Contents

Introduction

Remarks

The background programs which can be found in our toolbox are examples with which simple customised tasks can be solved in a day to day practice.
On one side these examples will show you a direct access to fully programmable components of the voltage regulator system REGSys™ via the programming language REG-L(anguage) and on the other side it shows you typical tasks which arise around transformers and choke coils.

It must be safeguarded that the standard functions of the regulators are not affected by the use of the following examples i.e. customised applications. For this reason 'a. eberle gmbh´ assumes no responsibility for damages arising from unqualified handling of REG-L commands.
However, the possibilities of having a self established or modified H-program tested by our engineers is available. Please send us an e-mail to (info@a-eberle.de **or** gerhard.seitz@a-eberle.de**). They will help you wherever they can.**

We wish you success and a lot of fun !

Pattern for a background program

The background program should be set up as shown below

```
#########################################################
# file name:            *.rgl                   date: **.**.00
# version:
# customer/project.:
# established by:                               a.eberle gmbh
# changed by:
#########################################################
# change prompt to your own identification code
rem,aa::
# delete all H- and P- program lines
H 0..31=''
P 0..31=''
# delete application menu
menuappn *="",menuapp *=-1
# stop H-Programm
hbreak+++

H 0='#       *.rgl'

#  first line for a new program
H 1=''

# change of summer/winter time
H 31='SOWI, IF, ZEIT-, +, ZEIT=.'

# start H-program
hbreak-
```

General tasks

Delete H- / P- programs

Task:
All H- and P- program lines should be deleted (for example before starting to program)

solution:
H 0..31=' '
P 0..31=' '

Loading of H-programs with specific identification codes

Task:
Similar H-programs should be loaded in several regulators (for example identification code "A", "B" and "C") which were all written in one download file.

Solution:
Change prompt to your own identification code so that all commands can be executed on your own regulator
Please note, that according to the "RegUpdate" program the 1st command should not include an identification code.
rem,aa::
for example the line H 2 should be different from the regulator
Kenn, 1,==,if, H 2='meld "Ich bin Regler A"'
Kenn,11,==,if, H 2='meld "Ich bin Regler B"'
Kenn,21,==,if, H 2='meld "Ich bin Regler C"'

Initialisation of the register

Task:
All registers should be deleted and the register b12 should be allocated a value.

Solution:
delete all registers
H 1='a *=0, b *=0'
assign register b12 to value 999
H 2='b12=999'

Toggle (switch) Boolschen (binary) value in a register

Task:
if a 1=1 => a 1=0
if a 1=0 => a 1=1

Solution:
H 1='a 1,^,a 1=.'

Output of register contents as a message on display

Task:
The current register value b 0 should be shown on the display.

Solution:
H 30='b 0,meld "Wert B0 = %!"'

Display of parallel-error or ELAN-error via LED

Task:
A LED should indicate an Elan-error or parallel error.

Solution:
LED 1 is on if Elan-error or parallel-error occurs
H 2='RegRelFW- 17,RegRelFW- 30,l,if,RegLED 1=1,else,RegLED 1=0'

Identification code enquiry in ELAN

Task:
a1=1, if the regulator with code B1 is included in the ELAN

Solution:
H 1='B1:indir*,a 1=.'

Generation of an impulse (transient signal)

Task:
If a BCD code occurs, a 3 s impulse should be generated on relay 4.

Solution:
H 1='RegStufe-,99,==,iff,a 1=1'
H 2='htd,a 1,if,a++ 63=.,a 63,3,>,if,a 1=0,endif,else,a 63=0'
H 3='a 1,RegRel 4=.'
or
H 1= 'RegStufe-,99,==,iff,zeit_*-,a63=.,a 1=1'
H 2='a1,if,3,zeit_*-,a63,-,>,a1=.'
H 3='a1,RegRel 4=.'

Variables:
a1: flag for tap position 99
a63: timer for an impulse of 3 s

Response on slopes and levels of a signal

General task:
if slope of signal is 0=>1 then => action1
if slope of signal is 1=>0 then => action2
if signal is =1 then => action3
if signal is =0 then => action4

General solution:
„**Signal**“ ,dup,iff, „**Aktion1**“ ,else, „**Aktion2**“ ,endif,if, „„**Aktion3**“ ,else, „„**Aktion4**“ '

Example:
Two parameter blocks should be managed. A block can be chosen via the binary input E7. This is shown below with the parameter “permissible system deviation” only.

