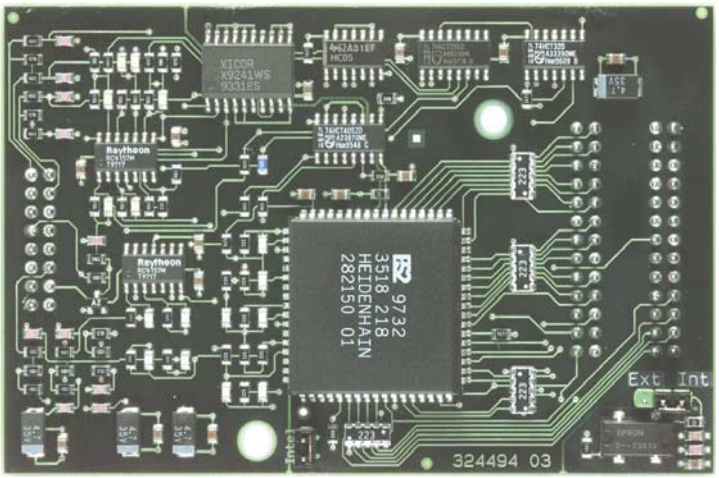


HEIDENHAIN

User´s Manual IK 410V

Counter Board with
Bus Interface for
Incremental 1 V_{PP} Encoders

11/97

Contents

Items Supplied

- IK 410 V counter card
- User's Manual
- Programming examples

Note on the EMC Guideline 89/336/EWG

Compliance with EMC Guideline 89/336/EWG was tested with a COMPAQ DESKPRO 386/20e computer.

Important

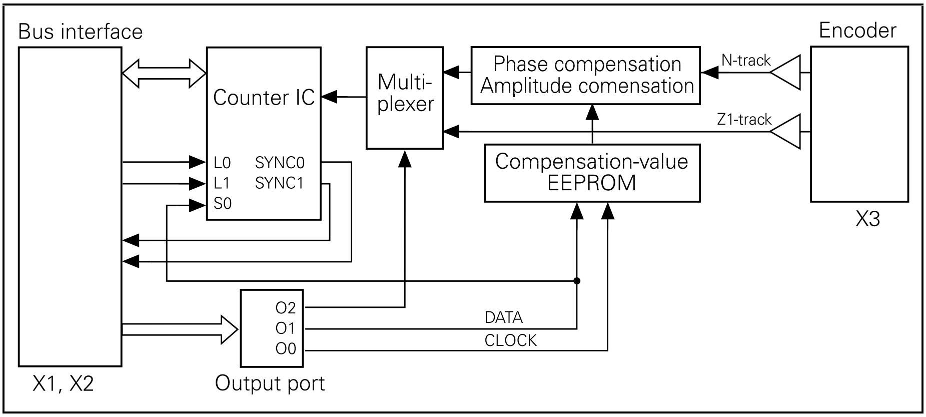
Danger to internal components!
When handling components that can be damaged by **electrostatic discharge (ESD)**, follow the safety recommendations in DIN EN 100 015. Use only antistatic packaging material. Be sure that the work station and the technician are properly grounded during installation.

Technical Description of the IK 410 V

The **IK 410 V** is an interpolation and counter card for distance and angular measurement using a HEIDENHAIN encoder with sinusoidal voltage signals. In addition to the input for the incremental signals, the card also has an input for the commutation track of a motor encoder (one sine/cosine per revolution). Thus the card is also suited to motor control.

The **IK 410 V** is inserted directly onto the PCB of a control or subsequent electronics. There are two operating modes. In the period counter mode the number of signal periods is counted in a 32-bit register. In the interpolation mode the encoder input signal is subdivided 1024-fold (10 bits) and is then added to the number of signal periods (32 bits). The measured values are called and latched either through external latch inputs or with software or timers, as well as by crossing over the reference marks with port addressing. The 42-bit measured value is formed from the interpolation value (10 bits) and the value of the period counter (32 bits). The measured values are stored in 48-bit registers, whereby the upper bits are expanded in two's complement representation with sign. The measured values can then be read directly via the 16-bit data bus of the control or the subsequent electronics.

Block diagram of the IK 410 V



The IK 410 V counter card has an interface similar to the ISA bus (connector X1 and X2), i.e. a 16-bit data bus,
6-bit address bus, as well as the signals -RD, -WR, -CS, -RESET and INTERRUPT.

The signals L0, L1 and SYNC0, SYNC1 are also available at the interface. With the signals L0 and L1 you can latch the counter value in the register banks 0 or 1. With the signals SYNC0 and SYNC1 it is possible to operate several IK 410 V cards simultaneously.

The compensation-value EEPROM can be described using an output port.

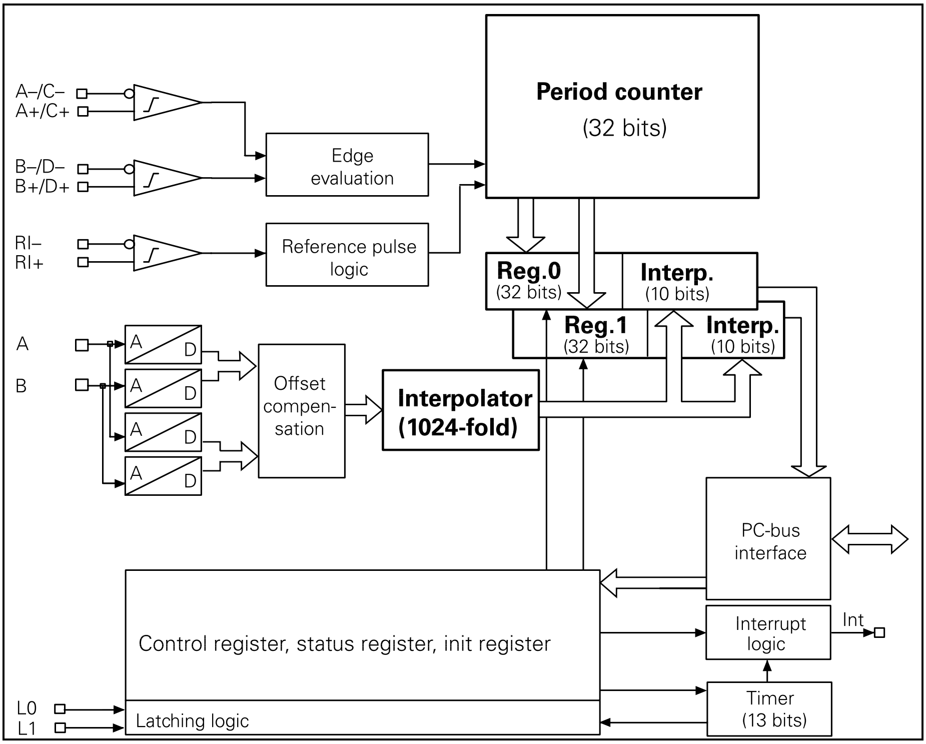
The offset of the sinusoidal encoder signals can be adjusted using software with data registers in the counter; the phase and amplitude with electronic potentiometers.

The output port also allows to switch between the 0° and 90° signals of the N and Z1 tracks of a HEIDENHAIN encoder using a multiplexer.

Time to access measured values

The time required for accessing the measured values is approx. 30 µs (depending on the clock frequency).

