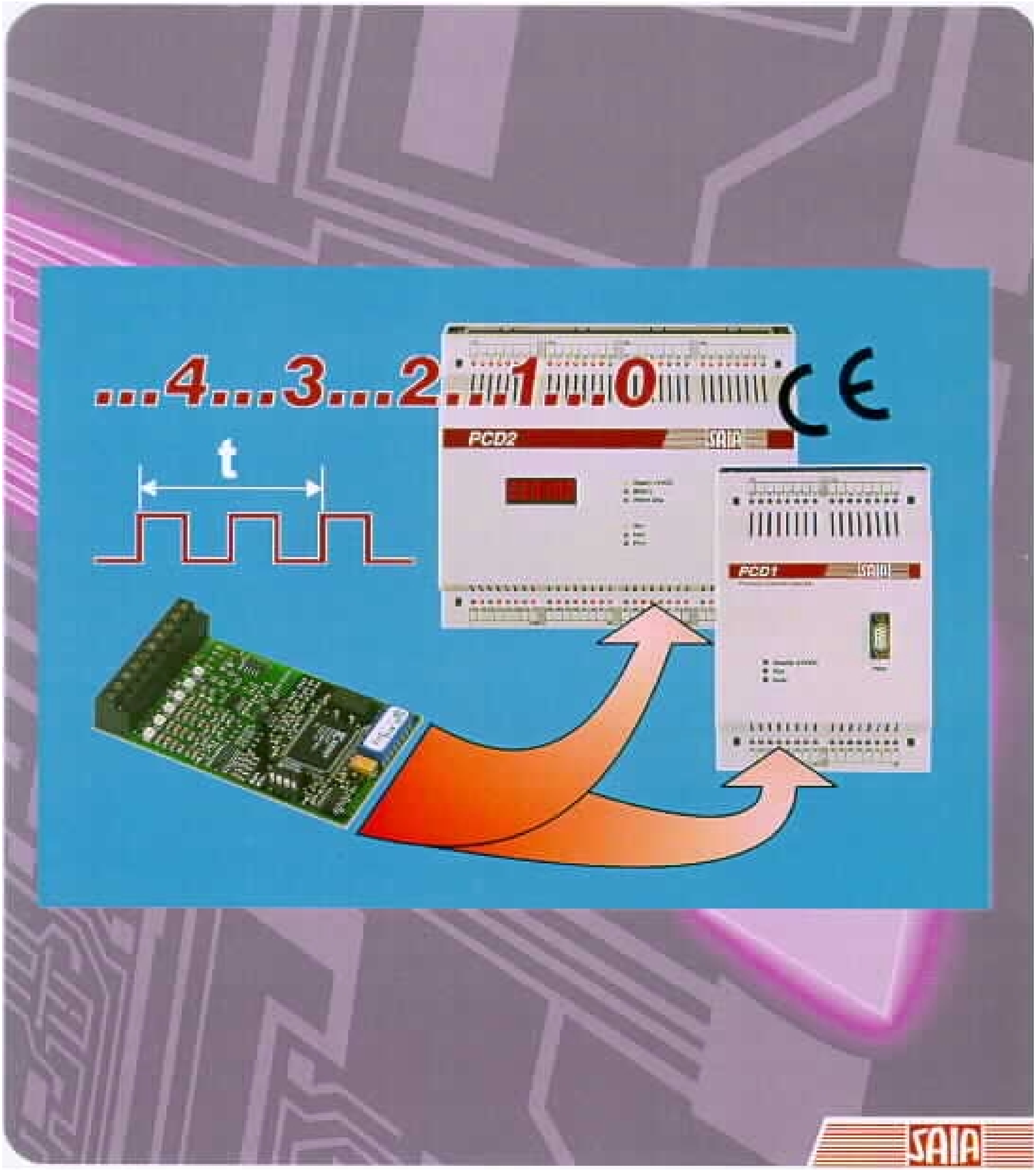
SAIA-Burgess Electronics

SWITCHES • MOTORS • CONTROLLERS

SAIA®PCD
Process Control Devices

PCD2.H110
Universal counting and measuring module



English edition 26/755 E2

SAIA-Burgess Electronics Ltd.
Bahnhofstrasse 18
CH-3280 Murten (Switzerland)
http;//www.saia-burgess.com

BA: Electronic Controllers   Telephone 026 / 672 72 72
Telefax 026 / 672 74 99

SAIA-Burgess Companies

Switzerland
SAIA-Burgess Electronics AG
Freiburgstrasse 33
CH-3280 Murten
☎ 026 672 77 77, Fax 026 670 19 83

France
SAIA-Burgess Electronics Sàrl.
10, Bld. Louise Michel
F-92230 Gennevilliers
☎ 01 46 88 07 70, Fax 01 46 88 07 99

Germany
SAIA-Burgess Electronics GmbH
Daimlerstrasse 1k
D-63303 Dreieich
☎ 06103 89 060, Fax 06103 89 06 66

Nederlands
SAIA-Burgess Electronics B.V.
Hanzeweg 12c
NL-2803 MC Gouda
☎ 0182 54 31 54, Fax 0182 54 31 51

Austria
SAIA-Burgess Electronics Ges.m.b.H.
Schallmooser Hauptstrasse 38
A-5020 Salzburg
☎ 0662 88 49 10, Fax 0662 88 49 10 11

Belgium
SAIA-Burgess Electronics Belgium
Avenue Roi Albert 1er, 50
B-1780 Wemmel
☎ 02 456 06 20, Fax 02 460 50 44

Italy
SAIA-Burgess Electronics S.r.l.
Via Cadamosto 3
I-20094 Corsico MI
☎ 02 48 69 21, Fax 02 48 60 06 92

Hungary
SAIA-Burgess Electronics Automation Kft.
Liget utca 1.
H-2040 Budaörs
☎ 23 501 170, Fax 23 501 180

Representatives

Great Britain
Canham Controls Ltd.
25 Fenlake Business Centre, Fengate
Peterborough PE1 5BQ UK
☎ 01733 89 44 89, Fax 01733 89 44 88

Portugal
INFOCONTROL Electronica e Automatismo LDA.
Praceta Cesário Verde, No 10 s/cv, Massamá
P-2745 Queluz
☎ 21 430 08 24, Fax 21 430 08 04

Denmark
Malthe Winje Automation AS
Håndværkerbyen 57 B
DK-2670 Greve
☎ 70 20 52 01, Fax 70 20 52 02

Spain
Tecnosistemas Medioambientales, S.L.
Poligono Industrial El Cabril, 9
E-28864 Ajalvir, Madrid
☎ 91 884 47 93, Fax 91 884 40 72

Norway
Malthe Winje Automasjon AS
Haukelivn 48
N-1415 Oppegård
☎ 66 99 61 00, Fax 66 99 61 01

Czech Republic
ICS Industrie Control Service, s.r.o.
Modranská 43
CZ-14700 Praha 4
☎ 2 44 06 22 79, Fax 2 44 46 08 57

Sweden
Malthe Winje Automation AB
Truckvägen 14A
S-194 52 Upplands Våsby
☎ 08 795 59 10, Fax 08 795 59 20

Poland
SABUR Ltd.
ul. Druzynowa 3A
PL-02-590 Warszawa
☎ 22 844 63 70, Fax 22 844 75 20

Suomi/ Finland
ENERGEL OY
Atomitie 1
FIN-00370 Helsinki
☎ 09 586 2066, Fax 09 586 2046

Australia
Siemens Building Technologies Pty. Ltd.
Landis & Staefa Division
411 Ferntree Gully Road
AUS-Mount Waverley, 3149 Victoria
☎ 3 9544 2322, Fax 3 9543 8106

Argentina
MURTEN S.r.l.
Av. del Libertador 184, 4° "A"
RA-1001 Buenos Aires
☎ 054 11 4312 0172, Fax 054 11 4312 0172

After sales service

USA
SAIA-Burgess Electronics Inc.
1335 Barclay Boulevard
Buffalo Grove, IL 60089, USA
☎ 847 215 96 00, Fax 847 215 96 06

Issue : 22.11.2000

SAIA® Process Control Devices

Universal counting and measuring module

PCD2.H110

Edition 26/755 E2 - 04.99

Subject to technical changes

Updates

Manual : PCD2.H110 - Universal counting and measuring module - Edition E2

Date	Chapter	Page	Description

Table of contents

		Page
1.	**Introduction**	
1.1	General	1-1
1.2	Function and application	1-2
1.3	Main characteristics	1-3
1.4	Typical areas of application	1-4
1.5	Programming	1-5
2.	**Technical data**	
2.1	Technical data of the hardware	2-1
2.2	Electrical specification	2-3
2.3	Function specific data	2-4
3.	**Presentation**	
4.	**Terminals, cable and meaning of the LED's**	
5.	**Brief introduction**	
5.1	Getting started with programming in IL	5-2
5.2	Getting started with programming in FUPLA	5-5
6.	**Programming**	
6.1	Programming in IL with FBs	6-2
6.1.1	The IL package (Installation of the FBs)	6-2
6.1.2	The individual FBs	6-5
6.2	Programming in FUPLA with FBoxes	
6.3	Programming in GRAFTEC with FBoxes	
7.	**Error handling and diagnostics**	
7.1	Definition errors checked by the assembler	7-1
7.2	Error handling in run	7-2

Page

8. PCD2.H110 for counting and motion control tasks

8.1 Block diagram of counter 8-1
8.2 Description of counter 8-2
8.3 Configuration of inputs 'EnableC', 'CCO', 'A', 'B' and the count mode 8-3

8.3.1 Configuration of the 'EnableC' input 8-3
8.3.2 Configuration of the 'CCO' output 8-4
8.3.3 Configuration of the inputs 'A' and 'B' 8-5
8.3.4 Configuration of count mode 8-6

8.4 Programmable counter functions 8-8
8.5 Programming principle 8-9
8.6 Application example no. 1: Counter in GRAFTEC 8-13
8.7 Application example no. 2: Motion control with incremental shaft encoder 8-18
8.8 Application example no.3: Measurement with counting 8-26

9. PCD2.H110 for measuring tasks

9.1 Frequency measurement 9-1

9.1.1 Block diagram 9-1
9.1.2 Description of frequency measurement 9-2
9.1.3 Configuration of frequency measurement 9-3
9.1.4 Programming principle 9-4
9.1.5 Combination of counting and frequency measuring 9-6

9.2 Period length measurement 9-7

9.2.1 Block diagram 9-7
9.2.2 Description of period length measurement 9-8
9.2.3 Configuration of period length measurement 9-9
9.2.4 Programming priciple 9-10

9.3 Pulse length measurement 9-13

9.3.1 Block diagram 9-13
9.3.2 Description of pulse length measurement 9-14
9.3.3 Configuration of pulse length measurement 9-15
9.3.4 Programming principle 9-16

Page

Appendix A: Summary of all software elements for programming in IL (FBs)

INIT	Initialisation FB	A-1
EXEC	Execution FB	A-3
LdCtPres	Load counter preset	A-4
LdRegPres	Load register preset	A-5
ModMsConf	Measure mode configuration	A-6
LdMsVal	Load measure value	A-7
RdCt	Read counter	A-9
RdMsImp	Read measure in impulsion	A-10
RdMsUnit	Read measure in unit	A-11
StartCt	Start counter	A-12
StartMs	Start measure	A-13
StopMs	Stop measure	A-14
RdIdent	Read module identification	A-15

Appendix B: Summary of all software elements for programming in FUPLA (FBoxes)

in preparation

Notes

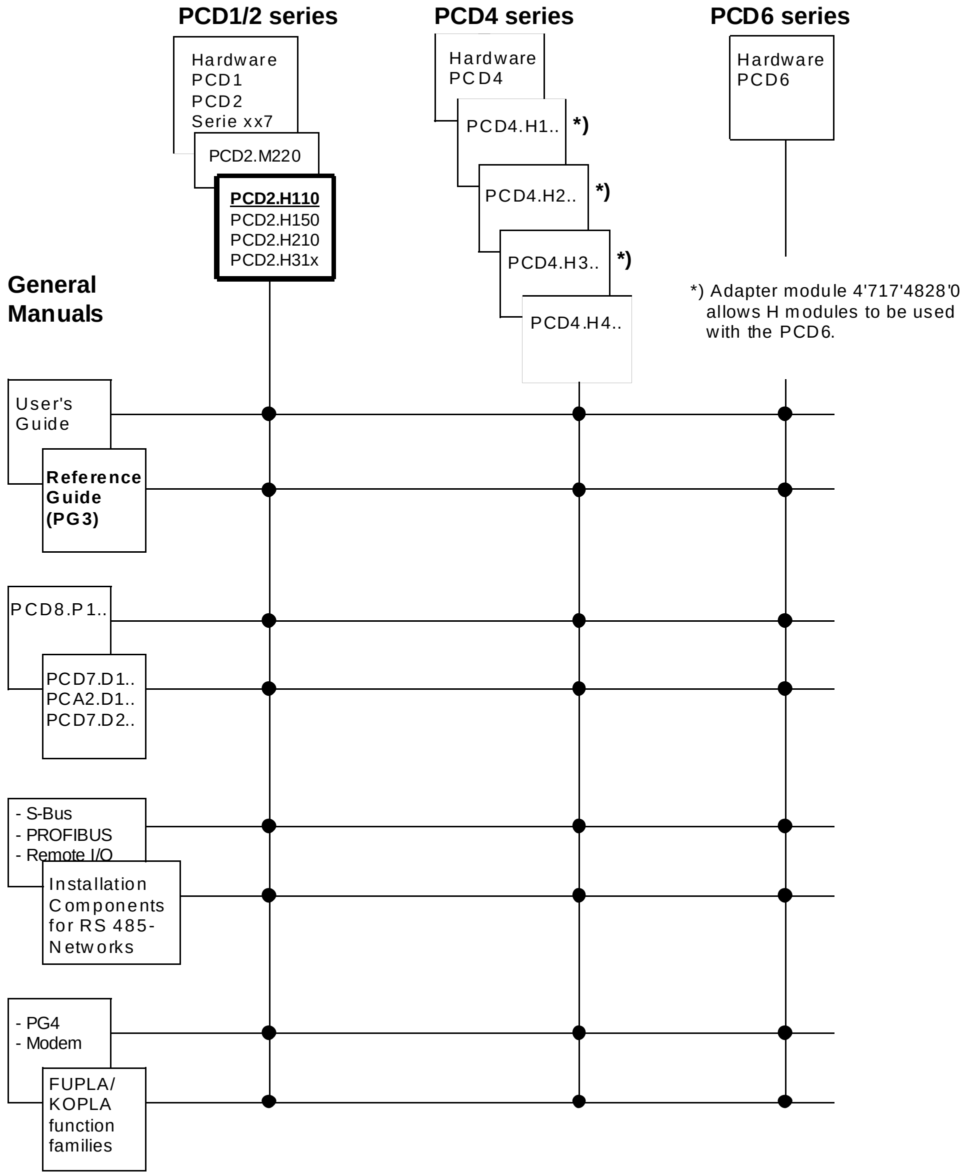
Please note:

A number of detailed manuals are available to aid installation and operation of the SAIA PCD. These are for use by technically qualified staff, who may also have successfully completed one of our workshops.

To obtain the best performance from your SAIA PCD, closely follow the guidelines for assembly, wiring, programming and commissioning given in these manuals. In this way, you will also become one of the many enthusiastic SAIA PCD users.

If you have any technical suggestions or recommendations for improvements to the manuals, please let us know. A form is provided on the last page of this manual for your comments.

Summary



Reliability and safety of electronic controllers

SAIA-Burgess Electronics Ltd. is a company which devotes the greatest care to the design, development and manufacture of its products:

- state-of-the-art technology
- compliance with standards
- ISO 9001 certification
- international approvals: e.g. Germanischer Lloyd, UL,
  Det Norske Veritas, CE mark ...
- choice of high-quality components
- quality control checks at various stages of production
- in-circuit tests
- run-in (burn-in at 85°C for 48h)

Despite every care, the excellent quality which results from this does have its limits. It is therefore necessary, for example, to reckon with the natural failure of components. For this reason SAIA-Burgess Electronics Ltd. provides a guarantee according to the "General terms and conditions of supply".

The plant engineer must in turn also contribute his share to the reliable operation of an installation. He is therefore responsible for ensuring that controller use conforms to the technical data and that no excessive stresses are placed on it, e.g. with regard to temperature ranges, overvoltages and noise fields or mechanical stresses.

In addition, the plant engineer is also responsible for ensuring that a faulty product in no case leads to personal injury or even death, nor to the damage or destruction of property. The relevant safety regulations should always be observed. Dangerous faults must be recognized by additional measures and any consequences prevented. For example, outputs which are important for safety should lead back to inputs and be monitored from software. Consistent use should be made of the diagnostic elements of the PCD, such as the watchdog, exception organization blocks (XOB) and test or diagnostic instructions.

If all these points are taken into consideration, the SAIA PCD will provide you with a modern, safe programmable controller to control, regulate and monitor your installation with reliability for many years.

1. Introduction

1.1 General

The standard equipment of SAIA® PCD process control devices already offers 1600 counting registers of 31 bits, although they can only capture frequencies up to approx. 20 Hz. Via the interrupt inputs, 1 kHz can be achieved and, with the ..H100 counting module, up to 20 kHz are possible.

The new ..H110 counting module not only extends the frequency range to 100 kHz, but also allows accurate measurement of frequencies up to 100 kHz and the length of periods or pulses up to one hour.

Its two counting inputs, A and B, enable it to recognize the rotatiorial direction of incremental shaft encoders, thus making the ..H110 module also capable of axis control, as long as regulated motion is not required from the module. For the regulated control of servo-motors with starting and braking ramps, we recommend the PCD2.H3.. motion control module.

The new ..H110 counting and measuring module uses a modern FPGA component (field programmable gate array), which can also be programmed for other specific OEM tasks by means of plug-in PROM. For this purpose, 4 inputs, 4 outputs and 2 x 4 LEDs are provided to the outside.

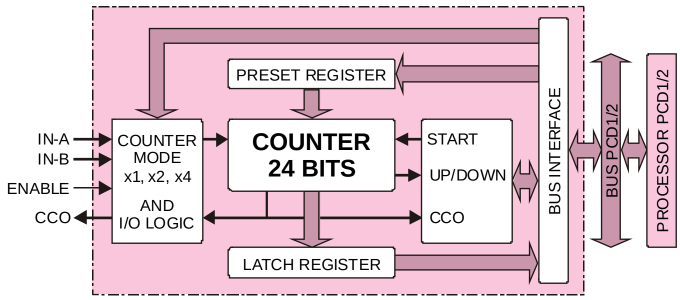
Function blocks and a comprehensive manual are available to the user for standard ..H110 functions.

1.2 Function and application

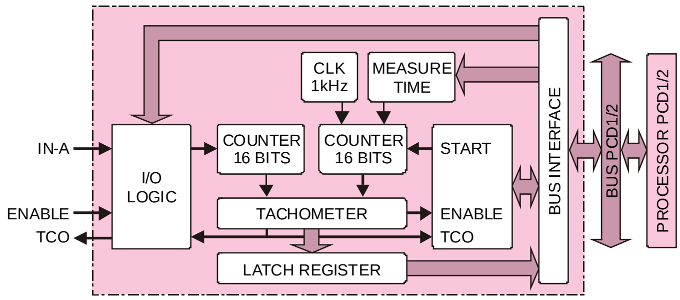
This low-cost module can be plugged into any I/O socket on a PCD1 or PCD2.