Solution:
- if E7 =0
  => follow the changes of the 1st block (copy current parameter values in value block 1)
- if slope of E7 is 0=>1
  => activate parameter block 2 (write parameter with the value block 2)
- if E7 =1
  => follow the changes of the 2nd block (copy current parameter values in value block 2)
- if slope of E7 is 1=>0
  => activate parameter block 1 (write parameter with the value block 1)

H 1=‘RgE- 7,dup,iff,b2,RegABW =.,else,b1,RegABW =.,endif,if,RegABW-,b2 =.,else,RegABW-,b1 =.‘

Variables:
b1: value block 1
b2: value block 2

Time delayed response on an event

Task:
LED 1 should be on if the voltage of the regulator U1 lies longer than 60 s above 120V.

Solution:
H 2='htd,**RegU1-,120,>**,if,b 0,^,if,a++ 31=.,a 31,**60**,>,b 0=., endif,else,b 0=0,a 31=0'
H 3='b 0,if,**RegLED 1=1,else,RegLED 1=0**'

Parameter:
b 0 =1if the time is exceeded and the condition remains
a 31 counter for the time delay

Enquiry command at a certain time

Task:
At 10.30 am it should be changed over to setpoint 2.

Solution:
H 10=‘hh,10,==,mm,30,==,&,iff,RegSWI =2‘

Processing of a measured value only after considerable changes

Task:
A measured value should only be taken over if it has changed by a previous set delta.
For example: RegU should be saved in register a 0, if the change (in b 0) exceeds 5V.

Solution:
H 0='1,iff,a 0=0,b 0=5'
H 1='RegU-,dup,a 0,-,abs,b 0,>=,if,a 0=.'

Conditional blocking of the keyboard and of certain inputs

Task:
The keyboard and selected binary inputs should be blocked via binary inputs.

Solution:
blocks keyboard, if E7=1
H 1='RegE- 7,if,RegSperreTas =2,else,RegSperreTas =0'
blocks the inputs E1...E6 as long as E8=1 (blocks via bit mask 0b00111111 = 63)
H 2='RegE- 8,if,RegSperre_E =63,else,RegSperre_E =0'

Simulation of key sequences

Task:
If the permissible system deviation is violated, the regulator should switch into regulator mode and change to maximum display. If the voltage lies within the system deviation again the maximum display should disappear.

Solution:
H 1='Regxw-,abs,1,>,iff,tt qqm1,RegBigDisp=1,else, RegBigDisp=0'

Adjustable REG-L register via application menu

Task:
The register a 2 should be changed via application menu.

Solution:
set default value
H 1='1,iff,a2=55‘
delete all menus
H 2='1,iff,menuappn *="",menuapp *=-1'
application menu, line 1
H 3='1,iff,menuappn 1=“Para a2“,menuapp 1=103'
define menu
P 3='menuedit a 2 ="a2 [1..1000]" 1 1000'

Variables:
a2: parameter to be set

Count of operation hours

Task:
The time of the regulator being connected to the net can be interrogated via application menu.
The time can be reset via „Master-Reset“ otherwise the value will be buffered via battery

Solution:
calculate operation time in seconds and convert into hours
H 1='1,iff,menuappn 1="Betriebsst.",menuapp 1=1'
P 1='zeit_-,3600,/,meld2 "Betriebsstunden" "%.2! h"'

Display of system deviation via LEDs

Task:if the system voltage is outside the tolerance band.