Block diagram of the counter IC



Description of signals *)

D0 to D15	Data bus
A0 to A5	Address bus
CLOCK	Input for clock (if jumper BR2 on EXT)
–RD	Input read strobe
–WR	Input write strobe
–CS	Input chip select
–RESET	Input reset
–INT	Output interrupt
–L0	Input hardware latch for register 0
–L1	Input hardware latch for register 1
–SYNC0	Output for synchronization (L0 assigned)
–SYNC1	Output for synchronization (L1 assigned)
A+	0° signal + (N track)
A–	0° signal – (N track)
B+	90° signal + (N track)
B–	90° signal – (N track)
RI+	Reference pulse +
RI–	Reference pulse –
C+	0° signal + (Z1 track)
C–	0° signal – (Z1 track)
D+	90° signal + (Z1 track)
D–	90° signal – (Z1 track)

*) – inverse signal

Power supply

+ 5 V ± 5%, 120 mA
+15 V ± 5%, 40 mA
–15 V ± 5%, 40 mA

Pin Layout

Interface to subsequent electronics

X1: 26-pin AMP connector (male)

Pin number	Signal
1a	Gnd
1b	*Free*
2a	Gnd
2b	*Free*
3a	–SYNC1
3b	*Free*
4a	–SYNC0
4b	*Free*
5a	–L1
5b	+5 V*
6a	–L0
6b	+5 V*
7a	–Int
7b	+5 V*
8a	–RD
8b	+5 V*
9a	–WR
9b	Gnd
10a	A5
10b	A4
11a	Gnd
11b	A3
12a	A2
12b	A1
13a	A0
13b	Gnd

* External power supply

X2: 26-pin AMP connector (male)

Pin number	Signal
1a	Gnd
1b	D15
2a	D14
2b	D13
3a	D12
3b	Gnd
4a	D11
4b	D10
5a	D9
5b	D8
6a	Gnd
6b	D7
7a	D6
7b	D5
8a	D4
8b	Gnd
9a	D3
9b	D2
10a	D1
10b	D0
11a	Gnd
11b	–CS
12a	–Res
12b	Gnd
13a	Clk
13b	Gnd

Encoder input

You can connect HEIDENHAIN linear or angle encoders with sinusoidal voltage signals A and B (N track) and reference mark signal R to the IK 410 V. In addition the commutation signals C and D (Z1 track) can also be evaluated with motor encoders.

Signal amplitudes A, B, C, D (0°, 90°) R (reference mark)	 $0.6\ V_{PP}$ to $1.2\ V_{PP}$ 0.2 V to 0.85 V
Signal levels for error message	$\leq 0.22\ V_{PP}$
Max. input frequency	350 kHz
Cable length[1)]	Max. 60 m (supply voltage 5.0 V

1) Cable lengths to 150 m are possible if it can be guaranteed that the encoder will be supplied by 5 V from an external power source.
In this case the input frequency is reduced to max. 250 kHz.

X3: 16-pin AMP connector (male)

Pin number	**Signal**
1a	A+
1b	A–
2a	Gnd
2b	B+
3a	B–
3b	Gnd
4a	RI+
4b	RI–
5a	Gnd
5b	C+
6a	C–
6b	D+
7a	D–
7b	+5 V*
8a	+15 V*
8b	–15 V*

* External power supply

Adjusting the Encoder Signals

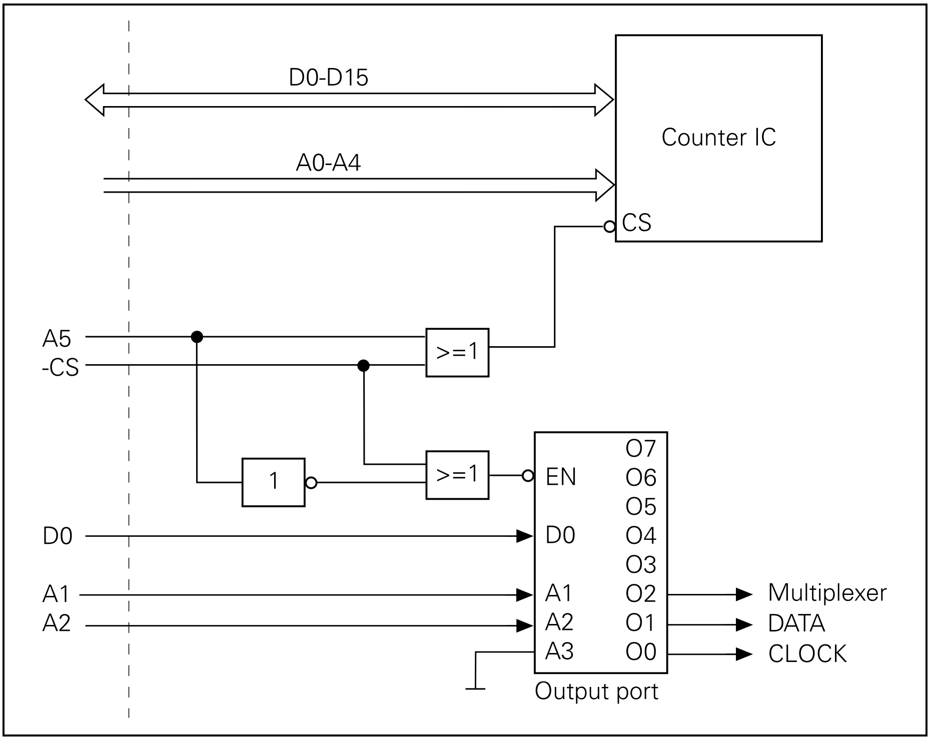
Encoder signals can be adjusted as follows:

- Phase and amplitude can be adjusted with electronic potentiometers
- Symmetry (offset) can be adjusted in the counters with offset registers

The potentiometer is I²C-bus-controlled. Examples on generating the control sequences can be found in the source code of the programs POTIS.CPP as well as POTIS.PAS and ADJUST.PAS.

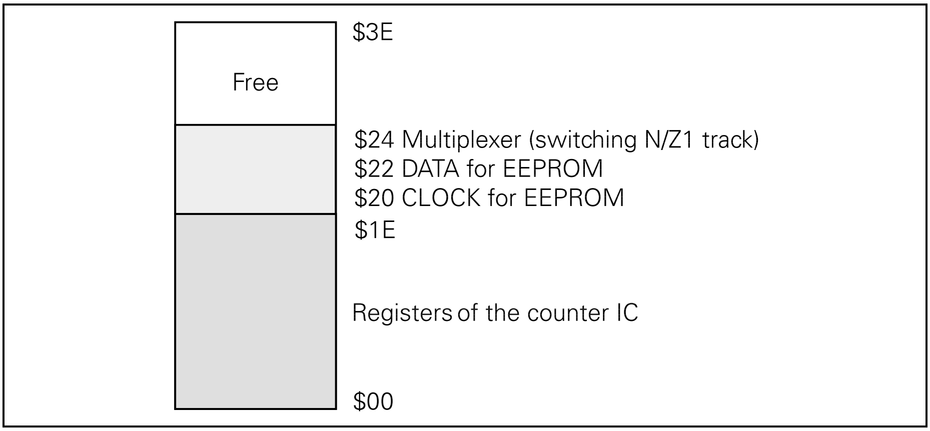
The compensation values for phase and amplitude are stored in the ICs of the electronic potentiometers in nonvolatile memory. The offset registers in the counter ICs, however, are **volatile** (the information will be lost when the power is removed). For this reason, the offset compensation values are stored in an EEPROM. After switch-on, the offset compensation values must be loaded from the EEPROM into the offset registers of the counters. Two functions in the files IIC.CPP and IIC.PAS fulfill these tasks. The Store Offset or store_offset function stores the offset compensation values in the EEPROM. The Load Offset or load_offset function copies the offset compensation values from the EEPROM into the offset register of the counters.

Address Space and Data Format



The counter IC requires the addresses A0 to A4 (address line A5=0), to read and write its registers, i.e. the address range: $0 to $1E.