The module can be used in different modes:

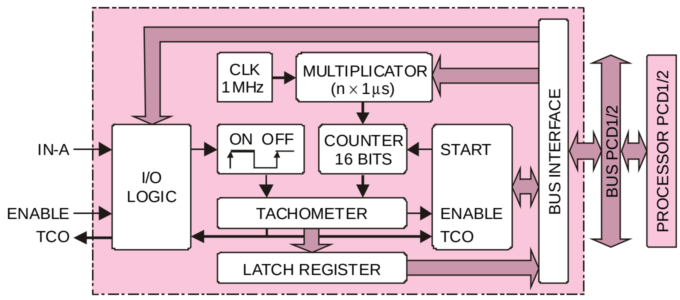
Block diagram as counting module



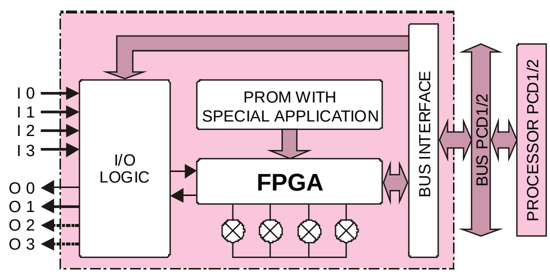
Block diagram for frequency measurement



Block diagram for measuring period or pulse length



Block diagram for special OEM versions



1.3 Main characteristics

- Up to 16 PCD2.H110 modules in parallel operation can be inserted in one PCD2, or up to 4 in one PCD1.
- Counting and measuring functions can be utilized simultaneously in the same module.

As a **counting module**

- Counting frequency up to 100 kHz
- Counting range 0 ... 16 777 215 (24 bit)
- Preset value 0... 16 777 215 (24 bit)
- Up or down counting to preset value
- 2 digital inputs A and B with recognition of rotational direction
- 1 direct counter output CCO
- Selectable counting modes

For frequency measurement

- Frequency range 500 Hz to 100 kHz
- Measurement range 0 ... 65 535 (16 bit)
- Accuracy ≥1‰. (depending on measurement time)
- The fast TCO output can be used at the end of a measurement, e. g. to trigger an interrupt

To measure period or pulse length

- Frequency range 0.27 mHz to 500 Hz
- Period or pulse lengths from 2 ms to 1h
- The fast TCO output can be used at the end of a measurement, e. g. to trigger an interrupt.

1.4 Typical areas of application

For small, basic PCD1 and PCD2 controllers, use of the new ..H-modules considerably extends the area of application. In particular, the ..H110 enables:

- fast pulse counting proportional to quantities (items, units of energy, etc.), placing little load on the basic CPU
- unregulated axis control of any drives with incremental shaft encoders
- quartz accuracy in determining velocity, rotary frequency, flow rate, etc.

Applications:

- Automatic handling- and assembly machines
- Pick and place functions
- Palletising equipment
- Automatic angle control, e.g. of cameras, headlamps, aerials, etc.
- Motion control of static axes (set-up)

1.5 Programming

Pre-programmed functional blocks make it possible to simply enter the parameters necessary for the desired count or measuring mode. These FBs (IL) and FBoxes (FUPLA) are used by the PG4 (Windows programming software). The present manual includes detailed descriptions of each function block, with associated practical examples. The use of the older programming tool "PG3" is possible only with special FBs)

lnitialization command

INIT	- Select the module number
	- Counter configuration
	- Counter preset
	- Register preset
	- Enable counter configuration
	- CCO configuration
	- IN-A configuration
	- IN-B configuration
	- Measuring configuration
	- Measuring value
	- Enable measuring configuration
	- TCO configuration

Execution command

EXEC	- Select the module number
	- command
	- Register for load value or result

The commands:

- LdCtPres	Load counter preset
- LdRegPres	Load register preset
- ModMsConf	Measure mode configuration
- LdMsVal	Load measure value
- RdCt	Read counter
- RdMsImp	Read measure in impulsion
- RdMsUnit	Read measure in unit (Hz or ms)
- StartCt	Start counter
- StartMs	Start measure
- StopMs	Stop measure
- RdIdent	Read module identification

Notes

2. Technical data

2.1 Technical data for the hardware

Digital inputs

Total	4
Nominal voltage:	24 V
Low range:	- 30 ... +5 V
High range:	+15 ... +30 V
Source mode only (positive logic)	
Input current (typical)	6.5 mA
Inputfilter	150 kHz
Switching type	galvanically connected

Digital outputs:

Total	2
Current range	5 to 500 mA (Leakage current max.: 1 mA) (Load resistance min.: 48 Ω in the voltage range from 5 to 24 V)
Short-circuit protection	no
Frequency	≤ 100 kHz
Voltage range	5 to 32 V (external supply)
Switching type	galvanically connected
Potential drop (typically)	< 0.5 V by 500 mA
Output delay	less than 1 µs with inductive loads, the delay is longer due to the protective diode

Power supply

Internal supply from PCD1/2 bus	5 VDC, max. 90 mA
External by user for all outputs	24 VDC (10 ... 32 VDC), max. 2 A smoothed ripple max. 10%

Operating conditions

Ambient temperature	operation: 0 ... +50°C without forced ventilation storage: -20 ... +85°C
Interference immunity	CE mark according to EN 50081-1 and EN 50082-2

LED displays

Total		6
	LED 0:	Status of input "A"
	LED 1:	Status of input "B"
	LED 2:	Status of input "EnableC"
	LED 3:	Status of input "EnableM"
	LED 4:	Status of "CCO" output
	LED 5:	Status of "TCO" output

Programming

Based on PCD user program (PG4) and pre-programmed functional blocks.

Ordering details

PCD2.H110	Universal counting and measuring module
PCD9.H11E	Software library with function blocks

2.2 Electrical specification

Internal power consumption

+5V:	max. 90 mA
Uext:	0 ... 10 mA (without load current)

External power supply

Terminals +/-: 10 ... 32 VDC smoothed, residial ripple max. 10%
TVS diode 39 V ±10%
max. 2 A for outputs not protected against wrong polarity!

Digital inputs

4 digital inputs (E0 ... E3)
(see chapter 2.1)

Digital outputs

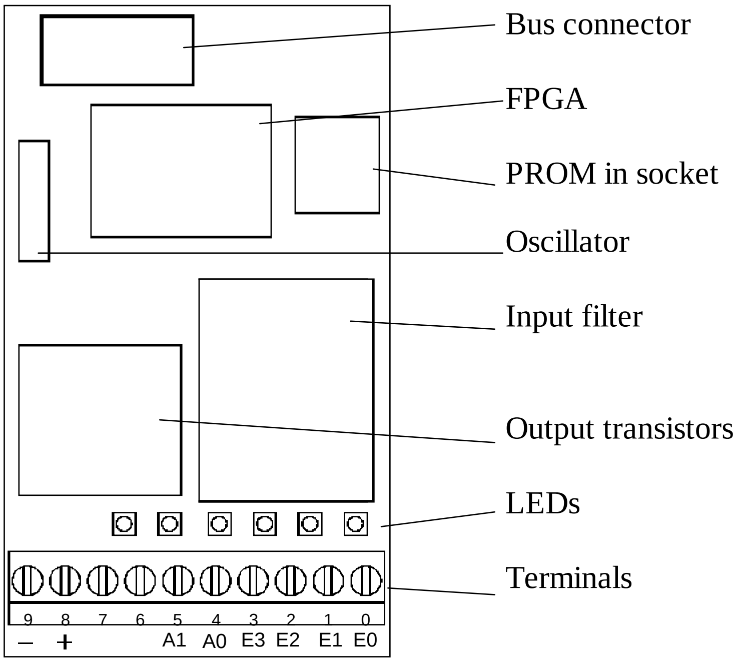
2 digital outputs (A0 and A1)
(see chapter 2.1)

2.3 Function specific data

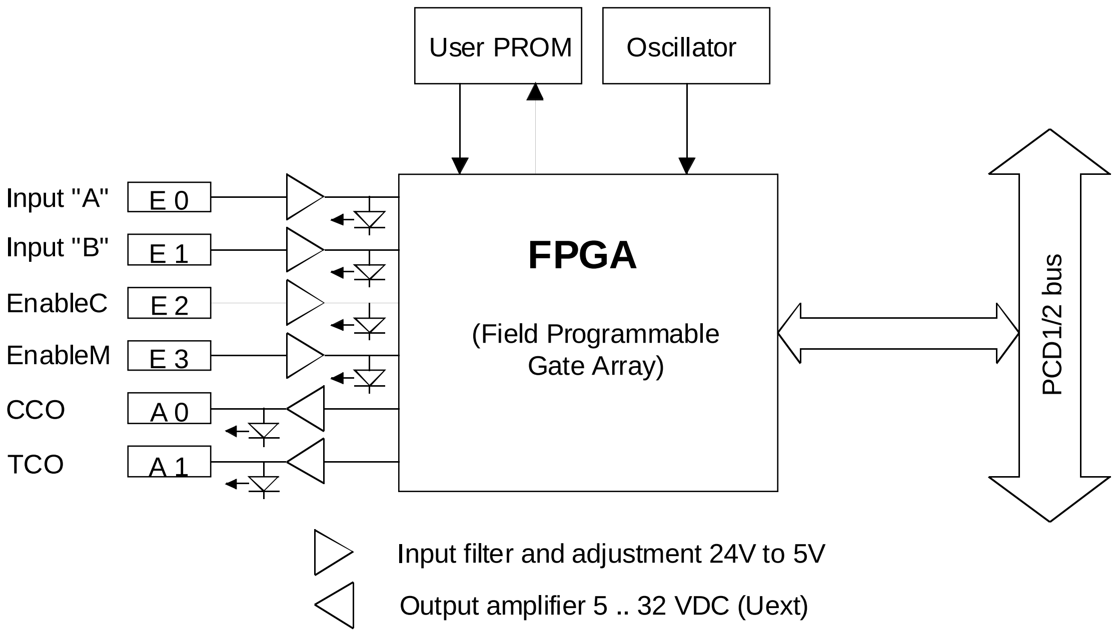
Number of systems	1
Counting range	0 ... 16 777 215 (24 bit)
Counting frequency	up to 100 kHz
Data protection	All data in this module is volatile (non-volatile registers in the PCD can be used)

3. Presentation

Equipped module



Block diagram

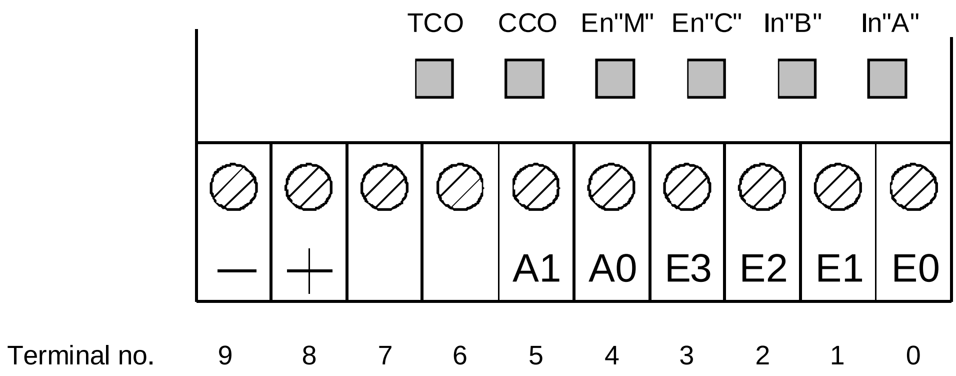


Notes

4. Terminals, cable and meaning of the LED's

Screw terminals

This picture shows the text on the print. The I/O connector is standard from 0 ... 9 (from right to left)



See also the block diagrams in chapter 1.2.

Inputs :

Total		4
Terminal 0 =	E0 :	Input "A" for counting and as measuring input
Terminal 1 =	E1 :	Input "B" for counting only
Terminal 2 =	E2 :	Input "Enable C" by use as counting module
Terminal 3 =	E3 :	Input "Enable M" by use as measuring module

Outputs :

Total		2
Terminal 4 =	A0 :	Output "CCO" for counter
Terminal 5 =	A1 :	Output "TCO" for measuring functions

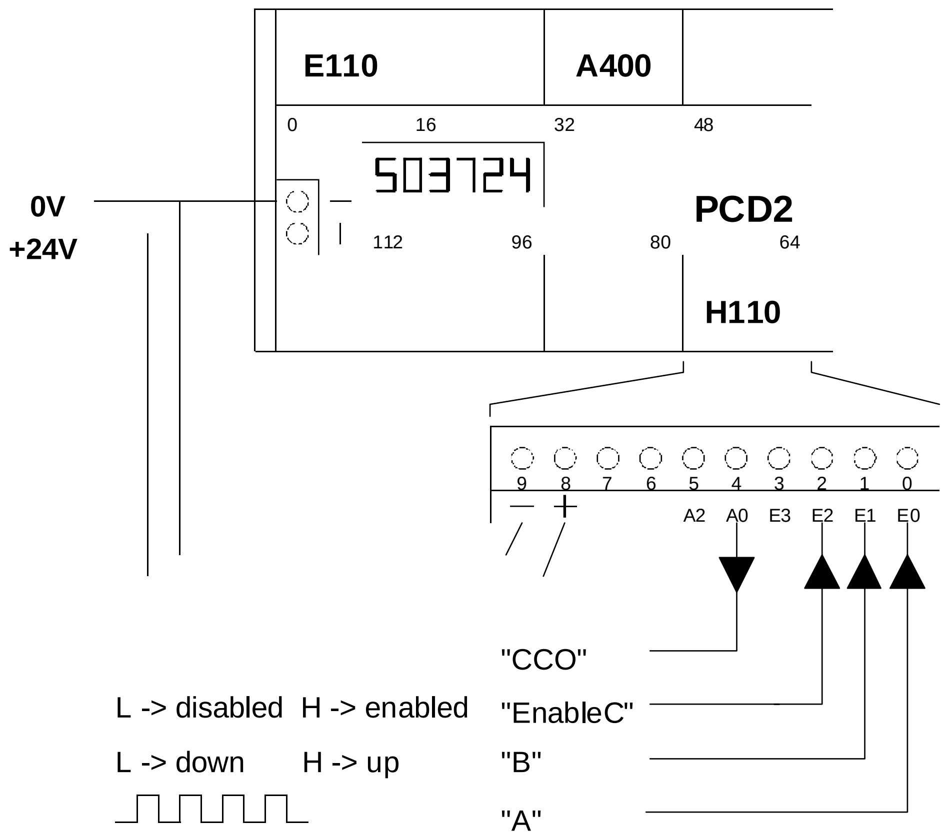
Supply:

Terminal 8 =	+	+ 24 VDC
Terminal 9 =	–	GND

LED displays

Total		6
	LED 0:	Status of input "A"
	LED 1:	Status of input "B"
	LED 2:	Status of input "EnableC"
	LED 3:	Status of input "EnableM"
	LED 4:	Status of outputs "CCO"
	LED 5:	Status of output "TCO"

5. Brief introduction



Minimum arrangement to operate a PCD2.H110 as up/down counter.

The individual elements are:

PCD2 (or PCD1) equipped with at least	1 PCD2.H110 (1 PCD2.F510/530) (1 PCD2.E110) (1 PCD2.A400)

No-bounce impulse contact

Supply device 24 VDC smoothed

An incremental shaft encoder (24V) can be connected to the input terminals "A" and "B". Depending from the rotation direction, pulses are counted automatically "up" or "down".

5.1 Getting started with programming in IL

The following minimal program is suggested to commission a PCD2.H110 module as up/down counter in the easiest way.

A proper user program should not contain wait loops. However, for the purposes of demonstrating the main instructions which drive a PCD2.H110, this example has been constructed with wait loops. In practice, a GRAFTEC or for the future a FUPLA structure should always be chosen for programs of this type.

Example task: After power-up of PCD, the counter of the counting module is to set on 1000. Signals on input "A" should increment (+) the counter, signals on input "B" should decrement (-) the counter. The actual counter value is to show on the display module or in the debugger.

Individual parameters and base address settings should be as in section 6.1 below. The user program can then take the following form:

(Detailed and well structured programs are shown in the individual chapters and are also located on the diskette PCD9.H11E).

The FBs (IL for PG4) are located on the diskette PCD9.H11E. To install the FBs on the PC follow the indications on the README.TXT on this diskette.

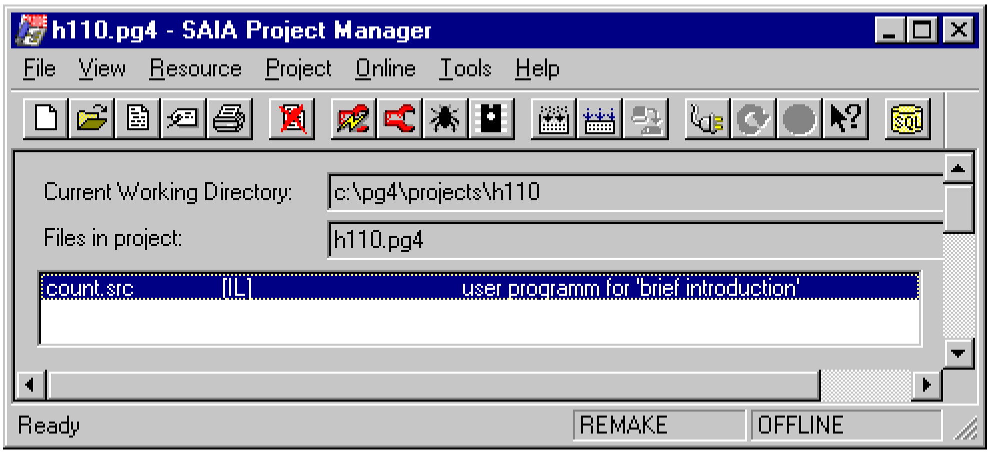
The number of modules (1) and the address of the PCD2.H110 module (64) is to indicate in the file 2D2H110_B.MBA:

```
NbrModules   EQU   1     ; No. of H110 modules used (0...16)

BA_1         EQU   64    ; Base address of module 1
```

This file (2D2H110_B.MBA) must be located in project directory of this example, i.e. the file is to copy manually from the diskette to the actual project directory.

```
$include d2h110_b.equ
$group h110

xob       16

ld      r 999      ; PCD register with
          1000     ;  start value for counter
ld      r 998      ; PCD register with value
          0        ;  for compare register

cfb       init     ; initialisation H110
        k 1        ;  par 1    module number
          0        ;  par 2    count mode "x1"
        r 999      ;  par 3    start value for counter
        r 998      ;  par 4    value for compare register
          0        ;  par 5    EnableC "static-normal"
          0        ;  par 6    CCO "static-normal"
          0        ;  par 7    input A "normal"
          0        ;  par 8    input B "normal"
          0        ;  par 9    not used for counter
          0        ;  par 10   not used for counter
          0        ;  par 11   not used for counter
          0        ;  par 12   not used for counter

cfb       exec     ; start counter
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

exob
; ------------------------------------------

cob       0        ; real user program
          0

cfb       exec     ; read counter
       k  1        ;  module number
          RdCt     ;  command: read counter
       r  777      ;  register 777

dsp    r  777      ; display on display module

$endgroup

ecob
```

The program is named "count.src" and is located in the project "h110".



With 'Project' - 'Build' the program will be assembled, linked, downloaded in the PCD and put in 'Run'. The actual counter value will immediately be displayed on the display module. If no display module is disposable the value of the counter in the PCD register R 777 can be viewed in the Debugger: (Display Register 777 <Space> Refresh <CR>).

5.2 Getting started with programming in FUPLA

in preparation

Notes

6. Programming

The standard PG4 programming tools are used to create a user program to manage the PCD2.H.. counting and motion control modules. (To use the older programming tool "PG3", special FBs are available).

Programming is either in IL (instruction list) with FBs (function blocks) or in FUPLA with FBoxes (in preparation). The FBs can be obtained on diskette using reference PCD9.H11E.

Since motion control tasks always concern sequential processes, it is preferable if user programs are written in GRAFTEC, while individual steps and transitions can be edited either in IL with FBs or in FUPLA with FBoxes. User programs, however, can also be written purely in BLOCTEC or FUPLA.

6.1 Programming in IL with FBs

6.1.1 The IL package (Installation of the FBs)

The ordering code of the diskette is PCD9.H11E. The diskette contains the following directories:

- APPSDIR : contains all helps
- FB : contains the .SRC and .EQU files of the H110
- FBOX : contains the FBoxes for the H110
- PG3_FB : contains all files for the FBs of PG3
- PG4_FB : contains examples and the .MBA file
- Readme : contains general information

The package is provided for the SAIA PG4 from version V2.0.70. For all other versions of PG4 consult the 'Readme' file. (The package also contains FBs for use with the older PG3, see 'Readme').