LED 1: voltage is above the tolerance band by the double
LED 2: voltage is above tolerance band
LED 3: voltage is below tolerance band
LED 4: voltage is below the tolerance band by the double

Solution:
H 1='RegXw-,1,>, if, RegLED 2 = 1, else, RegLED 2 = 0'
H 2='RegXw-,-1,<, if, RegLED 3 = 1, else, RegLED 3 = 0'
H 3='RegXw-,2,>, if, RegLED 1 = 1, else, RegLED 1 = 0'
H 4='RegXw-,-2,<, if, RegLED 4 = 1, else, RegLED 4 = 0'

short form:
H 1='RegXw-,1,>,RegLED 2 =.'
H 2='RegXw-,-1,<,RegLED 3 =.'
H 3='RegXw-,2,>,RegLED 1 =.'
H 4='RegXw-,-2,<,RegLED 4 =.'

in addition to that, the signals for inhibit are suppressed:
H 0='RegRelFW- 13,b00=.'
H 1='RegXw-,1,>,b00,^,&, RegLED 2 =.'
H 2='RegXw-,-1,<,b00,^,&, RegLED 3 =.'
H 3='RegXw-,2,>,b00,^,&, RegLED 1 =.'
H 4='RegXw-,-2,<,b00,^,&, RegLED 4 =.'

Information if the recording memory overflows

Task:
Relay 5 should pick up if the recording memory is more than 80% full.

Solution:
H 2='1,iff,RegRelFu 5=1'
H 3='FSize- rec p,80,>,RegRel 5=.'

Activation of automatic operation via binary input

Task:
The Auto-operation should be activated via input E7. It should switch into Manual-operation if the input is deactivated, but a change between Manual/Auto via keyboard should be possible again.

Solution:
```
# permanent switch to Auto
H 1='RegE- 7,if,RegAUTO=1'
# switch to manual only once
H 2='RegE- 7,^,iff,RegAUTO=0'
```

Deactivation of current influence if the active power is below zero

Task:
The current influence (current program) should be deactivated if the active power is below zero until the active power is above zero again.

Solution:
```
# the current influence should be deactivated if power P < 0.
H 1='RegP-,0,<,if,RegStromPR=0,else,RegStromPR=1'
```

Automatic generation of a group list

Task:
For all regulators (up to 10, connected via ELAN) it should be recognised whether they feed on bus bar 1 (input 7) or feed on bus bar 2 (input 8). The group lists should be made up accordingly. The parallel operation should be activated if more than two regulators are working on the same bus bar .

Solution:
```
# set parameters for inputs
H 1='RegEFu 7+8=1'
# flag condition of E7 and E8
H 2='RegE- 7,a1=.,RegE- 8,a2=.'
# analyse switching conditions
H 3='a1,if,a2,if,a10=0,else,a10=1,eif,else,a2,if,a10=2,else,a10=0'
# generate group list
H 4='a11=1'
H 5='a10,if,all r,a10,aa:a10,==,if,kenn,aa:,a11,RegParaGr .=.,a++ 11,endif,nexta'
H 6='a11,10,fori,i,RegParaGr .="---",nexti'
# activate parallel operation
H 7='a10,a11,2,>,&,if,RegParaPROGA=1,else,RegParaPROGA=0'
```

Variables:
a01: flag for input E7 (buffer for SS1)
a02: flag for input E8 (buffer for SS2)
a10: pointer for switching combinations
0= transformer is connected to no bus bar or to two bus bars
1= transformer is connected to bus bar 1
2= transformer is connected to bus bar 2
a11: counter for current number in group list

Activation Master / Slave function via input

Task:
Regulator A: or B: should become Master and Slave via E8.

Solution:
```
H 1='RegEFu 8=1,RegLEDFu 2=1'

H 3='A:RegE- 8,if,B:RegE- 8,if,a0=3,else,a0=1,endif,else,B:RegE- 8,if,a0=2,else,a0=0'
H 5='Kenn,1,==,if,a0,1,==,if,P 0,else,P 1'
H 7='Kenn,11,==,if,a0,2,==,if,P 0,else,P 1'
H 9='Kenn,1,==,Kenn,11,==,|,^,if,meld "falsche Kennung",P 1'
H 11='a0,3,==,if,meld2 "Signalzustand an" "E 8 ist falsch"'
P 0='RegParaPr =6,RegParaGr 1=  A,RegParaGr 2=  B,RegParaProgA =1,RegLED 2=1'
P 1='RegParaPr =0,RegParaGr 1=---,RegParaGr 2=---,RegParaProgA =0,RegLED 2=0'
```

Variables:
a0: condition of the inputs E8 on regulator A: and B:

Determination of the controlled variable and transfer to the regulator

Task:
The mean value of the voltages U1 and U2 should be used as a controlled variable.
If one of the two voltages drops below 80% of the setpoint, only the other voltage should then be used for the control.

