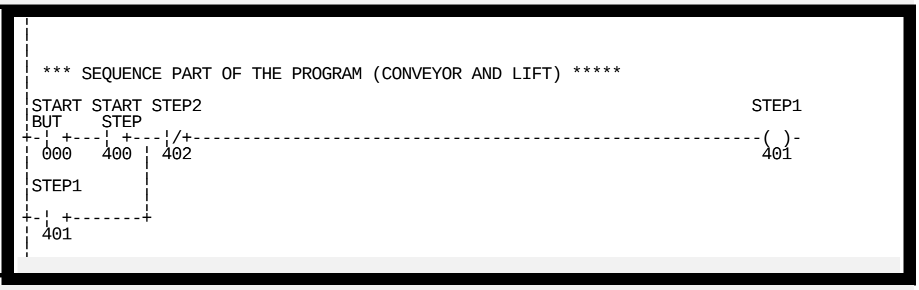
Monitor of the program:

When the program is transfered the program starts and the screen shows monitor of status on markers, inputs, outputs etc.
If they are active, the video will be reversed in these places.



Correcting the Program:

Let's say that we discover that that we have to insert a contact in the first block in serial with STARTBUT. Go to that location.

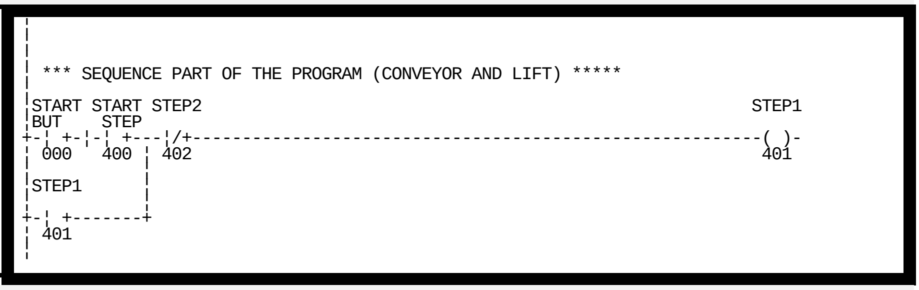


```
¦
¦
¦ *** SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
¦
¦START START STEP2                                                   STEP1
¦BUT    STEP
+-¦ +---¦ +---¦/+------------------------------------------------------( )-
¦ 000   400 ¦ 402                                                     401
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
```

Press F2 and choose the alternative "Horizontal expansion" (or press <H>).

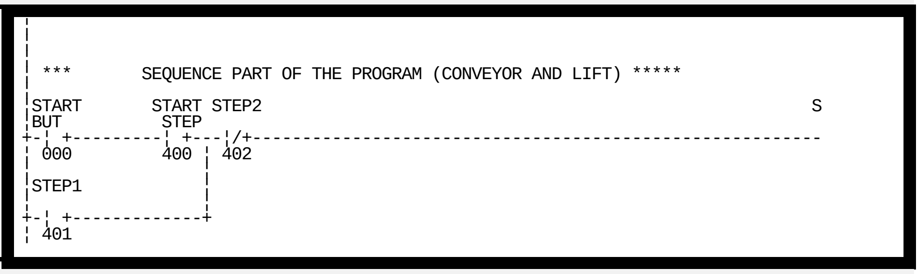
F2

H



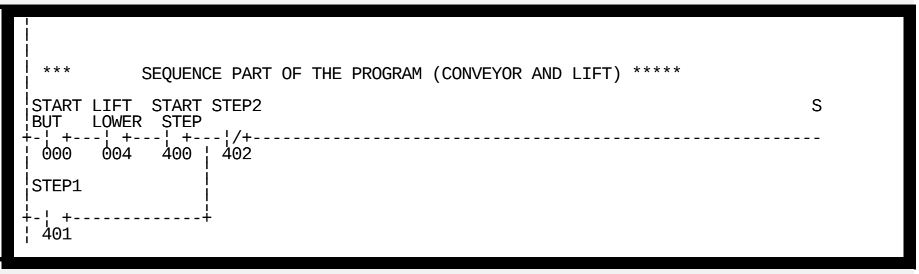
```
¦
¦
¦ *** SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
¦
¦START START STEP2                                                   STEP1
¦BUT    STEP
+-¦ +-¦-¦ +---¦/+------------------------------------------------------( )-
¦ 000   400 ¦ 402                                                     401
¦           ¦
¦STEP1      ¦
¦           ¦
+-¦ +-------+
¦ 401
¦
```

The Block expands.



```
¦
¦
¦ ***       SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
¦
¦START       START STEP2                                                   S
¦BUT          STEP
+-¦ +---------¦ +---¦/+-----------------------------------------------------
¦ 000         400 ¦ 402
¦                 ¦
¦STEP1            ¦
¦                 ¦
+-¦ +-------------+
¦ 401
```

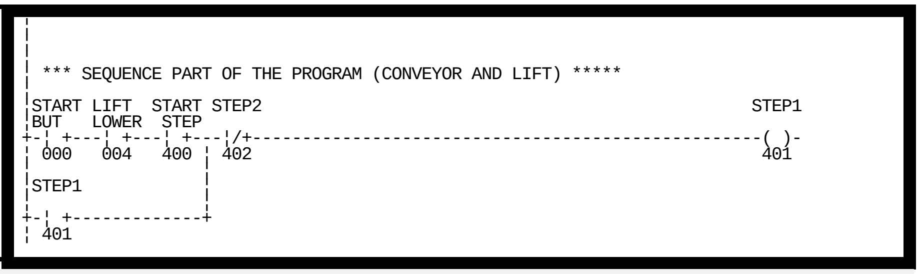
Type the new contact in the same way as earlier.



```
¦
¦
¦ ***       SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
¦
¦START LIFT  START STEP2                                                   S
¦BUT   LOWER  STEP
+-¦ +---¦ +---¦ +---¦/+-----------------------------------------------------
¦ 000   004   400 ¦ 402
¦                 ¦
¦STEP1            ¦
¦                 ¦
+-¦ +-------------+
¦ 401
```

Press <*> , which means modify block. (Not <Insert>)

*

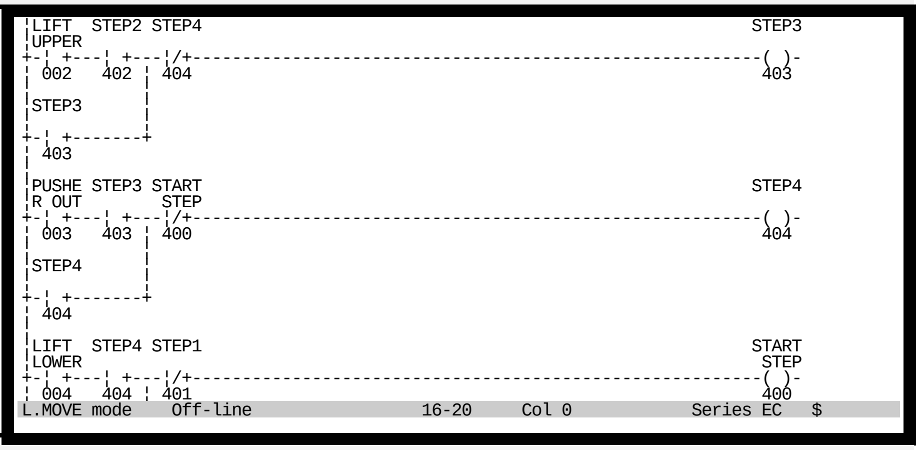


```
¦
¦
¦ *** SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
¦
¦START LIFT  START STEP2                                                    STEP1
¦BUT   LOWER  STEP
+-¦ +---¦ +---¦ +---¦/+-----------------------------------------------------( )-
¦ 000   004   400 ¦ 402                                                      401
¦                 ¦
¦STEP1            ¦
¦                 ¦
+-¦ +-------------+
¦ 401
```

Lets say that we also discover that we need a time delay before the lift starts to go down.
We will insert a timer, which delays the condition for entering STEP4.
Go to Block 4 ,
Press F2 and choose **"Vertical expansion**"

F2

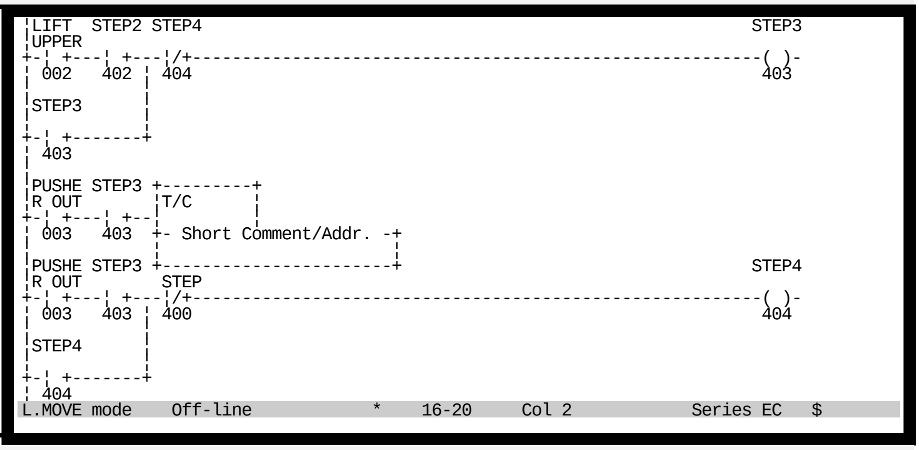
V



```
¦LIFT  STEP2 STEP4                                                          STEP3
¦UPPER
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 002   402 ¦ 404                                                            403
¦           ¦
¦STEP3      ¦
¦           ¦
+-¦ +-------+
¦ 403
¦
¦PUSHE STEP3 START                                                          STEP4
¦R OUT        STEP
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 003   403 ¦ 400                                                            404
¦           ¦
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
¦
¦LIFT  STEP4 STEP1                                                          START
¦LOWER                                                                       STEP
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 004   404 ¦ 401                                                            400
L.MOVE mode    Off-line                 16-20     Col 0            Series EC   $
```

Create the new block, which consists of a timer with the input condition Pusher out in step 3.
Press F8 to create the timer.
Name the timer e.g. "LIFT DELAY".

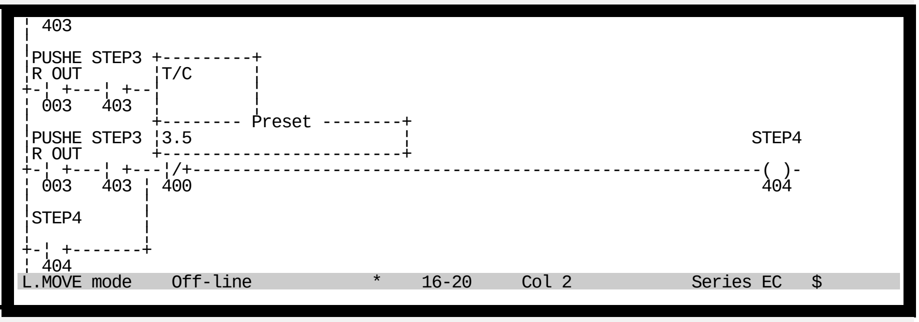
F8



```
¦LIFT  STEP2 STEP4                                                          STEP3
¦UPPER
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 002   402 ¦ 404                                                            403
¦           ¦
¦STEP3      ¦
¦           ¦
+-¦ +-------+
¦ 403
¦
¦PUSHE STEP3 +---------+
¦R OUT       ¦T/C      ¦
+-¦ +---¦ +--¦         ¦
¦ 003   403  +- Short Comment/Addr. -+
¦            ¦                       ¦
¦PUSHE STEP3 +-----------------------+                                      STEP4
¦R OUT        STEP
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 003   403 ¦ 400                                                            404
¦           ¦
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
L.MOVE mode    Off-line            *    16-20     Col 2            Series EC   $
```

You will now get a question about the preset value of the timer.
Enter the time in seconds or tenth of seconds.

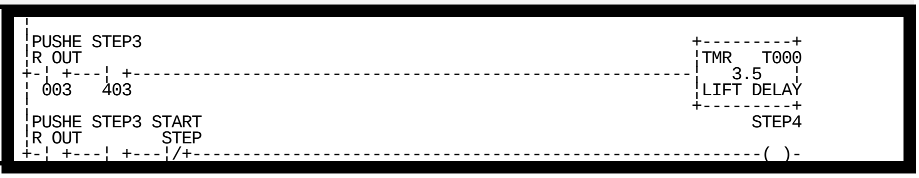
E.g. 3.5 <Enter>



```
¦ 403
¦
¦PUSHE STEP3 +---------+
¦R OUT       ¦T/C      ¦
+-¦ +---¦ +--¦         ¦
¦ 003   403  ¦         ¦
¦            +-------- Preset --------+
¦PUSHE STEP3 ¦3.5                     ¦                                     STEP4
¦R OUT       +------------------------+
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
¦ 003   403 ¦ 400                                                            404
¦           ¦
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
L.MOVE mode    Off-line            *    16-20     Col 2            Series EC   $
```

Press <Ins>.

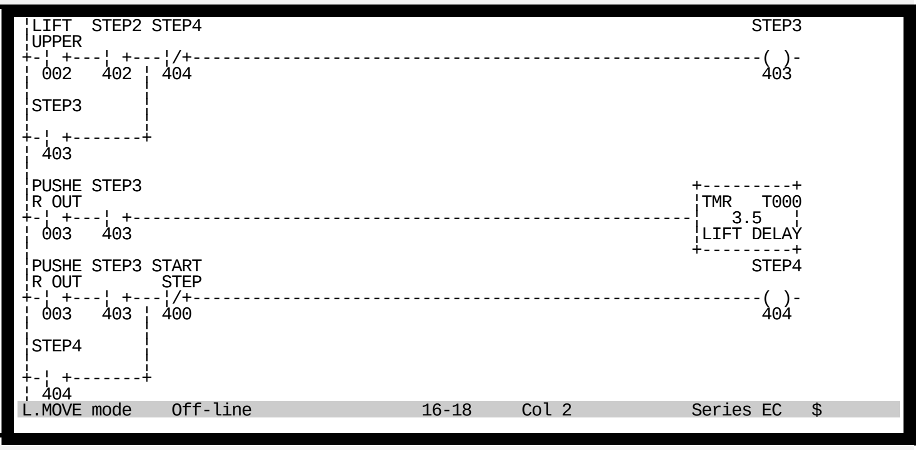
Insert



```
¦
¦PUSHE STEP3                                                           +---------+
¦R OUT                                                                 ¦TMR   T000
+-¦ +---¦ +------------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                            ¦LIFT DELAY
¦                                                                      +---------+
¦PUSHE STEP3 START                                                          STEP4
¦R OUT        STEP
+-¦ +---¦ +---¦/+-----------------------------------------------------------( )-
```

Change the serial connection Pusher Out and Step3 in the old block to the output of the timer by drawing over (replacing) the first contact and drawing a line on the second. After that, Press <*>.
Type a comment.

*

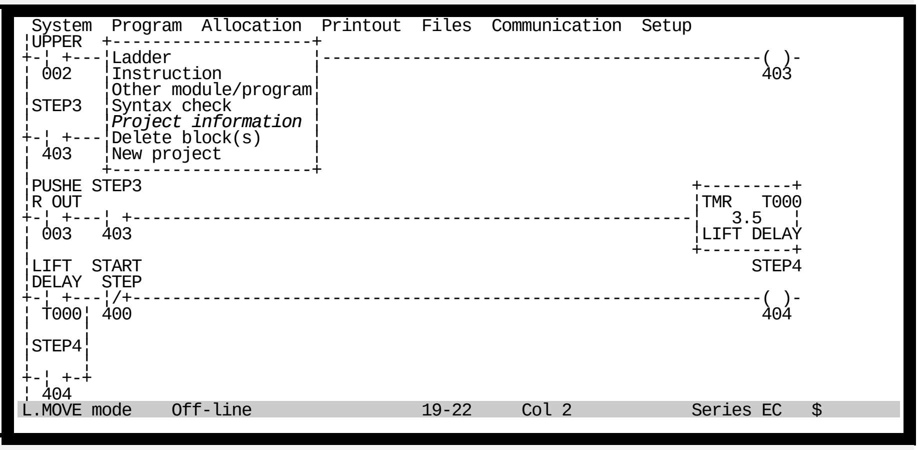


```
¦LIFT  STEP2 STEP4                                                       STEP3
¦UPPER
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 002   402 ¦ 404                                                         403
¦           ¦
¦STEP3      ¦
¦           ¦
+-¦ +-------+
¦ 403
¦
¦PUSHE STEP3                                                       +---------+
¦R OUT                                                             ¦TMR   T000
+-¦ +---¦ +--------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                        ¦LIFT DELAY
¦                                                                  +---------+
¦PUSHE STEP3 START                                                       STEP4
¦R OUT        STEP
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 003   403 ¦ 400                                                         404
¦           ¦
¦STEP4      ¦
¦           ¦
+-¦ +-------+
¦ 404
L.MOVE mode    Off-line                 16-18     Col 2            Series EC   $
```

Documentation:

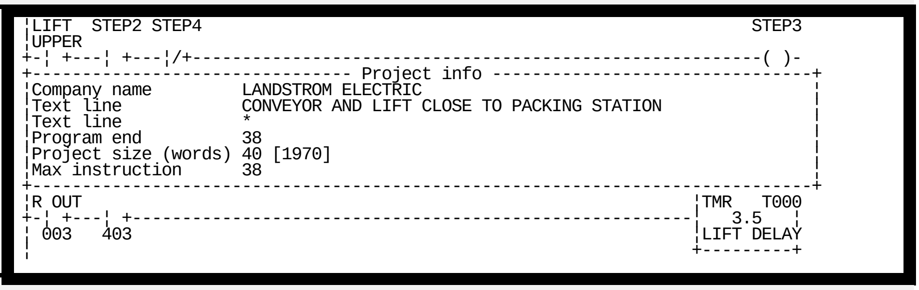
The program now works as we want it to and it is time for documentation.

Start to type the project information in the "Program" menu.



```
 System  Program  Allocation  Printout  Files  Communication  Setup
¦UPPER  +--------------------+
+-¦ +---¦Ladder              ¦--------------------------------------------( )-
¦ 002   ¦Instruction         ¦                                            403
¦       ¦Other module/program¦
¦STEP3  ¦Syntax check        ¦
¦       ¦Project information ¦
+-¦ +---¦Delete block(s)     ¦
¦ 403   ¦New project         ¦
¦       +--------------------+
¦PUSHE STEP3                                                       +---------+
¦R OUT                                                             ¦TMR   T000
+-¦ +---¦ +--------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                        ¦LIFT DELAY
¦                                                                  +---------+
¦LIFT  START                                                             STEP4
¦DELAY  STEP
+-¦ +---¦/+---------------------------------------------------------------( )-
¦ T000¦ 400                                                               404
¦     ¦
¦STEP4¦
¦     ¦
+-¦ +-+
¦ 404
L.MOVE mode    Off-line                 19-22     Col 2            Series EC   $
```

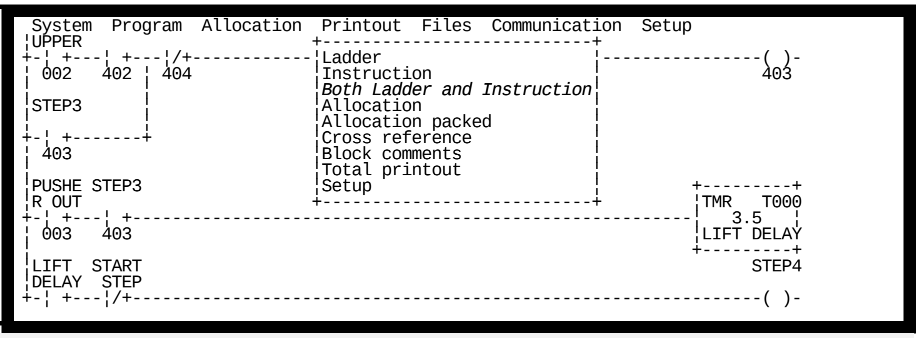
Beside typing of the project name you can here get information of the size of the project.



```
¦LIFT  STEP2 STEP4                                                       STEP3
¦UPPER
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
+-------------------------------- Project info --------------------------------+
¦Company name         LANDSTROM ELECTRIC                                       ¦
¦Text line            CONVEYOR AND LIFT CLOSE TO PACKING STATION               ¦
¦Text line            *                                                        ¦
¦Program end          38                                                       ¦
¦Project size (words) 40 [1970]                                                ¦
¦Max instruction      38                                                       ¦
+------------------------------------------------------------------------------+
¦R OUT                                                             ¦TMR   T000
+-¦ +---¦ +--------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                        ¦LIFT DELAY
¦                                                                  +---------+
```

We can now go to "Documentation".