FBoxes for FUPLA are not yet available.

Installation of package for PG4

The simplest method of installation is with the PG4 program 'Setup Extra Files':

Insert the diskette PCD9.H11E into drive A:
<Start> <Programs> <SAIA PG4> <Setup Extra Files>. The FBs and 'Help' file are installed on the hard disk in the 'PG4' directory.

The following files are installed:

D2H110_B.SRC	FB source code	read-only file
D2H110_B.EQU	FB definitions	read-only file

These 2 files are copied from the diskette and located in the PG4 directory ...\PG4\FB.

FB_LIB.HLP	FB library data
D2H110_B.HLP	FB help

These 2 files are located in the directory A:\APPSDIR and are copied into the PG4 directory ...\PG4.

The file **D2H110_B.MBA** (module base addresses) must be copied **by hand** from the diskette into the current project directory.

For the user the file **'D2H110_B.MBA'** is important and is shown here:

File: **D2H110_b.mba** (mba = module base address)

```
;
; This file can be modified by the user
;
; Base addresses defined by the user
; ---------------------------------
$group H110
NbrModules      EQU     1       ; No. of H110 modules used (0...16)

;
; Module base addresses (only the used modules must be defined)

BA_1            EQU    32       ;Base address of module 1
BA_2            EQU     0       ;Base address of module 2
BA_3            EQU     0       ;Base address of module 3
BA_4            EQU     0       ;Base address of module 4
BA_5            EQU     0       ;Base address of module 5
BA_6            EQU     0       ;Base address of module 6
BA_7            EQU     0       ;Base address of module 7
BA_8            EQU     0       ;Base address of module 8
BA_9            EQU     0       ;Base address of module 9
BA_10           EQU     0       ;Base address of module 10
BA_11           EQU     0       ;Base address of module 11
BA_12           EQU     0       ;Base address of module 12
BA_13           EQU     0       ;Base address of module 13
BA_14           EQU     0       ;Base address of module 14
BA_15           EQU     0       ;Base address of module 15
BA_16           EQU     0       ;Base address of module 16
$endgroup
```

The number of PCD2.H110 modules must be specified and the hardware base addresses of PCD2.H110 modules used should then be entered.

Since the '.mba' file does not appear in Project Manager's file list, a text editor (e.g. SEDIT32) must be used for modification.

The modules are to be numbered successively beginning with 'BA_1'. For example, if three H110 modules are used in a project, 'BA_1', 'BA_2' and 'BA_3' should be used. The places in the PCD can be freely chosen. Example:

```
NbrModules      EQU     3       ; No. of H110 modules used (0...16)
;
; Module base addresses (only the used modules must be defined)

BA_1            EQU    64       ;Base address of module 1
BA_2            EQU   208       ;Base address of module 2
BA_3            EQU   112       ;Base address of module 3
BA_4            EQU     0       ;Base address of module 4
BA_5            EQU     0       ;Base address of module 5
```

The base addresses of registers, flags and FBs are assigned automatically and can be viewed in the resource list under 'View' - 'Resource List'.

The project to be created should have project name "TEST-H11" and the actual user program module should be entitled "count-01.sfc". The files are arranged like this:

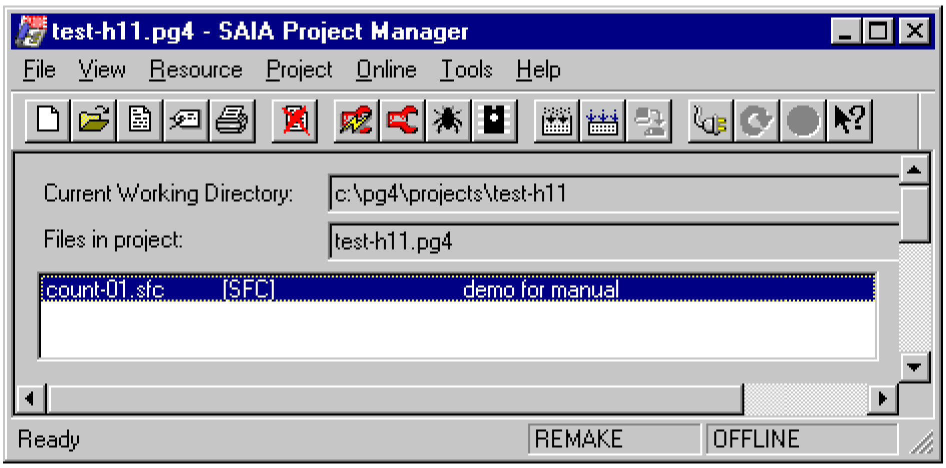
```
C:\PG4 \FB          \D2H110_b.equ
                    \D2H110_b.src
       \...
       \FBOX        \...
       \GALEP3      \...
       \PROJECTS    \FUP_E                  (Demo example PG4)
                    \GRAF_E                 (Demo example PG4)
                    \TEST-H11    \D2H110_b.mba
                                 \count-01.sfc
       \...
       \D2H110_b.hlp
```

The user program for the H110 part is structured as follows:

```
$include D2H110_b.equ
$group H110

XOB      16


PCD code


ecob
$endgroup
```

If the program is written in GRAFTEC, the assembler directives "$include" and "$group" are placed in the first step (ST), normally the initial step (IST). "$endgroup" comes at the end of the last transition (TR).

If everything has been correctly installed, the user program edited and all parameters defined, the program can be processed and downloaded to the PCD with the 'Project' - 'Build' command.



6.1.2 The individual FBs

The complete package consists of two main FBs with parameters:

- INIT Initialisation FB with 12 parameters
- EXEC Execution FB with 3 parameters

The call of the FB 'INIT' is as follows (as example):

```
CFB      init
    k    1          ;  par. 1:   Module number              (k 1 - k 16)
         0          ;  par. 2:   Counter configuration           (0 - 4)
    r    100        ;  par. 3:   Counter preset         (0 - 16777215)
    r    101        ;  par. 4:   Register preset        (0 - 16777215)
         0          ;  par. 5:   Enable counter config.          (0 - 3)
         0          ;  par. 6:   CCO configuration               (0 - 3)
         0          ;  par. 7:   Input A configuration           (0 - 3)
         0          ;  par. 8:   Input B configuration            (0, 1)
         0          ;  par. 9:   Measure configuration           (0 - 5)
         0          ;  par. 10:  Measure value               (0 - 65535)
         0          ;  par. 11:  Enable measure config           (0 - 3)
         0          ;  par. 12:  TCO configuration               (0 - 3)
```

The call of the FB 'EXEC' for some examples is as follow:

```
CFB      exec
    k    1          ;  par. 1:   Module number              (k 1 – k 16)
         RdCt       ;  par. 2:   Command: read counter
    r    555        ;  par. 3:   Target register

CFB      exec
    k    1          ;  par. 1:   Module number              (k 1 - k 16 )
         LdCtPres   ;  par. 2:   Command: load counter
    r    0          ;  par. 3:   not used (empty register)
```

Three parameters must always be defined, even if only two are required for a function. The third parameter can be defined as 'rNotUsed' or as any register.

A list of all command follows on the next page.

Commands (functions) for FB 'Exec' (parameter 2):

LdCtPres	; Load counter preset
LdRegPres	; Load register preset
ModMsConf	; Measure mode configuration
LdMsVal	; Load measure value
RdCt	; Read counter
RdMsImp	; Read measure in impulsion
RdMsUnit	; Read measure in unit (Hz or ms)
StartCt	; Start counter
StartMs	; Start measure
StopMs	; Stop measure
RdIdent	; Read module identification

Readable elements for the user

Element	**Address**	**Description**
Cstart_x	8 + BA_x	If counter runs, element = H. ('EnableC' = H and 'Start-Counter' executed)
CCO_x	9 + BA_x	Image of the output 'CCO' (in dynamic modes not visible, too short)
UpDown_x	10 + BA_x	If counter counts up, element = H.
EnableM_x	12 + BA_x	Image of the input 'EnableM'
TCO_x	13 + BA_x	Image of the output 'TCO' (in dynamic modes not visible, too short)
OverFlow_x	14 + BA_x	If overflow in measuring mode, element = H.
EndMes_x	15 + BA_x	If measuring is ended, element = H. The element is reset at the end of a read command (RdMsImp or RdMsUnit)
fPar_Err	see resourcelist	H, if parameter in the FB 'Init' is outside the range
fError_x	see resource list	H, by a division by zero in the conversion of a measuring result
rDiag	see resource list	see chapter 7: Error handling and diagnosis.

'_x' corresponds to the module number

The effective addresses of the elements are shown in the resource list (for debug purposes).

6.2 Programming in FUPLA with FBoxes

in preparation

Notes

6.3 Programming in GRAFTEC with FBoxes

in preparation

Notes

7. Error handling and diagnosis

7.1 Definition errors checked by the assembler

The following definition errors in file D2H110_b.MBA are checked during assembly:

- If the number of modules (NbrModules) is lower than 1, no code is assembled and the following warning is displayed in the 'Make' window:

"Remark : No H110 used (NbrModules = 0 in D2H110_B.MBA)"

- If the number of modules (NbrModules) is greater than 16, no code is assembled and the following error message is displayed in the 'Make' window:

"Error : more than 16 Modules H110 defined (NbrModules = 0...16)"

- If an incorrect instruction code is used for FB 'Exec' (e.g. RdIdenti instead of RdIdent), the assembler reports an error:

"Symbol not defined 'H110.RdIdenti'"
(generating the printout 'H110' from $group h110)

- If the definition $group H110 is absent, then for each instruction and each register/flag used in the program the assembler reports:

"Symbol not defined"

7.2 Error handling in run

Wrong parameter

In FB 'Exec' only the command code is checked. Parameter 1 (module no.) and parameter 3 (source/destination register) are not checked, to avoid making execution times longer.

In FBs 'Init' and 'Home' the values of all parameters are checked to verify whether they fall within the permitted range (e.g. counter mode = 0, 1, 2, 3, 4). If a parameter is outside a range, it is set to the minimum value, the error flag 'fPar_Err' is set and diagnostic register 'rDiag' is loaded with the corresponding error code.

Flag 'fPar_Err' is not reset inside the FB. This should take place in XOB 16 or in the initial step (IST).

The error code is composed as follows:

```
rDiag    bit 31 . . . . . . . 24 23 . . . . . . . 16 15 . . . . . . .  8 7 . . . . . . . 0
                 \  Reserve  /      \  FB No.  /     \  Par. No.  /   \ Mod. No./
                                    (Init    =  FB 1)
                                    (Exec  =  FB 2)
```

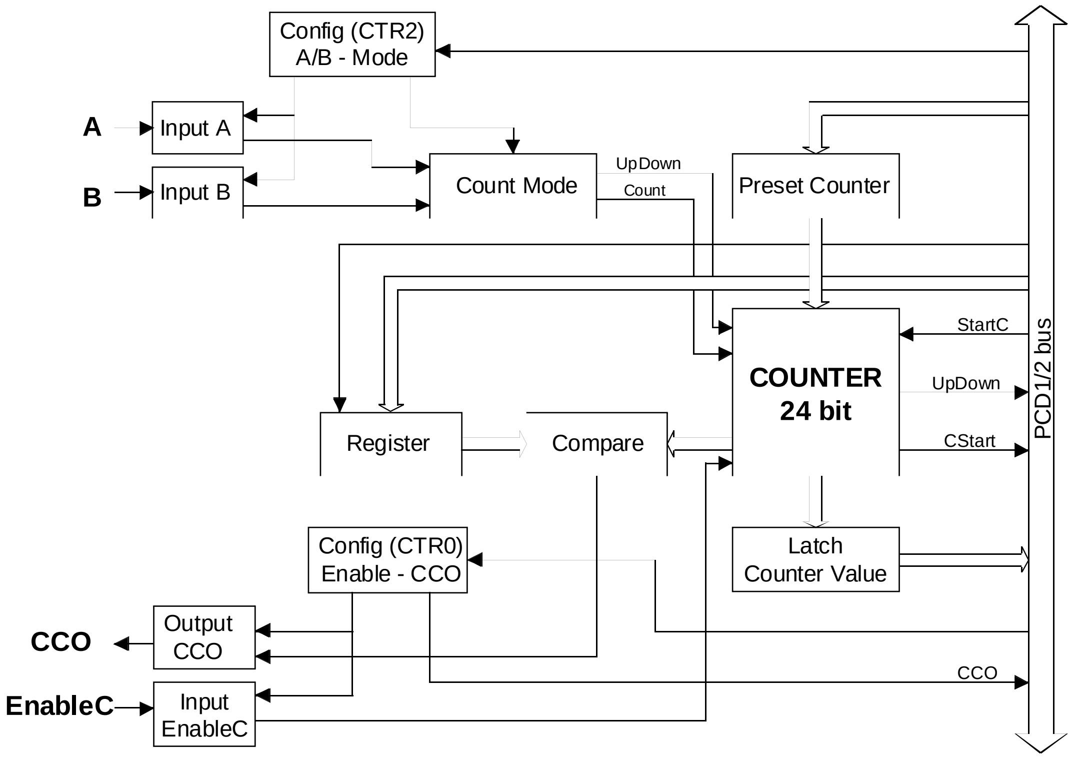
Example : If the configuration of the CCO output (parameter 6) in FB 'Init' of module 2 is incorrect, register 'rDiag' is set to 00 01 06 02 hex.

The diagnostic register is overwritten with each incorrect parameter and always contains the last error. It should therefore be evaluated as soon as flag 'fPar_Err' signals an error. The absolute addresses of 'rDiag' and 'fPar_Err' can be viewed in file 'project.MAP'. This can be useful during commissioning with the debugger to locate an error :

- Run until flag 'fPar_Err' = H
- Display register 'rDiag' hex
- Delete flag 'fPar_Err'

8. PCD2.H110 for counting and motion control tasks

8.1 Block diagram of counter



8.2 Description of counter

The core of the circuit is a 24 bit counter. This counter is loaded from the preset counter. The preset counter itself is loaded by the user program from a PCD register as three 8-bit values. The 'StartCt' command loads the preset counter value into the counter and starts it.

The register is loaded in a similar way. The user program loads the 24-bit value from a PCD register into the H110's register as three 8-bit values. When inputs 'A' and 'B', the 'CCO' direct counter output and the 'EnableC' input have all been configured (this is described in a later section), and if register and counter match, the 'CCO' can be switched according to its configuration so that the process and the user program can be controlled.

The status of 'CCO' (CCO_x), 'EnableC' (CStart_x) and 'UpDown' (UpDown_x) can be read by the user program.

8.3 Configuration of inputs 'EnableC', 'CCO', 'A', 'B' and the count mode

8.3.1 Configuration of the 'EnableC' input

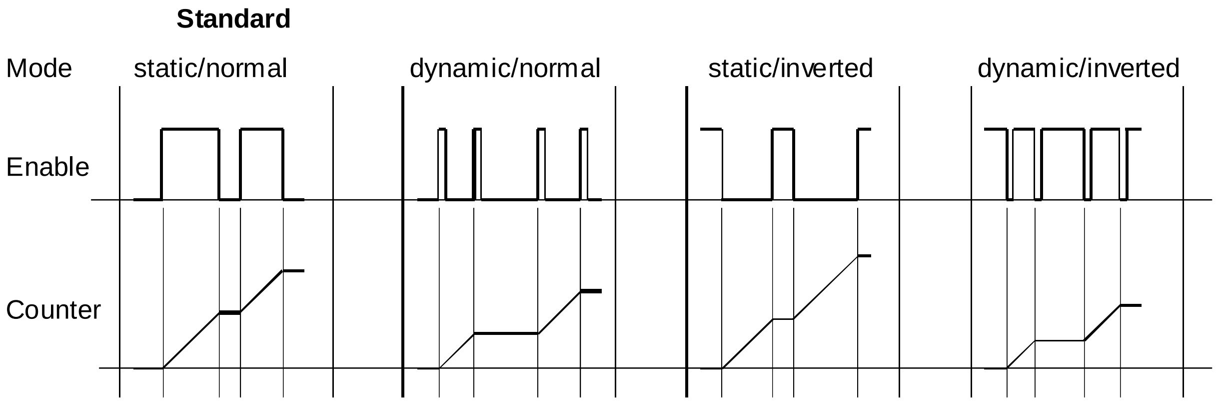
Standard: "static / normal" Init parameter 5 = 0
While the 'EnableC' input is 'H', counting is allowed.
While the 'EnableC' input is 'L', counting is stopped.

Additional possibilities:

"static / inverted" Init parameter 5 = 1
While the 'EnableC' input is 'L', counting is allowed.
While the 'EnableC' input is 'H', counting is stopped.

"dynamic / normal" Init parameter 5 = 2
The 'EnableC' input is 'L'. The first positive edge (H) switches 'EnableC' on, the next switches it off again, etc.

"dynamic / inverted" Init parameter 5 = 3
The 'EnableC' input is 'H'. The first negative edge (L) switches 'EnableC' on, the next switches it off again, etc.



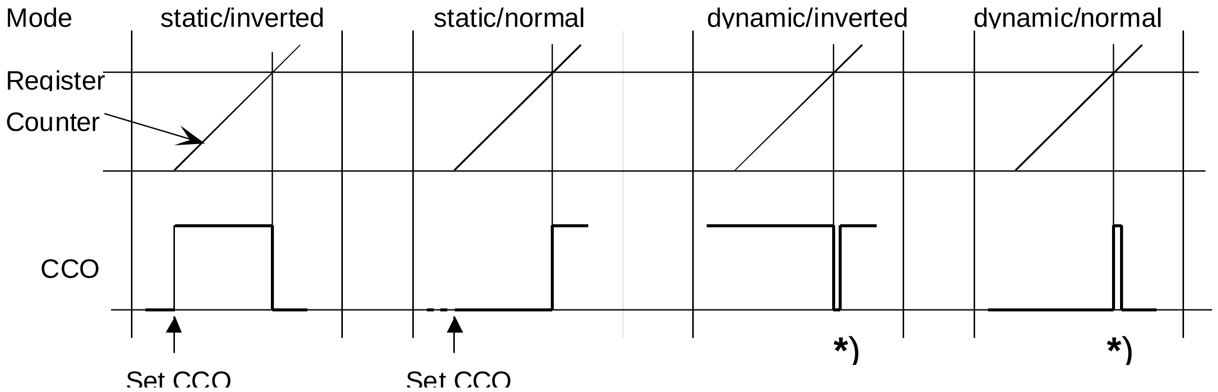
8.3.2 Configuration of the 'CCO' output

"static / normal" Init parameter 6 = 0
The 'CCO' is activated by the user program and becomes or remains L. If the register and counter are equal, the 'CCO' is switched 'H' and remains 'H' until a new activate command is received from the user program.

"static / inverted" Init parameter 6 = 1
The 'CCO' is switched on by the user program and becomes 'H'. If the register and counter are equal, the 'CCO' is switched 'L' and remains 'L' until a new command to switch on is received from the user program.

"dynamic / normal" Init parameter 6 =2
The 'CCO' is activated by the user program and becomes or remains 'L'. If the register and counter are equal, the 'CCO' becomes 'H' *) for 25 .. 100 µs. At each subsequent agreement of register and counter, the behaviour of the 'CCO' is repeated, without any new instructions from the user program.

"dynamic / inverted" Init parameter 6 = 3
The 'CCO' is switched on by the user program and becomes 'H. If the register and counter are equal, the 'CCO' becomes 'L' *) for 25 .. 100 µs. At each subsequent agreement of register and counter, the behaviour of the 'CCO' is repeated, without any new instructions from the user program.



*) Evaluation of this short pulse is via the PCD1/2 interrupt inputs and XOB 20 or 25.

8.3.3 The configuration of the inputs 'A' and 'B'

Inputs 'A' and 'B' can be inverted individually.

