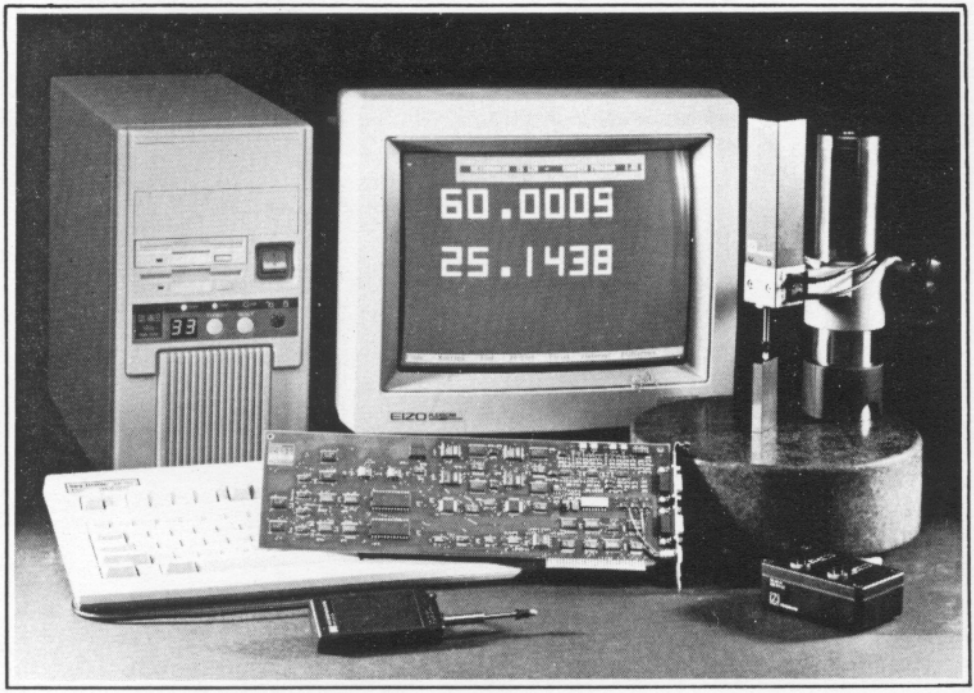
HEIDENHAIN

User Manual

IK 120

PC Counter Board



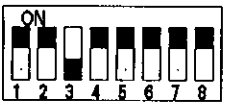
7/92

Contents

Quick start

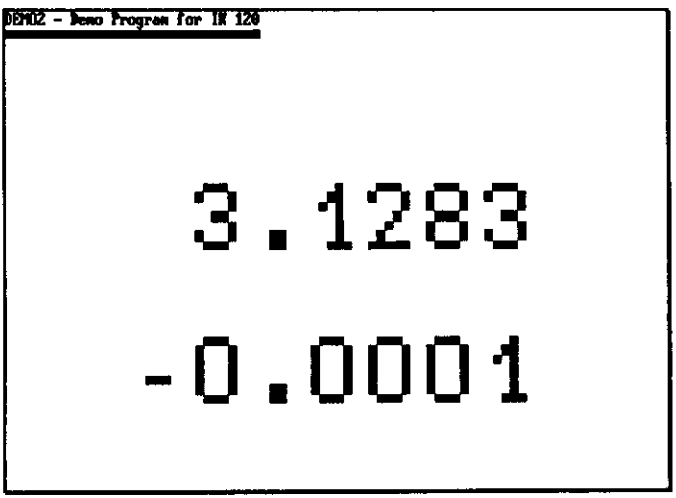
Carry out the following steps to get the IK 120 running for the first time:

Set the DIP switches as shown.



- **Remove the interrupt jumper so that no interrupt is assigned to the IK 120.**
- **Disconnect the power supply from your PC.**
- **Install the IK 120 into your PC.**
- **Connect a measuring system with sine wave output signals to the input marked "X1" on the IK 120. If necessary use the HEDENHAIN adapter cable (Id.No. 267 269).**
- **Insert the driver software disk supplied with the IK 120.**
- **Type the following DOS commands to display the screen shown below:**

```
> A:
> CD \TP
> DEMO2
```



If movement of the encoder is not registered on the screen refer to the section "What to do if it doesn't work".

General Topics

How to use this manual

This operating manual has the following sections:

Quick start
Instructions to get the IK 120 up and running on your PC as quick as possible.

General topics
A general description of the functions of the IK 120 so that you can assess if the IK 120 is suitable for a particular application.

Hardware
This section contains a description of the hardware of the IK 120, specification of the measuring system inputs, external functions, interpolation, jumper and DIP switch settings.

Register description
The IK 120 is programmed by writing and reading data to and from the board in the same way as with normal PC RAM memory. This section describes exactly what data can be read from the board and how to program the board.

Programming the IK 120
Description of the programming steps required to program the various functions required in typical applications such as the interval counter, using interrupts or using reference marks. How to program the IK 120 to latch measuring values from more than one counter simultaneously.

Software description
How to use the driver software supplied with the IK 120. Description of the functions included in the sample programs supplied with the IK 120.

What to do if it doesn't work
Hints to help you to get the IK 120 to work with your PC configuration.

Technical specification
Technical specifications for the IK 120.

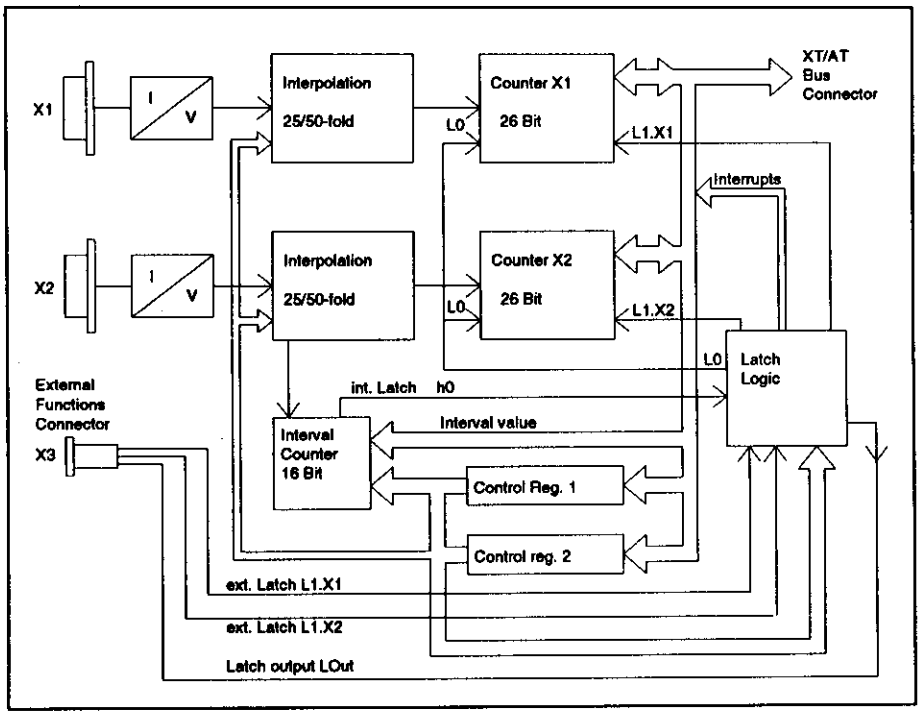
Description of the various types of text used in this manual:

Text shown like this is a WARNING!

Text shown like this contains INSTRUCTIONS.