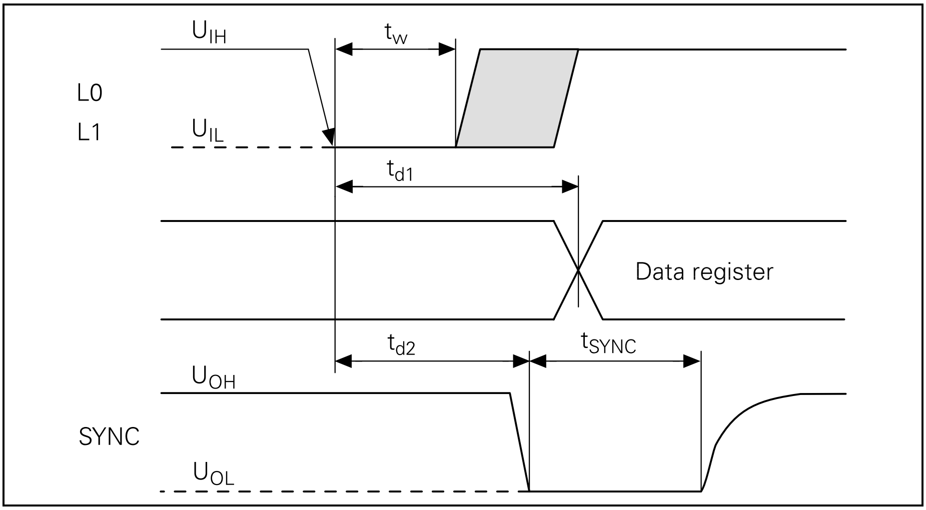
The output port is contacted via the addresses A1 and A2 and the data line D0 (address line A5=1), i.e. address range: $20 to $3E.


$3E
Free
$24 Multiplexer (switching N/Z1 track)
$22 DATA for EEPROM
$20 CLOCK for EEPROM
$1E
Registers of the counter IC
$00

Data format
You can select whether you wish to work with INTEL or MOTOROLA data format via jumper BR2 (see Dimensions).

Clock, Time Diagram and Voltage Levels

You can select via the jumper BR1 whether the IK 410V will have the internal clock frequency (20 MHz) or an external clock frequency (from 3 MHz to 20 MHz) (see Dimensions).



Designation	**Min.**	**Max.**
$t_{CLK\ low}$	20 ns	–
$t_{CLK\ high}$	20 ns	–
f_{CLK}	3 MHz	20 MHz
U_{IL} (V)	–3.0	0.9
U_{IH} (V)	3.15	30
t_{d1} (µs)	–	$t_{d1} = 8 * 1/CLK + 10*t_{AD} + t_{DIFF}$1)
t_{d2} (ns)	6x t_{CLK}	8x t_{CLK}
t_w (ns)	250	–
t_{SYNC}(µs)	4x t_{CLK}	–
U_{OL} (V)	0	0.8 (U_{OH}=4 V – 12 V); 1.0 (U_{OH}=12 V – 32 V)
U_{OH} (V)	4	32
I_{OL} (mA)	–	40

1) t_{AD} = 2 020 ns CLK: clock in (1/s)
For CLK > 16 MHz: t_{DIFF} = 38 *1/CLK
For CLK ≤ 6 MHz: t_{DIFF} = 32*1/CLK
Example: CLK = 20 MHz t_{max} = 0.4 µs + 20.2 µs + 1.9 µs = 24.3 µs

Registers

The circuit diagram of the counters on the last page will help you in the following description.

Important note for programmers:
The IK 410 V is accessed by reading and writing data words stored in registers. For this reason, only even port addresses may be addressed with word-write and word-read commands.

Overview of registers

Registers of the counter

Address(hex) B0 to B4	Write mode	Read mode
00		Data Register 0, LS-Word
02	No function	Data Register 0
04		Data Register 0, MS-Word
06		Data Register 1, LS-Word
08	No function	Data Register 1
0A		Data Register 1, MS-Word
0C Low Byte	Initializing Register 1	Initializing Register 1
High Byte	Initializing Register 2	Initializing Register 2
0E Low Byte	Control Register 1	Status Register 1
High Byte	No function	Status Register 2
10 Low Byte	Reference Mark Register	No function
High Byte	No function	Amplitude Register
12 Low Byte	Enable Register for Measured Value Latching	No function
High Byte	Axis Cascading	No function
14 Low Byte	Interrupt Enable Register	Interrupt Status Register 1
High Byte	No function	Interrupt Status Register 2
16 Low Byte	Offset Register for 0° Signal	Amplitude for 0°signal
High Byte	No function	Amplitude for 0° signal
18 Low Byte	Offset Register for 90° Signal	Amplitude for 90° signal
High Byte	No function	Amplitude for 90° signal
1A Low Byte	Timer Register, LSB	No function
High Byte	Timer Register, MSB	No function
1C Low Byte	Control Register 2	Status Register 3
High Byte	No function	Code Register
1E Low Byte	Control Register 3	Status Register 4
High Byte	No function	No function

Registers of the output port

Address (hex) A0 to A5	**Write mode**	**Read mode**
20	I^{2}C-bus line SCL: clock for EEPROM (signals must be programmed inverted)	–
22	I^{2}C-bus line SDA: data for EEPROM (signals must be programmed inverted)	–
24	Switching the multiplexer for the encoder signals	–

Data registers for the counters

The measured values are stored by the IK 410 V in 48-bit registers. Two data registers are available for each axis: Data Register 0 (00h to 04h) and Data Register 1 (06h to 0Ah). The measured values of the IK 410 V are composed of the 10-bit interpolation value and the 32-bit value of the period counter. Only 42 bits of the 48-bit registers are thus used for the measured value. The upper 6 bits are expanded with sign in two's complement form.

The data width of 48 bits can be shortened to 32 bits via Initializing Register 1 (0Ch), bit D6.

Initializing Register 1 (0Ch), bit D7 can also be used to define whether the measured value is formed only from the value of the period counter (D0 to D9 are not defined) or from the value of the period counter and the interpolation value.
The counter values can be stored in the data registers in the following ways:

- Software latch
- External inputs
- Timers
- Reference marks

The circuit diagram of the counters (see last page) illustrates the action of the various latch signals.

Bit D0 or D1 in Status Register 1 (0Eh) can be used to poll whether the measured value was stored in the data registers. As long as bit D0 or D1 is set, no further measured value can be stored until the most significant word of the measured value is read. (Exception: via Control Register 2, bit D6 or D7, the latch is enabled without the measured value having been read out.) In 48-bit mode these are the Data Registers 04h or 0Ah, and in 32-bit mode the Data Registers 02h or 08h. When the measured value has been read, bit D0 or D1 in Status Register 1 (0Eh) is reset.

If the counter is stopped, or the value stored through crossing over the reference marks, bits D0 to D9 will contain the fixed value 256.

0Ch: Initializing Register 1 (write mode)

Bit	Function
D0	**Operation with /without interpolation** 0 = Operation as period counter (w/o interpolation – data bits D0 to D9 are not defined) 1 = Measured value is formed from the value of the period counter and the interpolation value.
D1	0
D2	**Timer** 0 = Reset and stop timer 1 = Start timer
D3	0
D4	0
D5	0
D6	**Latch enable** 0 = Mode: 32-bit Register Reading bits D24 to D31 resets status bit D0 or D1 in Status Register 1 (0Eh). 1 = Mode: 48-bit Register Reading bits D40 to D47 resets status bit D0 or D1 in Status Register 1 (0Eh).
D7	**Counting direction** The counting direction determines whether the counters count positive (normal) or negative (inverse) when the traverse direction is positive. 0 = Normal counting direction 1 = Inverse counting direction Inverse counting direction is permitted only in the period counter mode. In operation with interpolation, inverse counting direction will result in faulty gating of the interpolation value and the period counter value.