Solution:
```
# to set parameters for voltages U1 and U2 on mapping via H-program
H 1='RegMiMap 1=-1,RegMiMap 2=-1'
# load register with value of voltages U1 and U2
H 2=',RegMiVal*- 1,b11=.,RegMiVal*- 2,b12=.,'
#check whether the voltage is below 80% of the setpoint.
H 3='RegSWI-,RegSW- .,b11,Pick 2,/,0.8,<,if,b12,b11=.
H 4='RegSWI-,RegSW- .,b12,Pick 2,/,0.8,<,if,b11,b12=.'
# calculation of the mean value
H 5='b11,b12,+,2,/,RegMiVal 1=.'
```

Parameter:
b11 RegMiVal* 1, unmapped voltage U1
b12 RegMiVal* 2, unmapped voltage U2

Changeover between controlled variable U1 and U2

Task:
The controlled variable voltage should be changed from U1 to U2 via input E7, the second setpoint should be used and Knu should be changed.

Please note:
The characteristic "three winding transformer " is necessary for this function.

Solution:
```
# if there is a level on input E7, it will be
#       - controlled via U2,  otherwise  U1
#       - setpoint will be used, otherwise SW1
#       - Knu of 300 (30KV)will be used, otherwise Knu of 100 (10KV)
H 1='RegEFu 7=1'
H 2='RegE- 7,if,Reg3WSELU=2,RegSWI=2,RegKnu=300,else,Reg3WSELU=1,RegSWI=1,RegKnu=100'
```

Addressing of single relays of a BIN-D card

Task:
Set relay 2 of the BIN-D card (change at hardware address 0)

Solution:
in general:
DevRel <adresse> <RelaisNr>=<bool>
or
DevRel* <adresse> =<byte> (wobei: bit0 => Relais1, bit1 => Relais2, ...)

H 1='DevRel 0 2=1'
or
H 1='DevRel* 0=0b00000010'

Change of Setpoints

Selection of setpoints 1..4 via binary inputs

Task:
4 setpoints of the regulator should be activated via continuous signals on the inputs BE1...BE4. If there is no signal or more than one at the inputs, setpoint 1 should be active.

Please note:
characteristic "4setpoints" must be active

Solution:
```
H 2=''
H 3='a 0=0,b 0=0'
H 4=''
H 5='RegBE- 1,if,b 0=1, a++ 0'
H 6='RegBE- 2,if,b 0=2, a++ 0'
H 7='RegBE- 3,if,b 0=3, a++ 0'
H 8='RegBE- 4,if,b 0=4, a++ 0'
H 9='a 0,1,==b 0,&,if,b 0,RegSWI=.,else,RegSWI=1'
```

Variables:
a0: counter for the number of inputs
b0: flag for the setpoint selector

Single changeover to setpoint 2

Task:
The standard setpoint SW2 should be active after PowerON and changeover from Manual->Auto:

Solution:
```
# if PowerOn and Manual->Auto, setpoint 2 is active
H 1='1,iff+,RgSWI=2'
H 2='RgAuto-,iff,RgSWI=2'
```

External setpoint shift with limitation

Task:
The current setpoint should be changed via the external signals on E3 and E4, however, only within certain limits:
A pulse on E3 increases the setpoint by 1V (limited to a maximum of 105 V)
A pulse on E4 reduces the setpoint by 1V (limited to a minimum of 95 V)

Solution:
```
# external setpoint shift with limitation
H 2='RegE- 3,iff,RgSWI-,RgSW- .,105,<,if,RgSWI-,RgSW- .,1,+,RgSWI-,RgSW- .=.'
H 3='RegE- 4,iff,RgSWI-,RgSW- .,95,>,if,RgSWI-,RgSW- .,1,-,RgSWI-,RgSW- .=.'
```

4 setpoints selectable via application menu

Task:
The active setpoint 1...4 should be selectable via application menu.