Description of the IK 120

The PC counter board IK 120 connects two HEIDENHAIN measuring systems with 11 μA_{pp} sine wave output signals directly to an XT, AT compatible personal computer (PC). Position values from the two measuring systems can be displayed using the demonstration programs supplied or processed using custom written software.



The interpolation electronics included on the IK 120 generates up to 200 counts per input signal period. This results in a finest measuring step of 1/200th part of the signal period.

As well as the interpolation and counting electronics, the IK 120 also has an interval counter which allows simultaneous storage of both counter values at predetermined distances.

Storage of both counter values can also be initiated independently for each axis by an external signal.

A demonstration program is supplied to show some of the features of the IK 120. Programming examples and driver software, written in Turbo Pascal and C, are supplied to simplify the programming of the IK 120.

The IK 120 is inserted directly into a full length expansion slot of an XT or AT compatible personal computer. It is ideal for applications requiring high-speed data acquisition or further processing of measuring data.

Order numbers

Id.No.	Unit
271 208 ..	IK 120 PC counter board including driver software.
267269 ..	Adapter cable for HEIDENHAIN standard incremental measuring systems. Please quote required length to nearest meter when ordering.
274 546 ..	Adapter cable for HEIDENHAIN measuring systems with APE unit (e.g. VM 101)
257 818 01	PCB adapter to output the input signal (X2) to another counter or control.
274 542 01	Adapter cable to feed input signal (X2) from PCB adapter to another counter or control.
265 775 02	Connector for the external functions connector (4 pin Lemosa connector).

Hardware

Input signal specification

Signal amplitude I_1, I_2 (0°, 90°) I_0 (Reference mark)	 7 to 16 μA_{pp} 3.5 to 8 μA_{pp}
Signal symmetry 0° and 90° signals	$s = \frac{\text{DC component}}{\text{AC component pp}}$ ≤ 6.5 %
Phase angle	90° ± 10°
Amplitude ratio	maximum 1: 0.8
Maximum input frequency	25-fold, 50 kHz ** 50-fold, 30 kHz
Reference mark signal	$\frac{\text{Current value at 360°}*}{\text{Usable component}} \leq 1.7$ Phase position = 135° ± 15°
Signal error level	≤ 2.5 µA

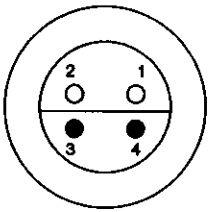
* 360° is a complete period of the input signal.

** for measuring systems with 3 dB limit of at least 100 kHz.

Connector pin assignments

9 pin sub-D input (female) for sine wave signals (X1, X2)

Pin number	Signal
1	I_1 -
2	Un 0 V
3	I_2 -
4	Inner ground
5	I_0 -
6	I_1 +
7	Up 5 V
8	I_2 +
9	I_0 +
connector housing	Outer ground and housing



4 pin external functions connector (X3)

Pin number	Signal
1	Latch input X1, L1.X1
2	Latch input X2, L1.X2
3	Latch output, LOut
4	0 V

9 pin sub-D output (male) for sine wave signals

Pin number	Signal
1	I_1 -
2	Un O V
3	I_2 -
4	not connected
5	I_0 -
6	I_1 +
7	not connected
8	I_2 +
9	I_0 +
connector housing	Outer ground and housing

10 pin AMP PCB connector for sine wave output signals

Pin number	Signal
1a	not connected
1b*	not connected
2a	not connected
2b	Un 0 V
3a	I_0 -
3b	I_0 +
4a	I_2 -
4b	I_2 +
5a	I_1 -
5b	I_1 +

b is the side of the connector with locking mechanism,
1 starts at the side of the connector with the indentation.

PC bus specification

The IK 120 can be used in 100% compatible XT, AT compatible PCs. HEIDENHAIN cannot guarantee that the IK 120 will function PC's that are not 100% XT or AT compatible. The IK 120 conforms to the IEEE P996 international standard (with zero wait states), defining the XT-, AT- and ISA-bus (industry standard architecture).

Bus width	8 bit
Supply voltages	+5 V ±5% +12 V ±5% - 12 V ±5%
Power requirement	typical 1 Watt maximum 1.5 Watt (without measuring system)
Slot type	Full length XT compatible, 8 bit slot
Required address space	1024 Bytes
Address range	C0000h - FFFFFh

Timing

Tests have shown that with an optimised assembler routine it is possible to read two count values into RAM in under **20 µs**. These tests were carried out on a PC with 80286 processor.

Signal subdivision

The input signals to the IK 120 are two incremental sine wave signals shifted by 90° to each other, the period of which corresponds to a particular linear or angular movement. The interpolation electronics used in the IK 120 (similar to the EXE 650B) converts one period of the input signals into 25 or 50 TTL signal periods.
These TTL signals are then fed to a counter IC which can be programmed to count either one, two, or four edges per TTL signal period.
This means that it is possible to generate up to 200 counts for each period of the input signals.

Interpolation	Evaluation	Subdivision / Number of counts
25-fold	1	25
	2	50
	4	100
50-fold	1	50
	2	100
	4	200

When a subdivision of 25, 50 or 100 is required use 25-fold interpolation with a maximum input frequency of 60 kHz.

Sine wave signal output from the IK 120

The signal on the input X2 is available at a 10 pin AMP connector on the IK 120. A cable assembly which connects the 10 pin AMP connector to a 9 pin sub-D connector (male) on a PC backstrip is available to feed this signal out of the PC housing. A special adapter cable is available to connect this output to a standard HEIDENHAIN counter or interpolation unit (EXE) (see "Order numbers").

External functions

The external functions connector (X3) is a 4 pin "LEMOSA" connector. The plug required for this connector is available from HEIDENHAIN (see "Ordering information").

Latch inputs (L1.X1, L1.X2)

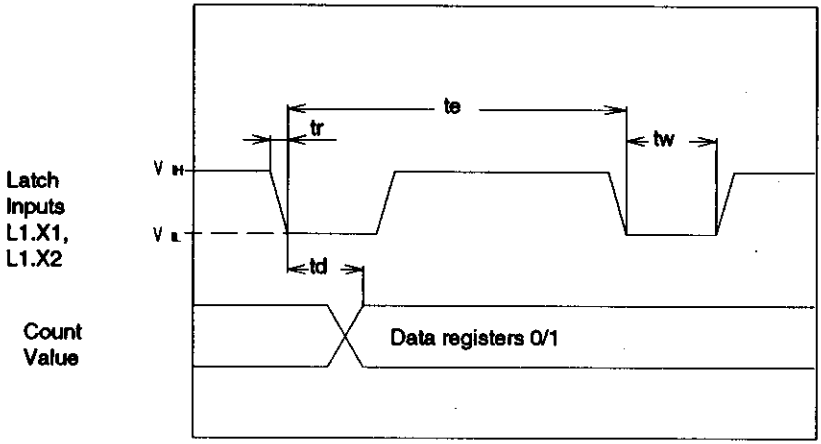
The IK 120 has an external function connector (X3) to allow the value of each counter to be stored in the data register 1 of each counter using an external signal.

An interrupt can also be generated with an external latch signal (L1.X1, L1.X2).

The input signals L1.X1 and L1.X2 are active low and are held high with a pull-up resistor on the IK 120. The input can be switched with standard TTL, LS or CMOS components.

The easiest way to activate the signal is to connect it to the 0 Volt (pin 4) with a switch. The switch must be debounced using hardware or software.