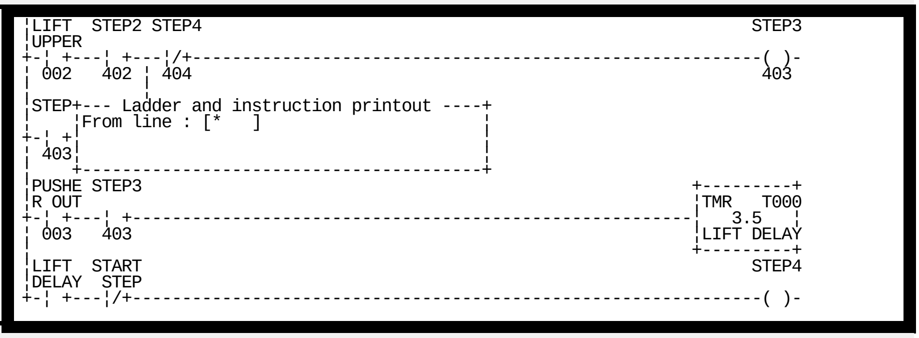
Here we can e.g. choose "Both Ladder and Instruction".



```
 System  Program  Allocation  Printout  Files  Communication  Setup
¦UPPER                       +---------------------------+
+-¦ +---¦ +---¦/+------------¦Ladder                     ¦----------------( )-
¦ 002   402 ¦ 404            ¦Instruction                ¦                403
¦           ¦                ¦Both Ladder and Instruction¦
¦STEP3      ¦                ¦Allocation                 ¦
¦           ¦                ¦Allocation packed          ¦
+-¦ +-------+                ¦Cross reference            ¦
¦ 403                        ¦Block comments             ¦
¦                            ¦Total printout             ¦
¦PUSHE STEP3                 ¦Setup                      ¦         +---------+
¦R OUT                       +---------------------------+         ¦TMR   T000
+-¦ +---¦ +--------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                        ¦LIFT DELAY
¦                                                                  +---------+
¦LIFT  START                                                             STEP4
¦DELAY  STEP
+-¦ +---¦/+---------------------------------------------------------------( )-
```

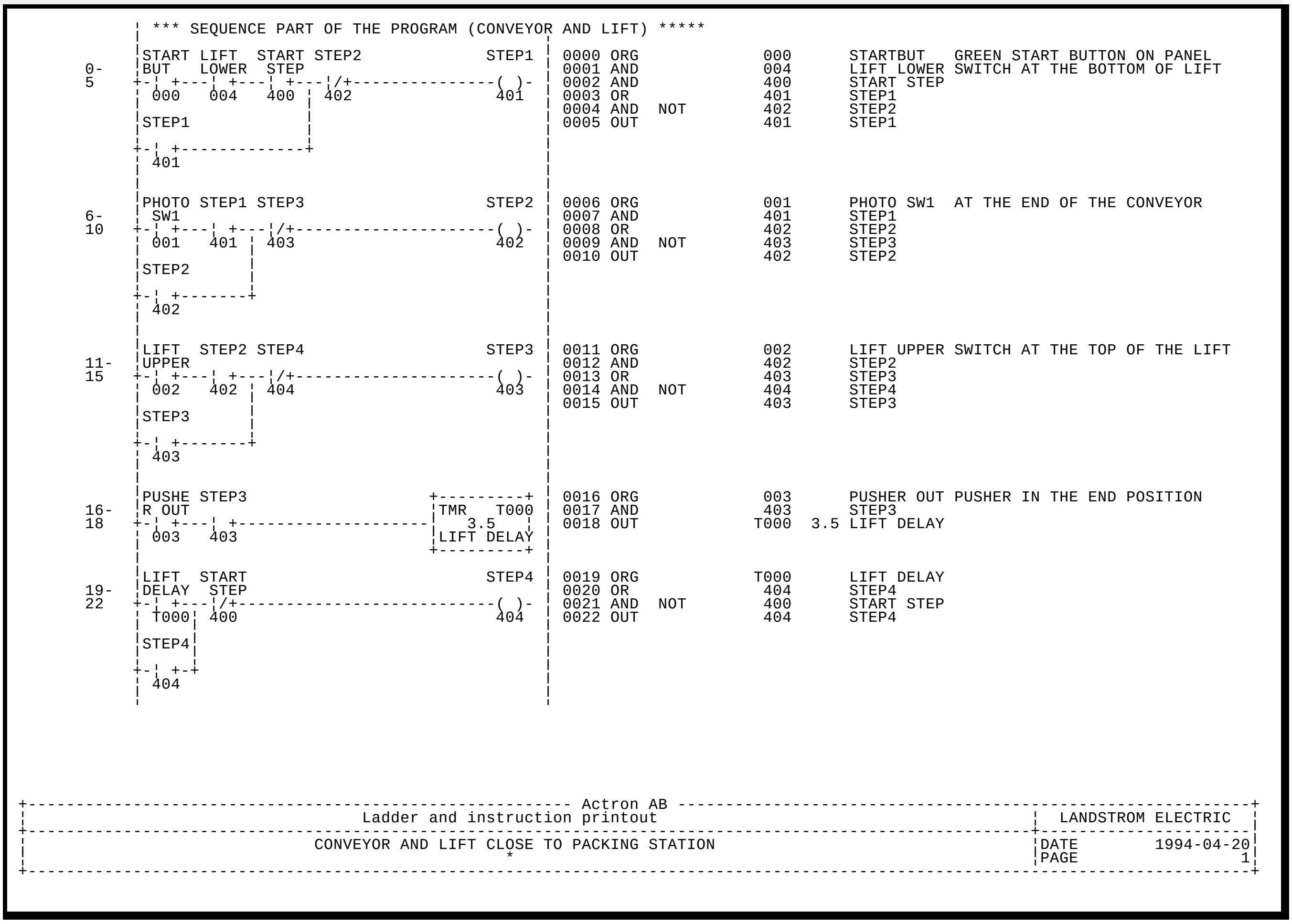
On the question "From line" and "To line" we can enter line numbers for a limited printout.

We can also enter "*"and get a printout of all the program.

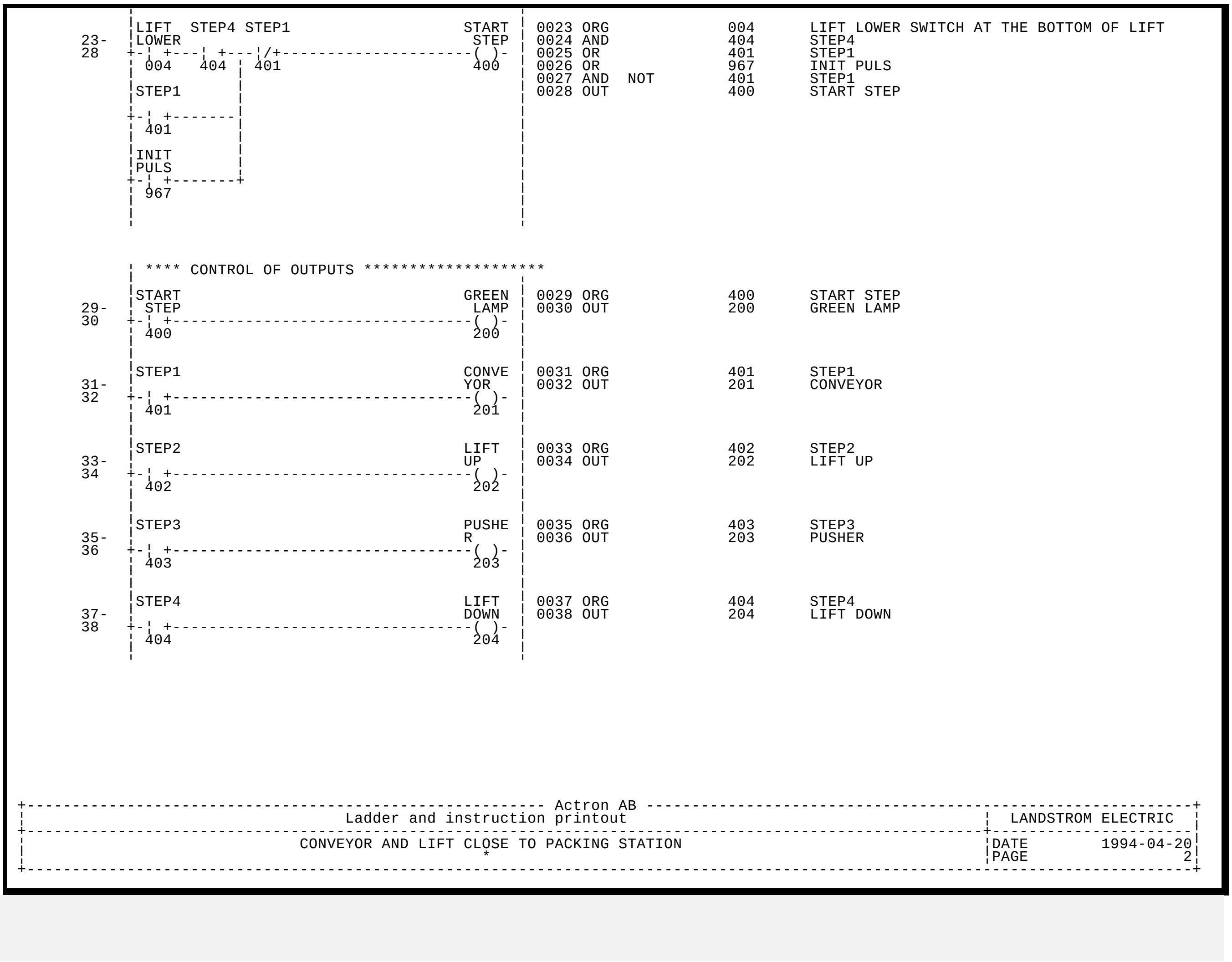


```
¦LIFT  STEP2 STEP4                                                       STEP3
¦UPPER
+-¦ +---¦ +---¦/+---------------------------------------------------------( )-
¦ 002   402 ¦ 404                                                         403
¦           ¦
¦STEP+--- Ladder and instruction printout ----+
¦    ¦From line : [*   ]                      ¦
+-¦ +¦                                        ¦
¦ 403¦                                        ¦
¦    +----------------------------------------+
¦PUSHE STEP3                                                       +---------+
¦R OUT                                                             ¦TMR   T000
+-¦ +---¦ +--------------------------------------------------------¦   3.5   ¦
¦ 003   403                                                        ¦LIFT DELAY
¦                                                                  +---------+
¦LIFT  START                                                             STEP4
¦DELAY  STEP
+-¦ +---¦/+---------------------------------------------------------------( )-
```

Printout



```
            ¦ *** SEQUENCE PART OF THE PROGRAM (CONVEYOR AND LIFT) *****
            ¦                                          ¦
            ¦START LIFT  START STEP2             STEP1 ¦ 0000 ORG             000      STARTBUT   GREEN START BUTTON ON PANEL
       0-   ¦BUT   LOWER  STEP                         ¦ 0001 AND             004      LIFT LOWER SWITCH AT THE BOTTOM OF LIFT
       5    +-¦ +---¦ +---¦ +---¦/+---------------( )- ¦ 0002 AND             400      START STEP
            ¦ 000   004   400 ¦ 402               401  ¦ 0003 OR              401      STEP1
            ¦                 ¦                        ¦ 0004 AND  NOT        402      STEP2
            ¦STEP1            ¦                        ¦ 0005 OUT             401      STEP1
            ¦                 ¦                        ¦
            +-¦ +-------------+                        ¦
            ¦ 401                                      ¦
            ¦                                          ¦
            ¦PHOTO STEP1 STEP3                   STEP2 ¦ 0006 ORG             001      PHOTO SW1  AT THE END OF THE CONVEYOR
       6-   ¦ SW1                                      ¦ 0007 AND             401      STEP1
       10   +-¦ +---¦ +---¦/+---------------------( )- ¦ 0008 OR              402      STEP2
            ¦ 001   401 ¦ 403                     402  ¦ 0009 AND  NOT        403      STEP3
            ¦           ¦                              ¦ 0010 OUT             402      STEP2
            ¦STEP2      ¦                              ¦
            ¦           ¦                              ¦
            +-¦ +-------+                              ¦
            ¦ 402                                      ¦
            ¦                                          ¦
            ¦LIFT  STEP2 STEP4                   STEP3 ¦ 0011 ORG             002      LIFT UPPER SWITCH AT THE TOP OF THE LIFT
       11-  ¦UPPER                                     ¦ 0012 AND             402      STEP2
       15   +-¦ +---¦ +---¦/+---------------------( )- ¦ 0013 OR              403      STEP3
            ¦ 002   402 ¦ 404                     403  ¦ 0014 AND  NOT        404      STEP4
            ¦           ¦                              ¦ 0015 OUT             403      STEP3
            ¦STEP3      ¦                              ¦
            ¦           ¦                              ¦
            +-¦ +-------+                              ¦
            ¦ 403                                      ¦
            ¦                                          ¦
            ¦PUSHE STEP3                   +---------+ ¦ 0016 ORG             003      PUSHER OUT PUSHER IN THE END POSITION
       16-  ¦R OUT                         ¦TMR   T000 ¦ 0017 AND             403      STEP3
       18   +-¦ +---¦ +--------------------¦   3.5   ¦ ¦ 0018 OUT            T000  3.5 LIFT DELAY
            ¦ 003   403                    ¦LIFT DELAY ¦
            ¦                              +---------+ ¦
            ¦LIFT  START                         STEP4 ¦ 0019 ORG            T000      LIFT DELAY
       19-  ¦DELAY  STEP                               ¦ 0020 OR              404      STEP4
       22   +-¦ +---¦/+---------------------------( )- ¦ 0021 AND  NOT        400      START STEP
            ¦ T000¦ 400                           404  ¦ 0022 OUT             404      STEP4
            ¦     ¦                                    ¦
            ¦STEP4¦                                    ¦
            ¦     ¦                                    ¦
            +-¦ +-+                                    ¦
            ¦ 404                                      ¦
            ¦                                          ¦



+--------------------------------------------------------- Actron AB ------------------------------------------------------------+
¦                                  Ladder and instruction printout                                       ¦  LANDSTROM ELECTRIC   ¦
+--------------------------------------------------------------------------------------------------------+-----------------------+
¦                              CONVEYOR AND LIFT CLOSE TO PACKING STATION                                ¦DATE         1994-04-20¦
¦                                                  *                                                     ¦PAGE                  1¦
+--------------------------------------------------------------------------------------------------------------------------------+
```



```
            ¦                                          ¦
            ¦LIFT  STEP4 STEP1                   START ¦ 0023 ORG             004      LIFT LOWER SWITCH AT THE BOTTOM OF LIFT
       23-  ¦LOWER                                STEP ¦ 0024 AND             404      STEP4
       28   +-¦ +---¦ +---¦/+---------------------( )- ¦ 0025 OR              401      STEP1
            ¦ 004   404 ¦ 401                     400  ¦ 0026 OR              967      INIT PULS
            ¦           ¦                              ¦ 0027 AND  NOT        401      STEP1
            ¦STEP1      ¦                              ¦ 0028 OUT             400      START STEP
            ¦           ¦                              ¦
            +-¦ +-------¦                              ¦
            ¦ 401       ¦                              ¦
            ¦           ¦                              ¦
            ¦INIT       ¦                              ¦
            ¦PULS       ¦                              ¦
            +-¦ +-------+                              ¦
            ¦ 967                                      ¦
            ¦                                          ¦
            ¦                                          ¦



            ¦ **** CONTROL OF OUTPUTS ********************
            ¦                                          ¦
            ¦START                               GREEN ¦ 0029 ORG             400      START STEP
       29-  ¦ STEP                                LAMP ¦ 0030 OUT             200      GREEN LAMP
       30   +-¦ +---------------------------------( )- ¦
            ¦ 400                                 200  ¦
            ¦                                          ¦
            ¦STEP1                               CONVE ¦ 0031 ORG             401      STEP1
       31-  ¦                                    YOR   ¦ 0032 OUT             201      CONVEYOR
       32   +-¦ +---------------------------------( )- ¦
            ¦ 401                                 201  ¦
            ¦                                          ¦
            ¦STEP2                               LIFT  ¦ 0033 ORG             402      STEP2
       33-  ¦                                    UP    ¦ 0034 OUT             202      LIFT UP
       34   +-¦ +---------------------------------( )- ¦
            ¦ 402                                 202  ¦
            ¦                                          ¦
            ¦STEP3                               PUSHE ¦ 0035 ORG             403      STEP3
       35-  ¦                                    R     ¦ 0036 OUT             203      PUSHER
       36   +-¦ +---------------------------------( )- ¦
            ¦ 403                                 203  ¦
            ¦                                          ¦
            ¦STEP4                               LIFT  ¦ 0037 ORG             404      STEP4
       37-  ¦                                    DOWN  ¦ 0038 OUT             204      LIFT DOWN
       38   +-¦ +---------------------------------( )- ¦
            ¦ 404                                 204  ¦
            ¦                                          ¦









+--------------------------------------------------------- Actron AB ------------------------------------------------------------+
¦                                   Ladder and instruction printout                                       ¦  LANDSTROM ELECTRIC  ¦
+---------------------------------------------------------------------------------------------------------+----------------------+
¦                              CONVEYOR AND LIFT CLOSE TO PACKING STATION                                 ¦DATE        1994-04-20¦
¦                                                  *                                                      ¦PAGE                 2¦
+---------------------------------------------------------------------------------------------------------+----------------------+
```

2.6. To program with ActGraph.

The purpose is to show how to get started and build an application from the start with ActGraph.

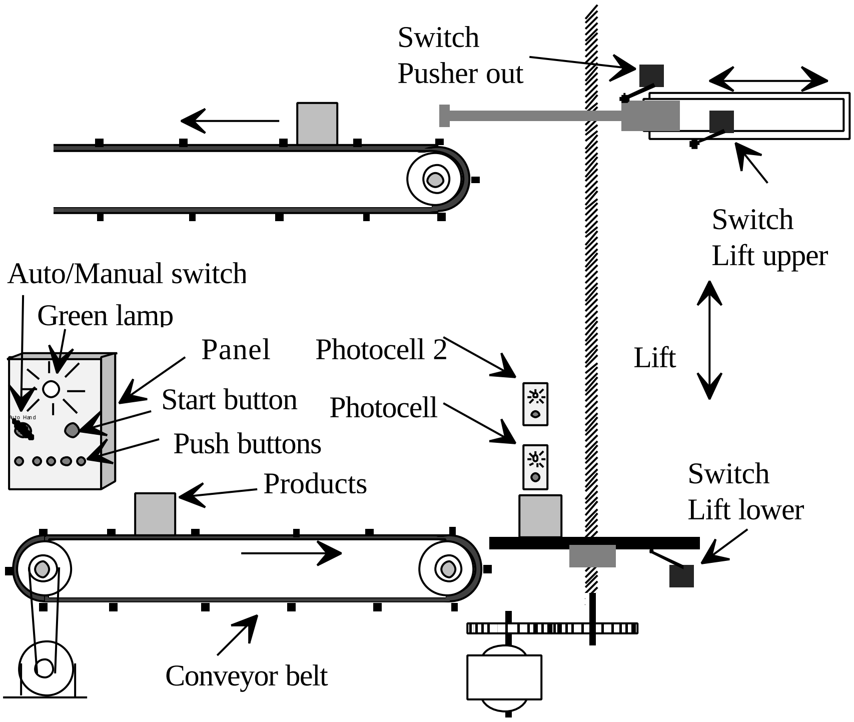
For more detailed description, see Manual for **ActGraph.**

We start from the same program example as in. But let us extend the example a little.

Example.

When the machine is energised a green lamp shall be turned ON. It will be ON until the operator pushes the Start button.
Then the lower conveyor shall start and move until the photocell in the end of the conveyor indicates.
Then the lift shall start going up and move until it reaches the top position.
Then the pusher moves until a switch indicates that it is out.
The lift moves down (while the pusher goes back automatically) until a switch indicates that it is down.
The sequence then repeats.

We also have an upper photocell, which indicates if the product is too big. We have also equipped the panel with an Auto / Manual switch and push buttons.