0Ch: Initializing Register 2 (write mode)

Bit	Function
D8 D9	**Only in operation as period counter:** **edge evaluation** The two incremental encoder signals (0°and 90° el.) provide a maximum of four edges for evaluation per signal period. The counters can be programmed to count one, two or four edges per signal period. 00 = 1-fold 01 = 2-fold 11 = 4-fold In operation with interpolation value, 1-fold evaluation is automatically set.
D10	**Only in operation as period counter:** **Counting mode** 0 = Linear counting mode -2^{41} to $+2^{41} - 1$ 1 = Angle counting mode as defined below in D11 For angle encoders with 36 000 or 360 000 lines per revolution.
D11	**Only with angle display:** **Counting mode** 0 = 17 999 to –18 000 1 = 179 999 to –180 000
D12	0
D13	0
D14 D15	**Measured value latch with reference pulse** 0 = 1st reference mark stores in Data Register 0 1 = 1st reference mark stores in Data Register 1 0 = 2nd reference mark stores in Data Register 0 1 = 2nd reference mark stores in Data Register 1

0Ch: **Read mode:** Bits D0 to D15: Read back of Initializing Registers 1 and 2

0Eh: Control Register 1 (write mode)

Bit	Function
D0	1 = Software latch: measured value in Data Register 0
D1	1 = Software latch: measured value in Data Register 1
D2	1 = Software latch in all data registers (must be enabled in Latch Enable Register)
D3	1 = Start counter
D4	1 = Stop counter
D5	1 = Delete counter
D6	1 = Clear encoder error (frequency exceeded)
D7	Delete amplitude value register
D8 D9 D10 D11 D12 D13 D14 D15	No function

0Eh: Status Register 1 (read mode)

Bit	**Function**
D0	Status for software latch in Register 0 1 = Measured value ready
D1	Status for software latch in Register 1 1 = Measured value ready
D2	No function
D3	No function
D4	1 = Counter is stopped
D5	I²C-bus line SDA (input)
D6	1 = Encoder error (frequency exceeded)
D7	No function

0Eh: Status Register 2 (read mode)

Bit	Function
D8	1 = Reference mark traverse is active
D9 D10 D11 D12	No function
D13	Logic level for 0° signal
D14	Logic level for 90° signal
D15	Logic level for reference mark

10h: Reference Mark Register (write mode)

HEIDENHAIN linear and angle encoders can feature one or several reference marks. HEIDENHAIN recommends in particular measuring systems with distance-coded reference marks.

In the case of power interruption the link between the encoder position and the displayed position value is lost. This reference can be reestablished after power-on with the encoder reference marks.

When a reference mark is traversed a signal is generated which marks this position on the scale as reference point. This reference point serves to restore the link between the axis positions and the display values last defined. For linear encoders with distance-coded reference marks a reference can be reestablished after max. 20 mm traverse.

Bit	Function while traversing reference mark
D0	1 = Start counter
D1	1 = Stop counter
D2	1 = Clear counter
D3	1 = Latch measured value
D4	1 = Latch measured value when second reference mark is crossed over
D5	1 = Clear counter each time a reference mark is crossed over
D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

Evaluating distance-coded reference marks

Measuring systems with distance-coded reference marks have reference marks spaced at a regular interval over the entire length of the measuring range. Between each two reference marks is another, whose distance to each of the first two changes with each interval (see below). Each of these distances is a multiple of the grating period and each is unique. Thus only two consecutive reference marks need to be traversed following a power interruption for the link between axis positions and display values to be reestablished.

The example below illustrates how these distance-coded reference marks are evaluated:
If the nominal increment is 1000 signal periods, the reference marks are arranged in the following pattern:

```
     501       499        502       498       503         497
|-----------|------------|-----------|-----------|------------|------------|----
0         501      1000      1502      2000       2503       3000
```

First you need to initialize the reference mark register 10h as follows:

- Cancel and start counter with traverse of the first reference mark (Bit D0 = 1 and D2 =1).
- Latch counter with traverse of the second reference mark (Bit D4 = 1).

To calculate the absolute position only the distance between the reference marks in signal periods is required. This distance in increments, called DIFF in the description below, must be divided by 1024.

DIFF = DISTANCE_IN_INCR : 1 024

To calculate the absolute position, the distance (OFFSET) between the 1st reference mark on the scale ("0" position in the drawing) and the 1st reference mark crossed over must be determined.

There are our possibilities:

1. Positive traverse direction and DIFF > 500
   OFFSET = (DIFF – 501) • 1 000
2. Positive traverse direction and DIFF < 500
   OFFSET = (500 – DIFF) • 1 000 – DIFF
3. Negative traverse direction and $|\text{DIFF}|$ > 500
   OFFSET = (DIFF – 501) • 1 000 + DIFF
4. Negative traverse direction and $|\text{DIFF}|$ < 500
   OFFSET = (500 – DIFF) • 1 000

The absolute position in increments can be calculated as follows:

ABS_POS_INCR = ACT_POS + PRESET + DISTANCE

ACT_POS: Distance (in increments) between the current position and the first traversed reference mark.

PRESET: Position (in increments) assigned to the first reference mark of the scale at datum set (0 position in the drawing).

DISTANCE: Distance (in increments) between the first reference mark on the scale and the first traversed reference mark.

DISTANCE = OFFSET • 1 024

The absolute position is calculated as follows — e.g. for a scale with signal period 0.02 mm:

$$\text{ABS\_POS\_MM} = \frac{\text{ABS\_POS\_INCR} \cdot 0.02}{1024}$$

For angle encoders:

$$\text{ABS\_POS\_DEGREE} = \frac{\text{ABS\_POS\_INCR} \cdot 360^\circ}{1024 \cdot \text{LINES\_PER\_REV.}}$$

You will find an application example in "TURBO PASCAL" for evaluating reference marks in the source code of the file CNT_2.PAS under (*Distance-coded ref.marks*).

10h: Amplitude Value Register (read mode)

Bit	Function
D0 D1 D2 D3 D4 D5 D6 D7	No function
D8 D9	**Current amplitude** A new amplitude value is determined with each measured value latch. D9 D8 0 0 Normal amplitude $0.47\ V_{PP} < U_e < 1.41\ V_{PP}$ 0 1 Low amplitude $0.23\ V_{PP} < U_e < 0.47\ V_{PP}$ 1 0 High amplitude $U_e > 1.41\ V_{PP}$ 1 1 Erroneously low amplitude $U_e < 0.23\ V_{PP}$ Before being read, the amplitude value should be frozen by bit D4 in Control Register 2. The Amplitude Value Register is reset by bit D7 in Control Register 1.
D10 D11	**Minimum value of the amplitude** Coding and read mode: see bits D8 and D9. If the current amplitude value is lower than the stored minimum value more than four times in a row, the IK will replace the old minimum value with the current amplitude value.
D12 D13 D14 D15	No function

12h: Enable Register for measured value latch (write mode)

Bit	Function
D0	Enable L0 for Data Register 0
D1	Enable L0 via delay circuit (125 ns) for Data Register 0
D2	1 = Enable "software latch in all data registers" for Data Register 0
D3	1 = Enable "software latch via timers " for Data Register 0
D4	Enable L1 for Data Register 1
D5	Enable L1 via delay circuit (125 ns) for Data Register 1
D6	1 = Enable "software latch in all data registers " for Data Register 1
D7	1 = Enable "software latch via timers " for Data Register 1

The circuit diagram of the counter ICs on page 46 shows the functions of the individual bits.