Latch input timing and levels



Signal / Time	Minimum	Maximum
Vih (V)	3.15	30
Vil (V)	-3.0	0.9
tr (ns)		500
td (ns)		570
te (µs)	40	
tw (µs)	20	

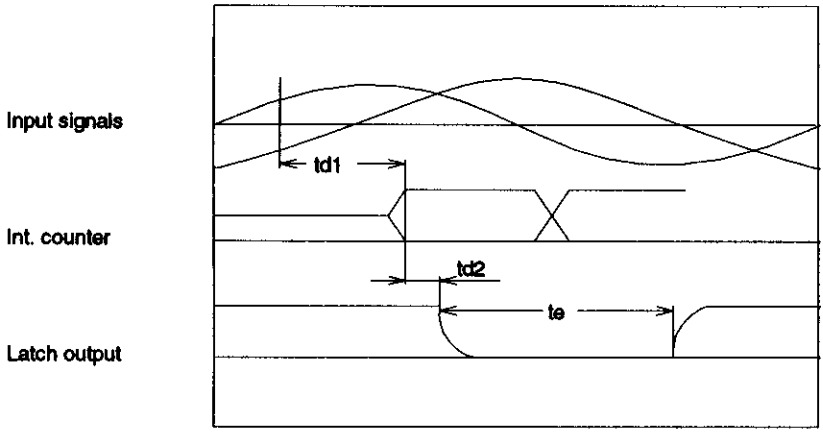
* with Vih = 5 V and Vil = 0V.

Latch output (LOut)

The latch signal LOut generated by the interval counter can be output over the external functions connector (X3). This is an open-collector output and if this output is held active the internal latch L0 is disabled.

This signal can be connected to the latch inputs L1.X1, L1.X2 of other IK 120s to store the values from counters on different IK 120s simultaneously.

Latch output timing and levels



Signal / Timing	Minimum	Maximum
V_{oh} (V)	4.0	32
V_{ol} (V)	0	0.8 with V_{oh}<12 1.0 with V_{oh}>12
I_{ol} (mA)		40
td1 (µs)	3.3	12 *
td2 (ns)		200
te (µs)	40	

* with V_{ih} = 5 V and V_{il} = 0V.

Interval counter

The IK 120 has a third counter which can be used to generate a latch signal at constant intervals e.g. every 1000 counts or every 0.1 mm. This signal can either be programmed to be gated to the latch1 input of each of the two counter ICs or to the LOut output on the external functions connector (X3).

Counter width	16 bit
Interval value	16 bit, from 2 to 65535
Reference mark functions	start, stop, reset, load, latch
Signal evaluation	one-fold, two-fold, four-fold
Input signal	only from input X2

Interrupts

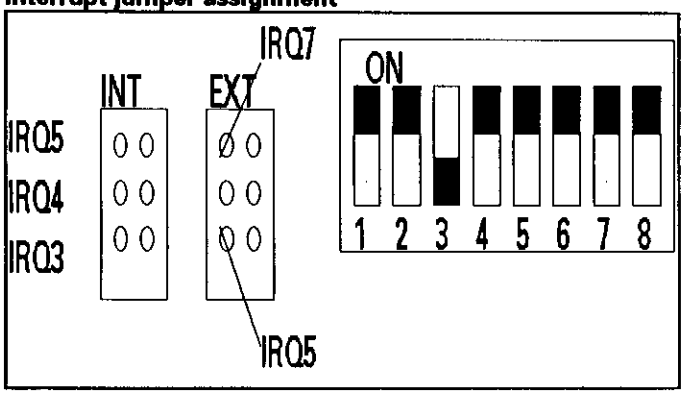
The IK 120 can generate one of the following PC interrupt signals: IRQ3, IRQ4, IRQ5, IRQ7.
The **internal** (INT) latch output from the interval counter LOut can be programmed to generate an interrupt signal on **IRQ3, IRQ4, or IRQ5.** The **Int0** flip-flop is set and should be reset by the PC at the end of the interrupt service routine.

The **external** (EXT) latch signal input L1.X1 or L1.X2 can be programmed to generate an interrupt signal on **IRQ5 or IRQ7.** The **Int1** flip-flop is set and should be reset by the PC at the end of the interrupt service routine.

PC Interrupt assignment

Interrupt signal	Interrupt number	Interrupt address	Normally assigned to
IRQ3	0Bh	002Ch	COM2
IRQ4	0Ch	0030h	COM1
IRQ5	0Dh	0034h	LPT2
IRQ7	0Fh	003Ch	LPT1

Interrupt jumper assignment



Only one interrupt signal, either internal or external, can be assigned on the board.
This interrupt signal can not used by any other cards in your PC.

DIP Switch address setting

Communication with the IK 120 is carried out through 8 bit RAM access in the upper memory range in an area of 1024 bytes, positioned between 768 kbyte and 1024 kbyte (C0000h to FFFFFh).

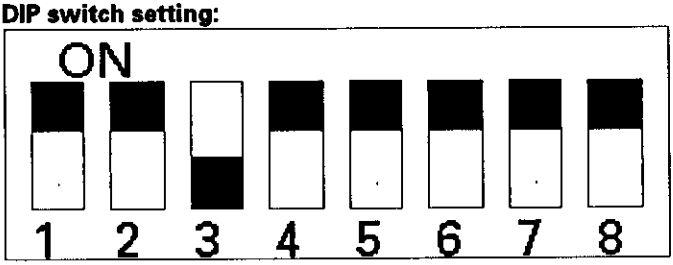
The IK 120 is delivered with a the base address setting C8000h, which means that the address range between C8000h and C8400h is occupied.

The base address can be adjusted with the DIP switches on the IK 120. The first megabyte of PC address space is addressed with 20 bits A0 to A19.

Address bits A19 and A18 are fixed at high (1). Address bits A17 to A10 can be adjusted with switches S1 to S8 **(0=ON, 1=OFF)**.
The table below shows how to adjust the DIP switches.

Address bit	A19 A18	A17 A16	A15 A14 A13 A12	A11 A10	A99 A8
Hexadecimal value	C		8	0	
Binary value	1 1	0 0	1 0 0 0	0 0	0 0
Switch	fixed	S1 S2	S3 S4 S5 S6	S7 S8	
Switch position		ON ON	**OFF** ON ON ON	ON ON	

DIP switch setting:



Sample base address settings:

Base address (Hex)	DIP Switches		
	S1 S2	S3 S4 S5 S6	S7 S8
C8000	ON ON	OFF ON ON ON	ON ON
C8400	ON ON	OFF ON ON ON	ON OFF
C8800	ON ON	OFF ON ON ON	OFF ON
CF000	ON ON	OFF OFF OFF OFF	ON ON
D0000	ON OFF	ON ON ON ON	ON ON
E0000	OFF ON	ON ON ON ON	ON ON

Check that the base address is not used by any other boards in your PC.

Two boards are not allowed to occupy the same address range.

C0000h - C6000h is usually occupied by the EGA or VGA graphics adapter.

Adapter boards (SCSI) for extra external storage units such as extra hard disks or optical disks ("GIGAFILE") often use the same address space as the IK 120, C8000h.

Do not use the address range F0000h to FFFFFh for the IK 120. This area is normally used by the BIOS ROM.