	Counter	**Measure**	**Parameter 7**
Input 'A'	normal	normal	0
Input 'A'	inverted	normal	1
Input 'A'	normal	inverted	2
Input 'A'	inverted	inverted	3

	Counter		**Parameter 8**
Input 'B'	normal	-	0
Input 'B'	inverted	-	1

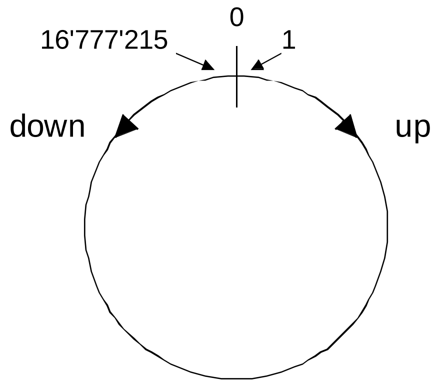
If a module is used as counter, inversion of a single input ('A' or 'B') results in the physical reversal of a drive's rotational direction.

8.3.4 Configuration of count mode

Mode 'x1' Init parameter 2 = 0 or 1

Use for simple counting tasks (without incremental shaft encoder):

- The signals to be counted are at input 'A'
- If input 'B' is 'L', count direction is downwards, if parameter 8 = 0.
  If input 'B' is 'H', count direction is upwards, if parameter 8 = 0.
  If parameter 8 = 1, counting direction is inverted.

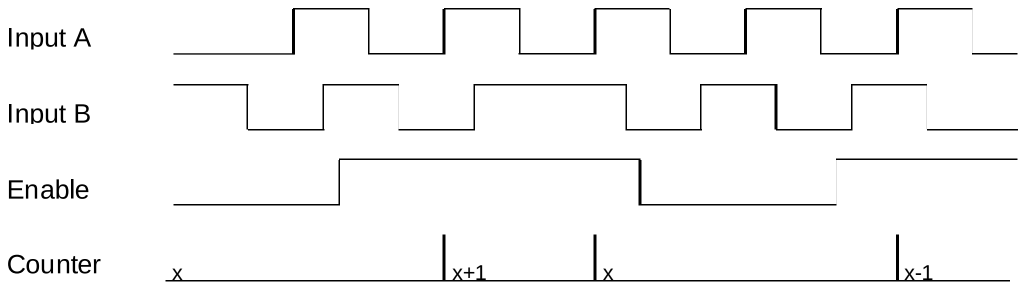


Counting range is 0 ... 16'777'215 (0 ... 2^{24} - 1)

If up-counting from 0 → 0, 1, 2 ...
If down-counting from 0 → 0, 16'777'215, 16'777'214 ...

There are no negative values and no overflow

Mode 'x1' Init parameter 2 = 0 or 1



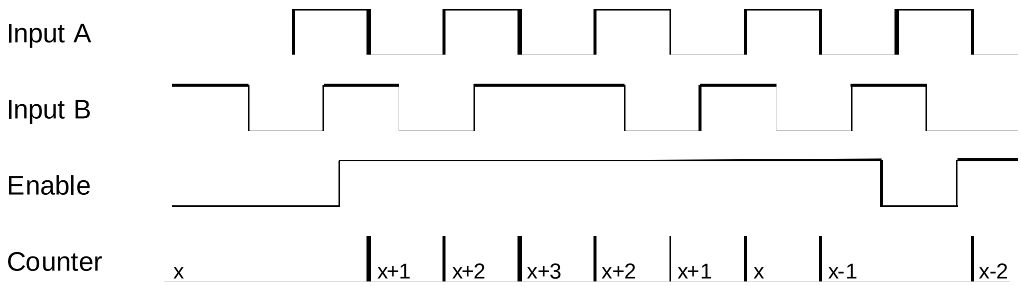
Only the rising edge of signal A is evaluated. Signal B in quadrature (phase shifted by 90°) defines the count direction.

Important: In mode 'x1' incremental shaft encoders should **<u>not be used</u>** because counting may be incorrect in some situations.

If 2-phase incremental shaft encoders are used, the following modes are available:

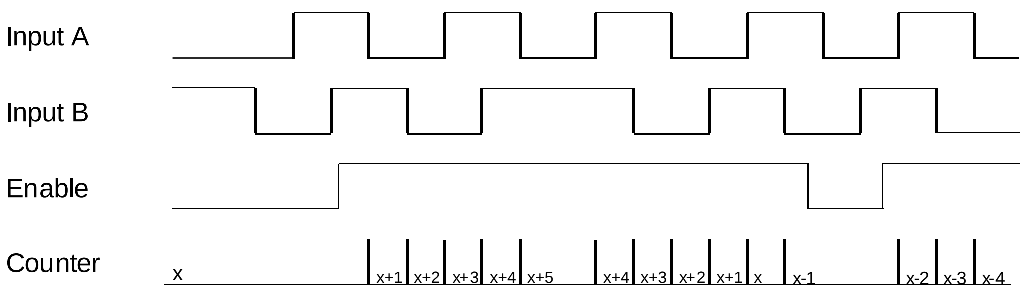
Mode 'x2' Init parameter 2 = 2

The rising and falling edges of signal 'A' are evaluated. Signal 'B' in quadrature defines the count direction.



Mode x4 Init parameter 2 = 4

The rising and falling edges of both signals 'A' and 'B' are evaluated. Signal 'B' in quadrature again defines count direction.



A mode 'x3' can also be selected (Init parameter 2 = 3), but has no practical uses and is not described here.

8.4 Programmable counter functions

- Definition of 'EnableC' and 'CCO'.
- Definition of counting mode.
- Definitions of inputs 'A' and 'B'.
- Counter initialization, i.e. adoption of definitions for EnableC', 'CCO' and counting mode.
- Loading counter values into a PCD register.
- Transfer of counter values from a PCD register into preset counter.
- Transfer of value from preset counter into counter.
- Loading the preset value into a PCD register
- Transfer of preset value from PCD register into preset register
- Starting the counter and activating 'CCO'.

- Reading counter value.
- Reading status of 'CCO' output (H = on, L = off)
- Reading status of 'EnableC' input (H = on, L = off)
- Reading count direction (H = up, L = down)

Programming is explained on the following pages, using a few examples.

Instead of several pages of description, **the programming principle** is illustrated using an unstructured example. This is a functional program which can, for example, be used to test a PCD2.H110.

Since counting tasks always have a sequential program flow:

- define module characteristics and counting task
- await end of counting
- evaluate counting

User programs should be consistently programmed in GRAFTEC.

The 3 typical examples which follow the demonstration of principle have therefore been edited in GRAFTEC.

The programs were written using "PG4" (GRAFTEC and IL).

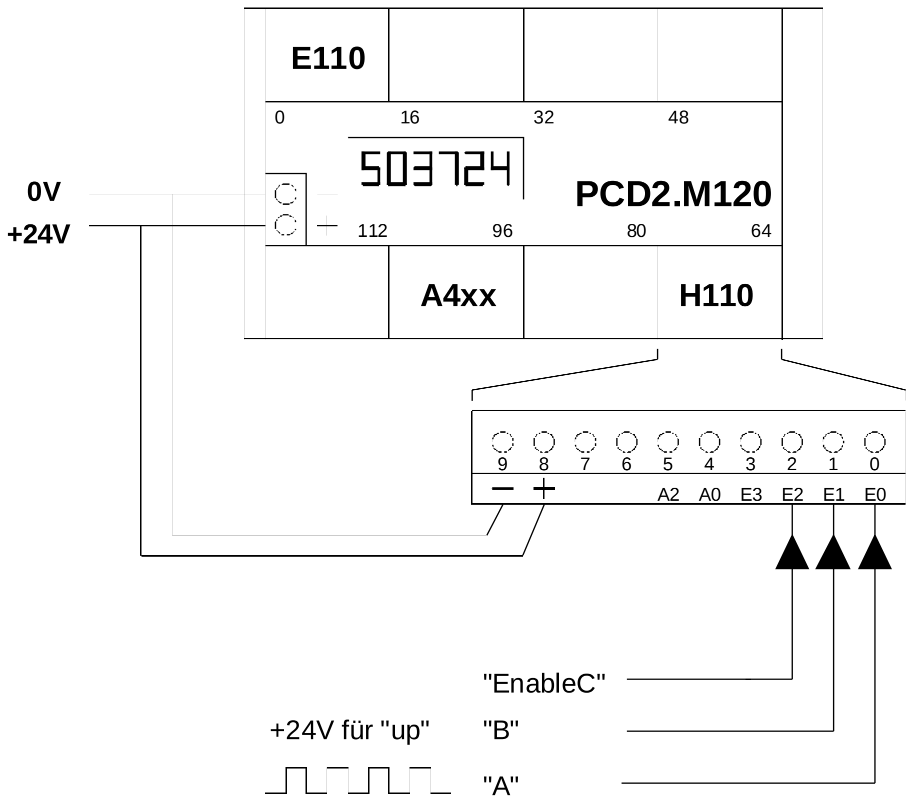
8.5 Programming principle

A simple example demonstrates the methodology for programming the counter of the PCD2.H110 module.

Task: The counter should be loaded with a start value of 500 and the preset register should be loaded with 900. The 'CCO' output should be defined as "static-normal" and the 'EnableC' input as "static-inverted". The count mode should be configured as 'x1' and the inputs 'A' and 'B' as "normal". After applying pulses to input 'A', the counter should count up to the value in the preset register (900). When the value is reached a digital PCD output should be complemented and the counter should be reloaded with its start value (500), and so on. The circuit can be seen as a frequency divider.

The program's name is "prinzip.src" and it is located in project "D2-H110". The file .MBA should also be copied into this project and the number of H110 modules (1) and the base address (64) should be indicated.

Arrangement for the use of the PCD2.H110 as counter in the defined example.



The elements are:

PCD1/2.M1x0 at least equipped with	1 PCD2.H110
	1 PCD2.A4xx
	1 PCD2.F510/530

```
; *********************************************************
; Basic user program for PCD2.H110 module as counter:
; prinzip.src
; *********************************************************

$include d2h110_b.equ
$group h110

xob       16

ld      r 999      ; PCD register with
          500      ;  start value for counter
ld      r 998      ; PCD register with
          900      ;  value for compare register
ld      r 0        ; scratch register
          0        ;  empty

cfb       init     ; initialisation H110
        k 1        ;  par 1    module number
          1        ;  par 2    count mode 'x1'
        r 999      ;  par 3    start value for counter
        r 998      ;  par 4    value for compare register
          1        ;  par 5    EnableC "static-inverted"
          0        ;  par 6    CCO "static-normal"
          0        ;  par 7    input A "normal"
          0        ;  par 8    input B "normal"
          0        ;  par 9    not used for counter
          0        ;  par 10   not used for counter
          0        ;  par 11   not used for counter
          0        ;  par 12   not used for counter

cfb       exec     ; start counter, set CCO
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

exob
; ---------------------------------------------------------

cob       0        ; real user program
          0

cfb       exec     ; read Counter
        k 1        ;  module number
          RdCt     ;  command: read counter
        r 777      ;  read value -> R 777

dsp     r 777      ; display on display module

sth       cco_1    ; polling CCO
cpb     h 25       ; if CCO = H, call PB 25

ecob
; ---------------------------------------------------------
```



```
pb        25       ; PB for new-start of counter

com     o 100      ; inverts on each R = C

cfb       exec     ; load counter with preset value
        k 1        ;  module number
          LdCtPres ;  command: load preset value
        r 999      ;  with value from register 999

cfb       exec     ; start counter, set CCO
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

epb

$endgroup
; -----------------------------------------------------------
```

Description of program

The file 'd2h110_b.equ' should be included at the beginning of the user program. The program between the directives '$group' and '$endgroup' is declared as code for PCD2.H110.

In the cold start routine 'XOB 16' a PCD register is loaded with the start value for the counter, e.g. R 999. If the start value is = 0, the register is loaded with 0. The next PCD register is loaded with the compare value of the H110, e.g. R 998. This register must be defined even if the value is not to be used in the program. An empty scratch register can also be prepared here, e.g. R 0.

The real configuration of the PCD2.H110 is done by calling the FB 'Init'. The call has 12 parameters. The meaning of these parameters is shown in the program example and in Appendix A of this manual. Parameter 5: 'EnableC' should be defined as "static-inverted" and parameter 6: 'CCO' should be defined as "static-normal". If counter and preset register are equal, the 'CCO' output goes = H.

After execution of the FB 'Init' the counter is started by the FB 'Exec' with parameter 2 as 'StartCt'. The 'CCO' will also be activated.

In the COB the counter value is read and transmitted to the display module and the 'CCO' is continuously polled. If CCO = H, i.e. "counter = register" PB 25 is called, a digital PCD output is complemented and the counter is loaded with the intial value and started again.

The program can now be processed with 'Project' - 'Build', loaded to the PCD and put in 'Run'. The function of the system can be followed on the display and on the activated PCD output.

It would also be imaginable to process the CCO dynamically, i.e. the CCO would be = H for approx. 100 µs on each "counter = register". This impulsion could be brought to the interrupt input 'INB1'. On each pulse (counter = register) the XOB 20 would be called. In the XOB 20 the counter could be loaded and restarted again.

This dynamic method should be used only be specialists, because conflicts between H110 FBs used in the COB and in XOB 20 may occur.

Other (more realistic) applications are presented in the following examples.

8.6 Application example no. 1: Counter in GRAFTEC

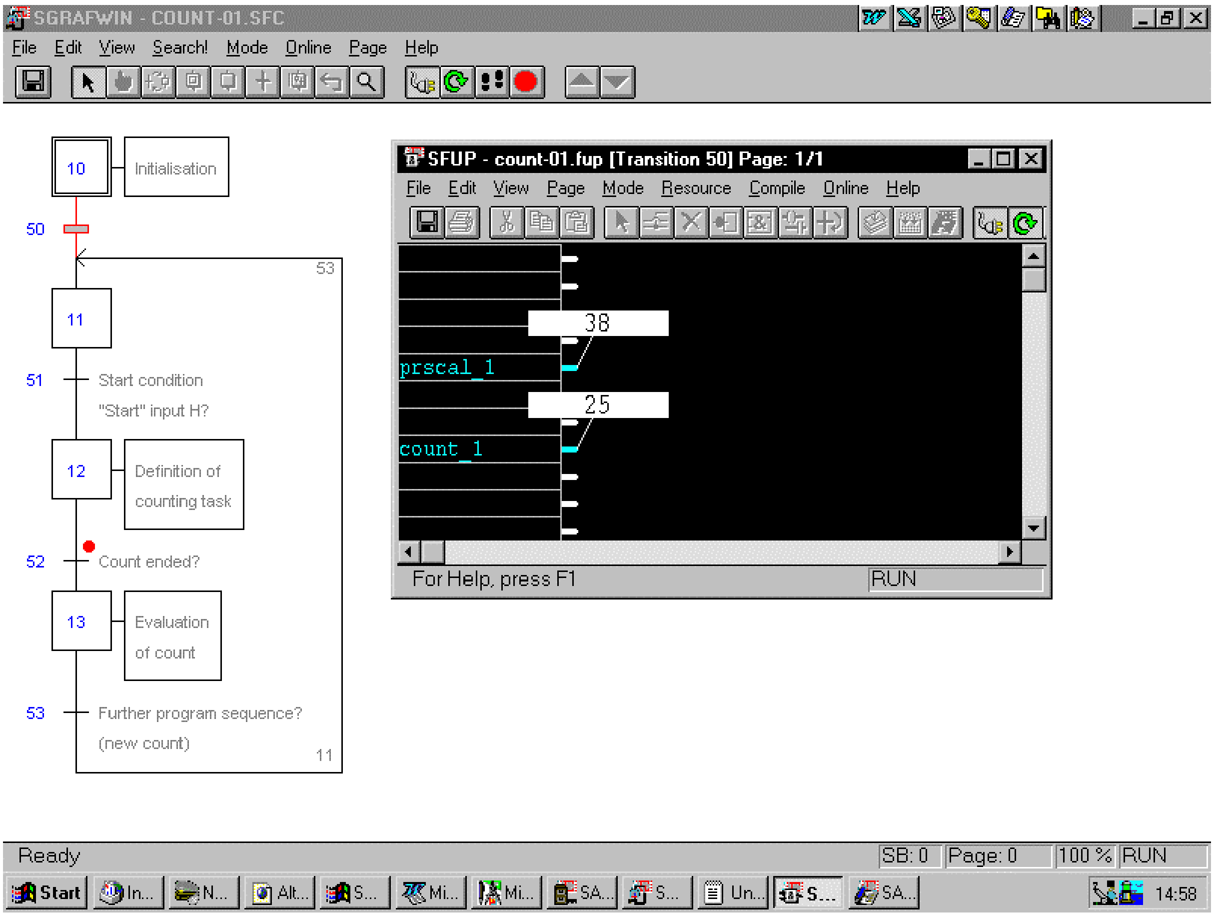
After switching on the "Start" input, the counter should be loaded at 0 and the register should be loaded at the value from the 2-digit BCD switch which has been wired to inputs 16 to 23.

The CCO output should be defined so that, at the start of each new count, it is switched H. When the counter reaches the preset register value, the CCO should switch to L and remain L until a new count is loaded.

Counter status should be displayed in the display window at every point in the program. It should also be possible to see the course of the program online, with register and counter values, on the programming unit screen.

The user program is edited in GRAFTEC. Within the "D2-H110" project, the program is entitled "COUNT-01.SFC".

The finished program will resemble the following:



Program code in "count-01.sfc"

(To obtain this representation, the file "count-01.sfc" should be renamed "count-01.src").

```
SB        0

;-------------------------------
IST       10       ;initialisation
    O 50
$include d2h110_b.equ
$group h110

ld      r 999      ; start value for counter
          0
ld      r 998      ; value for compare register
          0
ld      r 0        ; empty scratch register
          0

cfb       init     ; initialisation H110
        k 1        ;  par 1  module number
          1        ;  par 2  count mode 'x1'
        r 999      ;  par 3  start value for counter
        r 999      ;  par 4  value for preset register
          1        ;  par 5  enable "static-inverted"
          1        ;  par 6  CCO "static-inverted"
          0        ;  par 7  input A "normal"
          0        ;  par 8  input B "normal"
          0        ;  par 9  not used for counter
          0        ;  par 10 not used for counter
          0        ;  par 11 not used for counter
          0        ;  par 12 not used for counter

EST                ; 10

;-------------------------------
ST        11
    I 50
    I 53           ; continue task ?
    O 51           ; start condition: input "start" = H ?

EST                ; 11
```

```
;-------------------------------
ST         12       ; definition of counting task
    I 51            ; start condition: input "start" = H ?
    O 52            ; count ended ?

digir      2        ; read BCD value
        i 16
        r 998

cfb        exec     ; load preset register
        k 1         ;  module number
          LdRegPres;  command: load preset register
        r 998       ;  register with load value

cfb        exec     ; load counter
        k 1         ;  module number
          LdCtPres ;  command: load counter
        r 998       ;  register with load value

cfb        exec     ; start counters, set CCO
        k 1         ;  module number
          StartCt  ;  command: start counter
        r 0         ;  empty register

EST                 ; 12

;-------------------------------
ST         13       ; evaluation of count
    I 52            ; count ended ?
    O 53            ; further program sequence ?

com     o 101       ; process

EST                 ; 13

;-------------------------------
TR         50
    I 10            ; initialisation
    O 11

;; SFUP
__TR00050

ETR                 ; 50

;-------------------------------
TR         51       ; start condition: input "start" = H ?
    I 11
    O 12            ; definition of counting task

cfb        exec     ; read counter
        k 1         ;  module number
          RdCt      ;  command: read counter
        r 777       ;  read value in R 777
dsp     r 777       ; display on display module
sth     i 0         ; PCD input "start"

ETR                 ; 51
```

```
;--------------------------------
TR         52       ; count ended ?
    I 12            ; definition of counting task
    O 13            ; evaluation of count

cfb        exec     ; read counter
        k 1         ;  module number
          RdCt      ;  command: read counter
        r 777       ;  read value in R 777
dsp     r 777       ; display on display module
stl        cco_1    ; polling CCO

 ETR                ; 52

;--------------------------------
TR         53       ; further program sequence ?
    I 13            ; evaluation of count
    O 11

cfb        exec     ; read counter
        k 1         ;  module number
          RdCt      ;  command: read counter
        r 777       ;  read value in R 777
dsp     r 777       ; display on display module
stl     i 0         ; PCD input "Start"

$endgroup

ETR                 ; 53

ESB                 ; 0
```


Explanatory notes

Knowledge of the PG4 in general and of GRAFTEC in particular is required.