The program can be described graphic in the following way:

AUTOMATIC MODE GRAPH

```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + START BUTT
¦+---++------------+
¦¦001+¦  CONVEYOR  ¦
¦+---++------------+
¦  +---------------------------+
¦  + PHOTOCELL*                ¦
¦  ¦ /PHOTOCELL2               ¦:PRODUCT IS TOO
¦+---++--------------------+   ¦:BIG, Manuel
¦¦002+¦D LIFT UP   [D=1.5s]¦   ¦:sorting out
¦+---++--------------------+   + PHOTO SW1
¦  + LIFT UPPER                ¦ PHOTO SW2
¦+---++-------------+        +---++-------------+
¦¦003+¦  PUSHER     ¦        ¦005+¦  ALARM      ¦
¦+---++-------------+        +---++-------------+
¦  + PUSH OUT                  + /PHOTO SW2
¦+---++-----------+            ¦
¦¦004+¦  LIFT DOWN¦            ¦
¦+---++-----------+            ¦
¦  + LIFT LOW                  ¦
¦  +---------------------------+
+--+
```

MANUAL MODE : (Boolean expressions)

```
CONVEYOR    = PUSH BUT1*/PHOTOELL
LIFT UP     = PUSH BUT2*/LIFT UP  E
LIFT DOWN   = PUSH BUT3*/LIFT LOW
PUSHER      = PUSH BUT4
```

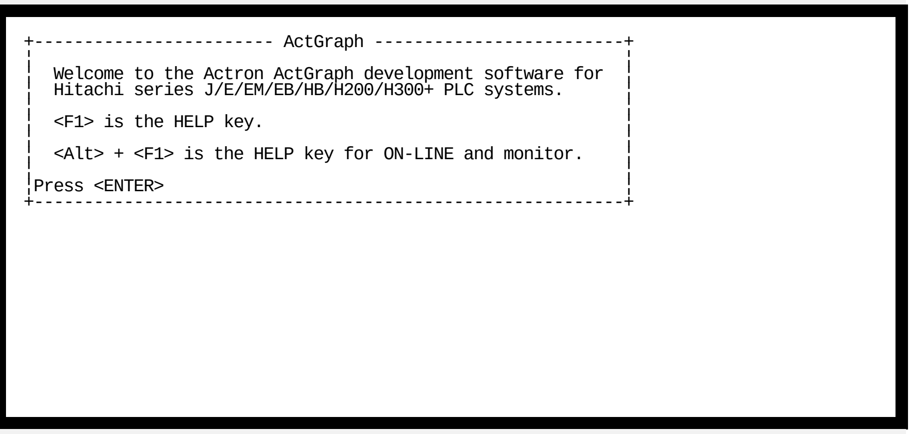
Let us make a program to achieve this.

Start of programming:

Start from DOS with the command < G > <Enter>.

Then a welcome window opens.
Note that F1 gives help and that <Alt>+ F1 gives help for ON-line.

ENTER

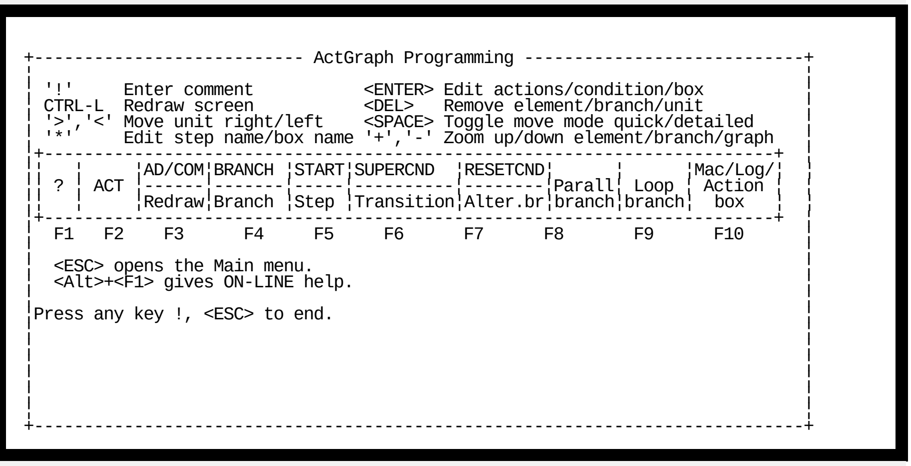


```
+------------------------ ActGraph ------------------------+
¦                                                          ¦
¦  Welcome to the Actron ActGraph development software for ¦
¦  Hitachi series J/E/EM/EB/HB/H200/H300+ PLC systems.     ¦
¦                                                          ¦
¦  <F1> is the HELP key.                                   ¦
¦                                                          ¦
¦  <Alt> + <F1> is the HELP key for ON-LINE and monitor.   ¦
¦                                                          ¦
¦Press <ENTER>                                             ¦
+----------------------------------------------------------+
```

(Set up of PC and PLC is done in the same way as in Actsip-Mini programming:)

Press F1 for Help. Note that a start step is <Shift>+ F5.

F1



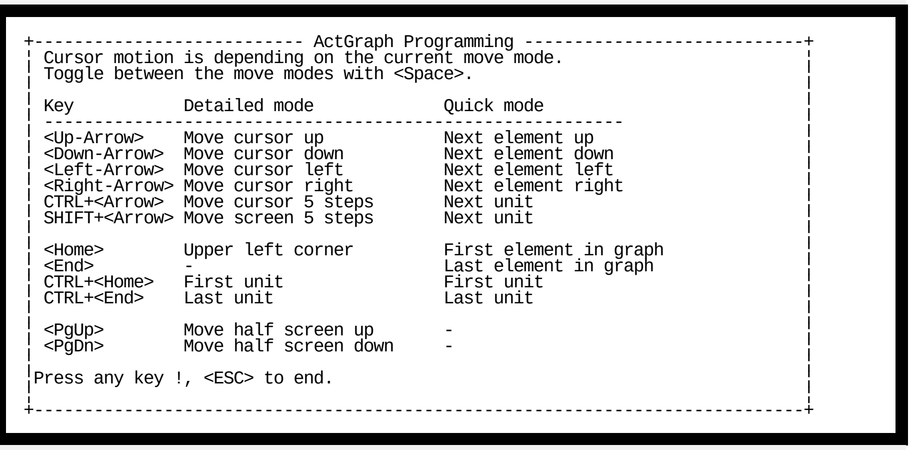
```
+-------------------------- ActGraph Programming ---------------------------+
¦                                                                           ¦
¦ '!'     Enter comment           <ENTER> Edit actions/condition/box        ¦
¦ CTRL-L  Redraw screen           <DEL>   Remove element/branch/unit        ¦
¦ '>','<' Move unit right/left    <SPACE> Toggle move mode quick/detailed   ¦
¦ '*'     Edit step name/box name '+','-' Zoom up/down element/branch/graph ¦
¦+-------------------------------------------------------------------------+¦
¦¦   ¦     ¦AD/COM¦BRANCH ¦START¦SUPERCND  ¦RESETCND¦      ¦      ¦Mac/Log/¦¦
¦¦ ? ¦ ACT ¦------¦-------¦-----¦----------¦--------¦Parall¦ Loop ¦ Action ¦¦
¦¦   ¦     ¦Redraw¦Branch ¦Step ¦Transition¦Alter.br¦branch¦branch¦  box   ¦¦
¦+-------------------------------------------------------------------------+¦
¦  F1   F2    F3      F4     F5     F6      F7      F8       F9      F10    ¦
¦                                                                           ¦
¦  <ESC> opens the Main menu.                                               ¦
¦  <Alt>+<F1> gives ON-LINE help.                                           ¦
¦                                                                           ¦
¦Press any key !, <ESC> to end.                                             ¦
¦                                                                           ¦
¦                                                                           ¦
¦                                                                           ¦
¦                                                                           ¦
¦                                                                           ¦
+---------------------------------------------------------------------------+
```

Press <Enter> to get more information.

ENTER

Press <Esc> to return to the drawing screen.

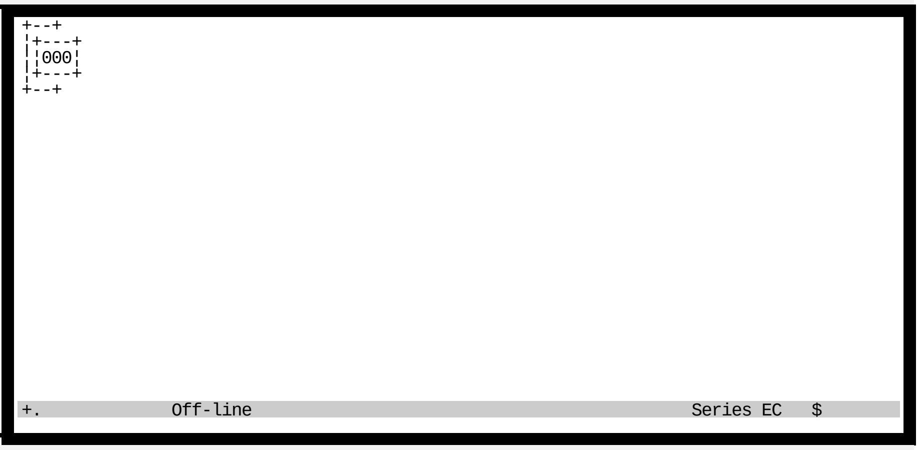
ESC



```
+-------------------------- ActGraph Programming ---------------------------+
¦ Cursor motion is depending on the current move mode.                      ¦
¦ Toggle between the move modes with <Space>.                               ¦
¦                                                                           ¦
¦ Key           Detailed mode             Quick mode                        ¦
¦ ------------------------------------------------------------              ¦
¦ <Up-Arrow>    Move cursor up            Next element up                   ¦
¦ <Down-Arrow>  Move cursor down          Next element down                 ¦
¦ <Left-Arrow>  Move cursor left          Next element left                 ¦
¦ <Right-Arrow> Move cursor right         Next element right                ¦
¦ CTRL+<Arrow>  Move cursor 5 steps       Next unit                         ¦
¦ SHIFT+<Arrow> Move screen 5 steps       Next unit                         ¦
¦                                                                           ¦
¦ <Home>        Upper left corner         First element in graph            ¦
¦ <End>         -                         Last element in graph             ¦
¦ CTRL+<Home>   First unit                First unit                        ¦
¦ CTRL+<End>    Last unit                 Last unit                         ¦
¦                                                                           ¦
¦ <PgUp>        Move half screen up       -                                 ¦
¦ <PgDn>        Move half screen down     -                                 ¦
¦                                                                           ¦
¦Press any key !, <ESC> to end.                                             ¦
¦                                                                           ¦
+---------------------------------------------------------------------------+
```

Press <Shift>+F5 and make a start step.

Shift
F5



```
+--+
¦+---+
¦¦000¦
¦+---+
+--+


+.             Off-line                                              Series EC   $
```

In the start step the output GREEN LAMP shall be active.

Press <Enter> and a window will open. Write GREEN LAMP and press <Enter> . (The cursor positions to the left and allows you the insert a ”detailed action” (see below) Press <Enter> until the window is closed.

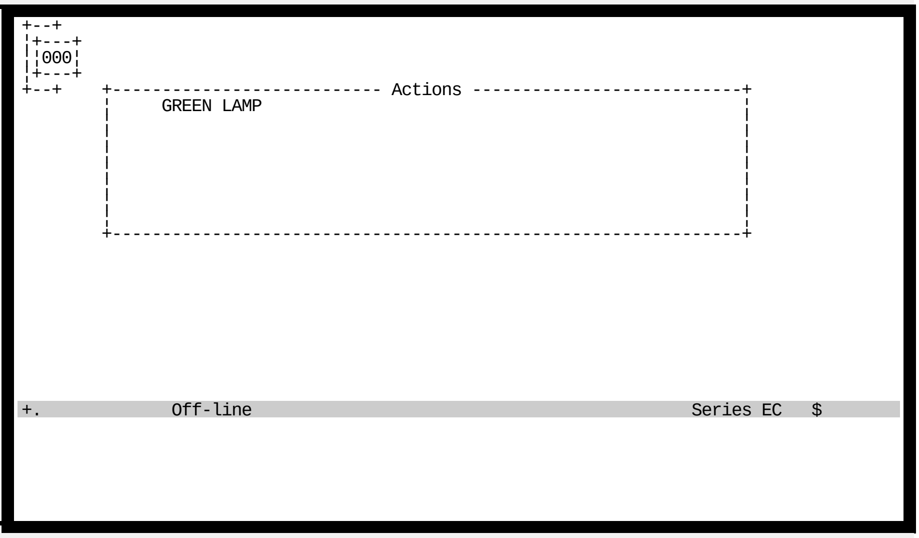
ENTER

GREE
N
LAMP

ENTER

ENTER

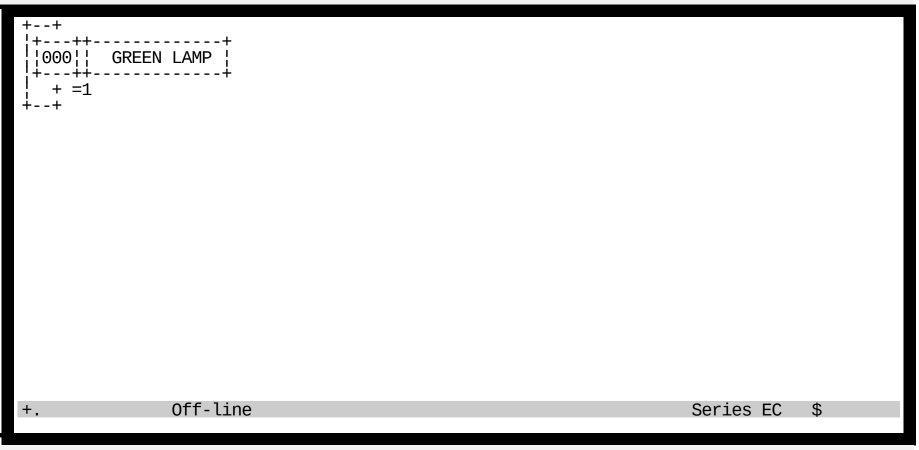
ENTER



```
+--+
¦+---+
¦¦000¦
¦+---+
+--+    +--------------------------- Actions --------------------------+
        ¦     GREEN LAMP                                               ¦
        ¦                                                              ¦
        ¦                                                              ¦
        ¦                                                              ¦
        ¦                                                              ¦
        ¦                                                              ¦
        +--------------------------------------------------------------+


+.             Off-line                                              Series EC   $
```

To create a transition, press F6.

F6



```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦   + =1
+--+


+.             Off-line                                              Series EC   $
```

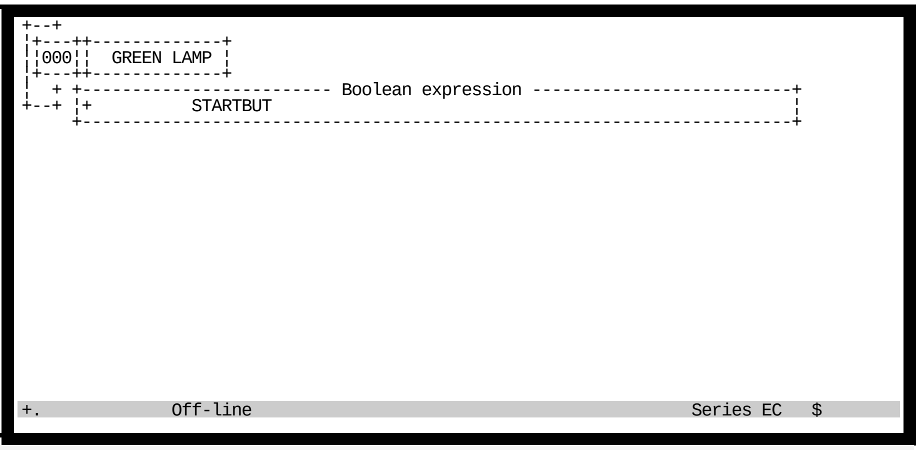
This transition shall start with the condition START BUTT . Press <Enter> and a new window opens.

Write START BUTT and press <Enter>.

ENTER

STAR
T
BUTT

ENTER



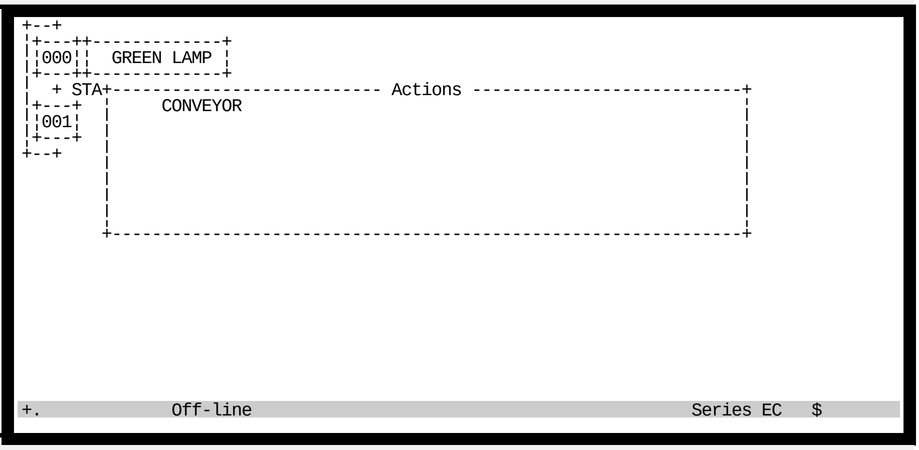
```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + +----------------------- Boolean expression --------------------------+
+--+ ¦+          STARTBUT                                                  ¦
     +---------------------------------------------------------------------+

+.              Off-line                                      Series EC   $
```

Continue with the next step through pressing F5.

Open the window with <Enter> and write CONVEYOR .

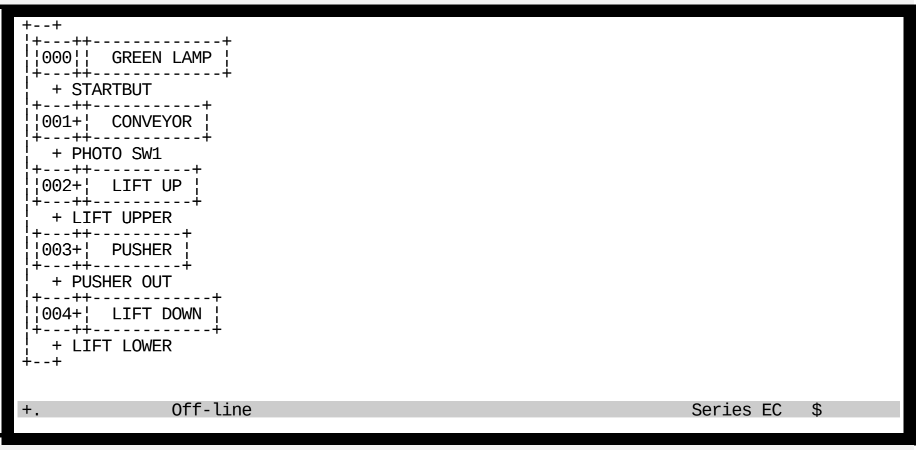
F5

ENTER



```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + STA+------------------------- Actions --------------------------+
¦+---+  ¦      CONVEYOR                                              ¦
¦¦001¦  ¦                                                            ¦
¦+---+  ¦                                                            ¦
+--+    ¦                                                            ¦
        ¦                                                            ¦
        ¦                                                            ¦
        ¦                                                            ¦
        +------------------------------------------------------------+

+.              Off-line                                      Series EC   $
```

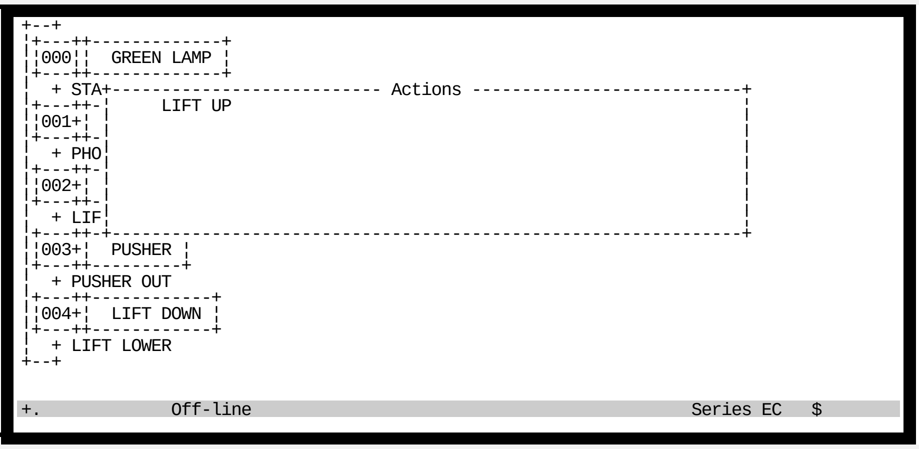
Complete the main sequence according to the above procedure.



```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + STARTBUT
¦+---++-----------+
¦¦001+¦  CONVEYOR ¦
¦+---++-----------+
¦  + PHOTO SW1
¦+---++----------+
¦¦002+¦  LIFT UP ¦
¦+---++----------+
¦  + LIFT UPPER
¦+---++---------+
¦¦003+¦  PUSHER ¦
¦+---++---------+
¦  + PUSHER OUT
¦+---++------------+
¦¦004+¦  LIFT DOWN ¦
¦+---++------------+
¦  + LIFT LOWER
+--+

+.              Off-line                                      Series EC   $
```

Detailed action. (Time delay)

We are now going to time delay LIFT UP . Directly after the type in of an action the cursor positions to the left to allow insertion of a ”detailed action”, e.g. time delay. Go to STEP 2, press <Enter> and open the action box. Rewrite the action.



```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + STA+--------------------------- Actions ---------------------------+
¦+---++-¦     LIFT UP                                                   ¦
¦¦001+¦ ¦                                                               ¦
¦+---++-¦                                                               ¦
¦  + PHO¦                                                               ¦
¦+---++-¦                                                               ¦
¦¦002+¦ ¦                                                               ¦
¦+---++-¦                                                               ¦
¦  + LIF¦                                                               ¦
¦+---++-+---------------------------------------------------------------+
¦¦003+¦  PUSHER ¦
¦+---++---------+
¦  + PUSHER OUT
¦+---++------------+
¦¦004+¦  LIFT DOWN ¦
¦+---++------------+
¦  + LIFT LOWER
+--+

+.              Off-line                                      Series EC   $
```

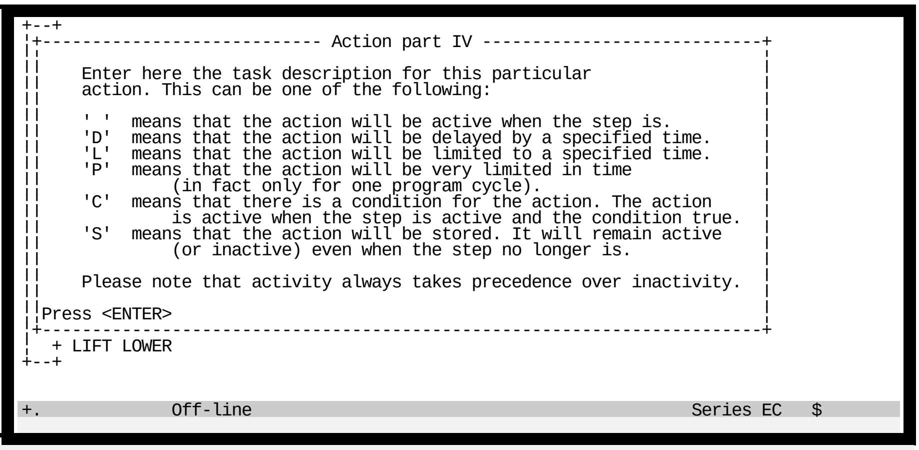
The cursor positions now to the left
Press F1 to see the different alternatives.