12h: Register for axis cascading (write mode)

Bit	Function
D8	1 = Enable pin L 0 for SYNC1
D9	1 = Enable software latch for SYNC1
D10	1 = Enable timer strobe for SYNC1
D11	0
D12	1 = Enable pin L1 for SYNC0
D13	1 = Enable software latch for SYNC0
D14	1 = Enable timer strobe for SYNC0
D15	0

12h: Read mode has no function

14h: Interrupt Enable Register (write mode)

The interrupt logic is programmed through the Interrupt Enable Register. The interrupt can be caused by the measured value latch in Register 0, Register 1 or the Timer strobe. Each interrupt source can be programmed independently of the others. If several interrupts are received at the same time, the following priorities are used:

- Highest priority: measured value latch via Register 0
- Second-highest priority: measured value latch via Register 1
- Lowest priority: measured value latch via timer strobe

After an interrupt, no further interrupts can follow until Interrupt Status Register 2 has been read.

Bit	Function
D0	1 = Enable interrupt 0 with measured value latch via Register 0
D1	1 = Enable interrupt 1 with measured value latch via Register 1
D2	1 = Enable interrupt 2 for timer strobe
D3	1 = Interrupt is generated (D0 = D1 = D2 = 0)
D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

The circuit diagram of the **counter ICs** on page 60 shows the functions of the individual bits.

14h: Interrupt Status Register 1 (read mode)

The currently active interrupt is displayed in this register (only one bit can be set). Reading this register resets the currently active interrupt and deletes the associated status bit so that the next interrupt can be triggered by a negative edge.

Bit	Function
D0	1 = Interrupt 0 is active, there was a measured value latch via Data Register 0
D1	1 = Interrupt 1 is active, there was a measured value latch via Register 1
D2	1 = Interrupt 2 is active; there was a measured value latch via the timer strobe
D3 D4 D5 D6 D7	No function

14h: Interrupt Status Register 2 (read mode)

In this register, all pending interrupts are shown (i.e., the active interrupt and the interrupts still to be executed). Several bits may therefore be set at the same time.

Bit	Function
D8	1 = Interrupt 0 is pending but not yet executed
D9	1 = Interrupt 1 is pending but not yet executed
D10	1 = Interrupt 2 is pending but not yet executed
D11 D12 D13 D14 D15	No function

16h: Offset Register for 0° signal (write mode)

This register contains the 7-bit offset value for the 0° signal in two's complement form. From this there results a maximum compensation of ± 63.

The offset values can only be written if one of the status bits D5 or D6 in Status Register 3 has the value 0.

Functional principle:

The IK 410 V adds the offset compensation values to the digital values of the 0° signal (0 to 1023) and the 90° signal. In case of overflow the value is limited to 1023 or 0.

Bit	Function
D0 D1 D2 D3 D4 D5 D6	Offset value for the 0° signal in two's complement form
D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

The offset register in the counters is **volatile**. For this reason the offset values are stored in an EEPROM. After switch-on, the offset values must be loaded from the EEPROM into the offset registers of the **counters** (see functions "store_offset" and "load_offset").

16h: Amplitude for 0°signal (read mode)

After each analog-digital conversion the result is stored by the IK 410 V. Before reading, the values must be frozen by bit D4 in Control Register 2.

Bit	Function
D0	
D1	
D2	
D3	
D4	
D5	Amplitude for the 0° signal
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	No function
D14	
D15	

18h: Offset Register for 90° signal (write mode)

This register contains the 7-bit offset value for the 90° signal. For a description of the function, see "16h: Offset register for 0° signal"

Bit	Function
D0 D1 D2 D3 D4 D5 D6	Offset value for the 90° signal in two's complement form
D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

18h: Amplitude for 90°signal (read access)

After each analog-digital conversion the result is stored by the IK 410 V. Before reading, the values must be frozen by bit D4 in Control Register 2.

Bit	Function
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9	Amplitude for the 90° signal
D10 D11 D12 D13 D14 D15	No function

1Ah: Timer Register (write mode)

The 13-bit timer value is stored in Registers 1Ah and 1Bh. The Timer Register can therefore hold values from 0 to 8191. The cycle time is indicated in µs; one µs must be subtracted from the desired cycle time.

Example

Desired cycle time: 1 ms (= 1000 µs)
Value to be programmed: 1000 – 1 = 999
Note that writing this register does not start the timer. The timer is started by setting bit D2 in Initializing Register 1 (0Ch). Furthermore, the corresponding bits must be set in Enable Registers 12h, 13h or 14h.

Bit	Function
D0	Timer value
D1	
D2	
D3	
D4	
D5	
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	No function
D14	
D15	

1Ah: Read mode has no function

1Ch: Control Register 2 (write mode)

Bit	Function
D0 D1 D2 D3	Program a fixed value ≥ 8
D4	1 = Freeze amplitude for 0° signal (16h) and 90° signal (18h), as well as Amplitude Value Register (10h High Byte). This bit must be set to prevent the register values from being changed during readout. Whether the register values are frozen can be read in bit D4 in Status Register 3.
D5	0
D6	1 = New latch possible via Data Register 0 without having to fetch the measured value beforehand. In this mode it is possible to change the measured value while reading.
D7	1 = New latch possible via Data Register 1 without having to fetch the measured value beforehand. In this mode it is possible to change the measured value while reading.
D8 D9 D10 D11 D12 D13 D14 D15	No function

1Ch: Status Register 3 (read mode)

Bit	Function
D0 D1 D2 D3	Cannot be read
D4	1 = Amplitude for 0° signal (16h) and 90° signal (18h) as well as Amplitude Value Register (10h High Byte) are frozen and can be read.
D5	0 = Offset Register for the 0° signal has been written.
D6	0 = Offset Register for the 90° signal has been written.
D7	No function

1Ch: Code Register (read mode)

Bit	Function
D8	
D9	
D10	
D11	IC code: 8 (Counter Chip G26: Id. Nr. 282 150-01)1)
D12	9 (Counter Chip G38: Id. Nr. 315 860-01)1)
D13	13 (Counter Chip G49: Id. Nr. 333 033-01)1)
D14	
D15	

1) Chip designation from HEIDENHAIN

1Eh: Control Register 3 (write mode)

Bit	Function
D0 D1	Fixed value 0
D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

1Eh: Status Register 4 (read mode)

Bit	Function
D0	No function
D1	Logic level on pin L0
D2	Logic level on pin L1
D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

Description of the output port registers

20h: I²C-bus line SCL: Clock for EEPROM (write mode)

Bit	Function
D0	0 = High levels at CLOCK input of the EEPROM 1 = Low levels at CLOCK input of the EEPROM

22h: I²C-bus line SDA: DATA for EEPROM (write mode)

Bit	Function
D0	0 = High levels at DATA input of the EEPROM 1 = Low levels at DATA input of the EEPROM

24h: Switching the multiplexer for the encoder signals (write mode)

Bit	Function
D0	0 = N-track 1 = Z1-track

Programming

A floppy disk with programming examples is supplied as an accessory with the IK 410 V. There are three directories on this disk:

BC
This directory contains simple examples in "BORLAND C". The hardware for these examples consists of an IBM PC-compatible AT-bus PCB with an IK 410 V.

BCPP
This directory contains programming examples in "BORLAND C++" with a RAM memory map of the registers. The hardware necessary for these examples is an IBM PC-compatible AT-bus PCB with two IK 410 Vs. The program is designed so that up to 16 PCBs can be controlled with two IK 410 Vs each.

TP
This directory contains programming examples in "TURBO PASCAL" with a RAM memory map of the registers. The hardware for these examples is also an IBM PC-compatible AT-bus PCB with two IK 410 Vs.