A call to sequential block SB 0 from a COB is generated automatically.

The course of the GRAFTEC program can be viewed online.

Initialization of the H110 module is done in IST 10. This IST is only processed when the SB is called for the first time, as with XOB 16. It is logical to carry out initialization of the H110 module in the IST belonging to the SB which handles that module, so that all the program for the module is in one place. XOB 16 is preferred for carrying out initializations which apply to the entire PCD.

Transition TR 50 is empty, but it could be edited in FUPLA so that the online values of the counter stand and the BCD preset value can be viewed. FUPLA's instruction list code is generated automatically by the PG4 and should not be modified.

In ST 12 the actual BCD value is loaded in the preset register, the counter is reset (set to zero), then the counter is started and with the same command the CCO is activated.

TR 52 queries whether counting is complete so that further program sequences can be triggered. The process itself is directly controlled from hardware with the CCO output. Before polling the switching condition (stl cco_1), the counter contents are read and displayed. This also applies to TR 51 and TR 53.

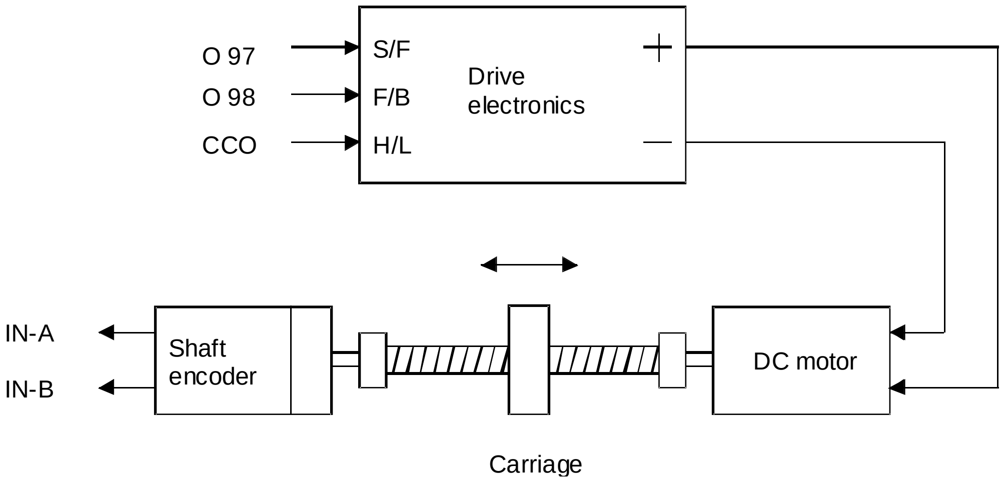
8.7 Application example no. 2: Motion control with incremental shaft encoder

The carriage of a working model (DC motor, spindle, sliding carriage, incremental shaft encoder and appropriate drive electronics) is to travel from a start position to another position and, after a pause, back to the start position. It should run at maximum acceleration to a preset speed, and continue at reducing speed to the destination position.

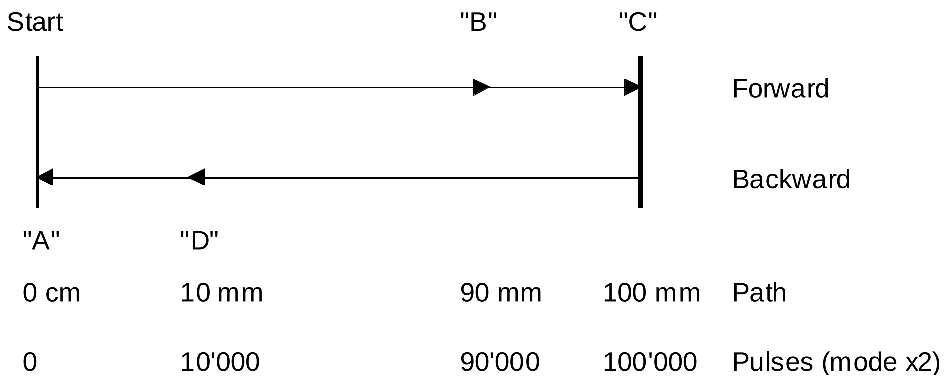
Some details of the working model (V-PCX 10):

DC motor with gears:	approx. 1200 rpm. at 24 VDC
Incremental shaft encoder:	500 pulses/rev., 2-phase, in quadrature
Spindle gradient	1.0 mm
Inputs for electronics: (V-PCX 11)	forward/backward (F/B): H = forward; L = backward slow/fast (S/F): L = slow; H = fast on/off : L = motor off, short-circuited H = Motor on, accordingly forward/backward, slow/fast
Outputs for electronics:	Motor (note polarity)
Supply for electronics:	24 VDC smoothed

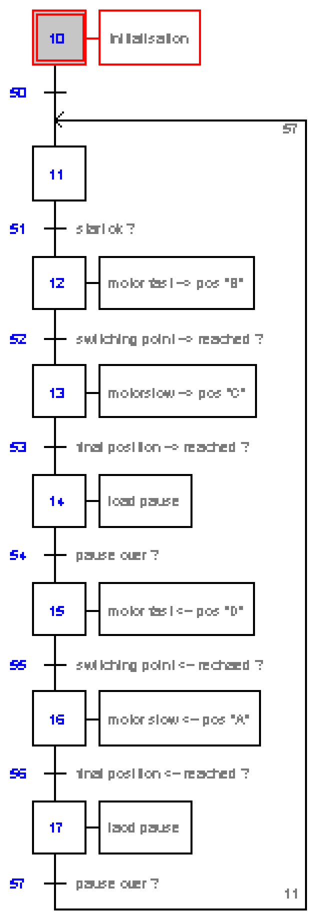
Arrangement and wiring of devices:



Motion:



The GRAFTEC structure looks like this:



Program code for "move-01.sfc"

(To obtain this representation, the file "move-01.sfc" should be renamed "move-01.src").

```
SB        0

;--------------------------------
IST       10        ; initialisation
    O 50
$include d2h110_b.equ
$group h110

ld      r 999       ; start value for counter
          0
ld      r 998       ; value for preset register
          0
ld      r 0         ; empty scratch register
          0
ld      r 995       ; scratch register for negative values
          16000000
ld      r 996       ; scratch register for negative value
          777215

cfb       init      ; initialisation H110
        k 1         ;  par 1  module number
          2         ;  par 2  count mode "x2"
        r 999       ;  par 3  start value for counter
        r 998       ;  par 4  value for preset register
          1         ;  par 5  enable "static-inverted"
          1         ;  par 6  CCO "static-inverted"
          0         ;  par 7  input A "normal"
          0         ;  par 8  input B "normal"
          0         ;  par 9  not used for counter
          0         ;  par 10 not used for counter
          0         ;  par 11 not used for counter
          0         ;  par 12 not used for counter

EST                 ; 10

;--------------------------------
ST        11
    I 50
    I 57            ; pause over?
    O 51            ; start ok ?
EST                 ; 11

;--------------------------------
ST        12        ; motor fast --> pos "B"
    I 51            ; start ok ?
    O 52            ; switching point --> reached ?

ld      r 998       ; load value for "B"
          8640
cfb       exec      ; load preset register
        k 1         ;  module number
          LdRegPres;  command: load preset register
        r 998       ;  PCD register with compare value
cfb       exec      ; start counter, set CCO
        k 1         ;  module number
          StartCt   ;  command: start counter
        r 0         ;  empty register

set     o 97        ; motor "fast"
set     o 98        ; motor "forward"

EST                 ; 12
```

```
;------------------------------
ST        13       ; motor slow --> pos "C"
    I 52           ; switching point --> reached ?
    O 53           ; final position --> reached ?

ld      r 998      ; load value for "C"
          9600

cfb       exec     ; load preset register
        k 1        ;  module number
          LdRegPres;  command: load preset register
        r 998      ;  PCD register with compare value

cfb       exec     ; start counters, set CCO
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

res     o 97       ; motor "slow"
set     o 98       ; motor "forward"

EST                ;13

;------------------------------
ST        14       ; load pause
    I 53           ; final position --> reached ?
    O 54           ; pause over ?

ld      t 0        ; pause
          50       ;  5 seconds

EST                ; 14

;------------------------------
ST        15       ; motor fast <-- pos "D"
    I 54           ; pause over ?
    O 55           ; switching point <-- reached ?

ld      r 998      ; load value for "C"
          960

cfb       exec     ; load preset register
        k 1        ;  module number
          LdRegPres;  command: load preset register
        r 998      ;  PCD register with compare value

cfb       exec     ; start counters, set CCO
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

set     o 97       ; motor "fast
res     o 98       ; motor "backward"

EST                ; 15
```

```
         ;------------------------------
         ST          16        ; motor slow <-- pos "A"
             I 55              ; switching point <-- reached ?
             O 56              ; final position <-- reached ?

         ld        r 998       ; load value for "A"
                     0

         cfb         exec      ; load preset register
                   k 1         ;  module number
                     LdRegPres;  command: load preset register
                   r 998       ;  PCD register with compare value

         cfb         exec      ; start counters, set CCO
                   k 1         ;  module number
                     StartCt   ;  command: start counter
                   r 0         ;  empty register

         res       o 97        ; motor "slow"
         res       o 98        ; motor "backward"

         EST                   ; 16

         ;------------------------------
         ST          17        ; load pause
             I 56              ; final position <-- reached ?
             O 57              ; pause over ?

         ld        t 0         ; pause
                     50        ;  5 seconds

         EST                   ; 17

         ;------------------------------
         TR          50
             I 10              ; initialisation
             O 11
         ETR                   ; 50

         ;------------------------------
         TR          51        ; start ok ?
             I 11
             O 12              ; motor fast --> pos "B"

         cfb         exec      ; read counter
                   k 1         ;  module number
                     RdCt      ;  command: read counter
                   r 777       ;  read value in R 777

         cmp       r 777       ; ¦
                   r 995       ; ¦ test if counter value
         jr        n next      ; ¦
         sub       r 777       ; ¦      below zero
                   r 995       ; ¦
                   r 777       ; ¦    if yes display
         sub       r 777       ; ¦
                   r 996       ; ¦    negative value
                   r 777       ; ¦

next:    dsp       r 777       ; display on display module

         sth       i 0         ; PCD input "start"

         ETR                   ; 51
```

```
        ;-------------------------------
        TR        52       ; switching point --> reached ?
            I 12           ; motor fast --> pos "B"
            O 13           ; motor slow --> pos "C"

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        cmp     r 777      ; ¦
                r 995      ; ¦ test if counter value
        jr      n next     ; ¦
        sub     r 777      ; ¦     below zero
                r 995      ; ¦
                r 777      ; ¦   if yes display
        sub     r 777      ; ¦
                r 996      ; ¦   negative value
                r 777      ; ¦

next:   dsp     r 777      ; display on display module

        stl       cco_1    ; polling CCO

        ETR                ; 52

        ;-------------------------------
        TR        53       ; final position --> reached ?
            I 13           ; motor slow --> pos "C"
            O 14           ; load pause

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        dsp     r 777      ; display on display module

        stl       cco_1    ; polling CCO

        ETR                ; 53

        ;-------------------------------
        TR        54       ; Pause over ?
            I 14           ; load pause
            O 15           ; Motor fast <-- pos "D"

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        dsp     r 777      ; disply on display module

        stl     t 0        ; polling timer

        ETR                ; 54
```

```
        ;-------------------------------
        TR        55       ; switching point <-- reached ?
            I 15           ; motor fast <-- pos "D"
            O 16           ; motor slow <-- pos "A"

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        dsp     r 777      ; display on display module
        stl       cco_1    ; polling CCO

        ETR                ; 55

        ;-------------------------------
        TR        56       ; final position <-- reached ?
            I 16           ; motor slow <-- pos "A"
            O 17           ; load pause

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        cmp     r 777      ; ¦
                r 995      ; ¦ test if counter value
        jr      n next     ; ¦
        sub     r 777      ; ¦      below zero
                r 995      ; ¦
                r 777      ; ¦    if yes display
        sub     r 777      ; ¦
                r 996      ; ¦    negative value
                r 777      ; ¦

next:   dsp     r 777      ; display on display module
        stl       cco_1    ; polling CCO

        ETR                ; 56

        ;-------------------------------
        TR        57       ; pause over ?
            I 17           ; load pause
            O 11

        cfb       exec     ; read counter
                k 1        ;  module number
                  RdCt     ;  command: read counter
                r 777      ;  read value in R 777

        cmp     r 777      ; ¦
                r 995      ; ¦ test if counter value
        jr      n next     ; ¦
        sub     r 777      ; ¦      below zero
                r 995      ; ¦
                r 777      ; ¦    if yes display
        sub     r 777      ; ¦
                r 996      ; ¦    negative value
                r 777      ; ¦

next:   dsp     r 777      ; display on display module
        stl     t 0        ; polling timer

        $endgroup

        ETR                ; 57

        ESB                ; 0
```

Explanatory notes

Knowledge of the PG4 in general and of GRAFTEC in particular is required.

A call to sequential block SB 0 from a COB is generated automatically.

The course of the GRAFTEC program can be viewed online.

Initialization of the H110 module is done in IST 10. This IST is only processed when the SB is called for the first time, as with XOB 16. It is logical to carry out initialization of the H110 module in the IST belonging to the SB which handles that module, so that all the program for the module is in one place. XOB 16 is preferred for carrying out initializations which apply to the entire PCD.

The counter is only loaded with zero on start-up, i.e. in the IST, after which it is not accessed by the program. All signals coming from the incremental shaft encoder are then counted. The counter therefore holds the exact carriage position, i.e. even overruns of the end position or manual rotation of the spindle are captured accurately.

Software querying of CCO in TR 52, 53, 55 and 56 is only used for GRAFTEC switching. Control of the process itself, in this case control of the carriage, is done directly by the CCO output.

The routine "Test if counter value below zero" in most TRs is only used to maintain the display and prevent the error flag being set, since only 6 digits can be displayed. If the counter value falls below zero, this routine subtracts 16,000,000 (R 999) before it is displayed, which makes the value displayable. The zero position could also, for example, be defined at 1000. This would be a way of getting round the problem of values "below zero".

The reader may notice that this example has not quite been properly programmed. When switching from "fast" to "slow" the motor is briefly switched off or short-circuited by the drive, since the CCO output is switched when positions "B" or "D" are reached and is only switched on again after the new value has been loaded. Strictly speaking, this switching ought to be bypassed with a normal output. However, this example is just to illustrate the principle.

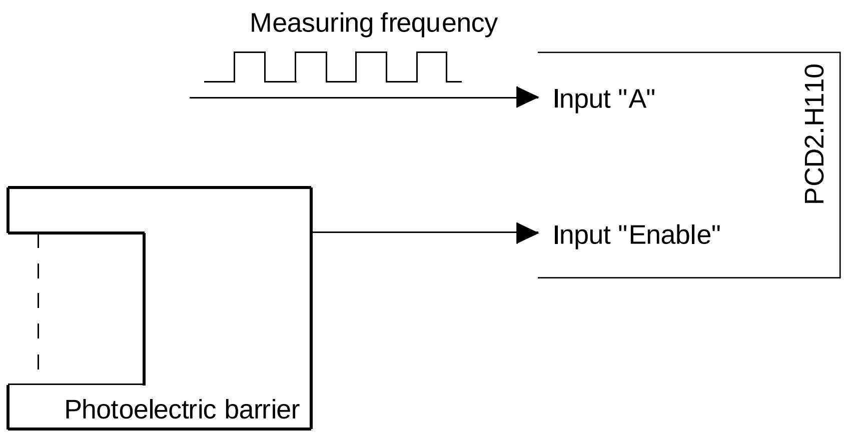
Section 9.1.5 shows how frequency can also be measured parallel to the motion control explained above.

8.8 Application example no. 3: Measurement with counting

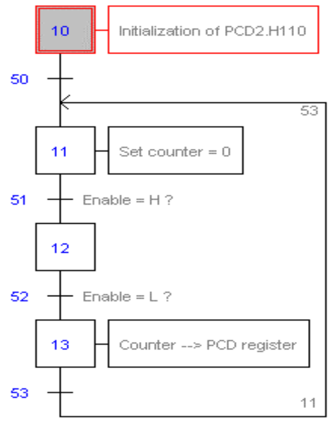
While a photoelectric barrier is covered by items being transported on a conveyor belt, pulses output in proportion to the speed of the conveyor are counted, thus measuring the size of items for sorting purposes. This simple method has been successfully used in the south of France to sort melons and apricots.

The task is therefore to count signals during a certain situation, e.g. photoelectric barrier covered, with the count being controlled directly by the photoelectric barrier using an input on the counting module, **not** via a digital PCD input.

For this purpose, the measurement pulses are carried to input "A" and the photoelectric barrier to the "Enable" input of the counting module, which already largely solves the problem.



The GRAFTEC structure looks like this:



Program code for "mess-01.sfc"

(To obtain this representation, the file "mess-01.sfc" should be renamed "mess-01.src").

```
SB        0

;--------------------------------
IST       10       ; initialisation
    O 50

$include d2h110_b.equ
$group h110

ld      r 999      ; start value for counter
          0
ld      r 998      ; value for preset register
          0
ld      r 0        ; empty scratch register
          0

cfb       init     ; initialisation H110
        k 1        ;  par 1  module number
          0        ;  par 2  count mode 'x1'
        r 999      ;  par 3  start value for counter
        r 998      ;  par 4  value for register (not used.)
          1        ;  par 5  enable "static-inverted"
          0        ;  par 6  CCO (not used)
          0        ;  par 7  input A "normal"
          1        ;  par 8  input B "inverted"
          0        ;  par 9  not used for counter
          0        ;  par 10 not used for counter
          0        ;  par 11 not used for counter
          0        ;  par 12 not used for counter

 EST               ; 10

;--------------------------------
ST        11       ; set counter = 0
    I 50
    I 53
    O 51           ; enable = H ?

cfb       exec     ; load counters (with 0)
        k 1        ;  module number
          LdCtPres ;  command: load counter
        r 998      ;  PCD register with load value

cfb       exec     ; start counter
        k 1        ;  module number
          StartCt  ;  command: start counter
        r 0        ;  empty register

EST                ; 11

;--------------------------------
ST        12
    I 51           ; enable = H ?
    O 52           ; enable = L ?
EST                ; 12
```

```
;-------------------------------
ST          13        ; counter -> PCD register
    I 52              ; enable = L ?
    O 53

cfb         exec      ; read counter (for result)
        k 1           ;  module number
          RdCt        ;  command: read counter
        r 777         ;  read value in R 777

putx    r 777
        r 2000
ini     k 1000

EST                   ; 13

;-------------------------------
TR          50
    I 10              ; initialisation
    O 11              ; set counter = 0
ETR                   ; 50

;-------------------------------
TR          51        ; enable = H ?
    I 11              ; set counter = 0
    O 12

cfb         exec      ; read counter (for display)
        k 1           ;  module number
          RdCt        ;  command: read counter
        r 777         ;  read value in R 777

dsp     r 777         ; display on display module

sth         Cstart_1 ; polling 'EnableC'

ETR                   ; 51

;-------------------------------
TR          52        ; enable = L ?
    I 12
    O 13              ; counter -> PCD register

cfb         exec      ; read counter (for display)
        k 1           ;  module number
          RdCt        ;  command: read counter
        r 777         ;  read value in R 777

dsp     r 777         ; display on display module

stl         Cstart_1 ; polling 'EnableC'

ETR                   ; 52

;-------------------------------
TR          53
    I 13              ; counter -> PCD register
    O 11              ; set counter = 0

$endgroup

ETR                   ; 53

ESB                   ; 0
```


Explanatory notes

Knowledge of the PG4 in general and of GRAFTEC in particular is required.