In this case we are going to use ”D”, which is a time delay.

Press <Enter>.

F1

ENTER



```
+--+
¦+--------------------------- Action part IV ---------------------------+
¦¦                                                                      ¦
¦¦    Enter here the task description for this particular              ¦
¦¦    action. This can be one of the following:                         ¦
¦¦                                                                      ¦
¦¦    ' '  means that the action will be active when the step is.       ¦
¦¦    'D'  means that the action will be delayed by a specified time.   ¦
¦¦    'L'  means that the action will be limited to a specified time.   ¦
¦¦    'P'  means that the action will be very limited in time           ¦
¦¦            (in fact only for one program cycle).                     ¦
¦¦    'C'  means that there is a condition for the action. The action   ¦
¦¦            is active when the step is active and the condition true. ¦
¦¦    'S'  means that the action will be stored. It will remain active  ¦
¦¦            (or inactive) even when the step no longer is.            ¦
¦¦                                                                      ¦
¦¦    Please note that activity always takes precedence over inactivity.¦
¦¦                                                                      ¦
¦¦Press <ENTER>                                                         ¦
¦+----------------------------------------------------------------------+
¦  + LIFT LOWER
+--+

+.             Off-line                                     Series EC   $
```

Write "D".
The cursor positions now far to the right and waits for the delay time (in seconds).

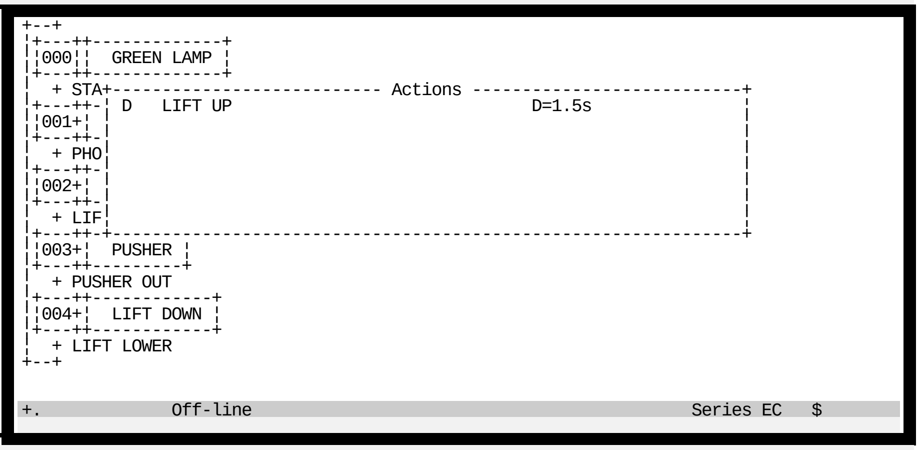
Press thereafter <Enter>

D

ENTER

1.5

ENTER



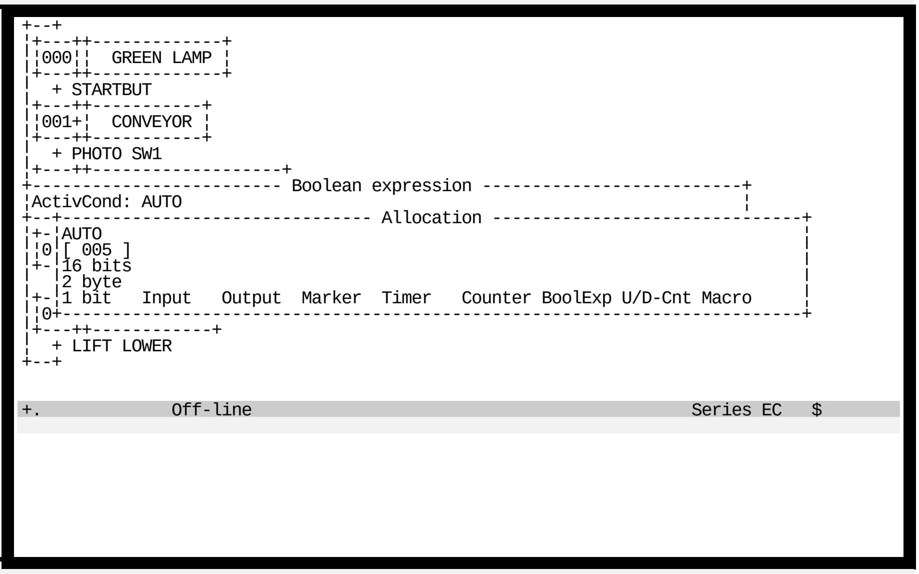
```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + STA+-------------------------- Actions --------------------------+
¦+---++-¦ D   LIFT UP                          D=1.5s                 ¦
¦¦001+¦ ¦                                                             ¦
¦+---++-¦                                                             ¦
¦  + PHO¦                                                             ¦
¦+---++-¦                                                             ¦
¦¦002+¦ ¦                                                             ¦
¦+---++-¦                                                             ¦
¦  + LIF¦                                                             ¦
¦+---++-+-------------------------------------------------------------+
¦¦003+¦  PUSHER ¦
¦+---++---------+
¦  + PUSHER OUT
¦+---++------------+
¦¦004+¦  LIFT DOWN ¦
¦+---++------------+
¦  + LIFT LOWER
+--+

+.             Off-line                                     Series EC   $
```

Super conditions

As the graph only will be active in automatic mode we can use the ”Activation condition”. You find this when you press <Shift>+F6. Write AUTO, which is the panel switch.
As we have not allocated a name to this address earlier the ”automatic allocation” suggests an non used address. Let us accept this address by pressing <Enter>

Shift⇑

F6

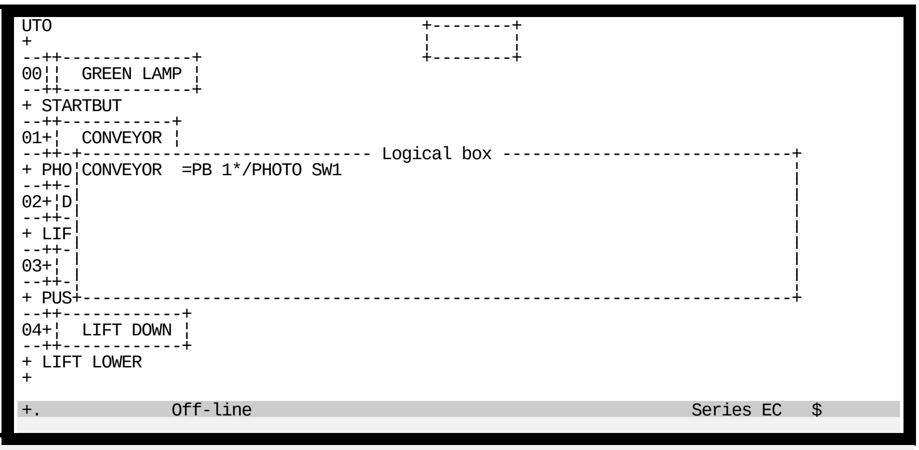


```
+--+
¦+---++-------------+
¦¦000¦¦  GREEN LAMP ¦
¦+---++-------------+
¦  + STARTBUT
¦+---++-----------+
¦¦001+¦  CONVEYOR ¦
¦+---++-----------+
¦  + PHOTO SW1
¦+---++--------------------+
+------------------------- Boolean expression -------------------------+
¦ActivCond: AUTO                                                       ¦
+--+------------------------------- Allocation -------------------------------+
¦+-¦AUTO                                                                      ¦
¦¦0¦[ 005 ]                                                                   ¦
¦+-¦16 bits                                                                   ¦
¦  ¦2 byte                                                                    ¦
¦+-¦1 bit   Input   Output  Marker  Timer   Counter BoolExp U/D-Cnt Macro     ¦
¦¦0+--------------------------------------------------------------------------+
¦+---++------------+
¦  + LIFT LOWER
+--+

+.             Off-line                                     Series EC   $
```

Create a logic box:

We are now also going to describe the ”manual conditions”. These are typically logic. We therefore open a logic box with F10.
Write the first logic expression for the different outputs in manual mode.

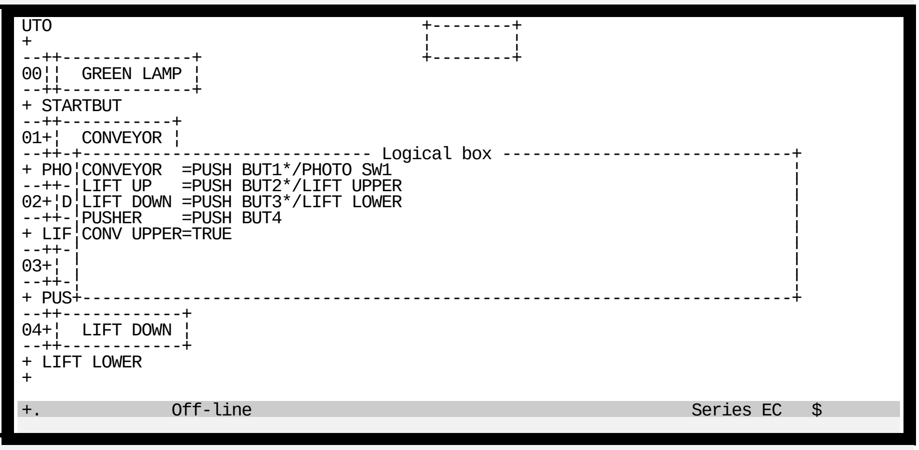
F10



```
UTO                                      +--------+
+                                        ¦        ¦
--++------------+                        +--------+
00¦¦  GREEN LAMP ¦
--++------------+
+ STARTBUT
--++----------+
01+¦  CONVEYOR ¦
--++-+------------------------------ Logical box -----------------------------+
+ PHO¦CONVEYOR  =PB 1*/PHOTO SW1                                              ¦
--++-¦                                                                        ¦
02+¦D¦                                                                        ¦
--++-¦                                                                        ¦
+ LIF¦                                                                        ¦
--++-¦                                                                        ¦
03+¦ ¦                                                                        ¦
--++-¦                                                                        ¦
+ PUS+------------------------------------------------------------------------+
--++-----------+
04+¦  LIFT DOWN ¦
--++-----------+
+ LIFT LOWER
+
+.             Off-line                                          Series EC   $
```

Complete with all outputs

Press <Enter> and a logic box appears. The condition for this to be active is that it is manual mode.
(or Not AUTO, which is written /AUTO)

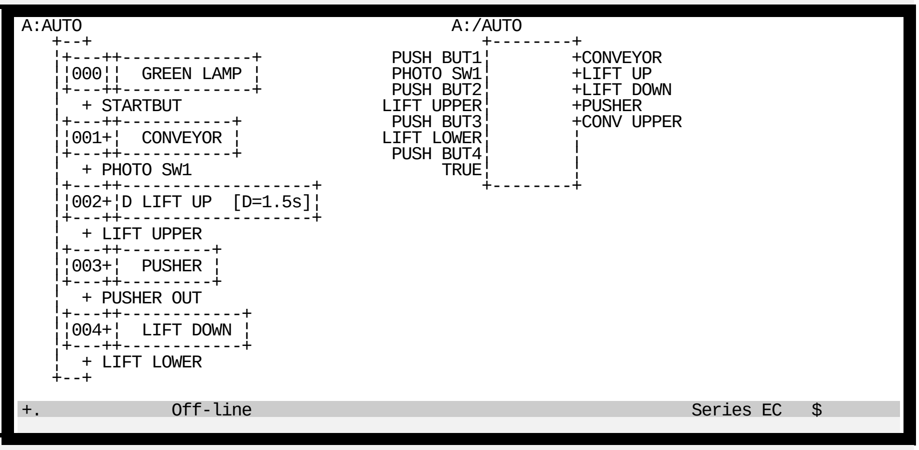


```
UTO                                      +--------+
+                                        ¦        ¦
--++------------+                        +--------+
00¦¦  GREEN LAMP ¦
--++------------+
+ STARTBUT
--++----------+
01+¦  CONVEYOR ¦
--++-+------------------------------ Logical box -----------------------------+
+ PHO¦CONVEYOR  =PUSH BUT1*/PHOTO SW1                                         ¦
--++-¦LIFT UP   =PUSH BUT2*/LIFT UPPER                                        ¦
02+¦D¦LIFT DOWN =PUSH BUT3*/LIFT LOWER                                        ¦
--++-¦PUSHER    =PUSH BUT4                                                    ¦
+ LIF¦CONV UPPER=TRUE                                                         ¦
--++-¦                                                                        ¦
03+¦ ¦                                                                        ¦
--++-¦                                                                        ¦
+ PUS+------------------------------------------------------------------------+
--++-----------+
04+¦  LIFT DOWN ¦
--++-----------+
+ LIFT LOWER
+
+.             Off-line                                          Series EC   $
```

Press <Shift>+F6 again to create this activation condition.
When this is inserted it will look like this.

Shift⇑

F6



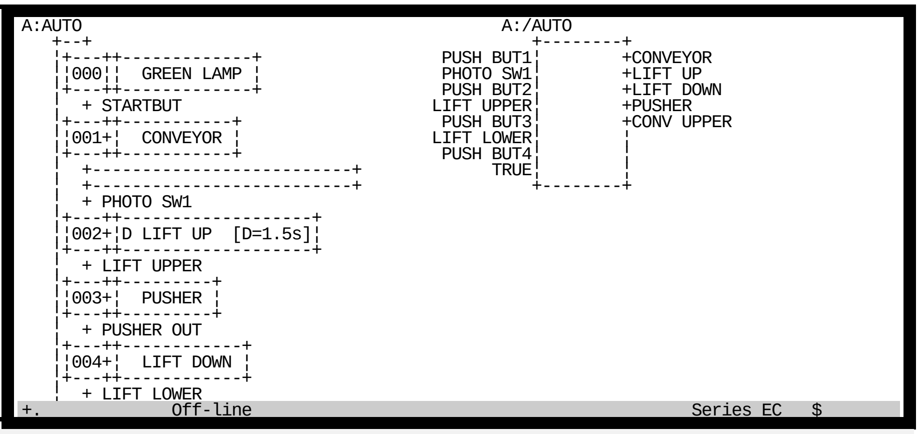
```
A:AUTO                                       A:/AUTO
   +--+                                         +--------+
   ¦+---++------------+              PUSH BUT1¦        +CONVEYOR
   ¦¦000¦¦  GREEN LAMP ¦             PHOTO SW1¦        +LIFT UP
   ¦+---++------------+              PUSH BUT2¦        +LIFT DOWN
   ¦  + STARTBUT                    LIFT UPPER¦        +PUSHER
   ¦+---++----------+                PUSH BUT3¦        +CONV UPPER
   ¦¦001+¦  CONVEYOR ¦              LIFT LOWER¦        ¦
   ¦+---++----------+                PUSH BUT4¦        ¦
   ¦  + PHOTO SW1                         TRUE¦        ¦
   ¦+---++-----------------+                  +--------+
   ¦¦002+¦D LIFT UP  [D=1.5s]¦
   ¦+---++-----------------+
   ¦  + LIFT UPPER
   ¦+---++---------+
   ¦¦003+¦  PUSHER ¦
   ¦+---++---------+
   ¦  + PUSHER OUT
   ¦+---++-----------+
   ¦¦004+¦  LIFT DOWN ¦
   ¦+---++-----------+
   ¦  + LIFT LOWER
   +--+
+.             Off-line                                          Series EC   $
```

Alternative branch:

We are now going to take care of the case when PHOTOCELL2 is indicating. (the item is too big)

Go to STEP 1, Press F7 and an alternative branch is created.

F7

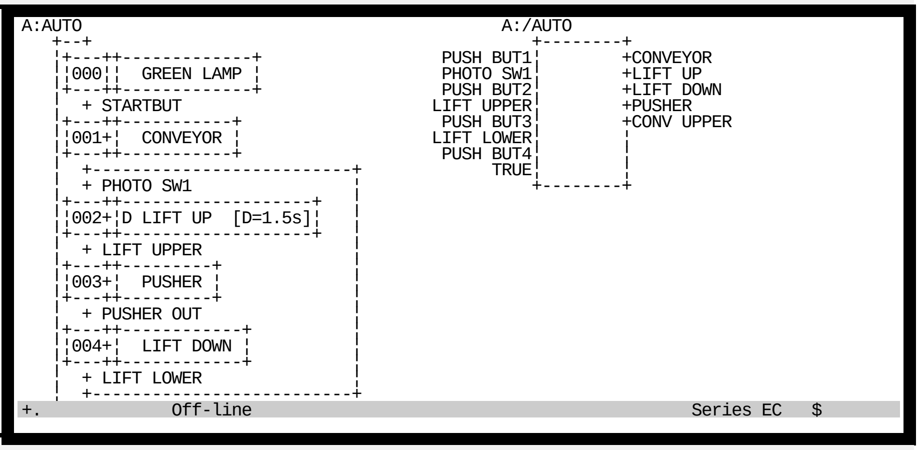


```
A:AUTO                                          A:/AUTO
   +--+                                            +--------+
   ¦+---++-------------+                 PUSH BUT1¦        +CONVEYOR
   ¦¦000¦¦  GREEN LAMP ¦                 PHOTO SW1¦        +LIFT UP
   ¦+---++-------------+                 PUSH BUT2¦        +LIFT DOWN
   ¦  + STARTBUT                        LIFT UPPER¦        +PUSHER
   ¦+---++-----------+                   PUSH BUT3¦        +CONV UPPER
   ¦¦001+¦  CONVEYOR ¦                  LIFT LOWER¦        ¦
   ¦+---++-----------+                   PUSH BUT4¦        ¦
   ¦  +-------------------------+             TRUE¦        ¦
   ¦  +-------------------------+                 +--------+
   ¦  + PHOTO SW1
   ¦+---++-------------------+
   ¦¦002+¦D LIFT UP  [D=1.5s]¦
   ¦+---++-------------------+
   ¦  + LIFT UPPER
   ¦+---++---------+
   ¦¦003+¦  PUSHER ¦
   ¦+---++---------+
   ¦  + PUSHER OUT
   ¦+---++------------+
   ¦¦004+¦  LIFT DOWN ¦
   ¦+---++------------+
   ¦  + LIFT LOWER
+.             Off-line                                            Series EC   $
```

We have to pull the lower part of the branch to the step where it is supposed to end.

This is done through using F4.
We pull the branch and pass STEP4 where the LIFT goes down.

F4
F4
F4
.
.