Please, note::
Characteristic „4 setpoints“ must be activated

Solution:
```
# delete all menus
H 2='1,iff,menuappn *="",menuapp *=-1'
# write menus
H 3='1,iff,menuappn 1="Sollwert1",menuapp 1=1'
H 4='1,iff,menuappn 2="Sollwert2",menuapp 2=2'
H 5='1,iff,menuappn 3="Sollwert3",menuapp 3=3'
H 6='1,iff,menuappn 4="Sollwert4",menuapp 4=4'
# set setpoints
P 1='RegSWI=1'
P 2='RegSWI=2'
P 3='RegSWI=3'
P 4='RegSWI=4'
```

Enquiry and change of tap position

Return to neutral tap position

(automatic tap changing to "Home"-position)

Task: If the binary input shows E4 = 1, the regulator should change into manual operation and should step automatically into a previously set tap position in the application menu.
If the binary input shows E4 = 0 again, the regulator should switch back to its previous operation mode (Manual or Auto).

Solution:

```
#establish application menu
H 2='1,iff,menuappn *="",menuapp *=-1'
H 3='1,iff,menuappn 1="Homing Stufe",menuapp 1=101'
# load register with chosen Homing tap position
P 1='menuedit a 10="1..19" 1 19'

#run regulator in Homing-tap position, free choice of tap position, default = 10
H 8='1, iff, a 10=10'
H 9='a 10,INTR,a 10=.'
H 10='RegE- 4,iff , RegAuto-, a 01=.,else, a 01, RegAuto=.'
H 11='RegE- 4, if, RegAuto = 0'
H 12='RegStufe-, a 10, <, b 02=., RegStufe-, a 10, >, b 03=.'
H 13='b 02, b 03, |, b 00=.'
H 14='RegE- 4, b 00, &, b 01=.'
H 15='b 01, iff, a 30=0, b 10=1, else,a 30=0'
H 16='htd, b 01, b 10, ^,&, if, a++ 30 =.,a 30, RegTLAUFL- ,1,+,>,iff,b 10=1,a 30=0'
H 17='b 01, b 10,&, if, b 10=0, b 02, if, RegHoeher* =1, endif, b 03, if, RegTiefer* =1'
```

Variables:
a 01: original operating condition (0:Manual, 1:AUTO)
a 10: neutral tap position, default = 10
a 30: timer for setting commands
b 00=1: tap position too low or too high
b 01=1: automatic setting procedure activated
b 02=1: tap position too low
b 03=1: tap position too high
b 10=1: execute setting command

Suppress tap changes which exceed the min./max. tap position

Task:
Setting commands which exceed the minimum/maximum limitation (for example tap position 5...20) should be suppressed. This does not work in manual operation mode!

Solution:

```
H 1='RegStufe-,5,<=,if,RegSperreT =3,else,RegSperreT =0'
H 2='RegStufe-,20,>=,if,RegSperreH =3,else,RegSperreH =0'
```

Adjustable limitation of min./max. tap position

Task:
Setting commands which would lead to a violation of the minimum/maximum limits (adjustable in the application menu) are suppressed. This does not work in manual operation mode!

Solution:
```
#establish application menu
H 1='1,iff,menuappn *="",menuapp *=-1'
H 2='1,iff,menuappn 1="min. Stufe",menuapp 1=111'
H 3='1,iff,menuappn 2="max. Stufe",menuapp 2=112'
#limitation of tap changes, free choice of tap position , default min. tap position = 4, max. tap position =15
H 6='1, iff, a 11=4, a 12=15'
H 7='a 11,INTR,a 11=.,a 12,INTR,a 12=.'
H 8='RegStufe-,a 11,<=,if,RegSperreT =3,else,RegSperreT =0'
H 9='RegStufe-,a 12,>=,if,RegSperreH =3,else,RegSperreH =0'
#set minimum tap position in application menu 1
P 1='menuedit a 11="Stufe 1..19" 1 19'
#set maximum tap position in application menu 2
P 2='menuedit a 12="Stufe 1..19" 1 19'
```

Variables:
a 11: minimum limitation of tap position, default = 4
a 12: maximum limitation of tap position, default = 15

Monitoring of the tap changer

Task:
The following events should be recognised:

General information:
-The monitoring functions can be loaded one by one and on demand as H-program modules in the REG-D or PAN-D.
- Inverse tap changers should be taken into account.

- Tap change without indication "tap changer in operation":
If there is no indication "tap changer in operation" after the setting command of the regulator within 3 s (in Auto or Manual operation).