A call to sequential block SB 0 from a COB is generated automatically.

The course of the GRAFTEC program can be viewed online.

Regarding definition of 'EnableC' and the 'CCO': The photoelectric barrier used in the example supplies a 'H' signal to the 'EnableC' input when it is not covered. The signal should be defined as "static" and "inverted".

Count mode is "up/down", i.e. mode 'x1'. With code '0', +24V must be applied to input 'B' for "up" counting. With code '1', input 'B' is inverted. It does not need +24V at the input 'B' terminal for "up" counting.

The program is controlled by software polling of the 'EnableC' input with "sth or stl Cstart_1". The counter is controlled directly using the "EnableC" input.

In ST 13 evaluation of the result is indicated. Each time the photoelectric barrier is released, the counter value is stored in consecutive PCD registers (from R 2000).

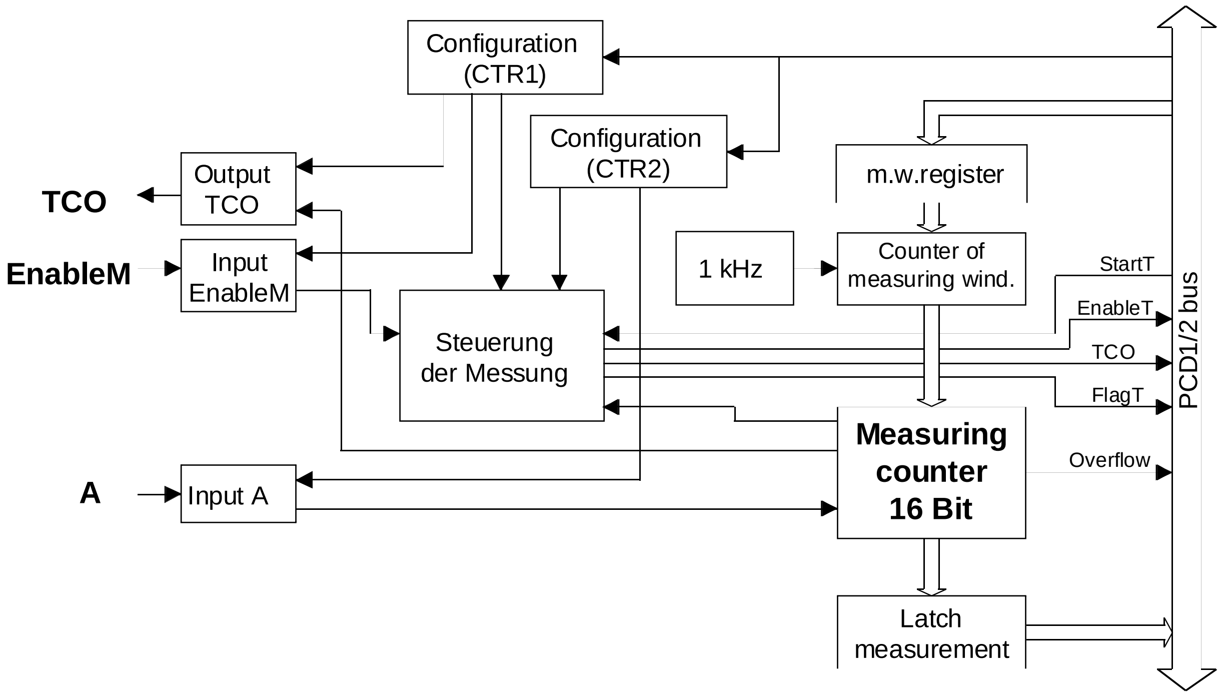
Instead of a photoelectric barrier, a bounce-free switch could also be used.

Notes

9. PCD2.H110 for measuring tasks

9.1 Frequency measurement

9.1.1 Block diagram



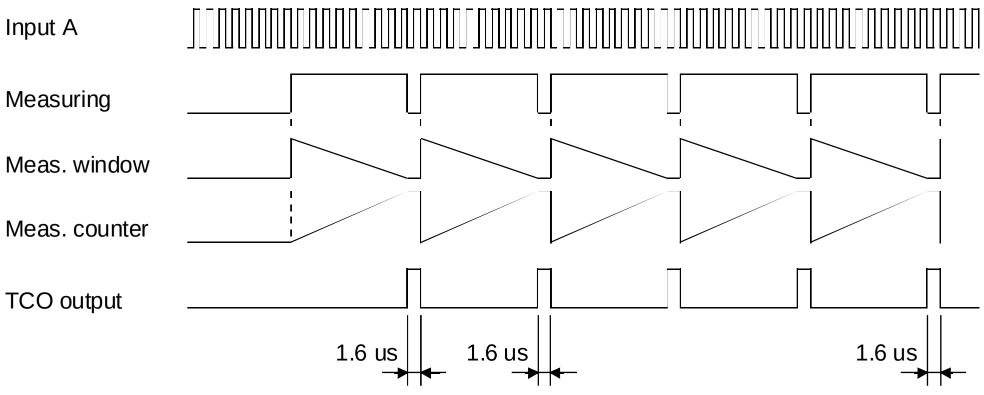
9.1.2 Description of frequency measurement

The frequency measurement range is from 500 Hz to 100 kHz.

Frequency measurement can be done in parallel with counting. It uses 2 counters of 16 bits each:

- One counter from this pair, the counter of measuring window, has a fixed clock of 1 kHz. This provides the time base in 1 ms steps for the programmable measuring window.
- The other counter, the measuring counter, counts the signals arriving at input 'A' during the time when the measuring window is open. If the measuring window has been defined as 1s (1000 ms) the result appears in the measuring counter directly in Hz or pulses per second.

Frequency measurement runs automatically, i.e. the time defined for the time window is measured, then the signal 'TCO' is activated, the measured value is latched and a new measurement is done, etc. The length of the signal 'TCO' the latch and the reset of the counter takes 1.6 µs.



9.1.3 Configuration of frequency measurement

The configuration takes place in FB 'Init'. For continuous frequency measurement, parameter 9 should be defined as '5' (frequency-automatic).

A "single shot" frequency measurement is possible but not very useful. To obtain this mode, parameter 9 should be defined as '4'. In this case only one measuring window is measured. Then the whole procedure has to be repeated for a new measurement.

Measuring range and time window

The measuring range is between 0 and 65 535 (16 bit).

To obtain a resolution of 1‰ at least 1000 signals must be captured per measurement. The measurement window’s open time depends on the frequency to be measured. To measure 100 kHz a minimum measurement time of 10 ms should be provided; to measure 500 Hz a measurement time of at least 2s is required.

If the frequency to be measured is even smaller, measurement time increases in length, which is not acceptable for every application. For frequencies below approx. 100 Hz, sufficient accuracy within an acceptable measurement time may call for period length measurement as described below, rather than frequency measurement.

Parameter 11 defines the behaviour of input 'EnableM', parameter 12 the behaviour of output 'TCO'.

= 9	Configuration measuring mode		integer	0 – 5	4 = frequency-manual 5 = frequency-autom.
= 10	Load value for measuring		integer	0 – 65535	length of time window for frequency measurement
= 11	Configuration of input EnableM		integer	0 – 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted
= 12	Configuration of output TCO		integer	0 – 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted

The complete table is shown in the appendix, page A-2.

9.1.4 Programming principle

Task: An impulse string is applied to input 'A'.
A continuous (automatic) frequency measurement with a time window of 1s (1000 ms) is to be realized. The result is to be shown as counting units (Hz) on the display module.

```
         ; ***********************************************
         ; Basic user program for the PCD2.H110 module
         ; for frequency measurement: frequ-01.src
         ; ***********************************************
         ;
         $include d2h110_b.equ
         $group h110

         xob       16

         ld      r 0        ; empty scratch register
                   0

         cfb       init     ; initialisation H110
                 k 1        ;  par 1  module number
                   0        ;  par 2  not used for measurement
                 r 0        ;  par 3  not used for measurement
                 r 0        ;  par 4  not used for measurement
                   0        ;  par 5  not used for measurement
                   0        ;  par 6  not used for measurement
                   0        ;  par 7  input A "normal"
                   0        ;  par 8  not used
                   5        ;  par 9  mode: "frequency-auto."
                   1000     ;  par 10 length time window in ms
                   0        ;  par 11 EnableM: "static-normal"
                   2        ;  par 12 TCO: "dynamic-normal"

         cfb       exec     ; start measure
                 k 1        ;  module number
                   StartMs  ;  command: start measure
                 r 0        ;  empty register

         exob
         ; -----------------------------------------------------------

         cob       0
                   0

  wait:  sth       EndMes_1 ; measure ended ?
         jr      l wait     ; if not, wait

         cfb       exec     ; read measure result
                 k 1        ;  module number
                   RdMsImp  ;  command: read measure in impulsion
                 r 777      ;  read value in R 777

         dsp     r 777      ; display on display module
;(wait:)
         ecob
         ; -----------------------------------------------------------

         xob       20       ; interrupt "INB1"
         com     o 101      ; inverts after each measure
         exob

         $endgroup
```

Program description

In hardware, a signal transmitter supplying the signals to be measured, e.g. a pulse generator, should be wired to input 'A'. Input 'B' remains open. The 'EnableC' input is not used at all for measurement and remains open. The 'EnableM' input ought really to receive +24V, but has been inverted during configuration and therefore remains open. *)

The 'TCO' output is wired to the PCD's interrupt input 'INB1'. At the end of each measurement, XOB 20 is called which, for this example, inverts the PCD output O 101. This is so that program function can be viewed better. In the example, this output is inverted every second.

The time base defined for this example is 1000, for a time window of 1000 ms = 1s. See also "Measuring range and time window".

The COB waits until the end of each measurement before reading the result from the measuring counter, where it can be displayed by the debugger or on a display module in integer format. The units of measurement are controlled by the time base definition, in this example it is in Hertz.

With 'RdMsUnit' instead of 'RdMsImp' the result is always converted into Hertz and can be viewed in the PCD register in floating point format.

With the command 'StopMs', a running measure can be broken off. The result is not valid. A new measure can be started again with 'StartMs'.

*) After a deactivate of 'EnableM' the result is not valid. A new measure can be started only with the command 'StartMs'

9.1.5 Combination of counting and frequency measuring

As already mentioned, frequency measurement can be used in parallel with counting. This is shown very well by a combination of the two examples "Motion control with incremental shaft encoder" (application example no. 2, section 8.7) and the present demonstration of the principle of frequency measurement. It is just necessary to remove the commands for displaying position (read counter) from the motion control example, since only one display is available and it is used to display frequency.

Since both functions are executed on the same H110, the initialisation is common for the whole module. In this example the initialisation is performed in the IST of the positioning program. The start of the measurement takes place directly after the initialisation.

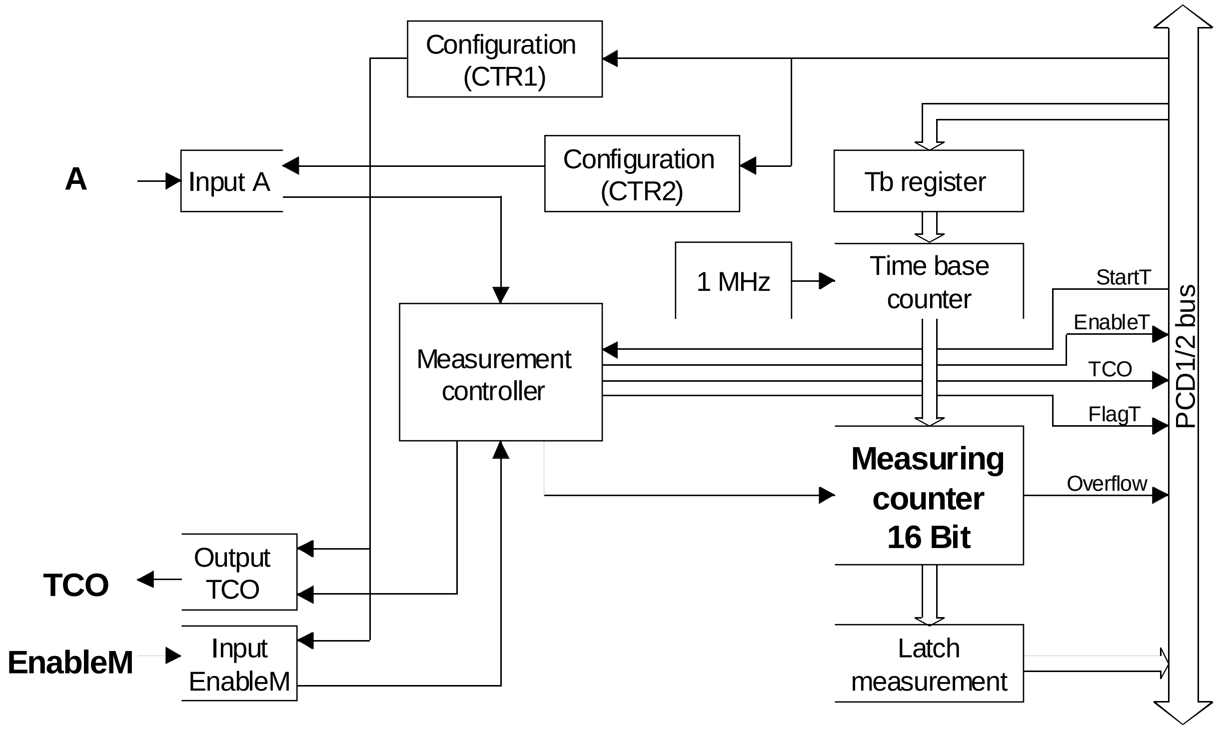
```
IST       10

$include d2h110_b.equ
$group h110

ld      r 0        ; empty scratch register
          0
ld      r 999      ; start value for counter
          0
ld      r 998      ; value for preset register
          0

cfb       init     ; initialisation H110
        k 1        ;  par 1  module number
          2        ;  par 2  count mode: "x2"
        r 999      ;  par 3  start value for counter
        r 998      ;  par 4  value for preset register
          1        ;  par 5  enable "static-inverted"
          1        ;  par 6  CCO "static-inverted"
          0        ;  par 7  input A "normal"
          0        ;  par 8  input B "normal"
          5        ;  par 9  measuring mode: "frequ.-auto."
          100      ;  par 10 length time window in ms
          0        ;  par 11 EnableM: "static-normal"
          2        ;  par 12 TCO: "dynamic-normal"

cfb     h exec     ; start measure
        k 1        ;  module number
          StartMs  ;  command: start measure
        r 0        ;  empty register

EST                ; 10
```

The project consists therefore of two programs: 'move-02.sfc' with the initialisation of the H110 and 'frequ-02' without XOB 16.

The motion control sequence can be viewed online in GRAFTEC with the relevant frequency on the display module.

9.2 Period length measurement

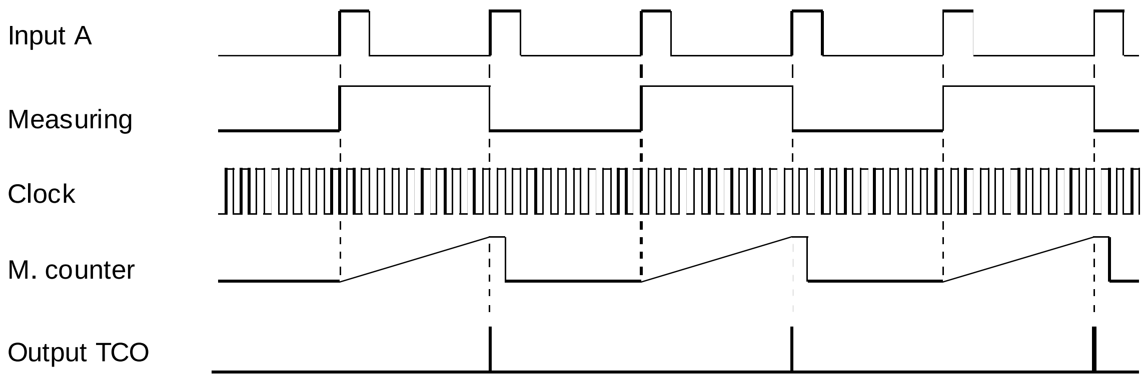
9.2.1 Block diagram



9.2.2 Description of period length measurement

Period length measurement uses 2 counters of 16 bits each:

- One of these two counters, the time base counter, has a fixed clock of 1 MHz, producing a fundamental time base of 1 µs. The user-defined time base is generated here.
- The other counter, the measurement counter, counts the time base pulses between two rising edges on input 'A'. Therefore, when there are consecutive pulses at input 'A', measurement is always between pairs of pulses, after which there is a pause to restore readiness for the next measurement.



9.2.3 Configuration of period length measurement

The configuration takes place in FB 'Init'. For continuous period length measurement, parameter 9 should be defined as '3' (period-automatic).

A manual period length measurement is possible but not very useful. To obtain this mode, parameter 9 should be defined as '2'. In this case only one period is measured. Then the whole procedure has to be repeated for a new measurement.

With parameter 7 the signal on input 'A' can be inverted.

Measuring range and time base

The measuring range is between 0 and 65 535 (16 bit).

The formula shown below can be used to calculate what value to enter for the time base:

$$n = \frac{T * 10^6}{clk} - 1$$

where: T = period length in seconds
clk = number of clock signals
n = value to enter

Example: Let period length equal 10s and the number of clock signals equal 10 000

$$n = \frac{10 * 10^6}{10\ 000} - 1 = 999$$

Parameter 11 defines the behaviour of input 'EnableM', parameter 12 the behaviour of output 'TCO'.

= 7	Configuration input A		integer	0 – 3	0: C=norm M=normal 1: C=invers M=normal 2: C=norm. M=invers 3: C=invers M=invers
= 9	Configuration measuring mode		integer	0 – 5	2 = period-manual 3 = period-automatic
= 10	Load value for measuring		integer	0 – 65535	Timebase for period length measurement
= 11	Configuration of input EnableM		integer	0 – 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted
= 12	Configuration of output TCO		integer	0 – 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted

The complete table is shown in the appendix, page A-2.

9.2.4 Programming principle

Task: A photoelectric barrier (or a no-bounce impulse contact) is to be wired to input 'A'. The time between two 'H' edges should be measured and displayed on the display module. The configuration should be designed so that for a measure time of 1 second, 10000 clock signals are counted. The timebase will be 99 (see formula).

```
; ***********************************************
; Basic user program for the PCD2.H110 module
; for period length measurement: peri-01.src
; ***********************************************
;
$include d2h110_b.equ
$group h110

xob       16

ld      r 0        ; empty scratch register
          0

cfb       init     ; initialisation H110
        k 1        ;  par 1  module number
          0        ;  par 2  not used for measurement
        r 0        ;  par 3  not used for measurement
        r 0        ;  par 4  not used for measurement
          0        ;  par 5  not used for measurement
          0        ;  par 6  not used for measurement
          3        ;  par 7  input A "inverted"
          0        ;  par 8  not used
          3        ;  par 9  mode: "period-auto."
          99       ;  par 10 timebase in µs
          0        ;  par 11 EnableM: "static-normal"
          2        ;  par 12 TCO: "dynamic-normal"

cfb       exec     ; start measure
        k 1        ;  modul number
          StartMs  ;  command: start measure
        r 0        ;  empty register

exob
; -----------------------------------------------------------

cob       0
          0

       sth       EndMes_1 ; measure ended ?
wait:  jr      l wait     ; if not, wait

cfb       exec     ; read measure result
        k 1        ;  module number
          RdMsImp  ;  command: read measure in impulsion
        r 777      ;  read value in R 777

dsp     r 777      ; display on display module

ecob
; -----------------------------------------------------------

xob       20       ; interrupt "INB1"
com     o 101      ; inverts after each measure
exob

$endgroup
```


Program description

In hardware, a signal transmitter supplying the signals to be measured, e.g. a photoelectric barrier, should be wired to input 'A'. Input 'B' remains open. The 'EnableC' input is not used at all for measurement and remains open. The 'EnableM' input ought really to receive +24V, but has been inverted during configuration and therefore remains open. *)

The 'TCO' output is wired to the PCD’s interrupt input 'INB1'. At the end of each measurement, XOB 20 is called which, for this example, inverts the PCD output O 101. This is so that program function can be viewed better.