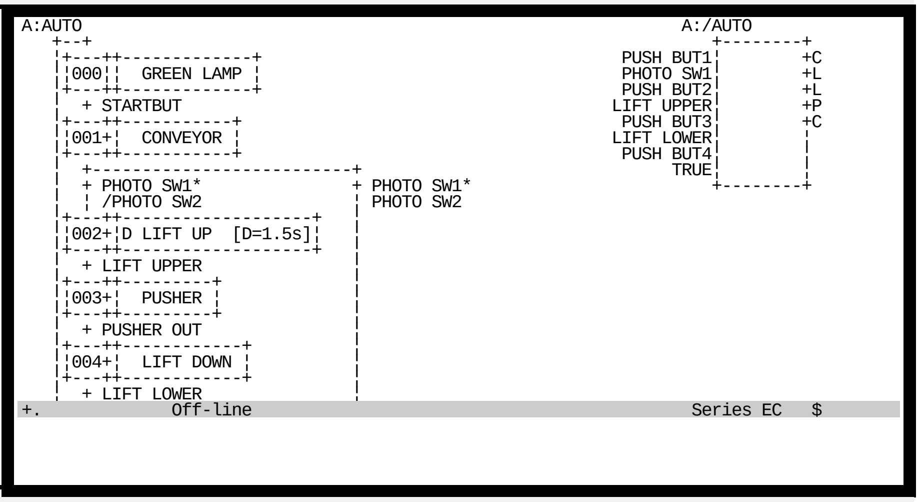


```
A:AUTO                                          A:/AUTO
   +--+                                            +--------+
   ¦+---++-------------+                 PUSH BUT1¦        +CONVEYOR
   ¦¦000¦¦  GREEN LAMP ¦                 PHOTO SW1¦        +LIFT UP
   ¦+---++-------------+                 PUSH BUT2¦        +LIFT DOWN
   ¦  + STARTBUT                        LIFT UPPER¦        +PUSHER
   ¦+---++-----------+                   PUSH BUT3¦        +CONV UPPER
   ¦¦001+¦  CONVEYOR ¦                  LIFT LOWER¦        ¦
   ¦+---++-----------+                   PUSH BUT4¦        ¦
   ¦  +-------------------------+             TRUE¦        ¦
   ¦  + PHOTO SW1               ¦                 +--------+
   ¦+---++-------------------+  ¦
   ¦¦002+¦D LIFT UP  [D=1.5s]¦  ¦
   ¦+---++-------------------+  ¦
   ¦  + LIFT UPPER              ¦
   ¦+---++---------+            ¦
   ¦¦003+¦  PUSHER ¦            ¦
   ¦+---++---------+            ¦
   ¦  + PUSHER OUT              ¦
   ¦+---++------------+         ¦
   ¦¦004+¦  LIFT DOWN ¦         ¦
   ¦+---++------------+         ¦
   ¦  + LIFT LOWER              ¦
   ¦  +-------------------------+
+.             Off-line                                            Series EC   $
```

Use the arrow keys to go to the transition between STEP1 and STEP2. Complete the transition condition with PHOTOCELL2 not indicating.
Create a new transition of the branch with F6 and write the condition for this, which is that both photocells are indicating.

↑

→

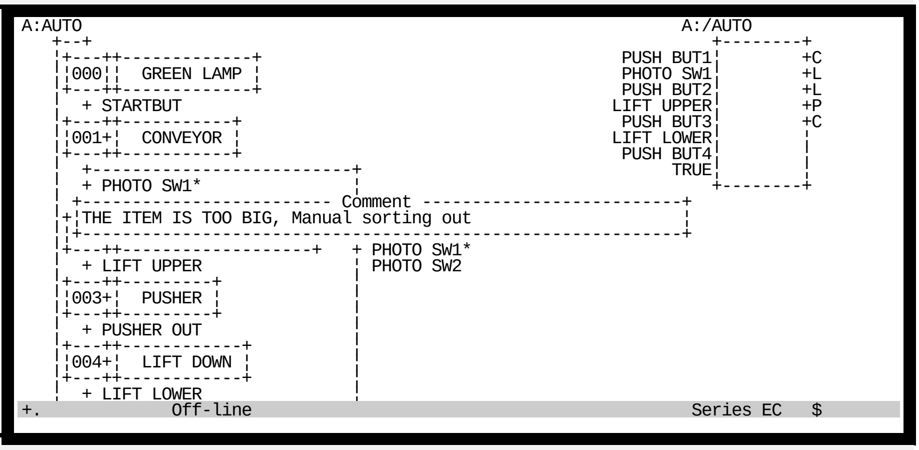


```
A:AUTO                                                            A:/AUTO
   +--+                                                              +--------+
   ¦+---++-------------+                                   PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                   PHOTO SW1¦        +L
   ¦+---++-------------+                                   PUSH BUT2¦        +L
   ¦  + STARTBUT                                          LIFT UPPER¦        +P
   ¦+---++-----------+                                     PUSH BUT3¦        +C
   ¦¦001+¦  CONVEYOR ¦                                    LIFT LOWER¦        ¦
   ¦+---++-----------+                                     PUSH BUT4¦        ¦
   ¦  +-------------------------+                               TRUE¦        ¦
   ¦  + PHOTO SW1*              + PHOTO SW1*                        +--------+
   ¦  ¦ /PHOTO SW2              ¦ PHOTO SW2
   ¦+---++-------------------+  ¦
   ¦¦002+¦D LIFT UP  [D=1.5s]¦  ¦
   ¦+---++-------------------+  ¦
   ¦  + LIFT UPPER              ¦
   ¦+---++---------+            ¦
   ¦¦003+¦  PUSHER ¦            ¦
   ¦+---++---------+            ¦
   ¦  + PUSHER OUT              ¦
   ¦+---++------------+         ¦
   ¦¦004+¦  LIFT DOWN ¦         ¦
   ¦+---++------------+         ¦
   ¦  + LIFT LOWER              ¦
+.             Off-line                                            Series EC   $
```

It is recommended to insert lots of comments in the program. Go to the place for the comment, press ”!”.
A window opens where the comment can be written.
Press <Enter> and the comment is written into the graph.

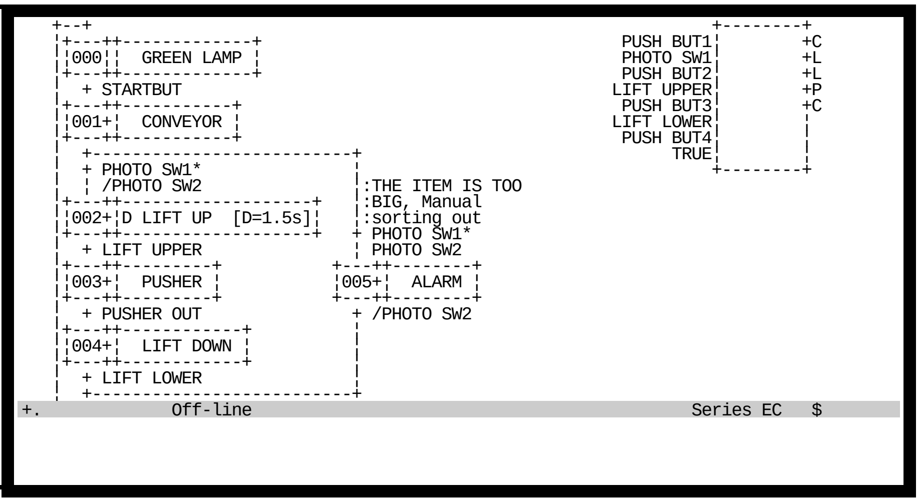
" ! "

↵ ENTER



```
A:AUTO                                                            A:/AUTO
   +--+                                                              +--------+
   ¦+---++-------------+                                   PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                   PHOTO SW1¦        +L
   ¦+---++-------------+                                   PUSH BUT2¦        +L
   ¦  + STARTBUT                                          LIFT UPPER¦        +P
   ¦+---++-----------+                                     PUSH BUT3¦        +C
   ¦¦001+¦  CONVEYOR ¦                                    LIFT LOWER¦        ¦
   ¦+---++-----------+                                     PUSH BUT4¦        ¦
   ¦  +-------------------------+                               TRUE¦        ¦
   ¦  + PHOTO SW1*              ¦                                   +--------+
   ¦ +------------------------- Comment --------------------------+
   ¦+¦THE ITEM IS TOO BIG, Manual sorting out                     ¦
   ¦¦+------------------------------------------------------------+
   ¦+---++-------------------+  + PHOTO SW1*
   ¦  + LIFT UPPER              ¦ PHOTO SW2
   ¦+---++---------+            ¦
   ¦¦003+¦  PUSHER ¦            ¦
   ¦+---++---------+            ¦
   ¦  + PUSHER OUT              ¦
   ¦+---++------------+         ¦
   ¦¦004+¦  LIFT DOWN ¦         ¦
   ¦+---++------------+         ¦
   ¦  + LIFT LOWER              ¦
+.             Off-line                                            Series EC   $
```

Complete with a new step and a transition. This means that when both photocells are indicating in STEP1 the flow goes to STEP5 where an alarm signal is activated. This does not stop before the item is removed. Then the flow continues to the STARTSTEP and starts from the beginning.



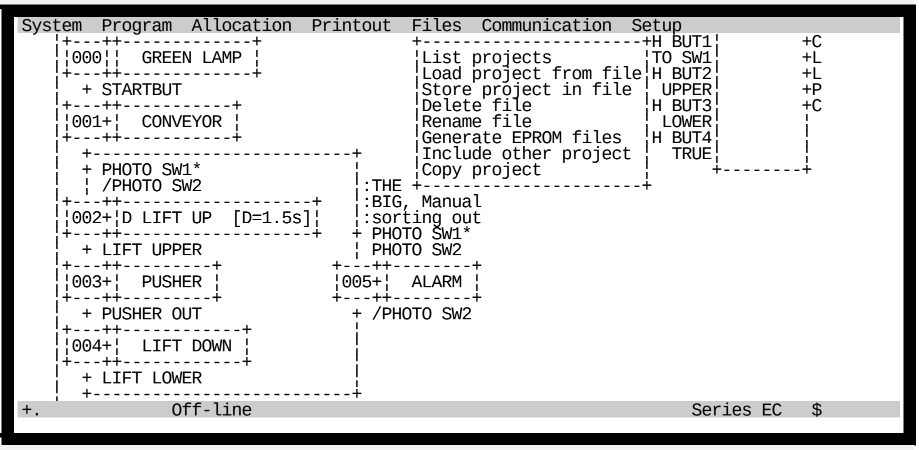
```
   +--+                                                              +--------+
   ¦+---++-------------+                                    PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +L
   ¦+---++-------------+                                    PUSH BUT2¦        +L
   ¦  + STARTBUT                                           LIFT UPPER¦        +P
   ¦+---++-----------+                                      PUSH BUT3¦        +C
   ¦¦001+¦  CONVEYOR ¦                                     LIFT LOWER¦        ¦
   ¦+---++-----------+                                      PUSH BUT4¦        ¦
   ¦  +--------------------------+                               TRUE¦        ¦
   ¦  + PHOTO SW1*               ¦                                   +--------+
   ¦  ¦ /PHOTO SW2               ¦:THE ITEM IS TOO
   ¦+---++-------------------+   ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦   ¦:sorting out
   ¦+---++-------------------+   + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                           Series EC   $
```

Save the project:

Now the project is ready for test.
We ought to save first.

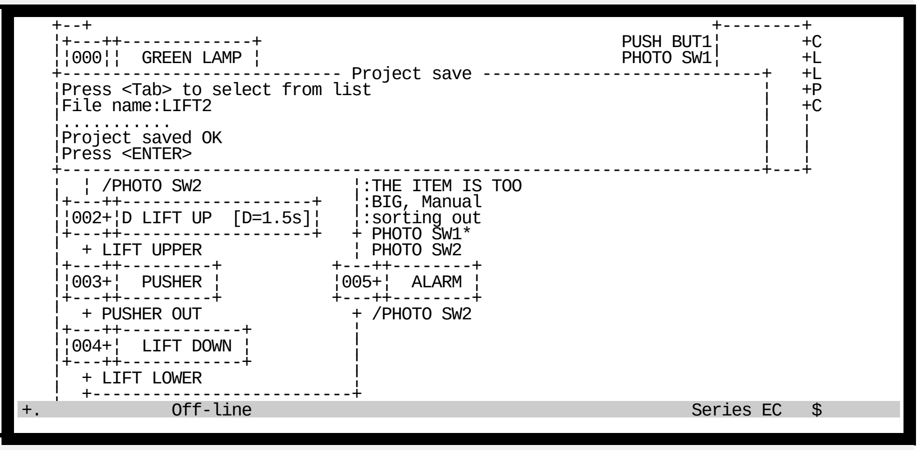
Press <Esc> and go to the menu "FILES".
Go down to "Save the project in file"
Press <Enter>

ESC



```
System  Program  Allocation  Printout  Files  Communication  Setup
   ¦+---++-------------+               +----------------------+H BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦               ¦List projects         ¦TO SW1¦        +L
   ¦+---++-------------+               ¦Load project from file¦H BUT2¦        +L
   ¦  + STARTBUT                       ¦Store project in file ¦ UPPER¦        +P
   ¦+---++-----------+                 ¦Delete file           ¦H BUT3¦        +C
   ¦¦001+¦  CONVEYOR ¦                 ¦Rename file           ¦ LOWER¦        ¦
   ¦+---++-----------+                 ¦Generate EPROM files  ¦H BUT4¦        ¦
   ¦  +--------------------------+     ¦Include other project ¦  TRUE¦        ¦
   ¦  + PHOTO SW1*               ¦     ¦Copy project          ¦      +--------+
   ¦  ¦ /PHOTO SW2               ¦:THE +----------------------+
   ¦+---++-------------------+   ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦   ¦:sorting out
   ¦+---++-------------------+   + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                            Series EC   $
```

Write the name of the project and press <Enter>



```
   +--+                                                              +--------+
   ¦+---++-------------+                                    PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +L
   +---------------------------- Project save ----------------------------+   +L
   ¦Press <Tab> to select from list                                       ¦   +P
   ¦File name:LIFT2                                                       ¦   +C
   ¦...........                                                           ¦   ¦
   ¦Project saved OK                                                      ¦   ¦
   ¦Press <ENTER>                                                         ¦   ¦
   +----------------------------------------------------------------------+---+
   ¦  ¦ /PHOTO SW2               ¦:THE ITEM IS TOO
   ¦+---++-------------------+   ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦   ¦:sorting out
   ¦+---++-------------------+   + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                            Series EC   $
```

Transfer the project to the PLC / Go ON-Line

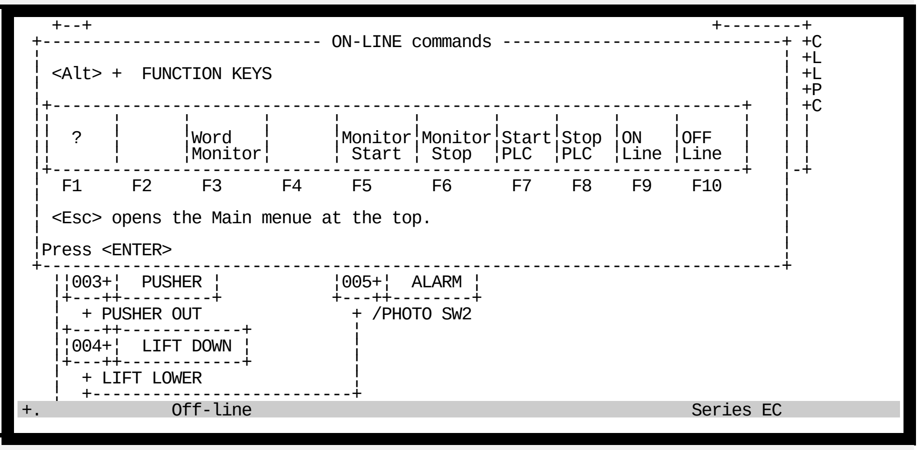
Press <Alt>+F1 for ON-Line help.

You can go step by step: ON-Line, Start PLC and Monitor PLC.

If you press Monitor PLC (F5) the complete chain will be performed.

Alt

F1

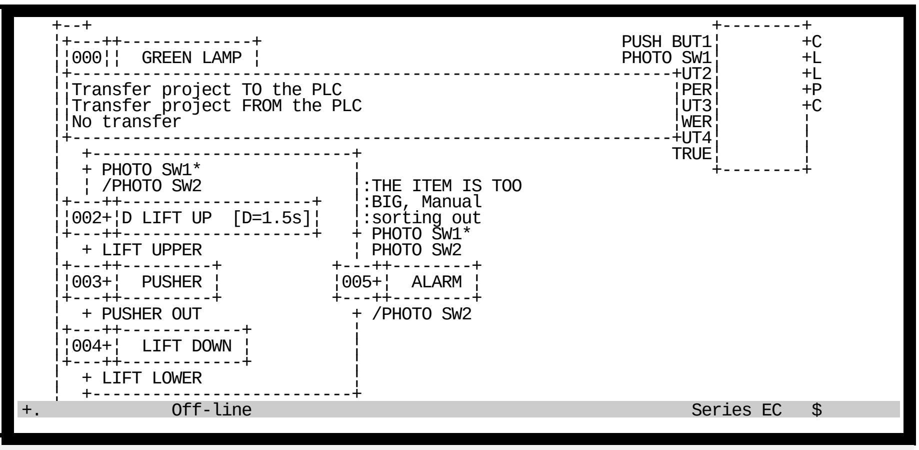


```
   +--+                                                              +--------+
 +---------------------------- ON-LINE commands ----------------------------+ +C
 ¦                                                                          ¦ +L
 ¦ <Alt> +  FUNCTION KEYS                                                   ¦ +L
 ¦                                                                          ¦ +P
 ¦+----------------------------------------------------------------------+  ¦ +C
 ¦¦      ¦      ¦       ¦      ¦       ¦       ¦     ¦     ¦      ¦      ¦  ¦ ¦
 ¦¦  ?   ¦      ¦Word   ¦      ¦Monitor¦Monitor¦Start¦Stop ¦ON    ¦OFF   ¦  ¦ ¦
 ¦¦      ¦      ¦Monitor¦      ¦ Start ¦ Stop  ¦PLC  ¦PLC  ¦Line  ¦Line  ¦  ¦ ¦
 ¦+----------------------------------------------------------------------+  ¦-+
 ¦  F1     F2     F3      F4     F5      F6      F7    F8    F9    F10      ¦
 ¦                                                                          ¦
 ¦ <Esc> opens the Main menue at the top.                                   ¦
 ¦                                                                          ¦
 ¦Press <ENTER>                                                             ¦
 +--------------------------------------------------------------------------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                            Series EC
```

Press <Alt>+F5, Press <Enter> for ”Yes” to transfer the project to the PLC.

Alt

F5



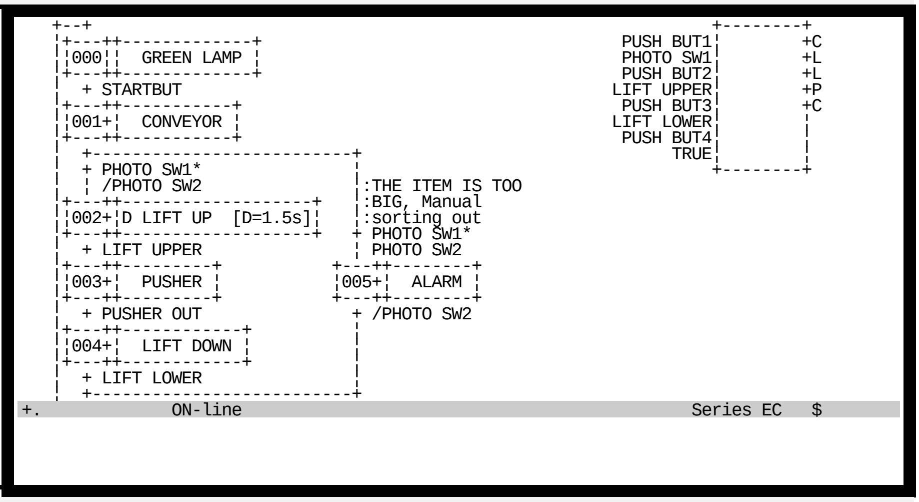
```
   +--+                                                              +--------+
   ¦+---++-------------+                                    PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +L
   ¦+------------------------------------------------------------+UT2¦        +L
   ¦¦Transfer project TO the PLC                                 ¦PER¦        +P
   ¦¦Transfer project FROM the PLC                               ¦UT3¦        +C
   ¦¦No transfer                                                 ¦WER¦        |
   ¦+------------------------------------------------------------+UT4¦        |
   ¦  +--------------------------+                               TRUE¦        |
   ¦  + PHOTO SW1*               ¦                                   +--------+
   ¦  ¦ /PHOTO SW2               ¦:THE ITEM IS TOO
   ¦+---++------------------+    ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦    ¦:sorting out
   ¦+---++------------------+    + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                            Series EC   $
```

In monitor you can follow what happens. You will see what steps and memories, which are active.

If you press <Alt>+ F5 once more you will get a window on the screen, where you can collect the bit and word memories etc. in the order you want.