- Tap change without setting command:
If there is an indication "tap changer in operation" without a setting command of the regulator.

Maximum time of "tap changer in operation" exceeded
- If the time for "tap changer in operation" is longer than the set time in the parameter.

- Tap change in the wrong direction :
If the BCD code does not get higher after a higher command.
If the BCD code does not get lower after a lower command.
.

- Wrong BCD-code:
If there is an invalid BCD code detected while using BCD 10 or the BCD 20 signal.

- Tap change without a BCD change:
If there is no change of the BCD code after a setting command.

program lines written for REG-D, for PAN-D the corresponding lines marked with "PAN-D have

to be replaced.

tap change without indication "tap changer in operation"
if tap changer is in operation
H 2='RegRel- 1,RegRel- 2,|,iff,a48=1,a60=0'
if the indication "tap changer in operation " is not received after 3 s -> error flag
H 3='htd,a48,if,a++ 60=.,RegLedFW- 38,iff,a48=0,endif,a60,3,>,if,a51=1,a48=0,endif,else,a60=0'
"PAN-D"
#H 3='htd,a48,if,a++ 60=.,RegRelFW- 19,iff,a48=0,endif,a60,3,>,if,a51=1,a48=0,endif,else,a60=0'

tap changes without a setting command
if tap changer is in operation
H 5='RegRel- 1,RegRel- 2,|,iff,a49=1,a61=0'
H 6='htd,a49,if,a++ 61=.,RegTLaufl-,a61,<,if,a49=0'
#if indication "tap changer in operation" is received in auto mode outside the set time -> error flag
H 7='RegLedFW- 38,iff,RegAuto-,a49,^,&,if,a52=1'
"PAN-D"
#H 7='RegRelFW- 19,iff,RegAuto-,a49,^,&,if,a52=1'

maximum time "tap changer in operation" exceeded
if the time "tap changer in operation" is exceeded -> error flag
H 9='htd,RegLedFW- 38,a53,0,==,&,if,a++ 62=.,RegTLaufl-,a62,<,if,a53=1,a62=0,endif,else,a62=0'
"PAN-D" # "RegRelFW- 08=1 includes the same function"

For these functions the tap position has to be fed back to the regulator

general part
inverse tap changer or standard tap changer
H 10='merkmal- invers,if,a57=-1,else,a57=1'
#record current tap position
H 11='RegStufe-,b50=.'
if HIGHER => expected tap position = present tap position + 1; "tap changer in operation" = 1
H 12='RegRel- 1,iff,b50,a57,+,b51=.,a57,a50=.'
"PAN-D"
#H 12='ar:RegRel- 1,iff,b50,a57,+,b51=.,a57,a50=.'
if LOWER => expected tap position = present tap position - 1; "tap changer in operation" = 1
H 13='RegRel- 2,iff,b50,a57,-,b51=.,a57,-1,*,a50=.'
"PAN-D"
#H 13='ar:RegRel- 2,iff,b50,a57,-,b51=.,a57,-1,*,a50=.'
calculate difference of tap positions
H 14='b50,b51,-,a50,*,b52=.'
if "tap changer in operation" && Differenz==0 => stop "tap changer in operation"
H 15='a50,0,!=,b52,0,==,&,if,a50=0'

tap changes in the wrong direction
include general part
if "tap changer in operation" && difference -2 =>stop "tap changer in operation" and set "tap change in wrong direction"
H 17='a50,0,!=,b52,-2,==,&,if,a50=0,a54=1'

wrong BCD-Code
include general part
if "tap changer in operation" && difference >0 | <-2 => stop "tap changer in operation and set "wrong BCD-Code"
H 19='a50,0,!=,b52,0,>,b52,-2,<,|,&,if,a50=0,a55=1'
if "present tap position" = 99 =>stop "tap changer in operation" and set "wrong BCD-Code"
H 20='b50,99,==,if,a50=0,a55=1'