The time base defined for this example is 99 for a result in 10000 ms for 1s measuring time. See also "Measuring range and time base"

Input 'A' is inverted (Init parameter 7 = '3'), as the photoelectric barrier supplies an inverted signal.

The COB waits until the end of each measurement before reading the result from the measuring counter, where it can be displayed by the debugger or on a display module in integer format. The units of measurement are controlled by the time base definition. In this example it is 1/10000 of seconds.

With 'RdMsUnit' instead of 'RdMsImp' the result is always converted into seconds and can be viewed in the PCD register in floating point format.

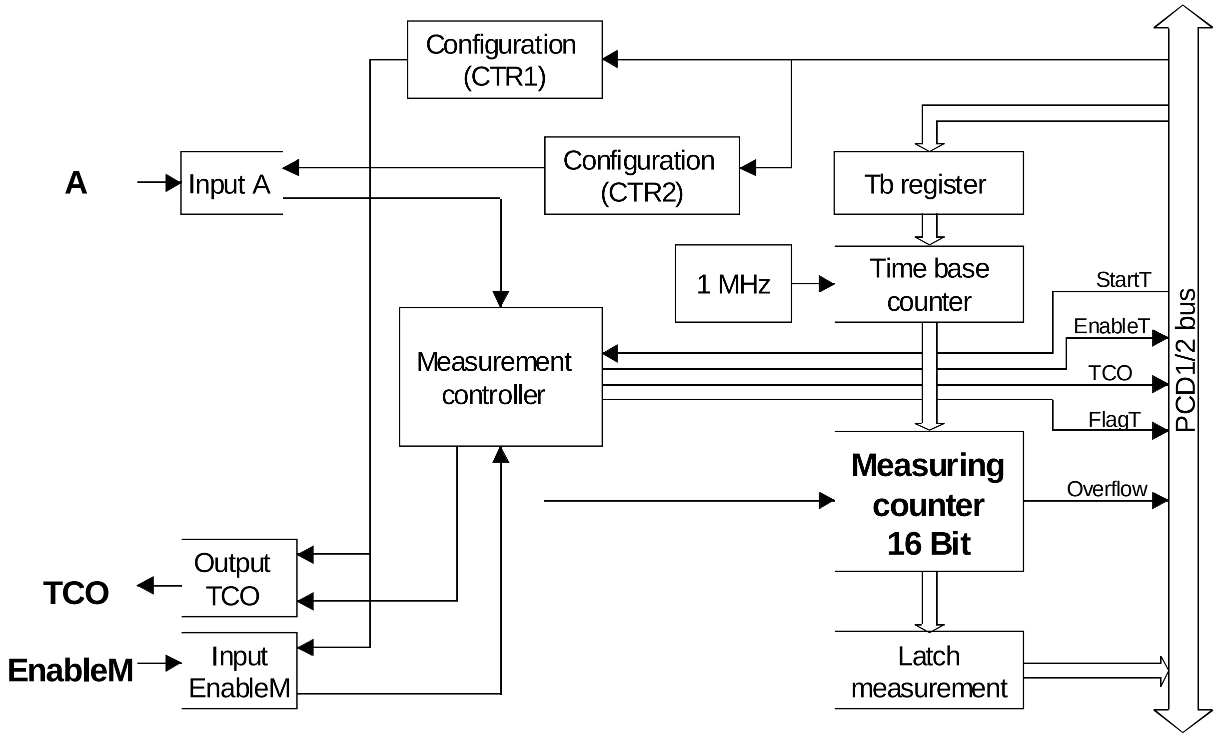
With the command 'StopMs', a running measure can be broken off. The result is not valid. A new measure can be started again with 'StartMs'.

*) After a deactivate of 'EnableM' the result is not valid. A new measure can be started only with the command 'StartMs'.

Notes

9.3 Pulse length measurement

9.3.1 Block diagram

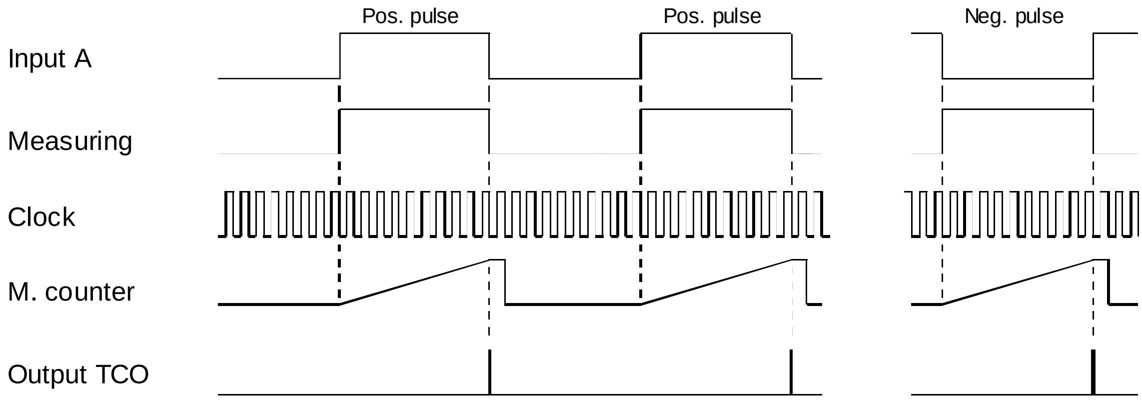


9.3.2 Description of pulse length measurement

Pulse length measurement uses 2 counters of 16 bits each:

- One of these two counters, the time base counter, has a fixed clock of 1 MHz, producing a fundamental time base of 1 µs. The user-defined time base is generated here.
- The other counter, the measurement counter, counts the time base pulses while input 'A' is H (positive or normal pulse length measurement) or while input 'A' is L (negative or inverted pulse length measurement).

Remark: Negative or inverted pulse length measurement is achieved by setting 'Init' parameter 7 = '3'.



9.3.3 Configuration of pulse length measurement

The configuration takes place in FB 'Init'. For continuous pulse length measurement, parameter 9 should be defined as '1' (impulse-automatic).

A manual pulse length measurement is possible but not very useful. To obtain this mode, parameter 9 should be defined as '0'. In this case only one pulse is measured. Then the whole procedure has to be repeated for a new measurement.

Measurement range and time base

The measurement range is between 0 and 65 535 (16 bit).

The formula shown below can be used to calculate what value to enter for the time base:

$$n = \frac{T * 10^6}{clk} - 1$$

where: T = pulse length in seconds
clk = number of clock signals
n = value to be entered

Example: Let the pulse length equal 10s and the number of clock signals be 10 000

$$n = \frac{10 * 10^6}{10\,000} - 1 = 999$$

Parameter 11 defines the behaviour of input 'EnableM', parameter 12 the behaviour of output 'TCO'.

= 7	Configuration input A		integer	0 – 3	0: C=norm M=normal 1: C=invers M=normal 2: C=norm. M=invers 3: C=invers M=invers
= 9	Configuration measuring mode		integer	0 - 5	0 = impulse-manual 1 = impulse-automatic
= 10	Load value for measuring		integer	0 - 65535	Timebase for pulse length measurement
= 11	Configuration of input EnableM		integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted
= 12	Configuration of output TCO		integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynamic-normal 3 = dynamic-inverted

The complete table is shown in the appendix, page A-2.

9.3.4 Programming principle

Task: A photoelectric barrier (or a no-bounce impulse contact) is to be wired to input 'A'. The time for the length of a pulse (input A = H or input A = L) is to be measured and displayed on the display module. The configuration is to be designed so that for a measure time of 1 second, 1000 clock signals are counted. The timebase will be 999 (see formula).

```
; ***********************************************
; Basic user program for the PCD2.H110 module
; for pulse length measurement: imp-01.src
; ***********************************************
;
$include d2h110_b.equ
$group h110

xob       16

ld      r 0        ; empty scratch register
          0

cfb       init     ; initialisation H110
        k 1        ;  par 1  module number
          0        ;  par 2  not used for measuring
        r 0        ;  par 3  not used for measuring
        r 0        ;  par 4  not used for measuring
          0        ;  par 5  not used for measuring
          0        ;  par 6  not used for measuring
          3        ;  par 7  input A "inverted"
          0        ;  par 8  not used
          1        ;  par 9  mode: "impulse-auto."
          999      ;  par 10 timebase in µs
          0        ;  par 11 EnableM: "static-normal"
          2        ;  par 12 TCO: "dynamic-normal"

cfb       exec     ; start measure
        k 1        ;  module number
          StartMs  ;  command: start measure
        r 0        ;  empty register

exob
; -----------------------------------------------------------

cob       0
          0

        sth       EndMes_1 ; measure ended ?
wait:   jr      l wait     ; if not, wait

cfb       exec     ; read measure result
        k 1        ;  module number
          RdMsImp  ;  command: read measure in impulsion
        r 777      ;  read value in R 777

dsp     r 777      ; display on display module

ecob
; -----------------------------------------------------------

xob       20       ; interrupt "INB1"
com     o 101      ; inverts after each measure
exob

$endgroup
```

Program description

In hardware, a signal transmitter supplying the signals to be measured, e.g. a photoelectric barrier, should be wired to input 'A'. Input 'B' remains open. The 'EnableC' input is not used at all for measurement and remains open. The 'EnableM' input ought really to receive +24V, but has been inverted during configuration and therefore remains open.

The 'TCO' output is wired to the PCD's interrupt input 'INB1'.At the end of each measurement, XOB 20 is called which, for this example, inverts the PCD output O 101. This is so that program function can be viewed better.

The time base defined for this example is 999 for a result in milliseconds. See also "Measuring range and time base"

Input 'A' is inverted (Init parameter 7 = '3'), as the photoelectric barrier supplies an inverted signal.

The COB waits until the end of each measurement before reading the result from the measuring counter, where it can be displayed by the debugger or on a display module in integer format. The units of measurement are controlled by the time base definition. In the present example it is milliseconds.

With 'RdMsUnit' instead of 'RdMsImp' the result is always converted into seconds and can be viewed in the PCD register in floating point format.

With the command 'StopMs', a running measure can be broken off. The result is not valid. A new measure can be started again with 'StartMs'.

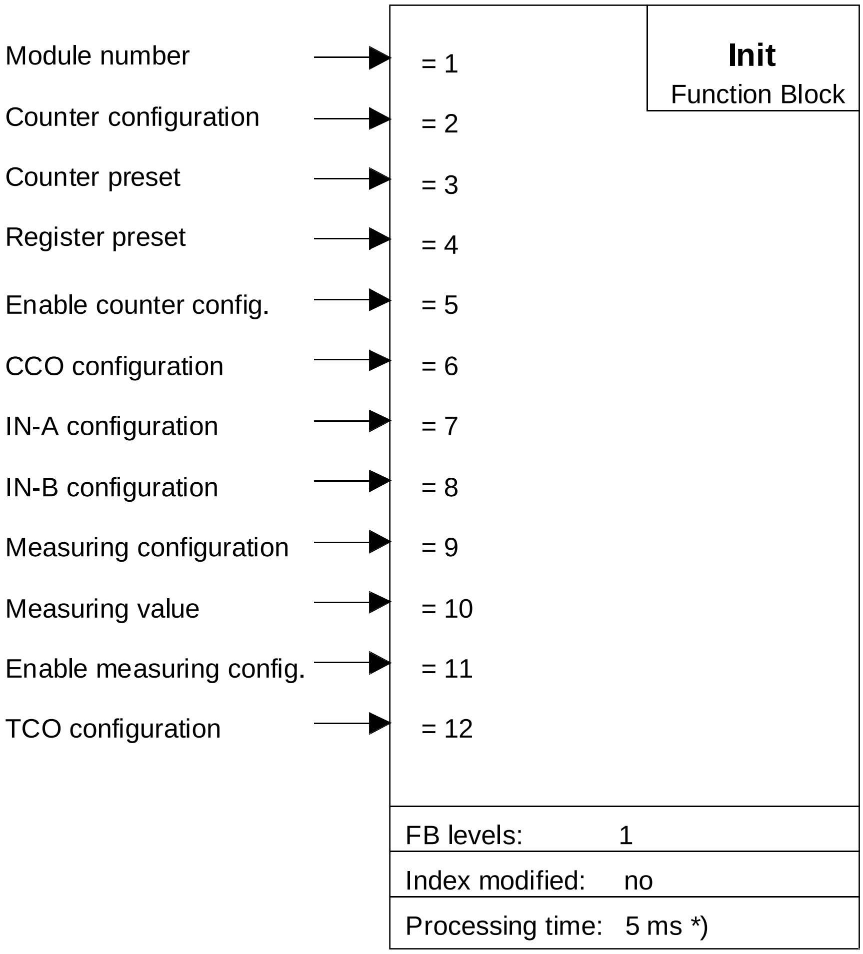
*) After a deactivate of 'EnableM' the result is not valid. A new measure can be started only with the command 'StartMs'.

Notes

Appendix A. Summary of all software elements for programming in IL

Function block 'Init'

Init **FB:** Initialisation of a H110 module



*) measured with PCD2.M120

Function description:

This FB defines the settings of the PCD2.H110 module and reads the base address from file D2H110_B.MBA.

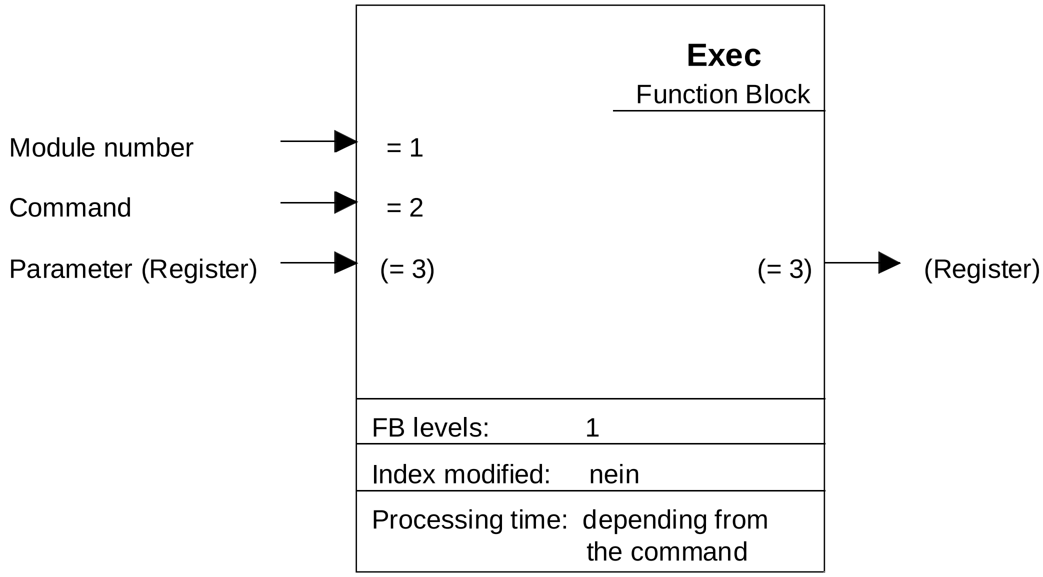
Parameter '1' must be given as a 'K' constant, parameter '3' and '4' are PCD register addresses (absolute or symbolic) and all other parameters as integer numbers.

Description of participating input/output elements:

Para-meter	Designation	Type	Format	Value	Comment
= 1	Module no.	K	K n	K 1–K 16	
= 2	Counter mode configuration		Integer	0 - 4	0 / 1 = mode x1 2 = mode x2 3 = mode x3 4 = mode x4
= 3	Load counter preset value	R	Integer	0 - 16777215	Start value of the counter
= 4	Load register preset value	R	Integer	0 - 16777215	CCO set value
= 5	Enable counter configuration		Integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynam.-normal 3 = dynam.-inverted
= 6	CCO configuration		Integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynam.-normal 3 = dynam.-inverted
= 7	Input A configuration		Integer	0 – 1	0 = normal (counter) 1 = invers (counter) 2 = normal (measure) 3 = invers (measure)
= 8	Input B configuration		Integer	0 - 1	0 = normal 1 = invers
= 9	Measure mode configuration		Integer	0 - 5	0 = impulse-manual 1 = impulse-autom. 2 = period-manual 3 = period-autom. 4 = frequency-manual 5 = frequency-autom.
= 10	Load measure value		Integer	0 - 65535	measure window in case of frequency measure and clock divider in case of impulsion length or period measure
= 11	Enable measure configuration		Integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynam.-normal 3 = dynam.-inverted
= 12	TCO configuration		Integer	0 - 3	0 = static-normal 1 = static-inverted 2 = dynam.-normal 3 = dynam.-inverted

Function block 'Exec'

Exec **FB:** Execution of a command for the H110 module



Function description:

This FB is used to send commands to the PCD2.H110 module.

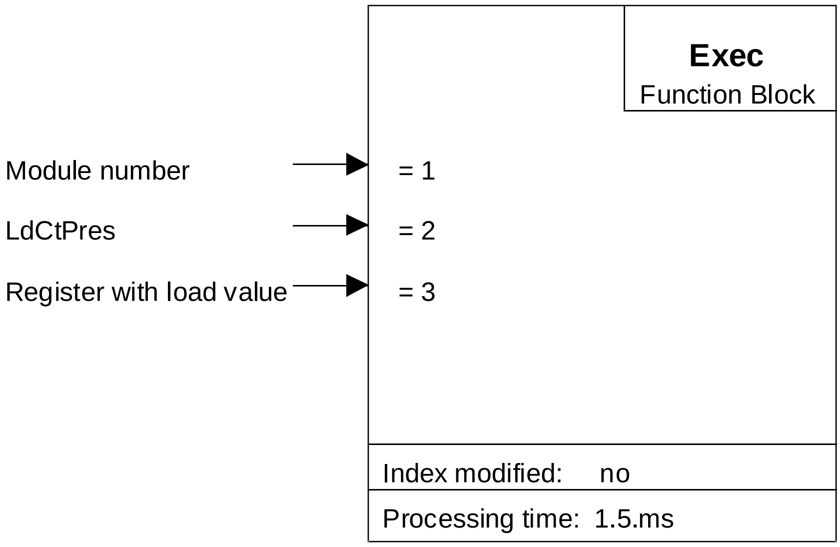
Module number (parameter 1) must be a 'K' constant (K 1…K 16).
The base address is defined in file 'D2H110_B.MBA'. These FBs support up to max. 16 PCD2.H110 modules per PCD system.

Individual commands (parameter 2) are described in the following pages.

The parameter of a command (e.g. counter preset value for command LdCtPres) is transferred from a register (parameter 3). If a command does not need a parameter (e.g. StartCt) any register or 'rNotUse' can be presented.

Individual instructions for PCD2.H110 (FB parameters)

LdCtPres **Command:** Load counter preset



Function description:

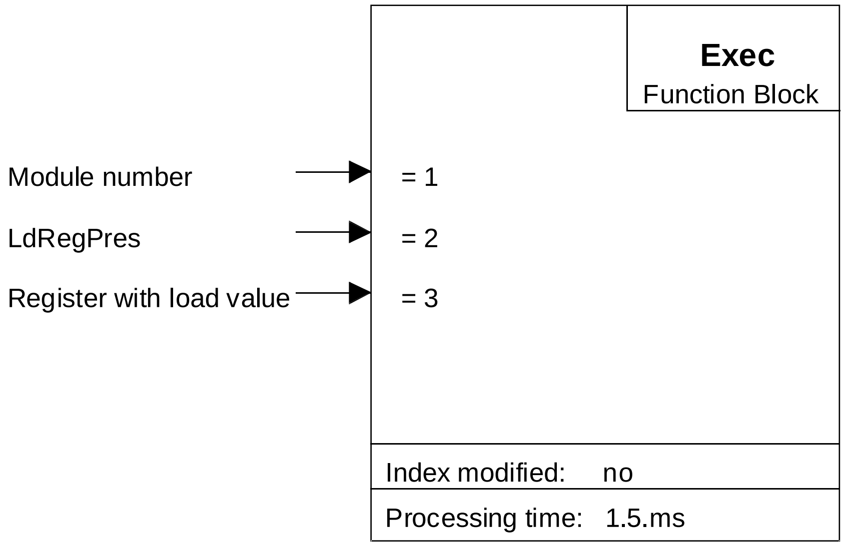
This command loads the preset counter value.