Alt

F5

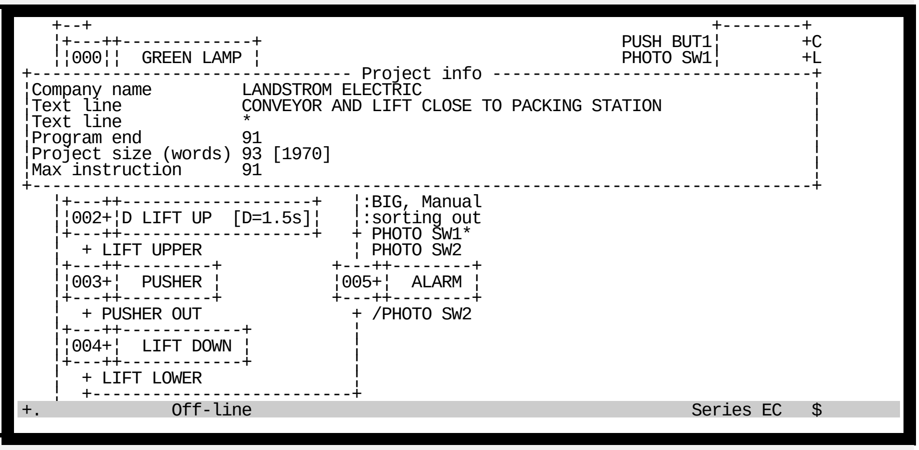


```
   +--+                                                              +--------+
   ¦+---++-------------+                                    PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +L
   ¦+---++-------------+                                    PUSH BUT2¦        +L
   ¦  + STARTBUT                                           LIFT UPPER¦        +P
   ¦+---++-----------+                                      PUSH BUT3¦        +C
   ¦¦001+¦  CONVEYOR ¦                                     LIFT LOWER¦        |
   ¦+---++-----------+                                      PUSH BUT4¦        |
   ¦  +--------------------------+                               TRUE¦        |
   ¦  + PHOTO SW1*               ¦                                   +--------+
   ¦  ¦ /PHOTO SW2               ¦:THE ITEM IS TOO
   ¦+---++------------------+    ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦    ¦:sorting out
   ¦+---++------------------+    + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             ON-line                                             Series EC   $
```

Document the program:

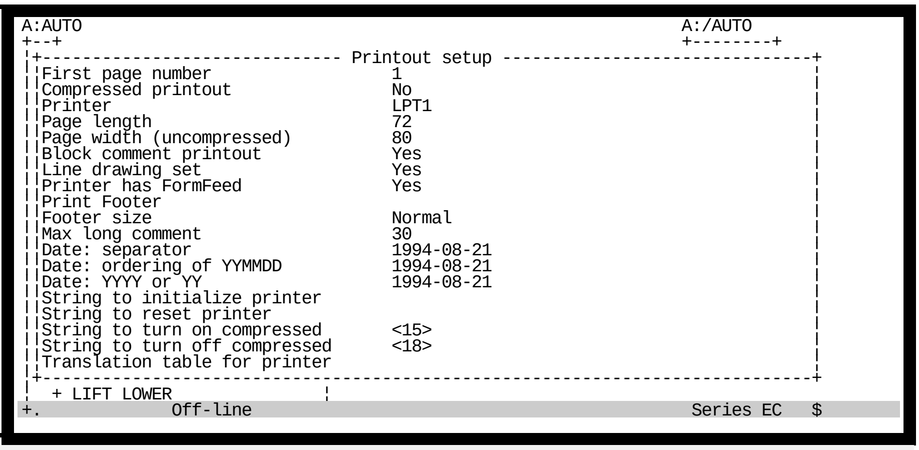
If the program works we should now create the documentation.

Go to the "Program-Project information" menu and write the project information.



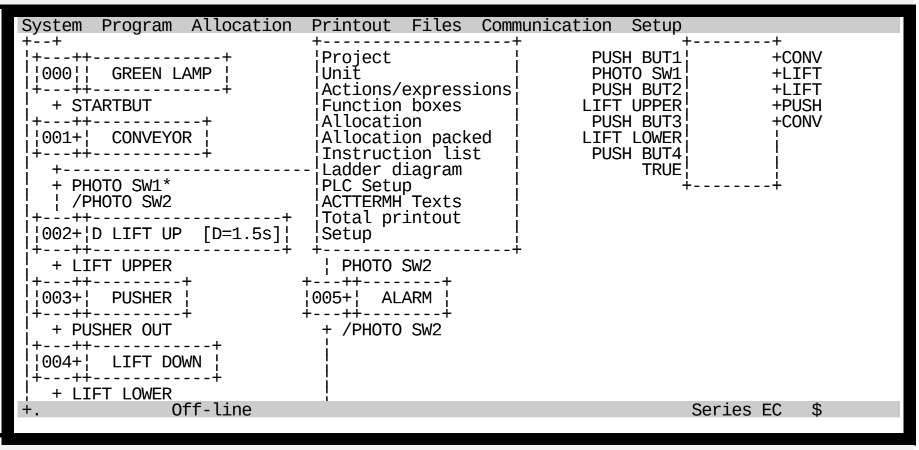
```
   +--+                                                              +--------+
   ¦+---++-------------+                                    PUSH BUT1¦        +C
   ¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +L
+-------------------------------- Project info --------------------------------+
¦Company name         LANDSTROM ELECTRIC                                       ¦
¦Text line            CONVEYOR AND LIFT CLOSE TO PACKING STATION               ¦
¦Text line            *                                                        ¦
¦Program end          91                                                       ¦
¦Project size (words) 93 [1970]                                                ¦
¦Max instruction      91                                                       ¦
+------------------------------------------------------------------------------+
   ¦+---++------------------+    ¦:BIG, Manual
   ¦¦002+¦D LIFT UP  [D=1.5s]¦    ¦:sorting out
   ¦+---++------------------+    + PHOTO SW1*
   ¦  + LIFT UPPER               ¦ PHOTO SW2
   ¦+---++---------+           +---++--------+
   ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
   ¦+---++---------+           +---++--------+
   ¦  + PUSHER OUT               + /PHOTO SW2
   ¦+---++------------+          ¦
   ¦¦004+¦  LIFT DOWN ¦          ¦
   ¦+---++------------+          ¦
   ¦  + LIFT LOWER               ¦
   ¦  +--------------------------+
+.             Off-line                                            Series EC   $
```

Go to "Setup-Printout" and set the printer parameters up.
For a normal parallel printer you do not need to make any setup except to choose between compressed and not compressed printout.



```
A:AUTO                                                            A:/AUTO
+--+                                                              +--------+
¦+------------------------------ Printout setup --------------------------------+
¦¦First page number                  1                                          ¦
¦¦Compressed printout                No                                         ¦
¦¦Printer                            LPT1                                       ¦
¦¦Page length                        72                                         ¦
¦¦Page width (uncompressed)          80                                         ¦
¦¦Block comment printout             Yes                                        ¦
¦¦Line drawing set                   Yes                                        ¦
¦¦Printer has FormFeed               Yes                                        ¦
¦¦Print Footer                                                                  ¦
¦¦Footer size                        Normal                                     ¦
¦¦Max long comment                   30                                         ¦
¦¦Date: separator                    1994-08-21                                 ¦
¦¦Date: ordering of YYMMDD           1994-08-21                                 ¦
¦¦Date: YYYY or YY                   1994-08-21                                 ¦
¦¦String to initialize printer                                                  ¦
¦¦String to reset printer                                                       ¦
¦¦String to turn on compressed       <15>                                       ¦
¦¦String to turn off compressed      <18>                                       ¦
¦¦Translation table for printer                                                 ¦
¦+------------------------------------------------------------------------------+
¦  + LIFT LOWER               ¦
+.             Off-line                                            Series EC   $
```

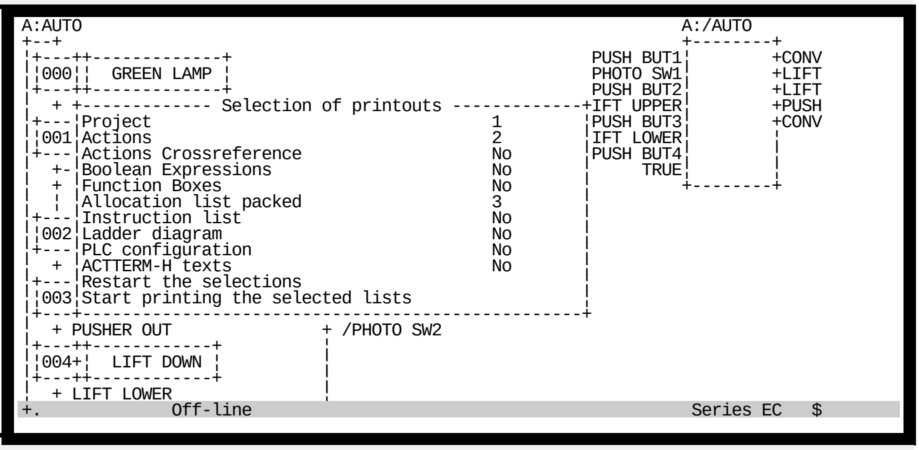
Go to Printout and choose "Total Printout"



```
System  Program  Allocation  Printout  Files  Communication  Setup
+--+                         +-------------------+                +--------+
¦+---++-------------+       ¦Project            ¦       PUSH BUT1¦        +CONV
¦¦000¦¦  GREEN LAMP ¦       ¦Unit               ¦       PHOTO SW1¦        +LIFT
¦+---++-------------+       ¦Actions/expressions¦       PUSH BUT2¦        +LIFT
¦  + STARTBUT                ¦Function boxes     ¦      LIFT UPPER¦        +PUSH
¦+---++-----------+          ¦Allocation         ¦       PUSH BUT3¦        +CONV
¦¦001+¦  CONVEYOR ¦          ¦Allocation packed  ¦      LIFT LOWER¦        ¦
¦+---++-----------+          ¦Instruction list   ¦       PUSH BUT4¦        ¦
¦  +-------------------------¦Ladder diagram     ¦            TRUE¦        ¦
¦  + PHOTO SW1*              ¦PLC Setup          ¦                +--------+
¦  ¦ /PHOTO SW2              ¦ACTTERMH Texts     ¦
¦+---++-------------------+  ¦Total printout     ¦
¦¦002+¦D LIFT UP  [D=1.5s]¦  ¦Setup              ¦
¦+---++-------------------+  +-------------------+
¦  + LIFT UPPER               ¦ PHOTO SW2
¦+---++---------+           +---++--------+
¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
¦+---++---------+           +---++--------+
¦  + PUSHER OUT               + /PHOTO SW2
¦+---++------------+          ¦
¦¦004+¦  LIFT DOWN ¦          ¦
¦+---++------------+          ¦
¦  + LIFT LOWER               ¦
+.             Off-line                                            Series EC   $
```

In the total printout you can choose between the printout alternatives, which are of interest and in what order.

At last, go to ”Print out the lists” and press <Enter>

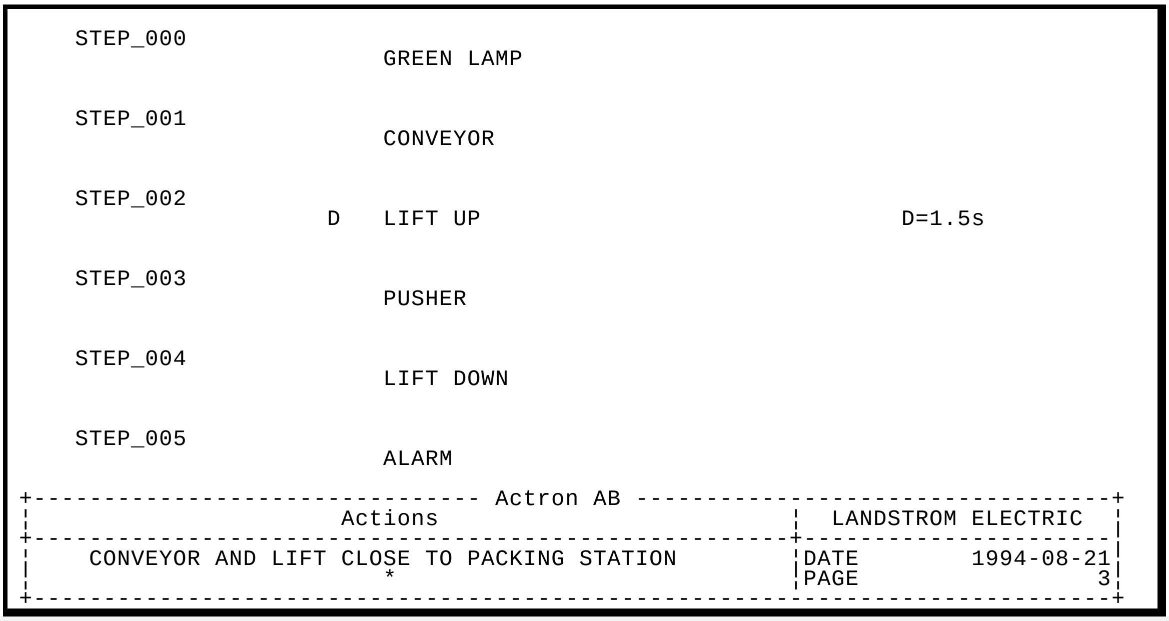


```
A:AUTO                                                            A:/AUTO
+--+                                                              +--------+
¦+---++-------------+                                    PUSH BUT1¦        +CONV
¦¦000¦¦  GREEN LAMP ¦                                    PHOTO SW1¦        +LIFT
¦+---++-------------+                                    PUSH BUT2¦        +LIFT
¦  + +------------- Selection of printouts -------------+IFT UPPER¦        +PUSH
¦+---¦Project                                  1        ¦PUSH BUT3¦        +CONV
¦¦001¦Actions                                  2        ¦IFT LOWER¦        ¦
¦+---¦Actions Crossreference                   No       ¦PUSH BUT4¦        ¦
¦  +-¦Boolean Expressions                      No       ¦     TRUE¦        ¦
¦  + ¦Function Boxes                           No       ¦         +--------+
¦  ¦ ¦Allocation list packed                   3        ¦
¦+---¦Instruction list                         No       ¦
¦¦002¦Ladder diagram                           No       ¦
¦+---¦PLC configuration                        No       ¦
¦  + ¦ACTTERM-H texts                          No       ¦
¦+---¦Restart the selections                            ¦
¦¦003¦Start printing the selected lists                 ¦
¦+---+--------------------------------------------------+
¦  + PUSHER OUT               + /PHOTO SW2
¦+---++------------+          ¦
¦¦004+¦  LIFT DOWN ¦          ¦
¦+---++------------+          ¦
¦  + LIFT LOWER               ¦
+.             Off-line                                            Series EC   $
```

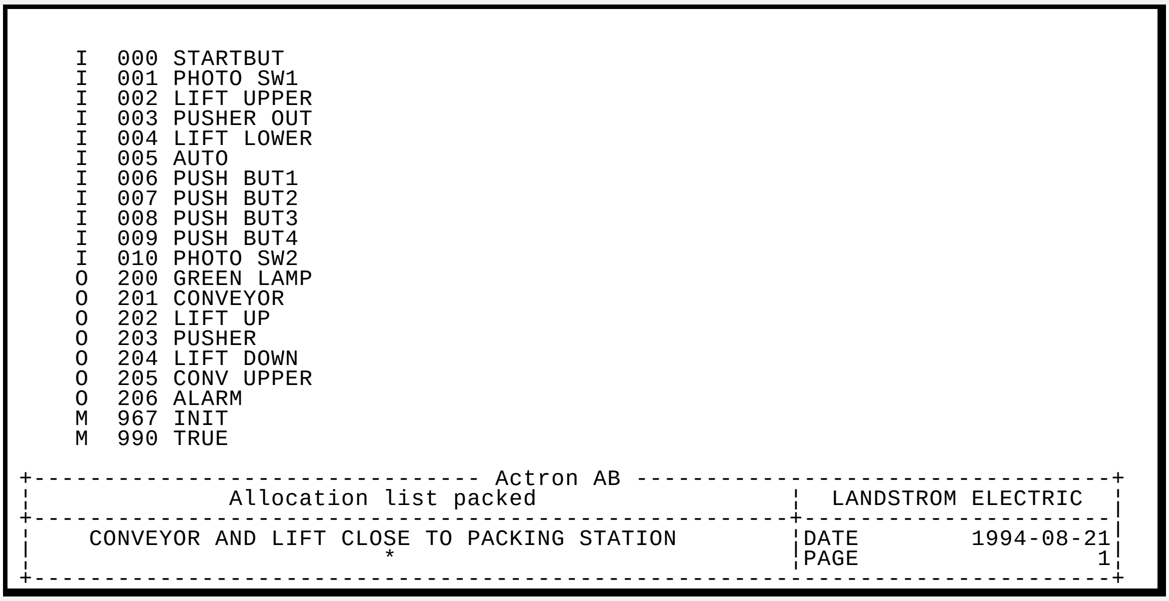
Printout

```
    A:AUTO
    +--+
    ¦+---++-------------+
    ¦¦000¦¦  GREEN LAMP ¦
    ¦+---++-------------+
    ¦  + STARTBUT
    ¦+---++-----------+
    ¦¦001+¦  CONVEYOR ¦
    ¦+---++-----------+
    ¦  +--------------------------+
    ¦  + PHOTO SW1*               ¦
    ¦  ¦ /PHOTO SW2               ¦:THE ITEM IS TOO
    ¦+---++-------------------+   ¦:BIG, Manual
    ¦¦002+¦D LIFT UP  [D=1.5s]¦   ¦:sorting out
    ¦+---++-------------------+   + PHOTO SW1*
    ¦  + LIFT UPPER               ¦ PHOTO SW2
    ¦+---++---------+           +---++--------+
    ¦¦003+¦  PUSHER ¦           ¦005+¦  ALARM ¦
    ¦+---++---------+           +---++--------+
    ¦  + PUSHER OUT               + /PHOTO SW2
    ¦+---++------------+          ¦
    ¦¦004+¦  LIFT DOWN ¦          ¦
    ¦+---++------------+          ¦
    ¦  + LIFT LOWER               ¦
    ¦  +--------------------------+
    +--+

+-------------------------------- Actron AB ----------------------------------+
¦                      Project                         ¦  LANDSTROM ELECTRIC  ¦
+------------------------------------------------------+----------------------¦
¦    CONVEYOR AND LIFT CLOSE TO PACKING STATION        ¦DATE        1994-08-21¦
¦                         *                            ¦PAGE                 1¦
+-----------------------------------------------------------------------------+
```

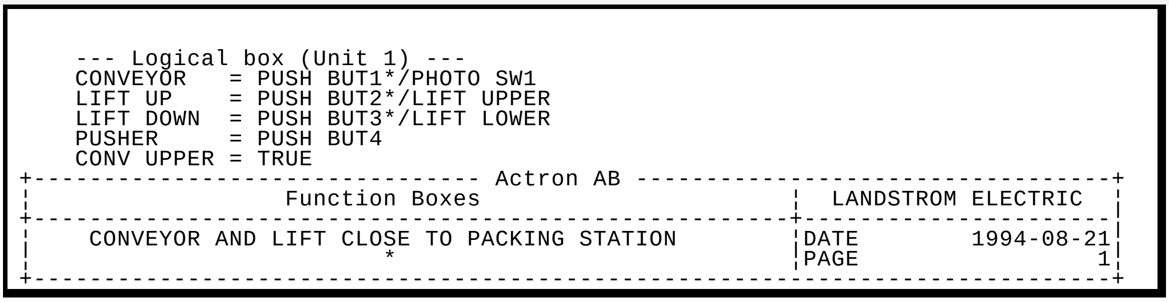
```
              A:/AUTO
              +--------+
     PUSH BUT1¦        +CONVEYOR
     PHOTO SW1¦        +LIFT UP
     PUSH BUT2¦        +LIFT DOWN
    LIFT UPPER¦        +PUSHER
     PUSH BUT3¦        +CONV UPPER
    LIFT LOWER¦        ¦
     PUSH BUT4¦        ¦
          TRUE¦        ¦
              +--------+
+-------------------------------- Actron AB ----------------------------------+
¦                      Project                         ¦  LANDSTROM ELECTRIC  ¦
+------------------------------------------------------+----------------------¦
¦    CONVEYOR AND LIFT CLOSE TO PACKING STATION        ¦DATE        1994-08-21¦
¦                         *                            ¦PAGE                 2¦
+-----------------------------------------------------------------------------+
```



```
   STEP_000
                         GREEN LAMP


   STEP_001
                         CONVEYOR


   STEP_002
                     D   LIFT UP                              D=1.5s


   STEP_003
                         PUSHER


   STEP_004
                         LIFT DOWN


   STEP_005
                         ALARM
+-------------------------------- Actron AB ----------------------------------+
¦                          Actions                     ¦  LANDSTROM ELECTRIC  ¦
+------------------------------------------------------+----------------------¦
¦    CONVEYOR AND LIFT CLOSE TO PACKING STATION        ¦DATE        1994-08-21¦
¦                          *                           ¦PAGE                 3¦
+-----------------------------------------------------------------------------+
```



```
   I  000 STARTBUT
   I  001 PHOTO SW1
   I  002 LIFT UPPER
   I  003 PUSHER OUT
   I  004 LIFT LOWER
   I  005 AUTO
   I  006 PUSH BUT1
   I  007 PUSH BUT2
   I  008 PUSH BUT3
   I  009 PUSH BUT4
   I  010 PHOTO SW2
   O  200 GREEN LAMP
   O  201 CONVEYOR
   O  202 LIFT UP
   O  203 PUSHER
   O  204 LIFT DOWN
   O  205 CONV UPPER
   O  206 ALARM
   M  967 INIT
   M  990 TRUE

+-------------------------------- Actron AB ----------------------------------+
¦                  Allocation list packed              ¦  LANDSTROM ELECTRIC  ¦
+------------------------------------------------------+----------------------¦
¦    CONVEYOR AND LIFT CLOSE TO PACKING STATION        ¦DATE        1994-08-21¦
¦                          *                           ¦PAGE                 1¦
+-----------------------------------------------------------------------------+
```

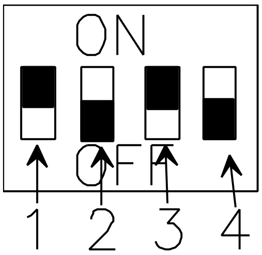


```
   --- Logical box (Unit 1) ---
   CONVEYOR   = PUSH BUT1*/PHOTO SW1
   LIFT UP    = PUSH BUT2*/LIFT UPPER
   LIFT DOWN  = PUSH BUT3*/LIFT LOWER
   PUSHER     = PUSH BUT4
   CONV UPPER = TRUE
+-------------------------------- Actron AB ----------------------------------+
¦                      Function Boxes                  ¦  LANDSTROM ELECTRIC  ¦
+------------------------------------------------------+----------------------¦
¦    CONVEYOR AND LIFT CLOSE TO PACKING STATION        ¦DATE        1994-08-21¦
¦                          *                           ¦PAGE                 1¦
+-----------------------------------------------------------------------------+
```

3. Programming of special functions:

3.1. ECL Link modules (Link)

3.1.0.1. Configuration of the link system



You can connect up to 8 ECL in a link system. One unit must be set up as the master. (This means that this unit in practice will control the communication between the units. But from the users point of view this has no importance.)
Below the front cover on the right side is a dip switch (4 poles). Switch 1 shall be ON for the master unit and OFF for the others.
If the ECL system is not in a link system all switches shall be OFF.
After a change of a dip switch the **power must be turned OFF and ON**.