tap changes without BCD changes
include general part
if "tap changer in operation" => increment timer else Timer =0

```
H 22='htd,a50,0,!=,if,a++ 63=.,else,a63=0'
# if "tap changer in operation" && difference -1 && Timer >max. time tap changer in operation =>stop "tap changer in operating";" set "idle tap change"
H 23='a50,0,!=,b52,-1,==,&,if,a63,RegTLaufl-,>,if,a50=0,a56=1'

##### Variables ##########
# a48   Flag "tap changer in operation " tap change without indication "tap changer in operation"
# a49   Flag "tap changer in operation " tap changes without a setting command
# a50   Flag "tap changer in operation" general part
# a51   error flag "tap change without indication "tap changer in operation"
# a52   error flag "tap change without setting command"
# a53   error flag "time for "tap changer in operation" exceeded"
# a54   error flag "tap change in the wrong direction"
# a55   error flag "wrong BCD-Code"
# a56   error flag "idle tap change"
# a60   timer for 3 s
# a61   timer for a "tap changer in operation" period
# a62   timer for "tap changer in operation"
# a63   timer for "tap changer in operation
# b50   actual tap position
# b51   expected tap position
# b52   difference of tap positions between actual and expected state
```

Count of tap changes per period with reset function

Task:
The following information should be available in the application menu:
„tap changes“ F1: "number of tap changes" and "time" since the last reset
„Reset“ F2: resets the counter for tap changes and the time

Solution:

```
# delete all menus
H 1='1,iff,menuappn *="",menuapp *=-1'
#generate menu
H 2='1,iff,menuappn 1=Stufungen,menuapp 1=1'
H 3='1,iff,menuappn 2=Reset,menuapp 2=2'
# reset counter after download of H-program
H 4='1,iff,zeit-,a01=.,a00=0'
# count of tap positions (a00)
H 5='RegRel- 1,RegRel- 2,|,iff,1,a00++=.'
# put out signal (executed in the application menu F1)
P 1='zeit-,a01,-,86400,/,dup,frac,86400,*,pick 2,int,a00,meld2 "Stufungen %!" "in %!Tagen %zz" 5'
# reset counter (executed in the application menu F1)
P 2='zeit-,a01=.,a00=0'
```

Variables:
a00: counter for tap changes
a01: flag for time after reset or download

Putting different regulators to the same tap position

Task:
To provide the condition for parallel operation the regulator B: has to be in the same tap position as regulator A: if regulator A: receives a signal on input A3. Both regulators have to be switched to manual mode.

Solution:

```
### If E3 --> then...
# ... flag for E3, set regulator A: and B: to manual operation
H 2='RegE- 3,iff,a1=1,RegAUTO=0,B:RegAUTO=0'
# ... register tap position of regulator A:
H 3='a1,iff,RegStufe-,a10=.'
# ... adjust tap position of transformer A: to B:
H 4='B:RegStufe-,a10,<,b02=.,B:RegStufe-,a10,>,b03=.'
H 5='b 02, b 03, |, b 00=.'
H 6='a1,b00,&,b01=.'
H 7='b01,iff,a30=0,b10=1,else,a 30=0'
H 8='htd,b01,b10,^,&,if,a++ 30 =.,a 30,RegTLAUFL-,1,+,>,iff,b10=1,a30=0'
H 9='b01,b10,&,if,b10=0,b02,if,B:RegHoeher* =1,endif,b03,if,B:RegTiefer* =1'
# ... flag for same tap position
H 10='b00,^,if,a2=1,else,a2=0'
# adjust tap positions, abort after 40 seconds
H 11='htd,b01,if,a++ 31=.,a31,40,>,if,a1=0,RegRel 5=1,eif,else,a31=0'
# application menu
H 12='1,iff,menuappn 1="Quittieren",menuapp 1=1'
# application menu ###
P 1='RegRel 5=0'
```

Variables:

a01: flag for input E3
a02: same tap position
a10: tap position of A: when E3 is active
a30: timer for setting commands
a31: timer - abort same tap position
b00: tap position too low or too high
b01: automatic setting operation active
b02: tap position too low
b03: tap position too high
b10: execute setting command

Analogue Inputs/Outputs

Analogue output in general

configuration of analogue output:

AnaModSel =	**AnaSSel =**
0: -10 V ... 0 ... 10 V	0: + / -
1: -20 mA ... 0 ... 20 mA	1: +
2: -5 mA ... 0 ... 5 mA	2: -
3: S0 ??	
4: 4 mA ... 20 mA	