This value will be the one from which the counter will count up or down after a start counter command.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: LdCtPres				24 bit counter
= 3	Register with load value	R	integer	0 - 16777215	

LdRegPres **Command:** Load register preset



Function description:

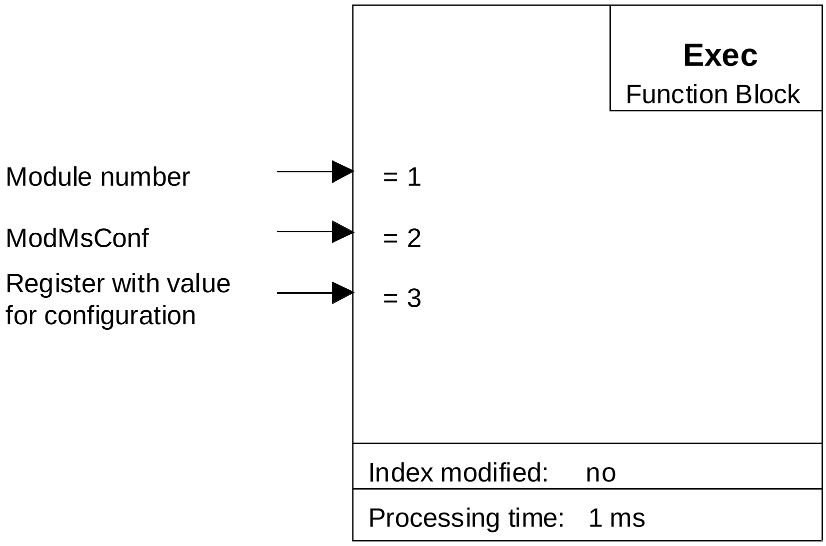
This command loads the register preset value.

This value will be compared to the counter value. When they'll be equal then the CCO will be set according to the CCO configuration.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: LdRegPres				24 bit counter
= 3	PCD reg. with load value	R	Integer	0 - 16777215	

ModMsConf **Command:** Measure mode configuration



Function description:

This command loads the measure register configuration with the mode of the measure chosen.

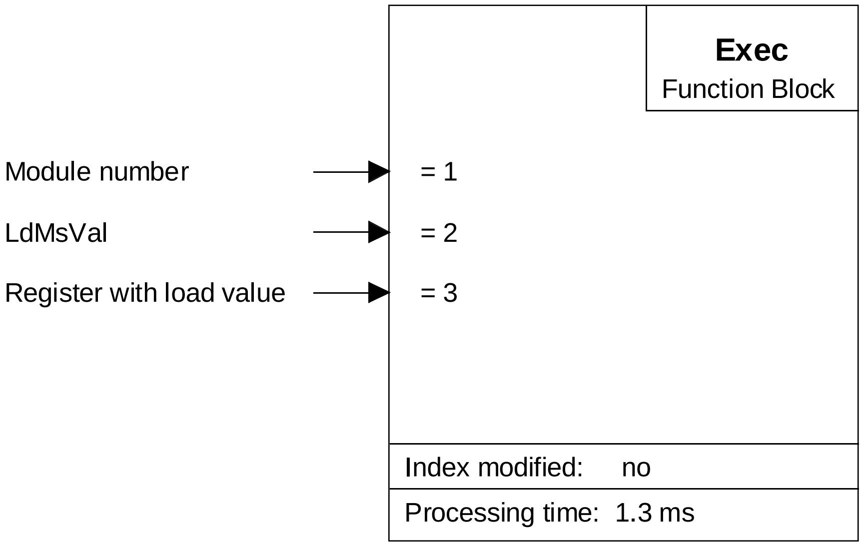
The manual mode means that after each measure a new start counter has to be done. In automatic mode the first start has to be done by the user program and then a condition like CCO is to be used to finish the measure.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: ModMsConf				
= 3	PCD register with value for the configuration	R	integer	0 - 5	default value = 0

This command is used only if the configuration should be changed during the program run. Normally, the configuration is done on the beginning of the program in the FB 'Init' (parameter 9).

LdMsVal **Command:** Load measure value



Function description:

This command loads the measure window in case of frequency measure and loads the clock divider value in case of period or impulsion length measure.

In case of frequency measure:

To get a resolution of 0.1%, you need to count 1000 impulsion on input A.
The measure window depend of the frequency to measure.
For instance to measure a frequency like:

100 kHz the measure window has to be:	10 ms
500 Hz the measure window has to be:	2000 ms

Formula:

f = frequency
t = time of measure window
R= resolution

$$t = \frac{1000 * R}{f}$$

In case of period or impulsion length measure, see next page:

In case of period or impulsion length measure

The time of the measure is always equal to a period of the signal or to the length of the impulsion.

For instance to measure a period of a frequency like:

500 Hz the measure time has to be 2 ms
270 µHz the measure time has to be 1 hour

With the formula as follow, it is possible to calculate the value of the time base. To get a resolution of 0.1 % you need at least 1000 impulsion.

t = period or impulsion length [s]
n = value to load
clk = clock number

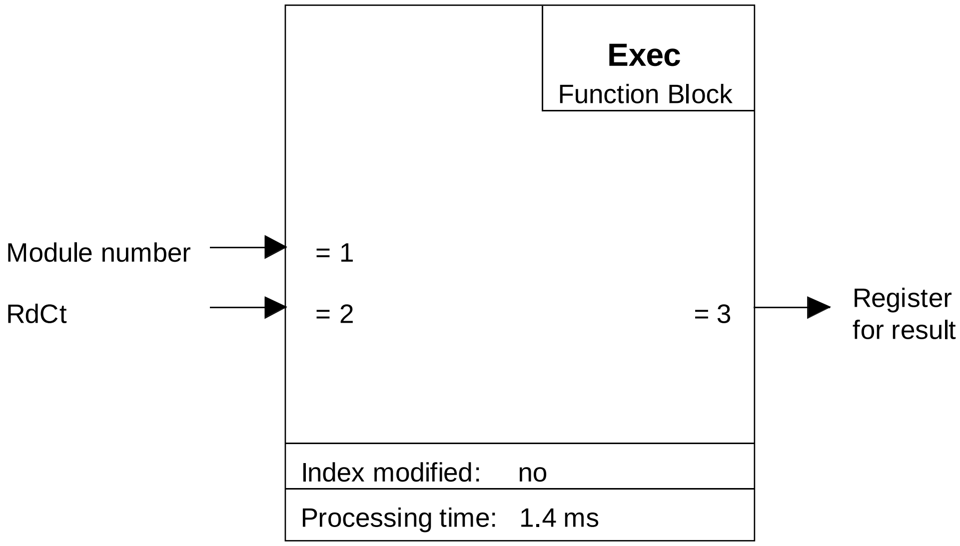
$$n = \frac{10^6 * t}{clk} - 1$$

You need to subtract 1 because when the counter "time base" reach 0, it needs 1µs to reinitialize itself.

Description of participating input/output elements:

Par.	**Designation/Function**	**Type**	**Format**	**Value**	**Com**
= 1	Module number	K		1 - 16	
= 2	Command: LdMsVal				
= 3	PCD reg. with load value	R	integer	0 - 65535	16 bit

RdCt **Command:** Read counter



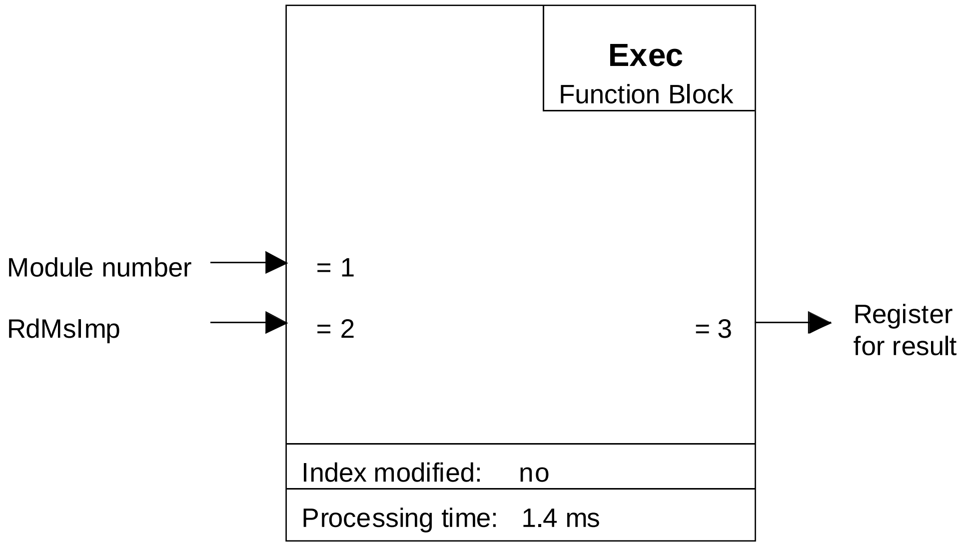
Function description:

This command reads the actual counter value.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Com.
= 1	Module number	K		1 - 16	
= 2	Command: RdCt				
= 3	PCD register for result	R	integer	0 - 16777215	24 bit

RdMsImp **Command:** Read measure in impulsion



Function description:

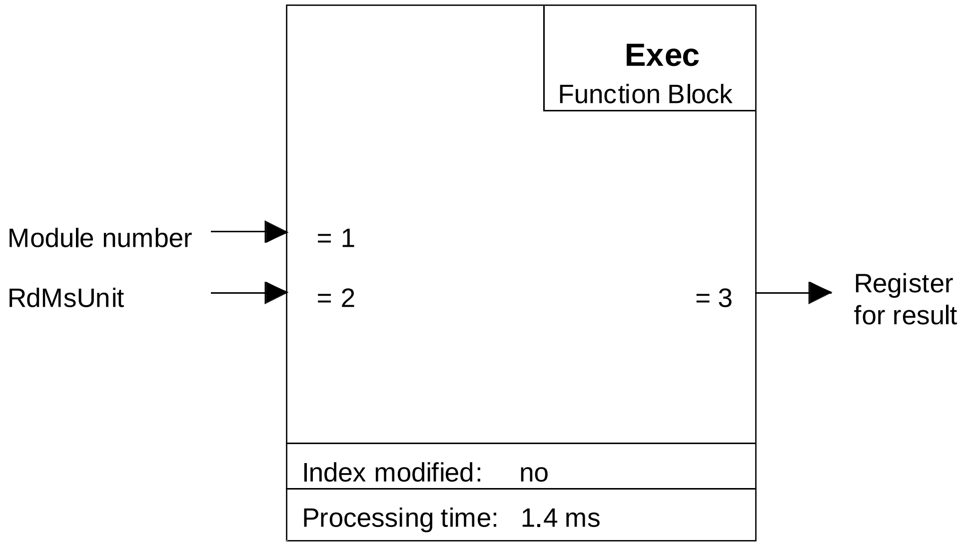
This command reads the result of the measure in impulsion.

For a frequency measure the result is the number of impulsion during the measure window was open. On the other hands for a period or impulsion length measurement the results is the number of impulsion between either two edge up for the period measure or one edge up and the edge down of the impulsion.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Com.
= 1	Module number	K		1 - 16	
= 2	Command: RdMsImp				
= 3	PCD register for result	R	integer	0 - 65535	16 bit

RdMsUnit **Command:** Read measure in Unit



Function description:

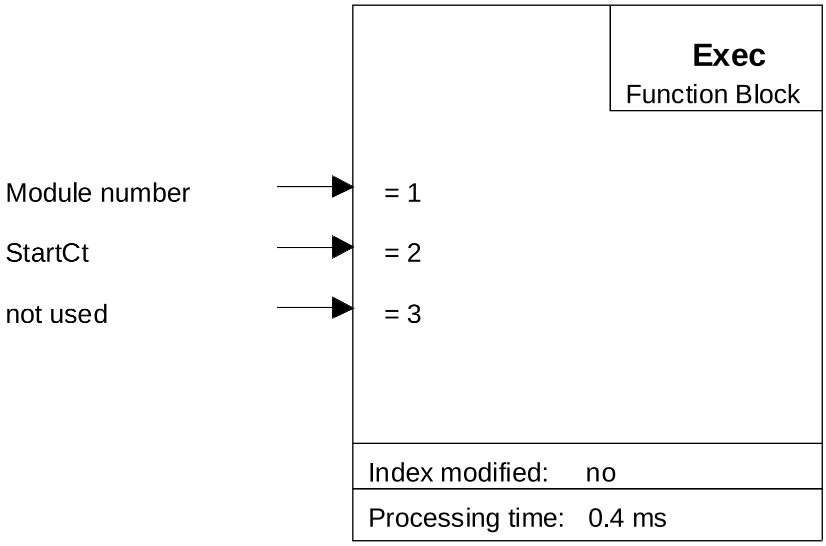
This command reads the result of the measure in specific unit.

For a frequency measure the result is in Hertz. For period or impulsion length measurements the results are seconds (s). In both cases the results is a floating point value.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Com.
= 1	Module number	K		1 - 16	
= 2	Command: RdMsUnit				
= 3	PCD register for result	R	FP		16 bit

StartCt **Command:** Start counter



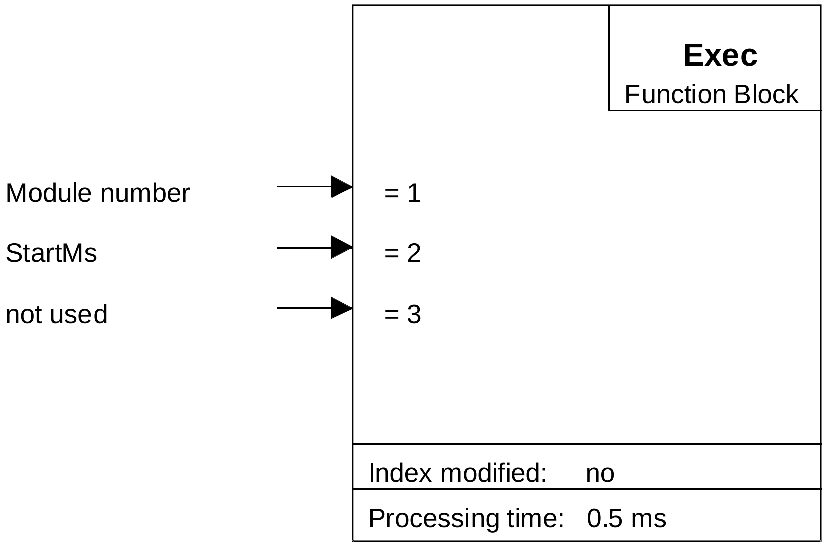
Function description:

This command starts the counter if the hardware enable is high or put the counter in a waiting position if the hardware enable is low.

In a manual mode this command has to be done after each measure, on the other hands in automatic mode the start has to be done once by the user program.

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: StartCt				
= 3	empty PCD register	R	integer	0	

StartMs **Command:** Start measure



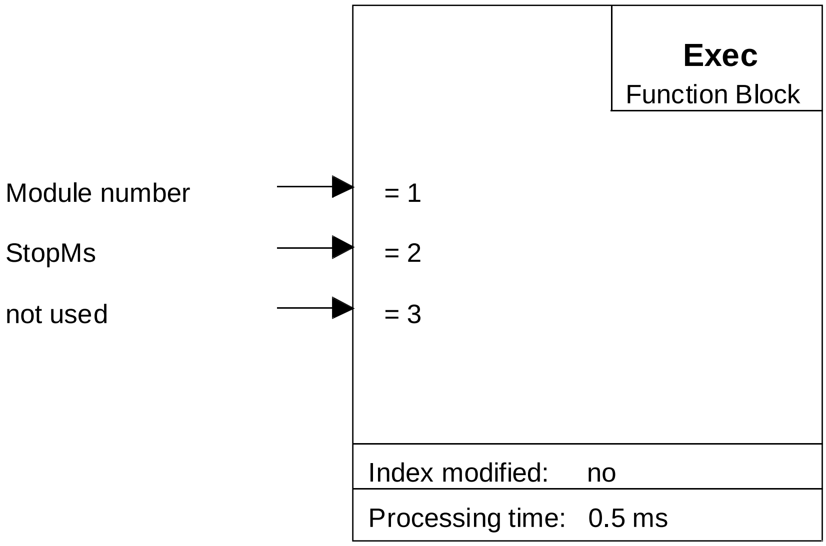
Function description:

This command starts the measure if the hardware enable is high or put the measure in a waiting position if the hardware enable is low.

In a manual mode this command has to be done after each measure, on the other hands in automatic mode the start has to be done once by the user program.

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: StartMs				
= 3	empty PCD register	R	integer	0	

StopMs **Command:** Stop measure



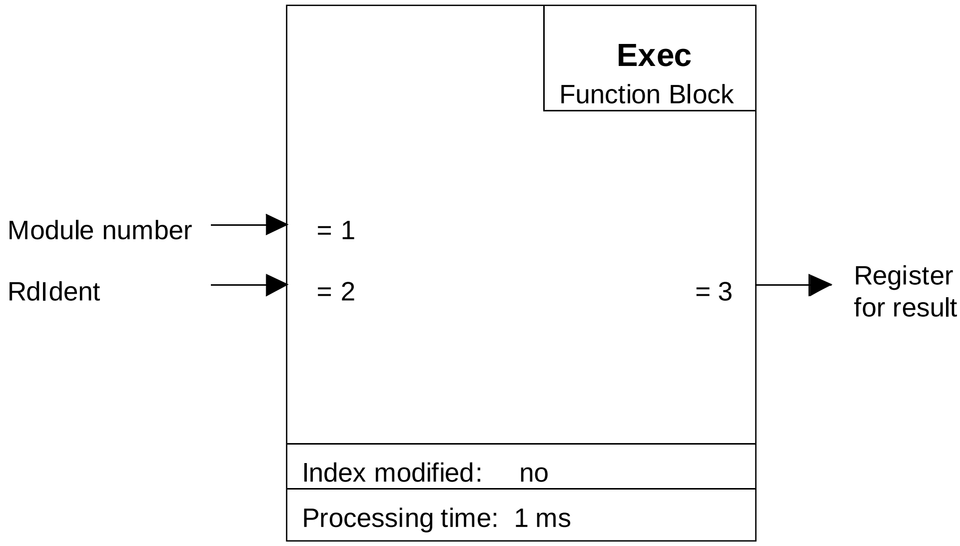
Function description:

This command stops the measure. The results in the two 16 bits counters are from the last measure finished or maybe completely wrong.

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: StopMs				
= 3	empty PCD register	R	integer	0	

For a new start, the command 'StartMs' is to execute again.

RdIdent **Command:** Read module identification



Function description:

This command can be used to check the correct function of the PCD2.H110 module and read the FPGA version. If the module is working properly, the value 17xx will be returned. See table below. If the module is faulty (or incorrectly addressed) the value 0 is read.

Description of participating input/output elements:

Par.	Designation/Function	Type	Format	Value	Comment
= 1	Module number	K		1 - 16	
= 2	Command: RdIdent				
= 3	PCD register for result	R	integer	12 bit	0 → faulty

Table of valid identifiers:

Value	FPGA version
2759	version HC0
1761	version HC1
1762	version HC2
1763	version HC3
...	...
3215	version HDF

Notes

Appendix B. Summary of all software elements for programming in FUPLA

In preparation

Notes

From :
Company :
Department :
Name :
Address :
Tel. :
Date :

To send back to :

SAIA-Burgess Electronics Ltd.
Bahnhofstrasse 18
CH-3280 Murten (Switzerland)
http://www.saia-burgess.com

BA : Electronic Controllers

Manual PCD2.H110

If you have any suggestions concerning the SAIA® PCD, or have found any errors in this manual, brief details would be appreciated.

Your suggestions :