Simple examples in "BORLAND C"

Basic functions for writing and reading the registers
The functions described in the following are on the supplied disk under the directory **BC** in the file **IK410_0.C** and the associated header file **IK_0.H**.
The two functions **write_g26**1) and **read_g26**1) write and read the data registers. These are the two basic functions for working with the IK 410 V.

1) g26 is the HEIDENHAIN designation for the counter IC

Function for writing to the registers

The following function writes a value to the 16-bit register of a counter. This function is adapted to the AT-bus PCB used for these examples. You must write your own function for your hardware in order to access the registers of the IK 410 V.

Function:	**write_g26**
Parameters	
baseadr:	Basic address
address:	Address of the register of the counter
datum:	Value to be written to the address

```
void write_g26 (unsigned int baseadr,
               unsigned int address, unsigned int datum)
{
/* Write <datum> to the counter */
outpw (baseadr | address, datum);
}
```

Function for reading the registers

The following function reads a value from the 16-bit register of a counter. This function is adapted to the AT-bus PCB used for these examples. You must write your own function for your hardware in order to access the registers of the IK 410 V.

Function:	**read_g26**1)
Parameters	
baseadr:	Basic address
address:	Address of the register of the counter
	datum: Value from the data register indicated under "address" as a 16-bit variable

```
unsigned int read_g26 (unsigned int baseadr,
                    unsigned int address)
{
/* Read datum from the counter */
return(inpw(baseadr | address));
}
```

Simple functions for measured value latch via software

Function for storing a measured value
The following functions store the counter status of the desired axis in Data Register 0 (soft_l0) or Data Register 1 (soft_l1).

Function: **soft_l0**
Parameters
baseadr: Basic address

```
void soft_l0 (unsigned int baseadr)
{
write_g26 (baseadr, 0x0e, 0x01);
}


/*-------------------------------------------------------
                         soft_l1
 -------------------------------------------------------
 This function reads the measured value and stores
 it in data register 1.
 -----------------------------------------------------*/
void soft_l1 (unsigned int baseadr)
{
write_g26 (baseadr, 0x0e, 0x02);
}
```

Function for checking whether the measured value was stored

The following function checks whether a measured value was stored.

Function:	**latched**
Parameters	
baseadr:	Basic address
reg:	0 = Data Register 0, 1 = Data Register 1
Result:	false = no measured value was stored true = a measured value was stored

```
unsigned char latched (unsigned int baseadr,
                    unsigned char reg)
{
unsigned char result;
switch (reg)
       {
       case 0:
       result = (unsigned char)
             (read_g26 (baseadr, 14) & 0x01);
       break;
       case 1:
       result = (unsigned char)
             (read_g26 (baseadr, 14) & 0x02);
       break;
       }
return (result);
}
```

Function for repeatedly checking whether the measured value was stored

The following function continues to poll whether a measured value was stored until a measured value is stored.

Function:	**poll_latch**
Parameters	
baseadr:	Basic address
reg:	0 = Data Register 0, 1 = Data Register 1

```
void poll_latch (unsigned int baseadr,
              unsigned char reg)
{
switch (reg)
       {
       case 0:
       while (latched (baseadr, 0) == 0)
       ;
       break;

       case 1:
       while (latched (baseadr, 1) == 0)
       ;
       break;
       }
}
```

Function for reading a 32-bit measured value
The following function reads a 32-bit measured value from a counter.

Function:	**read_count_value32**
Parameters	
baseadr:	Basic address
reg:	0 = Data Register 0, 1 = Data Register 1
Result:	Integer variable of the **long** type

```
long read_count_value32 (unsigned int baseadr,
                     unsigned char reg)
{
union  mapper
        {
        long field0;
        unsigned int field1[2];
        }buffer;

switch (reg)
       {
       case 0:
       buffer.field1[0] = read_g26 (baseadr, 0);
       buffer.field1[1] = read_g26 (baseadr, 2);
       break;
       case 1:
       buffer.field1[0] = read_g26 (baseadr, 6);
       buffer.field1[1] = read_g26 (baseadr, 8);
       break;
       }
return (buffer.field0);
}
```

Function for reading a 48-bit measured value

The following function reads a 48-bit measured value from a counter.

Function:	**read_count_value48**
Parameters	
baseadr:	Basic address
reg:	0 = Data Register 0, 1 = Data Register 1
Result:	Floating point variable of the **double** type

```
double read_count_value48 (unsigned int baseadr,
                       unsigned char reg)
{
unsigned int field[3];
double count_value48;

switch (reg)
       {
       case 0:
       field[0] = read_g26 (baseadr, 0);
       field[1] = read_g26 (baseadr, 2);
       field[2] = read_g26 (baseadr, 4);
       break;
       case 1:
       field[0] = read_g26 (baseadr, 6);
       field[1] = read_g26 (baseadr, 8);
       field[2] = read_g26 (baseadr, 10);
       break;
       }

if (field[2] & 0x8000)
       count_value48 = (double)((field[0]-65535) +
       65536.0*(field[1]-65535)+
       4294967296.0*(field[2]-65535)-1);
       else
       count_value48 = (double)(field[0] +
       65536.0*field[1] +
       4294967296.0*field[2]);


return (count_value48);
}
```

Simple program for measured value latching via software

The following examples employ the previously defined functions (given and defined in the files IK410_0.H and IK410_0.C). The examples are included on the supplied diskette under the directory **BC** in the files **SAMPLE32.C** and **SAMPLE48.C.**
In these examples the counter status is displayed on the screen in millimeters. The IK 410 V can of course also display angular degrees. The conversion is shown by the following two formulas.

Converting the count into millimeters

$$\text{Value [mm]} = \text{count} \bullet \frac{\text{grating period [mm]}}{1024}$$

Example: grating period = 20 µm

$$\text{Value [mm]} = \text{count} \bullet \frac{0.020\ \text{[mm]}}{1024}$$

Converting the count into degrees

$$\text{Value } [\circ] = \frac{\text{count} \bullet 360\ [\circ]}{1024 \bullet \text{lines / rev.}}$$

Example: lines/rev. = 36 000

$$\text{Value } [\circ] = \frac{\text{count} \bullet 360\ [\circ]}{1024 \bullet 36\,000}$$

```
/*----------------------SAMPLE32.C----------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

 A simple program for the IK 410.
 Measured value with 32 bits.

 V 1.00
 October 1997

 Project files:        IK410_0.C, SAMPLE32.C
 Include files:        IK410_0.H
 ------------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include "ik410_0.h"

#define base_address 0x0340

int main()
{
double c_value_0;

cls;
   /* Initialise the board in interpolation mode */
write_g26 (base_address, 0x0c, 0x0001);
   /* Reset error bit, start counter */
write_g26 (base_address, 0x0e, 0x0048);
   /* Write to control register 2 */
write_g26 (base_address, 0x1c, 0x0008);
   /* Switch to encoder position signals */
write_g26 (base_address, 0x24, 0x0000);

   /*Cursor off*/
_setcursortype(_NOCURSOR);

while(!kbhit())
       {
          /* Software latch in register 0 */
       soft_l0 (base_address);
          /* Poll whether latched */
       poll_latch (base_address, 0);
          /* Read position value */
       c_value_0 = (double)read_count_value32
                             (base_address, 0);

       /* Display measured values */
       printf("\r\t%16.4f",c_value_0*0.02/1024);
       }

   /*Cursor on*/
_setcursortype (_NORMALCURSOR);

return (0);
}
```

```
/*----------------------SAMPLE48.C----------------------
 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

 A simple program for the IK 410.
 Measured value with 48 bits.

 V 1.00
 October 1997

 Project files:        IK410_0.C, SAMPLE48.C
 Include files:        IK410_0.H
 -------------------------------------------------------*/
#include <stdio.h>
#include <conio.h>
#include "ik410_0.h"

#define base_address 0x0340

int main()
{
double c_value_0;

cls;
   /* Initialise the board in interpolation mode */
write_g26 (base_address, 0x0c, 0x0041);
   /* Reset error bit, start counter */
write_g26 (base_address, 0x0e, 0x0048);
   /* Write to control register 2 */
write_g26 (base_address, 0x1c, 0x0008);
   /* Switch to encoder position signals */
write_g26 (base_address, 0x24, 0x0000);

   /* Cursor off */
_setcursortype (_NOCURSOR);

while(!kbhit())
       {
          /* Software latch in register 0 */
       soft_l0 (base_address);
          /* Poll whether latched */
       poll_latch (base_address, 0);
          /* Read Position value */
       c_value_0 = read_count_value48
                        (base_address, 0);


       /* Display measured values */
       printf("\r\t%16.4f",c_value_0*0.02/1024);
       }

   /* Cursor on */
_setcursortype (_NORMALCURSOR);

return (0);
}
```