Address settings when using several IK 120s

When using more than one IK 120 in a PC the base address of each board should be adjusted so that each board occupies an adjacent 1024 bytes of memory space:

Interface board	Base address	Address space	S1 to S8
1st IK 120	C8000h	C8000h - C83FFh	0 0 1 0 0 0 0 0
2nd IK 120	C8400h	C8400h - C87FFh	0 0 1 0 0 0 0 1
3rd IK 120	C8800h	C8800h - C8BFFh	0 0 1 0 0 0 1 0
.....			

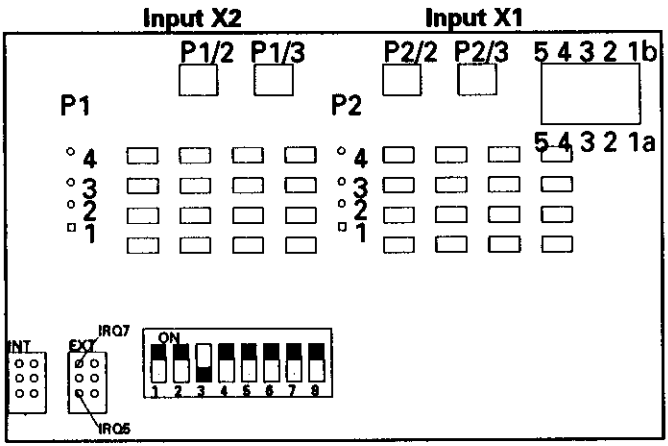
Signal adjustment

The symmetry of the input signals I_1 and I_2 (0° and 90°) can adjusted using potentiometers on the IK 120. I_1 can be measured across the measuring points
1 and 2, I_2 across 1 and 3 and should be adjusted to 2.5 V using potentiometers P2/3 and P2/2 respectively
Voltage on pin 1 (U_0) is not identical to the 0V of the PC and hence measurements should be made with a potential free measuring instrument e.g. battery powered voltmeter.

Potentiometers and measuring points P1 are for input X2

Potentiometers and measuring points P2 are for input X1

Measuring point	Signal	Values
1	U_0	0V- U_0 2.5 V ±1.5%
2	I_2 (Pot. P?/2)	pin 1/2 2.5 V ±1.5%
3	I_1 (Pot. P?/3)	pin 1/3 2.5 V ±1.5%
4	I_0 or REF mark	



Register Description

Address conventions for PCs

The address for a particular register on the IK 120 can be calculated as follows:

Address = Base address + address offset

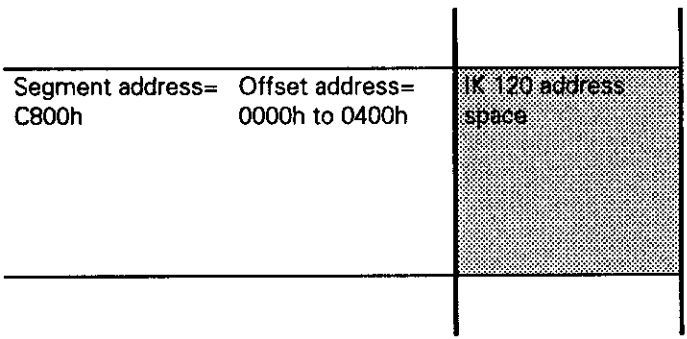
Where necessary for clarity this description will switch between using the base address and offset, segment address and only offset address.

Segment address

The 20 bit (5 hexadecimal characters) address range of a PC is split so that the most significant 16 bits **A19 to A4** are defined as the segment address. The segment address is 16 bits and defines an area of memory 64 kbytes large that can be addressed with an offset address.

Offset address

This is the lower 16 bits of the address **A15 to A0**.



Register overview

The IK 120 has the following registers:

Address (Hex)	Register
00 to 0F	Counter IC, Input X1
00	Data register 0, LSB
01	Data register 0
02	Data register 0
03	Data register 0, MSB
04	Data register 1, LSB
05	Data register 1
06	Data register 1
07	Data register 1, MSB
08	Software latch 0
09	Software latch 1
0A	Reference mark register
0B	Counter command register
0C	Operation mode register
0D	Reset status register
0E	Status register
0F	Initialising register (Write mode)
0F	Scan register (Read mode)
10 to 1F	Counter IC, Input X2 As above from 10 to 1F
20	Control register 1
30	Control register 2
40	Interval value LSB
50	Interval value MSB
60	Strobe register for interval counter
70	Clear interrupt

Address range 00 to 0F is the address space occupied by the counter IC for input X1, 10 to 1F is the address space occupied by the counter IC for input X2.
Both ICs are identical.

20h: Control register 1

The individual bits of control register 1 have the following meaning:

Bit	Function	Value
D0 D1	Interpolation, input X1 Interpolation, input X2	0 = 25-fold, 1 = 50-fold
D2 D3	Reset signal error, X1* Reset signal error, X2*	Bits D2 or D3 must be programmed from 1 to 0 to 1 to reset or 1 for no effect.
D4 D5	Signal evaluation of the interval counter	00(D4D5) = four fold 10 = two fold 01 or 11 = one fold
D6 D7	Interrupt Int0 Interrupt Int1	0 = not active 1= active

* When the interpolation electronics detects an error in the input signals a bit is set which can be read in the status register of each counter IC. This bit error bit will remain set until it has been reset by programming the sequence 1 - 0 - 1. in bits D2 or D3 of control register 1.

30h: Control register 2

Function	Hex	Bits	Meaning
Latch functions		**D3D2D1D0**	*
	00	0 0 0 0	Latch signals not activated
	01	0 0 0 1	L0
	02	0 0 1 0	L1.X1
	03	0 0 1 1	L0, L1.X1
	04	0 1 0 0	L1.X2
	05	0 1 0 1	L0, L1.X2
	06	0 1 1 0	L1.X1, L1.X2
	07	0 1 1 1	L0, L1.X1, L1.X2
	09	1 0 0 1	L0, LOut
	0B	1 0 1 1	L0, L1.X1, LOut
	0D	1 1 0 1	L0, L1.X2, LOut
	0F	1 1 1 1	L0, L1.X1, L1.X2, LOut
	08	1 0 0 0	A "low" signal on either of the latch inputs X3/1 or X3/2 causes both axes to be latched simultaneously.
Interval counter command		**D7D6D5D4**	**
	00	0 0 0 0	Interval counter has no function
	10	0 0 0 1	Start interval counter
	20	0 0 1 0	Stop
	30	0 0 1 1	Start when strobed
	40	0 1 0 0	Stop when strobed
	50	0 1 0 1	Start when reference mark is detected
	60	0 1 1 0	Stop when reference mark is detected
	70	0 1 1 1	Generate latch signal
	80	1 0 0 0	Zero interval counter
	90	1 0 0 1	Zero when strobe signal is detected
	A0	1 0 1 0	Zero when reference mark is detected
	B0	1 0 1 1	Load interval value
	C0	1 1 0 0	Load when strobed
	D0	1 1 0 1	Load when reference mark is detected
	E0	1 1 1 0	Generate latch signal when strobed
	F0	1 1 1 1	Generate latch when ref. mark detected

*L0 = Activate simultaneous latching of both counters from the interval counter.

L1.X1 = Activate latch input from pin 1 of external functions connector X3 to latch input X1 in data register 1.

L1.X2 = Activate latch input from pin 2 of external functions connector X3 to latch input X2 in data register 1.

LOut = Activate output of the latch signal to the external functions connector X3 pin 3.