Scaling of analogue output:
Ana = **AnaN** * **AnaFactor** + **AnaOffset**
however, AnaModSel=1: AnaN: –1 ... 1 => output: -20 mA ... 20 mA

Task 1:
The standardised system voltage U1N should be put out as a mA-signal at the analogue output Ana1. The output should be scaled as follows: U1N: 0 ... 120 V => Ana 1: 0 ... 20 mA

Solution:
```
# configuration of Ana 1
H 1='1,iff,AnaModSel 1 =1,AnaSSel 1 =1,AnaFACTOR 1 =20,AnaOFFSET 1 =0'
#scaling of Ana 1
H 2='RegU1N-,6,/,Ana 1=.'
```

Task 2:
The standardised system voltage U1N should be put out as a mA-signal at the analogue output Ana 1. The output should be scaled as follows: U1N: 0 ... 100 V => Ana 1: 4 ... 10 mA

Solution:
```
# configuration of AnaN 1 (0...100V => 4...10mA => 0,2...0,5AnaN)
H 1='1,iff,AnaModSel 1 =1,AnaSSel 1 =0 '
#scaling of AnaN 1
H 2='RegU1N-,333.333,/,0.2,+,AnaN 1=.'
```

Analogue output, kinked

Task:
The standardised system voltage U1N should be put out as a mA-signal at the analogue output Ana1. The output should be scaled as follows:
U1N = 0 ... 80 ... 120 V
Ana 1 = 0 ... 4 ... 20 mA

Solution:
```
H 1='1,iff,AnaModSel 1 =1,AnaSSel 1 =0,AnaFACTOR 1 =20,AnaOFFSET 1 =0'
H 2='RegU1N-,80,<,if,RegU1N-,20,/,Ana 1=.,else,RegU1N-,2.5,/,28,-,Ana 1=.'
```

2nd example with universal scaling:
for example:
UoN: 0...20 ...100 V => Ana1: 0...18 ...20 mA
in general:
UoN: 0...a00...a01 V => Ana1: 0...b00...b01 mA

Solution (for the example):
H 1='a00=20, a01=100, b00=18, b01=20'
H 2='b00,a00,/,a10=.'
H 3='b01,b00,-,a01,a00,-,/,a11=.'
H 4='a01,b00,*,a00,b01,*,-,a01,a00,-,/,b11=.'
H 5='RegU1N-, a00,<,if,RegU1N-, a10,*,Ana 1=.,else,RegU1N-, a11,*,b11,+,Ana 1=.'

Variables:
a00 kink of the input voltage
a01 rated value of the input voltage
b00 kink at the output signal
b01 rated value of the output signal
a10 ratio at the kink
a11 ratio above the kink
b11 calculated offset

Analogue input in general

Configuration of the analogue input:

AnaModSel =	**AnaSSel =**
0: -10 V ... 0 ... 10 V	0: + / -
1: -20 mA ... 0 ... 20 mA	1: +
2: -5 mA ... 0 ... 5 mA	2: -
3: S0 ??	
4: 4 mA ... 20 mA	

Scaling of the analogue input:
Ana = **AnaN** * **AnaFactor** + **AnaOffset**
however, AnaModSel=1: AnaN: –1 ... 1 => Eingang: -20 mA ... 20 mA

Task:
The analogue input Ana2 (0..20mA) should be used for the simulation of a transducer input (0..100V).

Solution:
configuration of Ana 2
H 1='1,iff,AnaModSel 2 =1,AnaSSel 2 =0,AnaFACTOR 2 =20,AnaOFFSET 2 =0'
transformer input U1 is replaced by RegMIVAL 1
H 2='1,iff,RegMiMap 1=-1'
scaling of Ana 2 / transformer input is set via RegMIVAL
H 3='Ana- 2,5,*,RegMiVal 1=.'
analogue input value is shown on the display
H 4='Ana- 2,meld "AnaEin 2 = %! mA"'

Analogue input for tap position

Task:
The analogue input Ana1 (for example 0..20mA) should be used to read the tap position (for example 1...19).

Solution:
Ana1 0..20 mA => tap position 1..19
H 1='1,iff,AnaModSel 1 =1,AnaSSel 1 =1'
H 2='Ana- 1,0.0555,/,1,+,INTR,RegStufe =.'