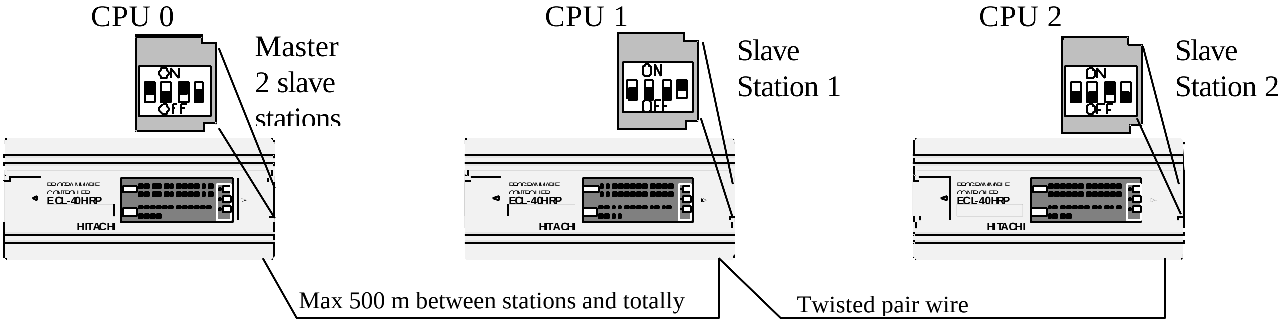
Configuration of Master: (switch 1 ON)

No. of Slaves	Switch No.		
	2	3	4
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON
ERROR	OFF	OFF	OFF

Configuration of Slave (switch 1 OFF)

Unit no.	Switch No.		
	2	3	4
1	OFF	OFF	ON
2	OFF	ON	OFF
3	OFF	ON	ON
4	ON	OFF	OFF
5	ON	OFF	ON
6	ON	ON	OFF
7	ON	ON	ON
ERROR	OFF	OFF	OFF

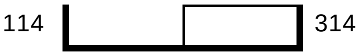
When the link is in RUN the COM lamp on the front of the ECL unit is on.

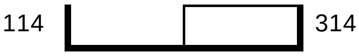


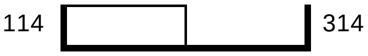
Addr.	Word in Read	Word out Write	Addr.
100		Write	300
102		CPU 0	302
104			304
106	Read		306
108	CPU 1		308
110	Read		310
112	CPU 2		312

Addr.	Word in Read	Word out Write	Addr.
100	Read		300
102	CPU 0		302
104			304
106		Write	306
108		CPU 1	308
110	Read		310
112	CPU 2		312

Addr.	Word in Read	Word out Write	Addr.	Chan-nel
100	Read		300	0
102	CPU 0		302	1
104			304	2
106	Read		306	3
108	CPU 1		308	4
110		Write	310	5
112		CPU 2	312	6







7

The link system communicates through the words on the inputs and outputs. This means that the common area is 8 words * 16 bits IN and 8 words * 16 bits OUT. Each link system is allocated a part of the memory, which is used as write area. This is the area where the system can send information to the other systems.

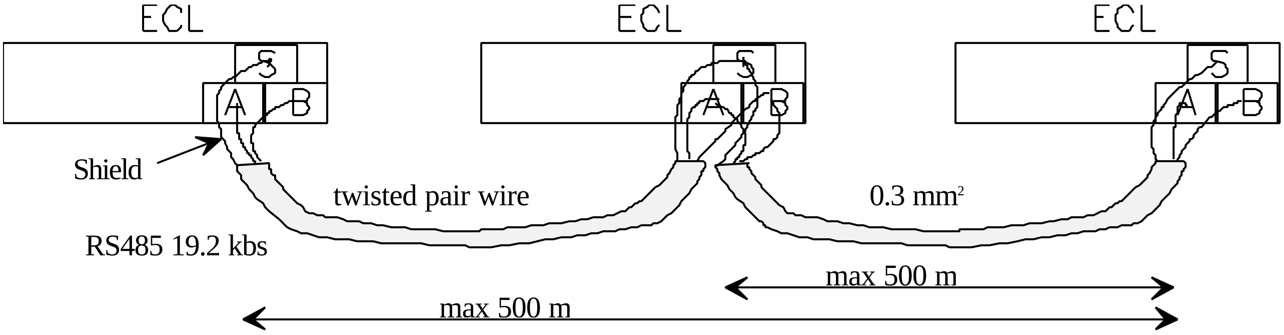
Response times:

The delay time of the information exchange can be calculated in the following way:

Max. Delay =
Scan time for the sending system +
+Up date time for the link (see below) +
Scan time for the receiving system

Up date time for link	**Total number of stations**
80 ms	2
120 ms	3
150 ms	4
190 ms	5
230 ms	6
270 ms	7
300 ms	8

3.1.0.2. Wiring of the link system.

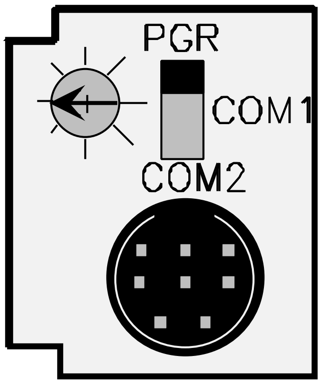


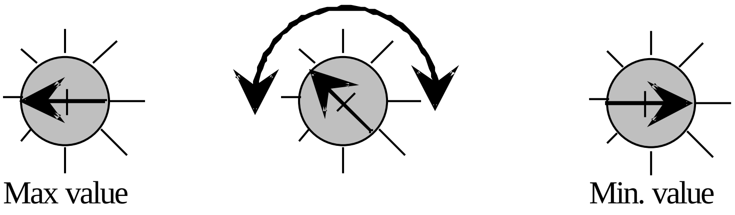
All CPUs can read the write areas of the other CPUs.
To use the complete read/write area, communication is done with the word instructions FUN 10 and FUN 21 (see page 16).
If you want all 128 bits directly accessible in the program you can copy all 2 byte addresses to bit addresses in the following way:

Memory	Part of the Program	Part of the Program continues
Write area: Bit Memory 500-547 Read CPU1: Bit Memory 400-431 Read CPU2: Bit Memory 450-481	FUN20 500 reflect the write area FUN21 300 FUN20 516 FUN21 302 FUN20 532 FUN21 304	FUN10 106 Reflect the read area CPU 1 FUN22 400 FUN10 108 FUN22 416 FUN10 110 Reflect the write area CPU 2 FUN22 450 FUN10 112 FUN22 466

3.2. Analogue timer (Counters) on ECL

The analogue timer is below the front cover on the left side
The timer has the address T/C95.
To use the analogue timer you must define it with FUN97. (see below)
If this is not done T/C95 will remain as a normal timer or Counter.





FUN 97 16 declares T/C95 as an analogue timer with 1 s as a base
The area is 0-999 s (or 0-9999 times as a Counter)

```
 ¦                       +---------+ ¦
 +-----------------------¦FUN97  16+-¦
 ¦                       +---------+ ¦
```

FUN97 **16**

FUN 97 17 declares T/C95 as an analogue timer with 0.1 s as a base
The area is 0-99.9 s (or 0-999 times as a Counter)

```
 ¦                       +---------+ ¦
 +-----------------------¦FUN97  17+-¦
 ¦                       +---------+ ¦
```

FUN97 **17**

FUN 97 18 declares T/C95 as an analogue timer with 0.1 s as a base
The area is 0-9.9 s (or 0-99 times as a Counter)

```
 ¦                       +---------+ ¦
 +-----------------------¦FUN97  18+-¦
 ¦                       +---------+ ¦
```

FUN97 **18**

FUN 97 19 declares T/C95 as an analogue timer with 0.01 s as a base
The area is 0-9.99 s

```
 ¦                       +---------+ ¦
 +-----------------------¦FUN97  19+-¦
 ¦                       +---------+ ¦
```

FUN97 **19**

The instruction FUN97 (with argument 16-19) can be written anywhere in the program (but only once)

The current value (T/C195) can be read in the same way as a normal timer/Counter.
(FUN 10 T/C 195 and. FUN21 T/C 195, page 38)
The preset value (T/C295) can also be read but not written. (FUN 10 T/C 295, page 38)

The accuracy of the timer is +/- 15%.

4. Trouble shooting

No.	Phenomenon	Check item	Check result	Remedy
1	Power lamp does not light when turning on the power supply	Check line voltage	Abnormal	Correct to normal line
			Normal	Exchange the product
2	Operation does not start though STA input turns on.	Connect STA and 24 V terminals to check if the lamp is on.	Abnormal	Correct external wiring around start switch
			Normal	Exchange the product
		Check programmer switch	Set to PROG	Set to TEST or RUN
		Conduct syntax Check by key in of [CLR] [SRC]	Error detected	Correct program
			Error not detected	Exchange the product
3	During operation RUN lamp went off and operation stopped. (Or RUN lamp went off shortly after start of operation.)	Check if ERR lamp is lit.	Lit	Eliminate noise source and recheck the program. Then restart the operation. (If error occurs after eliminating the noise source the product must be exchanged with a new one.)
		Check if a shorter program can run	Can run	The scan time is probably longer that 100 ms. Make the program shorter.
			Cannot run	Exchange the product
4	Input lamp stays OFF	Connect the relevant input terminal and 24 V terminal to check if the lamp lights up.	Lights up	Correct external wiring or replace external input device.
			Does not light up	Utilise unassigned input terminal or exchange the product.
5	Input lamp won’t go off.	Open-circuit the relevant input terminal and check if the lamp goes off	Goes off	Correct external wiring or replace external input device.
			Does not go off	Utilise unassigned input terminal or exchange the product.
6	Output lamp will not come on or go off	Monitor the relevant output and confirm that the lamp status matches the monitored contents.	Matches	Correct the program
			Does not match	Utilise unassigned output terminal or exchange the product
7	Output lamp does not match load ON/OFF status	Check for conductivity across relevant output terminal and C terminal using a tester	Output lamp matches conductive status.	Correct external wiring or exchange external output device.
			Output lamp does not match conductive status.	Utilise unassigned output terminal or exchange the product. (If the contacts of internal relay are fused because of excessively large load current an intermediate relay is required)

5. Technical description.

5.1. Technical data

Basic specification

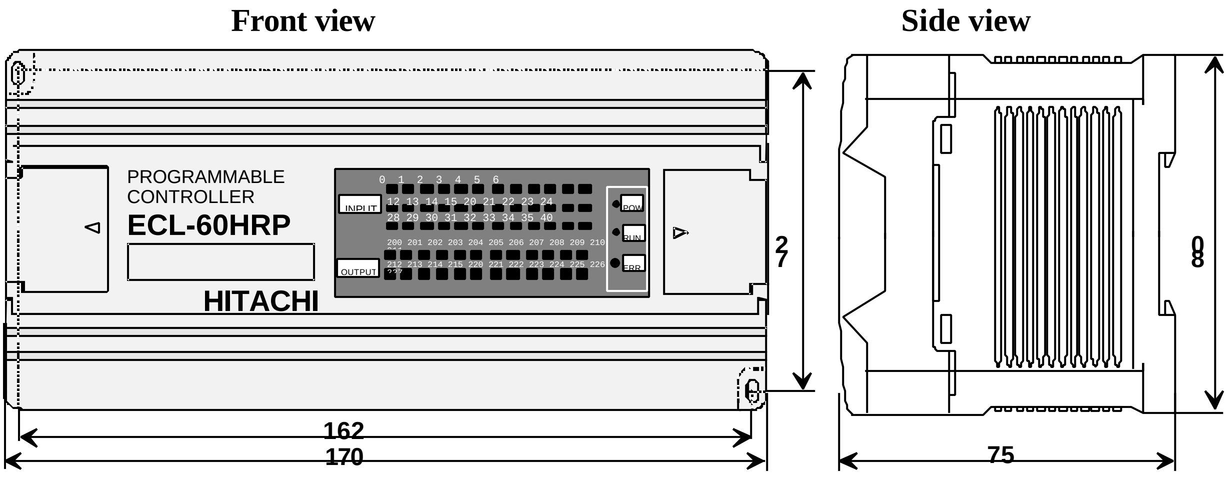
		EC standard	ECL (link type)
CPU-specifica-tion	Process method	Cyclic processing	
	Cycle time	1.5 µs basic instruction	
	Program size	1948 program instructions EEPROM	
Program-instruc-tions	Logic instructions	12 (ORG, STR, AND, OR, OR STR, AND STR, OUT etc.)	
	Arithmetic (word-) instructions	36 (Word load, Word out, + - x /, Word OR, Word AND etc.)	
	Application instructions	21 (Edge detection, Jump, Master Control etc.)	
Input/ /Output handling	External Inputs	24 V DC, PNP, loads 7 mA	
	External Outputs	Relay outputs, 1 A at resistive load. 4 outputs per common per	
	Retentive memories	256 (128 words)	
	Non-retentive memories	256 (128 words)	
	Special memories	12 + 4 words. E.g. time, error information etc.	
	Counters/Timer type	Up Counting	
	Counters/Timer amount	96	
	Timer preset	0.01-9.99 s, 1-999 s, (10 timers 1-9999)	
	Counters preset	1-999 (10 counters 1-9999)	
	Analogue timer	-	1
	High speed counter	1 (2-phase) Up-/down Counter, 10 k Hz, 8 BCD digits (input X0-X2 if defined by program) For a single phase subtraction the speed is reduced to 1.5 kHz	
	CPU link	-	128 bits / 8 Words (RS-485)
	External interrupt	1 (input X3 if defined by program)	
External communi-cation	RS232-port is standard, Connection directly to PC/terminal for programming and control.		

Backup of memory during power off	2 weeks with built in capacitor (at 25 °C)
Voltage supply	19.2 V to 30 V DC (maximum ripple xxxx %) or 85 V to 250 V AC
Dielectric strength	1500 V AC during 1 min. between the input and output terminals (inclusive the supply terminal) and ground terminal
Insulation resistance	>= 20 M0 during 1 min. between the input and output terminals (inclusive the supply terminal) and ground terminal at 500 V DC
Operating/Storage temperature	0 to 55 º C / -10 to 75 º C
Working /storage humidity	20 to 90 % Relative humidity (no condensing) / 10 to 90% (No condones)
Vibration resistance	Conforms to JIS C0911 IIB 3rd class on condition that vibration with frequency 10 to 55 Hz and amplitude 0.5 mm is applied for 2 hours in each of X, Y and Z directions.
Shock resistance	Conforms to JIS C0912 on condition that shock of 10 G is applied twice an each of X, Y and Z directions.
Noise resistance	Noise voltage 1500 V p-p, pulse width 1 µs
Environment	Must be free from corrosive gas and dust.
Altitude	2000 m or less
Grounding	100 0 max.

Type	Voltage supply	Article no.	Description
Standard type	24 V DC	EC-D20HRP	12 Inputs 24 V DC, 8 Outputs Relay 1 A.
		EC-D28HRP	16 Inputs. 24 V DC, 12 Outputs. Relay 1 A.
		EC-D40HRP	24 Inputs. 24 V DC, 16 Outputs. Relay 1 A.
		EC-D60HRP	40 Inputs. 24 V DC, 24 Outputs. Relay 1 A.
	85 - 240 V AC	EC-20HRP	12 Inputs. 24 V DC, 8 Outputs. Relay 1 A.
		EC-28HRP	16 Inputs. 24 V DC, 12 Outputs. Relay 1 A.
		EC-40HRP	24 Inputs. 24 V DC, 16 Outputs. Relay 1 A.
		EC-60HRP	40 Inputs. 24 V DC, 24 Outputs. Relay 1 A.
Link Type (ECL)	24 V DC	ECL-D20HRP	12 Inputs. 24 V DC, 8 Outputs. Relay 1 A.
		ECL-D40HRP	24 Inputs. 24 V DC, 16 Outputs. Relay 1 A.
		ECL-D60HRP	40 Inputs. 24 V DC, 24 Outputs. Relay 1 A.
	85 - 240 V AC	ECL-20HRP	12 Inputs. 24 V DC, 8 Outputs. Relay 1 A.
		ECL-40HRP	24 Inputs. 24 V DC, 16 Outputs. Relay 1 A.
		ECL-60HRP	40 Inputs. 24 V DC, 24 Outputs. Relay 1 A.
Programming tools	ACT-MINI	Ladder (Ladder), Instruction code	
	ACTGRAPH	Grafcet according to IEC848	
Hand programming units	PGMJ	Standard unit	
	PGMJ-R2	Universal unit with EPROM-programming and printout possibilities	
Cable	CNPG-15	Cable between EC and PGMJ or PGMJ-R/2 (active cable)	
	CBL-EC	Cable between EC and computer	

5.2. Installation.

5.2.1. Dimensions



5.2.2. Mounting in general.

-Reserve a distance of min 50 mm from top to bottom of the control system.

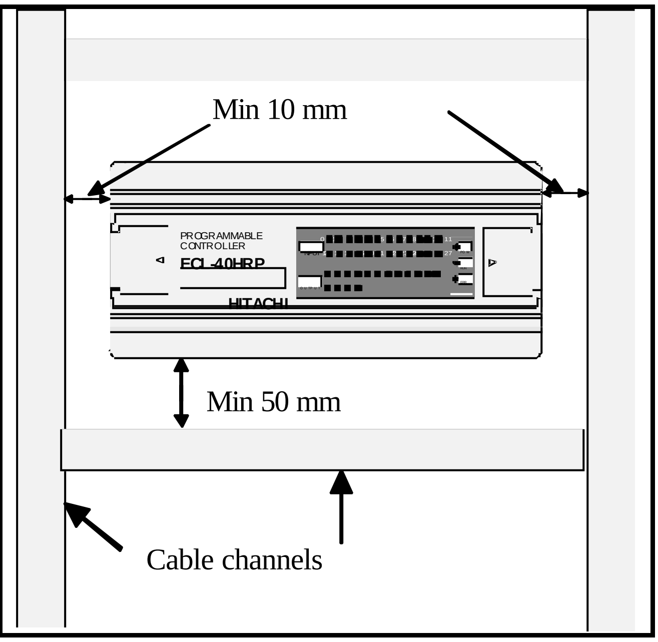
-Take care that metal filings (e.g. from hole drilling) or similar do not fall into the PLC.

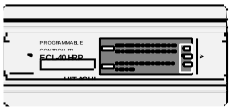
-Avoid installing the PLC directly above a heat source, e.g. a transformer or an power resistor.

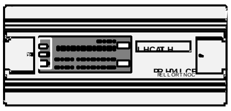
-Keep a good distance to high voltage wiring etc.

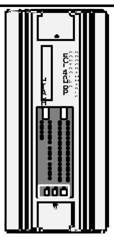
- Avoid installation in direct sun shine and places with condensation, dust, oil, smoke and corrosive gas

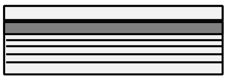
- Avoid installation where much vibrations or shocks occur.