** The interval counter commands are only valid for the interval counter and not the counters for inputs X1 and X2.

40h-50h: Interval value

This is the number of counts that the interval counter is programmed to count before giving out a latch signal L0. The value is a 16 bit positive integer value which is written to:

- Address 40h, Interval value LSB
- Address 50h, Interval value MSB

e.g. Calculate interval value to generate a signal every millimeter with 10 µm signal period and four fold evaluation.

$$\frac{1\ mm}{0.010} * 200\ counts = 20000\ counts\ per\ mm$$

20000 in hexadecimal notation is 4E20h
Lower byte (LSB) interval value = 20h
Higher byte (MSB) interval value = 4Eh

60h: Strobe register for interval counter

A write access to this register generates a stobe signal for the interval counter. The interval counter reacts on the strobe signal depending on how it has been programmed (see control register 2).
The interval counter can be programmed to "latch when strobed" so that a write access to the strobe register latches both counters simultaneously.

70h: Clear interrupt register

This register must be written to at the end of the interrupt service routine to reset the interrupt flip flop.

Single axis counter registers

The single axis counter counts 90° phase shifted incremental signals in a 26 bit counter that counts either from either -2^{25} to $+2^{25}-1$ or -180 000 to +179 999 and presents the count value as a 32 bit twos complement number.

00h-03h: Data register 0
When the count value is latched with a **software latch0 or from the interval counter** it is stored in the four bytes of data register 0.

04h-07h: Data register 1
When the count value is latched with a **software latch1 or from an external latch signal** it is stored in the four bytes of data register 1.

Once a count value has been stored in a data register the latch signal is no longer active until the MSB of the count value has been read, address 03h or 07h, bit D0 or D1 in status register 0Eh is also reset.

08h: Software latch0
A write access to this register stores the count value in data register 0.

09h: Software latch1
A write access to this register stores the count value in data register 1.

0Ah: Ref. mark command register

This register allows you to program the counter to carry out a particular action, start, stop, zero, when the reference mark is detected.

Value (hex)	Action to be carried out when reference mark is detected
00	Reference mark has no function
01	Start counter
02	Stop counter
03	Reference mark has no function
04	Zero counter
05	Zero and start counter
06	Zero and stop counter
07	Zero counter
08	Latch count value to data register
09	Latch and start counter
0A	Latch and stop counter
0B	Latch count value to data register
0C	Latch and zero counter
0D	Latch, zero and start counter
0E	Latch, zero and stop counter
0F	Latch and zero counter

0Bh: Command register

This register allows you to start, stop or zero the counter. Actions are carried out immediately, independent from the reference mark.

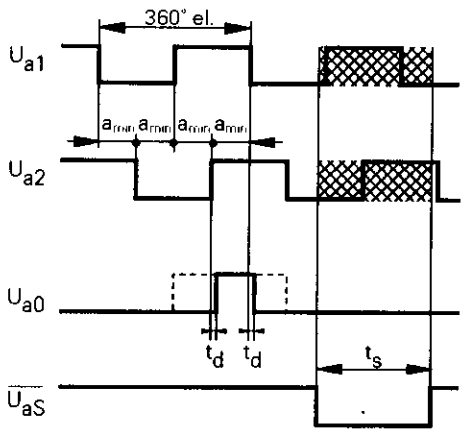
Value (hex)	Action
01	Start counter
02	Stop counter
04	Zero counter
05	Zero and start counter
06	Zero and stop counter
07	Zero counter

0Ch: Operating mode register

This register allows you to change the following three counter functions:

Edge Evaluation

The two incremental signals 0° and 90° have a total of four edges per cycle. The counter can be programmed to count one, two or four edges per cycle.



Count direction
The count direction of the counter can be inverted.

Counter mode
There are two counter modes:

Linear:	-2^{25} to $+2^{25}-1$
Angle:	-180 000 to 179 999

Value (Hex)	Evaluation	Count direction	Count mode
00	one fold	normal	linear
02	two fold	▪	▪
03	four fold	▪	▪
04	one fold	▪	angle
05	two fold	▪	▪
07	four fold	▪	▪
08	one fold	inverse	linear
09	two fold	▪	▪
0B	four fold	▪	▪
0C	one fold	▪	angle
0D	two fold	▪	▪
0F	four fold	▪	▪

0Dh: Reset status register
Write the value FFh to this register to reset the status register.

0Eh: Status register

The status register is used to store and report various occurrences in the counter IC.

Bit	Meaning
D0	Set to 1 when count value is stored in data register 0. Set to 0 when the MSB of the count has been read
D1	Set to 1 when count value is stored in data register 1. Set to 0 when the MSB of the count has been read
D2	0 = counter stopped 1 = counter started
D3	Signal error bit. Set to 1 when an amplitude or frequency error is detected on the input signal Set to 0 by writing to delete status bits register 0Dh
D4	0
D5	Set to 1 when a command is written to the reference mark command register 0Ah. Set to 0 when a reference mark is detected.
D6	Level of the 0° signal at the time of the last internal latch.
D7	Level of the 90° signal at the time of the last external latch.

0Fh: Initialising register (Write only)

Bit	Meaning
D0	0
D1	0
D2	0 = Latch0 pin active (interval counter) 1 = Latch0 pin inactive
D3	0 = Latch1 pin active (external latching) 1 = Latch1 pin inactive
D4	0 = Reference mark latches to data register 0. 1 = Reference mark latches to data resister 1.
D5	0 = Reference mark triggers on signal edge. 1 = Reference mark triggers on signal level.
D6	0 = Intel data format
D7	0 or 1

0Fh: Scan register (Read only)

This register contains the signal level of various pins on the counter IC.

Bit	Meaning
D0	Level of signal on latch0 pin
D1	Level of signal on latch1 pin
D2	0 = Latch0 active (interval counter) 1 = Latch0 inactive
D3	0 = Latch1 active (external latching) 1 = Latch1 inactive
D4	0 = Reference mark latches to data register 0 1 = Reference mark latches to data resister 1
D5	Level of signal on the reference mark pin
D6	Level of signal on the 0° pin
D7	Level of signal on the 90° pin

Programming the IK 120

The IK 120 is memory addressed and can be programmed with any programming language that allows you to read and write data to and from the upper RAM memory area between addresses C0000h and FFFFFh (768k and 1024k) of a PC.

There are two identical counter ICs on the IK 1120 with following offset address:

- Address 00h to 0Fh, Counter X1
- Address 10h to 1Fh, Counter X2

In the following sample programs all addresses are offsets from the base address, in hexadecimal notation and preceded with the "@" symbol.

Complete address = base address + offset address.

Initialising the control registers

Below is a list of the instructions necessary to initialise the IK 120.

		Control register 1
WRITE	@20h 0Fh	Interpolation 50-fold and
WRITE	@20h 03h	program bits D2 and D3 with 1 - 0 - 1 to reset the signal error bit.
WRITE	@20h 0Fh	
		Control register 2
WRITE	@30h 00h	Only software latching

Below is a list of instructions to initialise the counter ICs.

			Counter X1
WRITE	@0Fh	00h	Enable latch inputs to the counter, REF mark latches to data register 0.
WRITE	@0Dh	FFh	Delete status bits
WRITE	@0Ch	03h	4 fold, normal direction, linear mode.
WRITE	@0Ah	00h	Reference mark inactive.
WRITE	@0Bh	05h	Zero and start the counter.
READ	@03h	DUMMY	Activate data register 0
READ	@07h	DUMMY	Activate data register 1
			Counter X2
WRITE	@1Fh	00h	
WRITE	@1Dh	FFh	
WRITE	@1Ch	03h	
WRITE	@1Ah	00h	
WRITE	@1Bh	05h	
READ	@13h	DUMMY	
READ	@17h	DUMMY	

Software latching

Once the counters are initialised the count values can be stored in data register 0 of the counter IC with the following command.