Application examples with a RAM memory map in "BORLAND C++"

Examples with a RAM memory map in "BORLAND C++" can be found in the directory BCPP. The data structures and functions used are given and defined in the following files:

IK410.H:	In this header file you can set the addresses for up to 16 PCBs, each with two IK 410 Vs.
IK410_1.H:	Data structures for a RAM memory map for the IK 410 V registers and explanations of the functions in the files IK410_1.CPP, IIC.CPP and POTI_1.CPP
IK410_1.CPP:	Basic functions for the IK 410 V
IIC.CPP:	Functions for data transfer via the I²C-bus.
POTI_1.CPP:	Functions for setting the electronic potentiometer.

In the file IK410_1.H, a RAM memory map for the registers of the IK 410 V is set up with the help of data structures. The data for the map is written to the registers of the IK 410 V with the help of the functions **InitHandler** and **CommHandler**.

DISPLAY.CPP

The program DISPLAY.CPP displays the contents of all the IK 410 V registers.

Source code:	DISPLAY.CPP, IK410_1.CPP, IIC.CPP
Header files:	IK410.H, IK410_1.H

POTIS.CPP

The program POTIS.CPP shows how to set the electronic potentiometer of the IK 410 V via the I²C-bus using software.

Source code:	POTIS.CPP, IK410_1.CPP IIC.CPP, POTI_1.CPP
Header files:	IK410.H, IK410_1.H

Application examples with a RAM memory map in "TURBO PASCAL"

Examples with a RAM memory map in "TURBO PASCAL" can be found in the directory **TP**. The data structures and functions used are given and defined in the following files:

IK410_0.TPU	Basic functions
IK410_1.TPU	Data structures and functions for a RAM memory map of the IK 410 V registers.
IK410_2.TPU	Functions for ADJUST.PAS, POTIS.PAS and SCOPE.PAS
IIC.TPU	Functions for data transfer via I^2C-bus
SCOPE_0.TPU	Functions for SCOPE.PAS
POTI_0.TPU	Functions for POTIS.PAS
ADJ_0.TPU	Functions for ADJUST.PAS

These files are integrated as **units** into further application programs with the help of the **uses** command. The files can be found on the supplied disk and have the file name extension *.PAS. The *.PAS files must be compiled into *.TPU **units**.

In the file IK410_1.TPU, a RAM memory map for the registers of the IK 410 V is set up with the help of data structures. The data for the map is written to the addresses of the IK 410 V with the help of the functions **init_handler** and **comm_handler**.
The programs require a BORLAND graphics interface (BOLAND Graphics Interface = *.BGI). Included among the IK 410 V accessories is the graphics interface EGAVGA.BGI. This interface must be stored in the same directory as the example programs.

SAMPLE32.PAS

This is a simple application program for showing the measured value latch via software.

Source code:	SAMPLE32.PAS	
Units:	IK410_0.TPU	Basic functions

SCOPE.PAS

This program displays the sinusoidal encoder signals either as in amplitude-time diagram or in XY representation. The potentiometers can be set with key commands that are explained on the screen.

Source code:	SCOPE.PAS	
Units:	IK410_0.TPU	Basic functions
	IK410_1.TPU	Functions for a RAM memory map
	IK410_2.TPU	Functions for ADJUST.PAS, POTIS.PAS and SCOPE.PAS
	IIC.TPU	Functions for data transfer via I^2C-bus
	SCOPE_0.TPU	Functions for SCOPE.PAS
	CNT_0.TPU	Window functions
	LOGO.TPU	Initial image after program start

POTIS.PAS

This program shows the settings of the electronic potentiometers for phase position and amplitude as well as the values of the offset registers. The potentiometers can be set with key commands that are explained on the screen.

Source code:	POTIS.PAS	
Units:	IK410_0.TPU	Basic functions
	IK410_1.TPU	Functions for a RAM memory map
	IK410_2.TPU	Functions for ADJUST.PAS
	IIC.TPU	Functions for data transfer via I^2C-bus
	POTI_0.TPU	Functions for POTIS.PAS
	CNT_0.TPU	Window functions
	LOGO.TPU	Initial image after program start

ADJUST.PAS

ADJUST.PAS automatically adjusts axis 1 (selection:1) or axis 2 (selection:2) for phase position (selection:p), amplitude (selection:a) and offset (selection:o) of the sinusoidal encoder signals. The compensation values are calculated by slow movement of the encoder. After 30 signal periods, a tone signals that a compensation value is ready and can be stored by pressing the S key.

Source code:	ADJUST.PAS	
Units:	IK410_0.TPU	Basic functions
	IK410_1.TPU	Functions for a RAM memory map
	IK410_2.TPU	Functions for ADJUST.PAS, POTIS.PAS and SCOPE.PAS
	ADJ_0.TPU	Functions for ADJUST.PAS
	CNT_0.TPU	Window functions
	LOGO.TPU	Initial image after program start

IK410.PAS

This is an example of a position display program in "TURBO PASCAL".

Source code:	IK410.PAS	
Units:	IK410_0.TPU	Basic functions
	IK410_1.TPU	Functions for a RAM memory map
	IK410_2.TPU	Functions for ADJUST.PAS, POTIS.PAS and SCOPE.PAS
	CNT_0.TPU	Window functions
	CNT_1.TPU	Position display
	CNT_2.TPU	Counter program
	SCOPE_0.TPU	Functions for SCOPE.PAS
	ADJ_0.TPU	Functions for ADJUST.PAS
	LOGO.TPU	Initial screen after program start
Include	IK410.WIN	Window definitions
files:	IK410.CNT	Initializing file
Help texts:	IK410.HLP	File with help texts

Specifications

Mechanical Data	
Dimensions	100 mm x 65 mm
Operating temp.	0°C to 55°C
Storage temp.	–30°C to 70°C

Electrical Data

Encoder input
X3: 16-pin AMP connector
– Input for incremental linear or angular encoders with sinusoidal voltage signals (1 V_{PP})
– Additional input for the commutation track of a motor encoder (one sine/cosine per revolution)

Max. input frequency: 350 kHz at $f_{CLK\ (min)}$ 3 MHz

Cable length: Max. 60 m (supply voltage 5.0 V)
Cable lengths to 150 m are possible if it can be guaranteed that the encoder will be supplied by 5 V from an external power source. In this case the input frequency is reduced to max. 250 kHz.