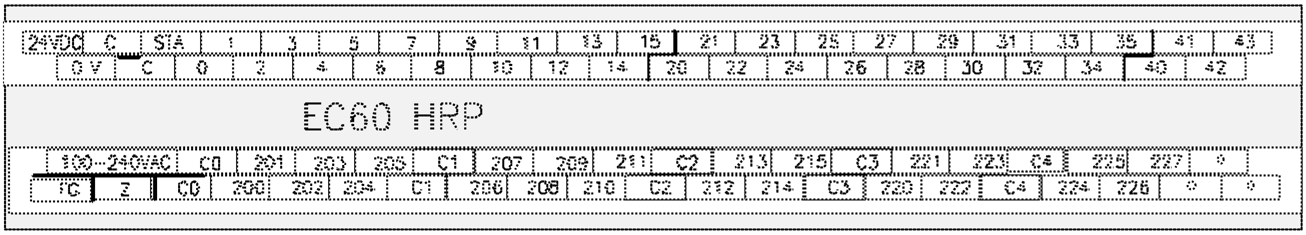

Normal monting

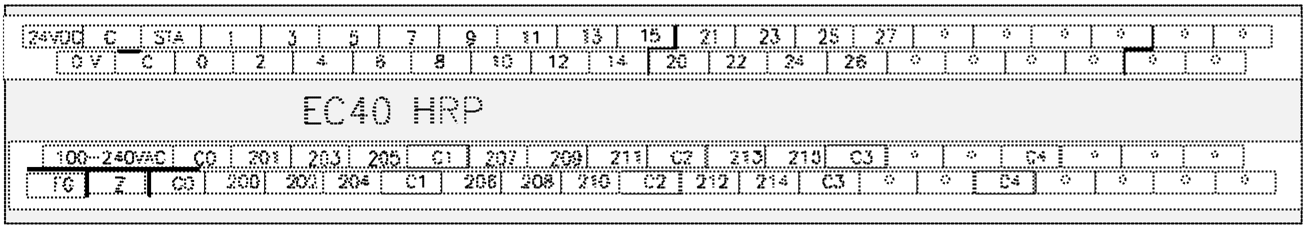

Upside down mounting

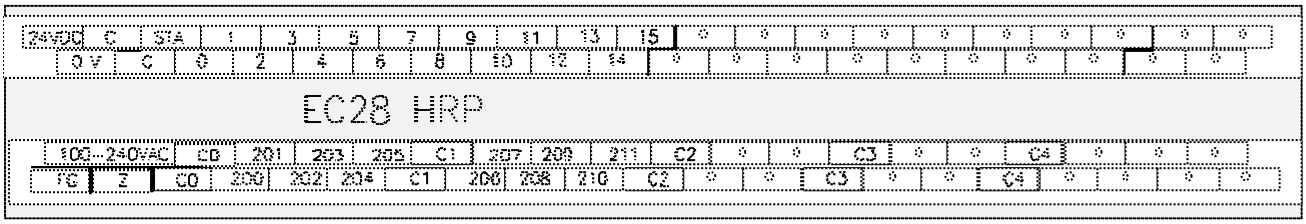

OK if the power supply is up

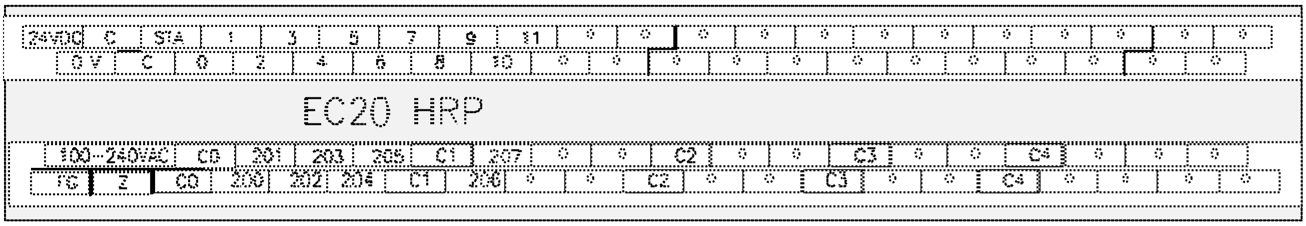

Only OK if the surounding temperature is < 40 °C

5.2.3. Screw terminal layout.

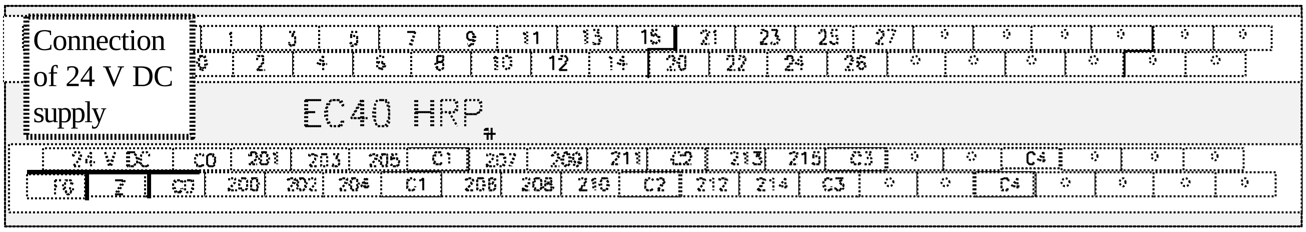

24VDC C STA 1 3 5 7 9 11 13 15 21 23 25 27 29 31 33 35 41 43
0 V C 0 2 4 6 8 10 12 14 20 22 24 26 28 30 32 34 40 42
EC60 HRP
100~240VAC C0 201 203 205 C1 207 209 211 C2 213 215 C3 221 223 C4 225 227 ○
FG Z C0 200 202 204 C1 206 208 210 C2 212 214 C3 220 222 C4 224 226 ○ ○


24VDC C STA 1 3 5 7 9 11 13 15 21 23 25 27 ○ ○ ○ ○ ○ ○
0 V C 0 2 4 6 8 10 12 14 20 22 24 26 ○ ○ ○ ○ ○ ○
EC40 HRP
100~240VAC C0 201 203 205 C1 207 209 211 C2 213 215 C3 ○ ○ C4 ○ ○ ○
FG Z C0 200 202 204 C1 206 208 210 C2 212 214 C3 ○ ○ C4 ○ ○ ○ ○

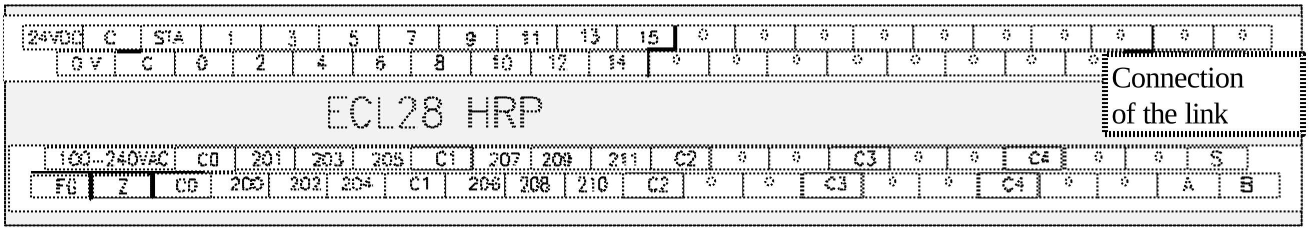

24VDC C STA 1 3 5 7 9 11 13 15
0 V C 0 2 4 6 8 10 12 14
EC28 HRP
100~240VAC C0 201 203 205 C1 207 209 211 C2 C3 C4
FG Z C0 200 202 204 C1 206 208 210 C2 C3 C4


24VDC C STA 1 3 5 7 9 11
0 V C 0 2 4 6 8 10
EC20 HRP
100~240VAC C0 201 203 205 C1 207 C2 C3 C4
FG Z C0 200 202 204 C1 206 C2 C3 C4

ECD-versions: (24 V DC power supply)


Connection of 24 V DC supply
1 3 5 7 9 11 13 15 21 23 25 27
0 2 4 6 8 10 12 14 20 22 24 26
EC40 HRP
24 V DC C0 201 203 205 C1 207 209 211 C2 213 215 C3 C4
FG Z C0 200 202 204 C1 206 208 210 C2 212 214 C3 C4

ECL-versions: (Link)


Connection of the link
24VDC C STA 1 3 5 7 9 11 13 15
0 V C 0 2 4 6 8 10 12 14
ECL28 HRP
100~240VAC C0 201 203 205 C1 207 209 211 C2 C3 C4 S
FG Z C0 200 202 204 C1 206 208 210 C2 C3 C4 A B

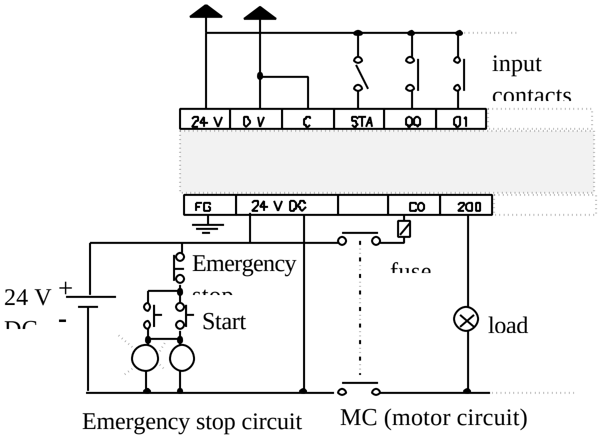
C stands for Common. E.g. the outputs 206-211 are fed fromcommon terminal C1.

5.2.4. Wiring

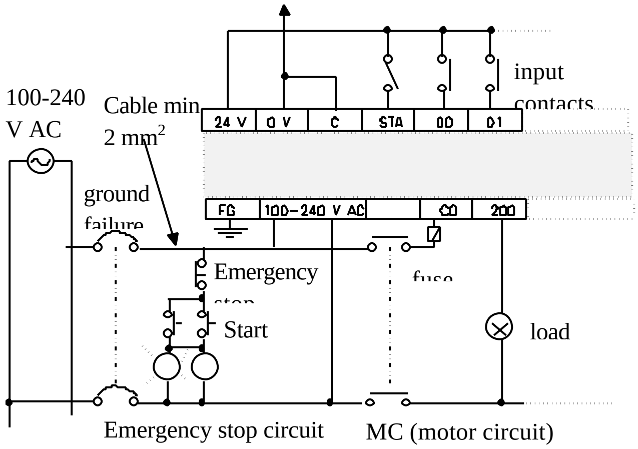
5.2.4.1. Wiring of power supply

Example of wiring:

ECDxxHRP 24 V DC.



ECxxHRP 100-240 V AC.

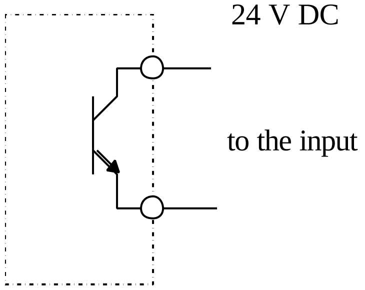
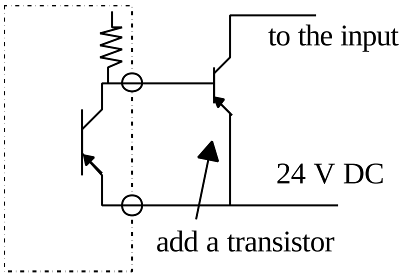


Current consumption:
(also valid for ECL versions)

		EC-20		EC-40	EC-60	With programming unit
Supply Voltage level	ECD xx 24 V DC	18-30 V DC				
	EC xx 240 V AC	93.5 - 250 V AC	50/60 Hz			
Current consumption	ECD xx 24 V DC	< = 400 mA		< = 600 mA	< = 700 mA	add 260 mA
	EC xx 240 V AC	< = 18 VA		< = 23 VA	< = 28 VA	add 10 VA
Total current to sensors from the 24 V terminal		I = 470 mA - (7 mA x amount of ON inputs) - (6 mA x amount of ON outputs) - - (100 mA for the programming unit)				
Max. allowed voltage interruption		20 ms				

5.2.4.2. Wiring of inputs

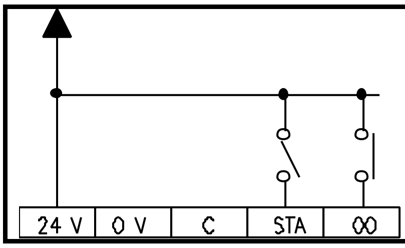
The EC Series has a 24 V DC power supply for external inputs. When an input is closed to 24 V DC a current of 7 mA runs through the input. You can connect inductive sensors, photocells, etc., which are of Source type (PNP). Sensors of Sink type (NPN) have to be connected via a transistor connection.

Normal connection with Source sensor.  	Normal connection with Sink sensor .  	The ripple on the 24 V DC output is less 200 mV p-p. Therefore it is normally no problem to feed external sensors. If there is a problem due to a sensitive sensor or very long distances, connect an aluminium electrolyte capacitor of 100 μF or more (50 V min) between 24 V connection and the 0 V connection.

Maximum length of the input cables is 100 m with the input cables separated from power cables, output cables etc. Otherwise max. 30 m
The total resistance shall not exceed 300 0.

Ripple on the inputs shall not exceed 10 %.

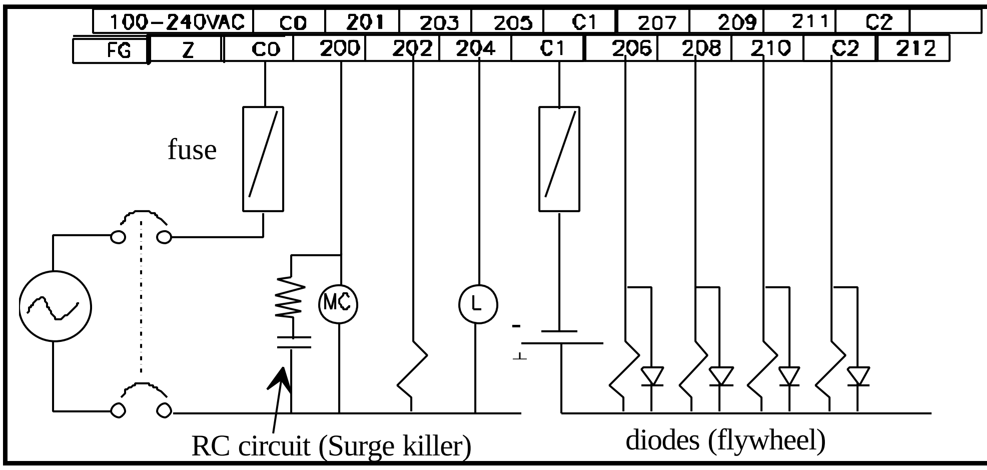
5.2.4.3. STA-input



The STA-input shall be closed during run.

This is also valid during computer programming.

5.2.4.4. Wiring of Outputs



For inductive load, connect a RC circuit (0.1 μF + 100 0) over the AC coil and a diode over the DC coil.

6. Appendix

6.1.2. Explanations

Bit:	Is an input, output or internal output, which can be represented by ”ON/OFF” or ”1”/”0” etc. (two possible conditions.)
Word:	16 bits in a row, which together will represent a value between 0 and 65535. The word can be a **16 bit word**. That means that the most significant bit of 16 addresses (with 8 bits behind every address) together will be a word. It can also be a **2 byte word** which means that 8 bits from two byte addresses together will be a word. The instruction type decides which type of word addressing it is.
Byte:	8 bits which represent a value between 0 and 255. A 2 byte word consists of two bytes (The memory is basically organised in bytes. Behind every memory address there is a byte available for memory handling. This is used in 2 byte word addressing.)
RR	The Result Register or Flag. The temporary logic condition during program execution makes this flag true or false. The output instructions will be effected according to the logic status of RR.
Logic stack	The logic status in RR is stored in the logic stack. This can be done in several levels. Every time the instructions STR or STR NOT are used the logic status of RR is pushed down one level in the stack and it is raised one level when AND STR or OR STR are used.
AR **Arithmetic register**	This contains 16 bits (”1” or ”0”). It is used for all temporary storage during word handling like copying of words between inputs, outputs or internal memory words, counters, timers etc.
ER **Extra register**	This contains 16 bits (”1” or ”0”). It is used for all temporary storage for operations where the AR register is not enough.
Internal memory area.	The memory is either divided into ”1” or ”0” bits, which is valid for normal internal outputs) or words (16 bits) which is valid for e.g. up / down counters, shift registers etc.
Special internal memories	Internal memories with special function, e.g. clock pulses and error status.
Inputs	Physical connected inputs from different sensors.
Outputs	Physical connected outputs to different devices.
b7	The most significant bit in a memory address. The only one which is read by a logic instruction. The rest of the bits (b0 - b6) can only be read by 2 byte word instructions.

6.2. BCD/HEX-code, etc. explanations

Handling of BCD code in the memory or inputs/outputs.

BCD code Binary code

Every digit (0-9) corresponds in the binary code of a combination of 4 bits (”1” or ”0”). In this way a 4 digit value can be represented in the memory by 16 bits. This is e.g. valid for Up /Down counters.

E.g. the value 5107 stored in the memories 620 to 635 is represented:

```
    5       1       0      7
+-------------------------------+
¦1 0 0 1¦0 0 0 1¦0 0 0 0¦0 1 1 1¦
+-------------------------------+
 ¦                             ¦
 635                          620  (address)
```

Address 635-632 is 0 1 0 1 which according to the table below corresponds to 5.
Address 631-628 is 0 0 0 1 which according to the table below corresponds to 1.
Address 627-624 is 0 0 0 0 which according to the table below corresponds to 0.
Address 623-620 is 0 1 1 1 which according to the table below corresponds to 7.

If this value was interpreted in binary, then the hexadecimal value 5107 will be:
5 * 4096 + 1 * 256 + 0 * 16 + 7 * 1 = 20743 in decimal code.

Hexa decimal representation:
Because of practical reason the hexadecimal representation is often used in stead of the binary to represent the register content. A hexadecimal value has the base ”16” instead of ”10”. That means that the least significant part of the value is a multiple of 16^0 (=1), the next will be a multiple of 16^1 (=16) and the next one a multiple of 16^2 (=256) etc. The conversion is done like the example above.
The conversion between binary and hexadecimal and vice versa is very simple as every hexadecimal digit corresponds to four binary bits.

Hexadecimal	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3´	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

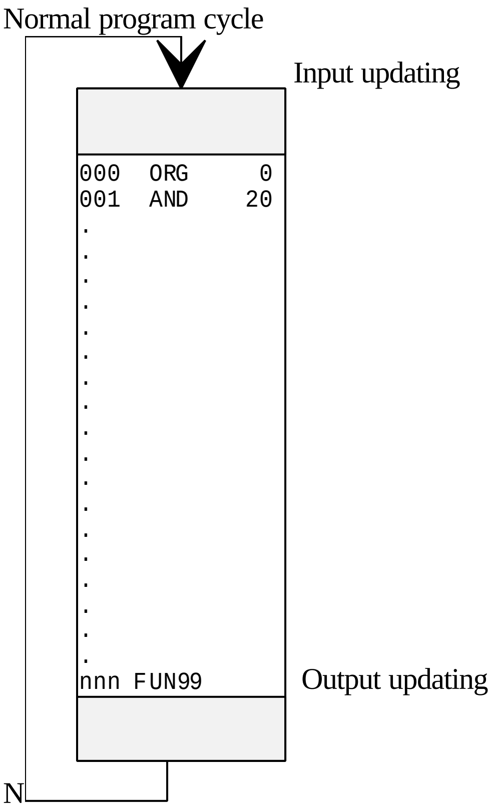
Explanations of comparison signs:

< :	Less than
> :	Greater that
<= :	Less than or equal to
>= :	Greater than or equal to

LSD :	Least Significant Digit
MSD :	Most Significant Digit
LSB :	Least Significant Bit (b0)
MSB:	Most Significant Bit (b7)

```
 MSD                              LSD
+----------------+----------------+
¦       ¦        ¦       ¦        ¦
+----------------+----------------+
 ^              ^ ^              ^
 ¦              ¦ ¦              ¦
b7     401     b0 b7     400     b0
```

6.3. Program processing principles



During normal a **program cycle** the updating of the inputs take place before the program scan.
This means that the physical inputs are **copied to the memory**., which is a copy of the input status.
Thereafter the program is executed instruction by instruction from row 0 until the last one (where it finds the END instruction FUN99).
Then the program scan is interrupted and the outputs are updated. That means that memory, which is a copy of the output status is copied to the physical outputs.

The logic program only works against memory copy of the inputs and outputs. (not directly on the physical inputs and outputs)

The **longest response time which can occur is close to 2 program cycles**. This happens if the input status changes right after the updating of the inputs. In this case it will take another program cycle before the right status is copied to the input copy.
Then the program scan is executed and the output status in the memory copy is changed due to a logic combination.
In the end of the scan the physical outputs are updated.
This will take almost 2 program cycles in worst case.

If a **refresh instruction** is used the updating of the input or output will take place when the instruction is executed.
Therefore it is possible to reduce worst case to one program cycle if the refresh instructions are used (see page 44).

7. Index