WRITE	@08h	FFh	Latches count X1 to data register 0
WRITE	@18h	FFh	Latches count X2 to data register 0

The count value can be read from data register 0 as follows:

READ	@00h	CountX1	Reads value to a 32 bit variable CountX1.
READ	@10h	CountX2	Reads value to CountX2.

Converting the count value to millimetres

The IK 120 can be programmed to carry out 25-fold or 50-fold interpolation, producing signal 25 or 50 TTL signal periods from one period of the input signals. The counter ICs can then also be programmed to generate 1, 2 or 4 counts per TTL signal periods.

To convert the number of counts to millimetres with a linear measuring system:

$$\text{Value [mm]} = \text{Count} * \frac{\text{Signal period [mm]}}{\text{Evaluation} * \text{Interpolation}}$$

e.g. Grating period 20 µm, four fold evaluation, 50-fold interpolation

$$\text{Value [mm]} = \text{Count} * \frac{0.020\ \text{[mm]}}{4 * 50}$$

Converting the count value to degrees

To convert the number of counts to degrees with a rotary encoder:

$$\text{Value [°]} = * \frac{\text{Count} * 360\ [°]}{\text{Evaluation} * \text{Interpolation} * \text{Lines/rev}}$$

e.g. 36000 lines per rev, four fold evaluation, 50-fold interpolation

$$\text{Value [°]} = \text{Count} * \frac{360\ [°].}{4 * 50 * 36000}$$

Example 1, Displaying count values

A simple program to display the count value is shown below:

```
                                    Initialise Control register 1
WRITE   @20h  0Fh                   Interpolation 50-fold and
WRITE   @20h  03h                   program bits D2 and D3 with
                                    1 - 0 - 1 to reset the signal error
                                    bit.
WRITE   @20h  0Fh

                                    Control register 2
WRITE   @30h  00h                   Only software latching

                                    Initialise Counter X1
WRITE   @0Fh  00h                   Enable latch inputs to the
                                    counter, REF mark latches to
                                    data register 0.
WRITE   @0Dh  FFh                   Delete status bits
WRITE   @0Ch  03h                   4 fold, normal direction, linear
                                    mode.
WRITE   @0Ah  00h                   Reference mark inactive.
WRITE   @0Bh  05h                   Zero and start the counter.
READ    @03h  DUMMY                 Activate data register 0
READ    @07h  DUMMY                 Activate data register 1

                                    Initialise Counter X2
WRITE   @1Fh  00h
WRITE   @1Dh  FFh
WRITE   @1Ch  03h
WRITE   @1Ah  00h
WRITE   @1Bh  05h
READ    @13h  DUMMY
READ    @17h  DUMMY

WHILE NOT KEYPRESSED DO
BEGIN
        WRITE @08h FFh
        WRITE @18h FFh
        READ  @00h CountX1          (32 bit)
        READ  @10h CountX2          (32 bit)
        PRINT CountX1 * 0.02 / 200
        PRINT CountX2 * 0.02 / 200
END
```

```
{Example program for software latching with IK 120 in Turbo Pascal.}
PROGRAM Example1;
USES CRT;
VAR   CountX1, CountX2: longint;
BEGIN
   mem[$C800:$20]:= $0F;          {Interpolation 50 fold}
   mem[$C800:$20]:= $03;          {Reset signal error bit}
   mem[$C800:$20]:= $0F;          {Reset signal error bit}
   mem[$C800:$30]:= $00;          {Software latching}
                                  {COUNTER X1}
   mem[$C800:$0F]:= $00;          {Enable counter latches}
   mem[$C800:$0D]:= $FF;          {Reset status bits}
   mem[$C800:$0C]:= $03;          {4 fold, normal, linear}
   mem[$C800:$0A]:= $00;          {REF mark inactive}
   mem[$C800:$0B]:= $05;          {Zero and start counter}
   CountX1:=      meml[$C800:$00];{Enable data register0}
   CountX1:=      meml[$C800:$04];{Enable data register1}

   mem[$C800:$1F]:= $00;            {COUNTER X2}
   mem[$C800:$1D]:= $FF;
   mem[$C800:$1C]:= $03;
   mem[$C800:$1A]:= $00;
   mem[$C800:$1B]:= $05;
   CountX2:=      meml[$C800:$10];
   CountX2:=      meml[$C800:$14];

   ClrScr;
   REPEAT
      mem[$C800:$08]:= $FF;              {CountX1 to data reg. 0}
      mem[$C800:$18]:= $FF;              {CountX2 to data reg. 0}
      CountX1:= meml[$C800:$00];          {Read  from data reg. 0}
      CountX2:= meml[$C800:$10];          {Read  from data reg. 0}
      GotoXY(10,10);
      WRITELN(  'X1= ',CountX1*0.02/200 :6:3,
              '  X2= ',CountX2*0.02/200 :6:3);
   UNTIL KEYPRESSED;
END.
```

```
'EXAMPLE1, Test program for the IK 110/120 in QBASIC
DEF SEG = &HC800
POKE &H20, &HF
POKE &H20, &H3
POKE &H20, &HF
POKE &H30, &H0

POKE &HF, &H0
POKE &HD, &HFF
POKE &HC, &H3
POKE &HA, &H0
POKE &HB, &H5
Dummy = PEEK(&H3)
Dummy = PEEK(&H7)

POKE &H1F, &H0
POKE &H1D, &HFF
POKE &H1C, &H3
POKE &H1A, &H0
POKE &H11B, &H5
Dummy = PEEK(&H13)
Dummy = PEEK(&H17)

CLS
PRINT "TESTIK Test Program for the IK 110/120 in QBASIC"
DIM COUNTX1 AS LONG
DIM COUNTX2 AS LONG

WHILE INKEY$ <> CHR$(27)
    POKE &H8, &HFF
    POKE &H18, &HFF
    P00 = PEEK(0)
    P01 = PEEK(1)
    P02 = PEEK(2)
    P03 = PEEK(3)
    P10 = PEEK(&H10)
    P11 = PEEK(&H11)
    P12 = PEEK(&H12)
    P13 = PEEK(&H13)

    IF (P03 AND &H80) = &H80 THEN
      COUNTX1& = (P00 - 255) + 256 * (P01 - 255) + 65536 * (P02 - 255) +
                 16777216 * (P03 - 255) - 1
    ELSE COUNTX1& = P00 + 256 * P01 + 65536 * P02 +  16777216 * P03
    END IF
```

```
    IF (P13 AND &H80) = &H80 THEN
      COUNTX2& = (P10 - 255) + 256 * (P11 - 255) + 65536 * (P12 - 255) +
                 16777216 * (P13 - 255) - 1
    ELSE COUNTX2& = P10 + 256 * P11 + 65536 * P12 + 16777216 * P13
    END IF

    LOCATE 9, 1
    PRINT USING "####.###"; COUNTX1& * .02 / 200; COUNTX2& * .02 / 200
WEND
```

Checking/Resetting signal error status bits

The signal error bit is set when the input signals drop below a particular level and remains set until the status register is deleted by writing to resister 0Dh and control register 1, bits 2 and 3.

```
READ    @0E   STATUSX1
READ    @1E   STATUSX2

IF  STATUSX1 BIT D3=1 THEN
BEGIN
        WRITE "SIGNAL ERROR X1"
        WRITE @20h 07h              Reset signal error bit
        WRITE @0Dh FFh
END

IF  STATUSX2 BIT D3=1 THEN
BEGIN
        WRITE "SIGNAL ERROR X2"
        WRITE @20h 0Fh              Reset signal error bit
        WRITE @0Dh FFh
END
```

Hardware latching

To use the external functions input to latch the count values with an external signal you must change the initialisation to include the following line:

```
WRITE   @30h 06h      Activate latch inputs for X1 and X2
```

Read the status register from each counter until bit D1 has been set to show that a latch signal has been detected and a count value has been stored in data register 1.