Interface to subsequent electronics
X1 and X2: 26-pin AMP connector
16-bit microcomputer interface corresponding to a static RAM

Control bus:	–RD , –WR, –CS, –Res, Clk (max. 20 MHz)
Data bus:	D0 to D15
Address bus:	A0 to A5
Power supply:	Gnd and +5 V / ±15 V (±5%) approx. 20 mA (without encoder)
Latch inputs:	–L0, –L1
Latch synchronization:	–SYNC0, –SYNC1 (for simultaneous latching via several cards)
Interrupt output:	–INT

Signal interpolation	1024-fold (10 bits)

Adjustment of encoder signals
Offset is adjusted with registers in the counters
Phase and amplitude are adjusted with electronic potentiometers

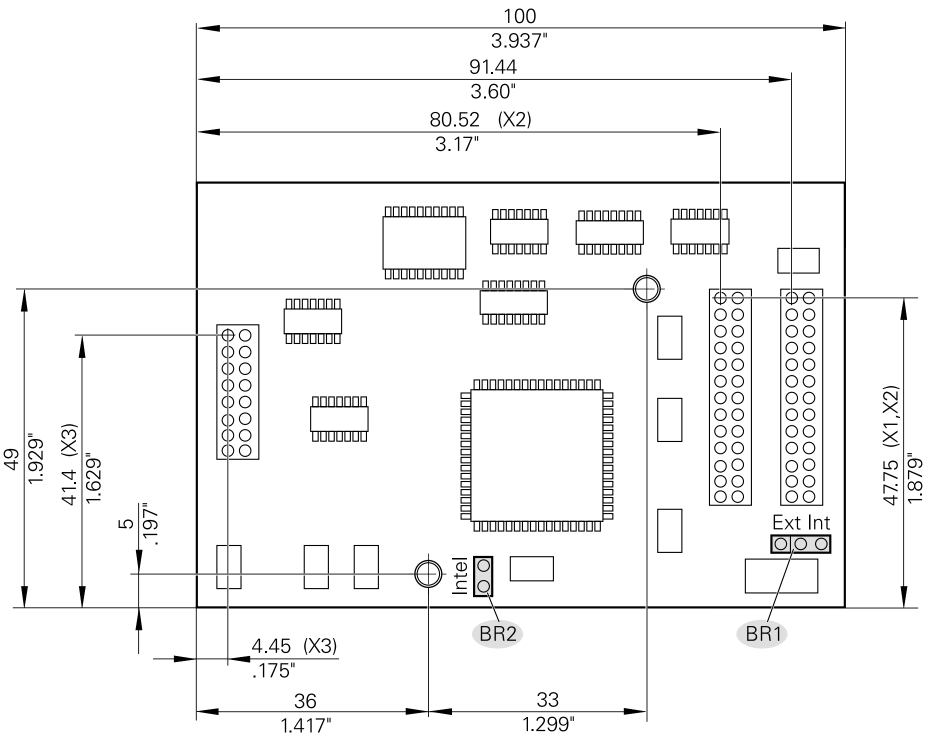
Counter functions	Two modes:
Period counter mode:	32-bit counter value
Interpolation mode:	32-bit counter value with 10-bit interpolation value
Data register:	48 bits of which only 42 are used for the measured value

Measured value latch
The latching procedure can be triggered by asynchronous latching signals (–L0 or –L1, edge-sensitive), by a software command or by traversing the reference mark. The time between latch request and actual latching procedure is 200 ns*) with a time uncertainty of 100 ns*).
After the measured value is stored (max. 25 µs), a status bit is set and/or an interrupt generated.

Data format	MOTOROLA or INTEL format

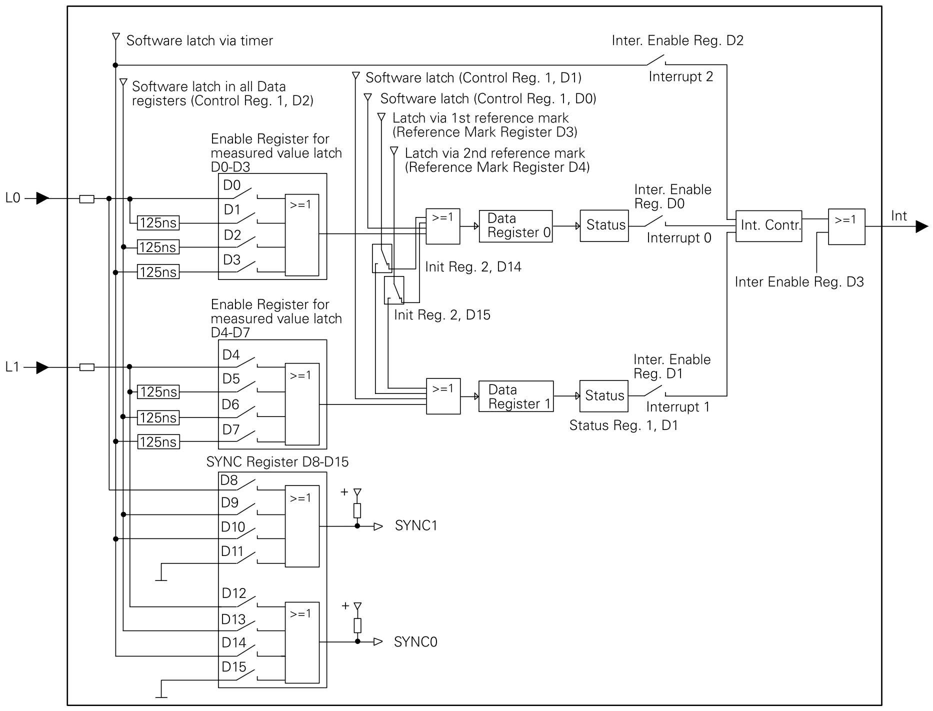
*) These times are valid for the maximum clock frequency of 20 MHz.

Dimensions



BR1: External or internal clock
BR2: INTEL data format: jumper connected
MOTOROLA data format: jumper open

Basic Circuit Diagram of the Latch Paths in the Counter ICs


Software latch via timer
Inter. Enable Reg. D2
Interrupt 2
Software latch in all Data registers (Control Reg. 1, D2)
Software latch (Control Reg. 1, D1)
Software latch (Control Reg. 1, D0)
Latch via 1st reference mark (Reference Mark Register D3)
Latch via 2nd reference mark (Reference Mark Register D4)
Enable Register for measured value latch D0-D3
D0
D1
D2
D3
>=1
125ns
L0
Data Register 0
Status
Inter. Enable Reg. D0
Interrupt 0
Int. Contr.
Int
Inter Enable Reg. D3
Init Reg. 2, D14
Init Reg. 2, D15
Enable Register for measured value latch D4-D7
D4
D5
D6
D7
L1
Data Register 1
Inter. Enable Reg. D1
Interrupt 1
Status Reg. 1, D1
SYNC Register D8-D15
D8
D9
D10
D11
D12
D13
D14
D15
SYNC1
SYNC0

HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH
Dr.-Johannes-Heidenhain-Straße 5
83301 Traunreut, Germany
☏ +49/8669/31-0
FAX +49/8669/5061
e-mail: info@heidenhain.de

Technical support	FAX	+49/8669/31-1000
Measuring systems	☏	+49/8669/31-3104
e-mail: service.ms-support@heidenhain.de		
TNC support	☏	+49/8669/31-3101
e-mail: service.nc-support@heidenhain.de		
NC programming	☏	+49/8669/31-3103
e-mail: service.nc-pgm@heidenhain.de		
PLC programming	☏	+49/8669/31-3102
e-mail: service.plc@heidenhain.de		
Lathe controls	☏	+49/711952803-0
e-mail: service.hsf@heidenhain.de		

www.heidenhain.de

326 959-21 · 1.5 · 4/2001 · S · Printed in Germany · Subject to change without notice