Hardware latching always latches to data register 1.

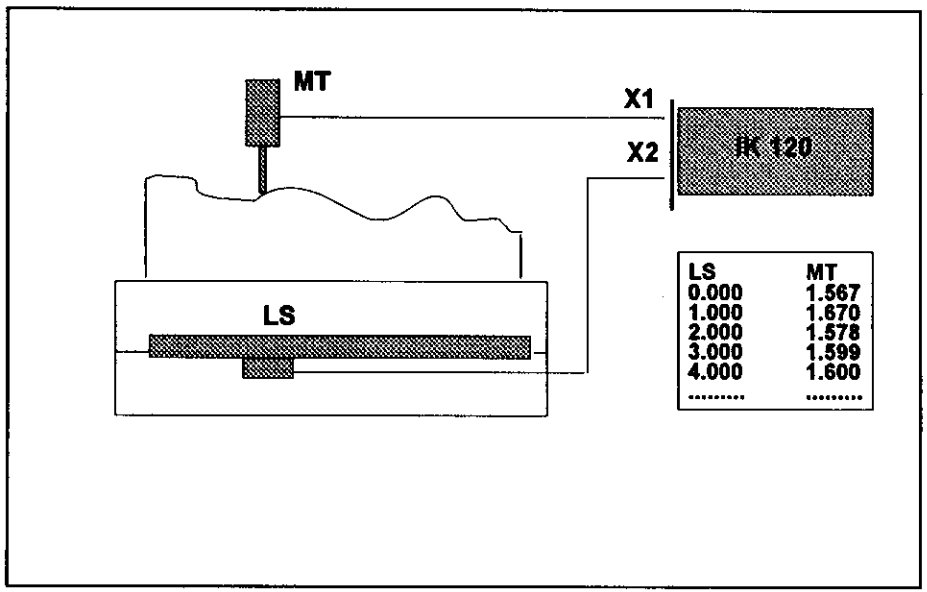
```
READ    @0Eh  STATUSX1
READ    @1Eh  STATUSX2

IF STATUSX1 BIT D0=1 THEN
BEGIN
        READ @04h  COUNTX1      'Data register 1
        PRINT 'X1 =', COUNTX1 * 0.02 / 200
END

IF STATUSX2 BIT D0=1 THEN
BEGIN
        READ @14h  COUNTX2      'Data register 1
        PRINT 'X2=', COUNTX2 * 0.02 / 200
END
```

Using the interval counter

The interval counter is a third counter which runs parallel to the counter for input X2. It can be used to latch count values at constant distance intervals. This is especially suitable for comparison or profile measurements.
The interval counter can be programmed to count down from a particular 16 bit value and generate a latch signal when it reaches zero. The latch signal (latch0) is connected to a programmable logic controller (PLD) which is programmed with bits D0-D3 of control register 2.
When "L0" and "LOut" is programmed the latch signal is gated to the latch0 pin of the counter ICs and to the LOut pin of the external functions connector.

Example

Program the interval counter to generate a latch signal every 1 mm and store the value from the METRO measuring gauge.



Calculating the interval value
Latch signal required every 1mm

$$\text{Counts per interval} = \frac{\text{Interval} * \text{Evaluation} * \text{Interpolation}}{\text{Signal period}}$$

$$\text{Counts per interval} = \frac{1\ \text{mm} * 4 * 50}{0.02\ \mu\text{m}}$$

$= 10000$
$= 2710\text{hex}$
Lower byte $= 10\text{h}$
Upper byte $= 27\text{h}$

Initialising the interval counter.

```
WRITE   @40h 10h        Set lower byte of interval value
WRITE   @50h 27h        Set upper byte of interval value

WRITE   @20h 0Fh        Set evaluation of interval
                        counter to 4-fold.
WRITE   @30h 8Fh        Reset interval counter and
                        enable latch0 from interval
                        counter.

WRITE   @30h 1Fh        Start interval counter.
```

Every 10000 counts (2710hex) or 1 mm the interval counter generates a latch signal which is connected to the latch0 input of both counters. Bit D0 of the status byte of the counters must be checked to see when a latch0 signal has occurred. Once it has occurred the value must be read from data register 0 before the next latch0 signal occurs.

```
READ    @0E   STATUSX1
IF STATUSX1 BIT D0 = 1  THEN
BEGIN
        READ @00h          COUNTX1  'Data register 0
        READ @10h          COUNTX2  'Data register 0
        PRINT 'X1 =', COUNTX1 * 0.01 / 200
        PRINT 'X2 =', COUNTX2 * 0.01 / 200
END
```

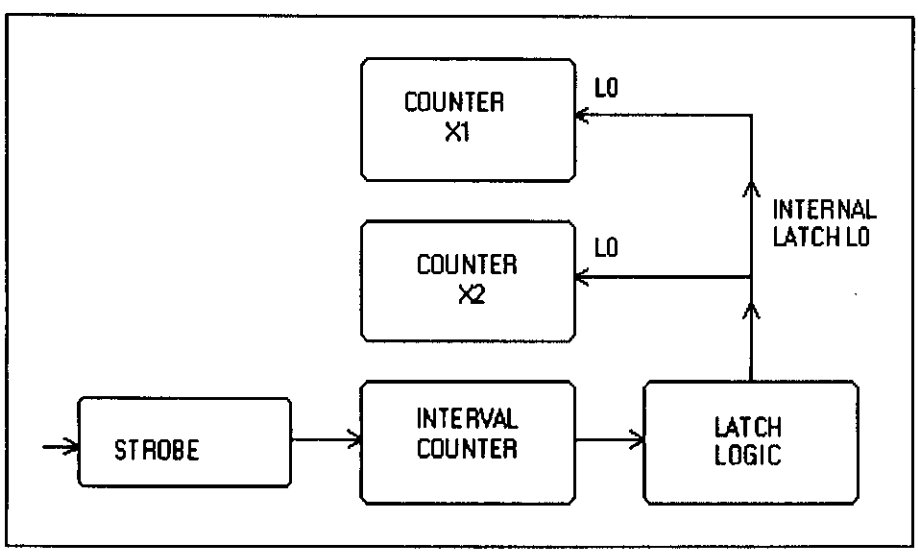
```
{Sample program using the interval counter on the IK 120 in Turbo Pascal.}
PROGRAM Example2;
USES CRT;
VAR   CountX1, CountX2: longint;
BEGIN
   mem[$C800:$20]:= $0F;          {Interpolation 50 fold}
   mem[$C800:$20]:= $03;          {Reset signal error bit}
   mem[$C800:$20]:= $0F;          {Reset signal error bit}
   mem[$C800:$30]:= $00;          {Software latching}
                                  {COUNTER X1}
   mem[$C800:$0F]:= $00;          {Enable counter latches}
   mem[$C800:$0D]:= $FF;          {Reset status bits}
   mem[$C800:$0C]:= $03;          {4 fold, normal, linear}
   mem[$C800:$0A]:= $00;          {REF mark inactive}
   mem[$C800:$0B]:= $05;          {Zero and start counter}
   CountX1:=  meml[$C800:$00];    {Enable data register0}
   CountX1:=  meml[$C800:$04];    {Enable data register1}

   mem[$C800:$1F]:= $00;          {COUNTER X2}
   mem[$C800:$1D]:= $FF;
   mem[$C800:$1C]:= $03;
   mem[$C800:$1A]:= $00;
   mem[$C800:$1B]:= $05;
   CountX2:=  meml[$C800:$10];
   CountX2:=  meml[$C800:$14];
                                  {INTERVAL COUNTER}
   mem[$C800:$40]:= $10;          {Set lower byte interval }
   mem[$C800:$50]:= $27;          {Set upper byte interval }
   mem[$C800:$20]:= $0F;          {4 fold evaluation interval }
   mem[$C800:$30]:= $8F;          {Enable latch0, Reset}
   mem[$C800:$30]:= $1F;          {Start interval counter}

   ClrScr;
   REPEAT
      IF (mem[$C800:$0E] AND $01) =1 THEN
      BEGIN
         CountX1:= meml[$C800:$00];  {Read CountX1}
         CountX2:= meml[$C800:$10];  {Read CountX2}
         WRITELN(   'MT= ',CountX1*0.01/200 :6:3,
                    '   LS= ',CountX2*0.02/200 :6:3);
      END;
   UNTIL KEYPRESSED;
END.
```

```
'EXAMPLE2, Test program for interval counter in QBASIC
DEF SEG = &HC800
POKE &H20, &HF
POKE &H20, &H3
POKE &H20, &HF
POKE &H30, &H0

POKE &HF, &H0
POKE &HD, &HFF
POKE &HC, &H3
POKE &HA, &H0
POKE &HB, &H5
Dummy = PEEK(&H3)
Dummy = PEEK(&H7)

POKE &H1F, &H0
POKE &H1D, &HFF
POKE &H1C, &H3
POKE &H1A, &H0
POKE &H11B, &H5
Dummy = PEEK(&H13)
Dummy = PEEK(&H17)

POKE &H40, &H10
POKE &H50, &H27
POKE &H20, &HF
POKE &H30, &H8F
POKE &H30, &H1F

CLS
PRINT "TESTIK Test Program for interval counter in QBASIC"
DIM COUNTX1 AS LONG
DIM COUNTX2 AS LONG

WHILE INKEY$ <> CHR$(27)
    IF (PEEK(&HE) AND 1) = 1 THEN
      P00 = PEEK(0)
      P01 = PEEK(1)
      P02 = PEEK(2)
      P03 = PEEK(3)
      P10 = PEEK(&H10)
      P11 = PEEK(&H11)
      P12 = PEEK(&H12)
      P13 = PEEK(&H13)
```

```
    IF (P03 AND &H80) = &H80 THEN
      COUNTX1& = (P00 - 255) + 256 * (P01 - 255) + 65536 * (P02 - 255) +
                 16777216 * (P03 - 255) - 1
    ELSE COUNTX1& = P00 + 256 * P01 + 65536 * P02 + 16777216 * P03
    END IF

    IF (P13 AND &H80) = &H80 THEN
      COUNTX2& = (P10 - 255) + 256 * (P11 - 255) + 65536 * (P12 - 255) +
                 16777216 * (P13 - 255) - 1
    ELSE COUNTX2& = P10 + 256 * P11 + 65536 * P12 + 16777216 * P13
    END IF
    PRINT USING "####.###"; COUNTX1& * .02 / 200;
                            COUNTX2& * .02 / 200
  END IF
WEND
```

Simultaneous latching of 2 or more counters

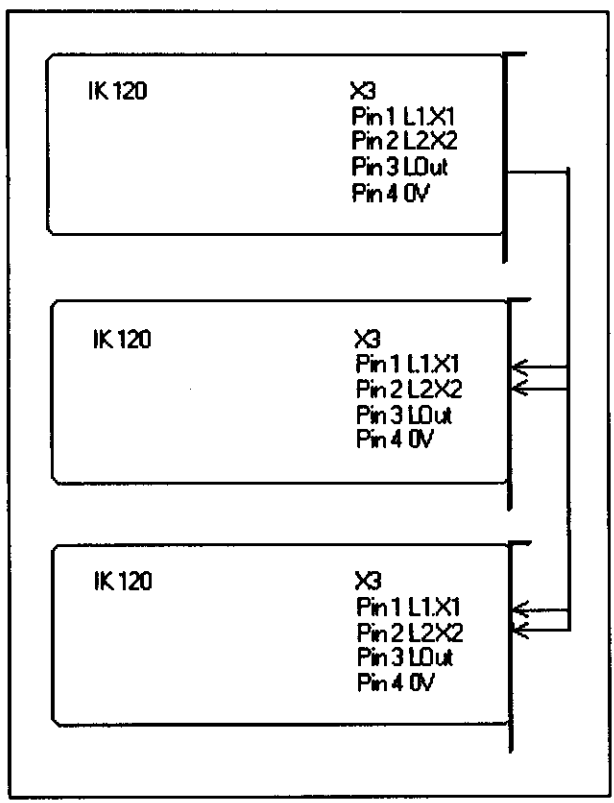
The two counters on the IK 120 can be latched simultaneously with one software write command to the strobe address. The interval counter must be programmed to generate a latch0 signal to both counters when a strobe signal is detected.



Initialise and latch the interval counter as shown below:

WRITE	@30h EFh	Latch when strobed, all latch signals enabled
WRITE	@60h FFh	Strobe signal
READ	@00h COUNTX1	'Data register 0
READ	@10h COUNTX2	'Data register 1

It is possible to simultaneously latch the counters on more than one IK 120 by connecting the LOut output of one IK to the latch inputs on the other IKs.



Initialise and latch the IK 120s as shown below:

```
WRITE   @30h EFh                Latch when strobed, all latch
                                signals enabled
WRITE   @430h 08h               Latch signals L1.X1 or L1.X2
                                enabled on 2nd IK 120

WRITE   @60 FFh                 Strobe signal
                                        '1st IK 120
READ    @00   COUNTX1           'Data register 0
READ    @00   COUNTX2           'Data register 0
                                        '2nd IK 120
READ    @404  COUNTX3           'Data register 1
READ    @404  COUNTX4           'Data register 1
```

Interrupt programming

The IK 120 can also generate an interrupt request signal on the PC bus when a latch signal has been generated by the interval counter or detected on the external latch inputs L1.X1 or L1.X2.

The interrupt signal can be used to call a software routine which reads the latched counter values.

We do not recommend using interrupt programming with the IK 120 for the following reasons:

- **Serial polling of the status byte is the fastest method of detecting when a latch signal has occurred.**
- **Very few applications require interrupt programming.**
- **Interrupt programming requires a great deal of expertise.**

Set control register 1, bit D6 to 1 to enable the interrupt signal from the interval counter, **Int0**.
Set control register 1, bit D7 to 1 to enable the interrupt signal from an external latch signal, **Int1**.

The interrupt jumper on the IK 120 must also be set appropriately, see sectiiion "HARDWARE, Interrupt jumper assignment".

For an example of interrupt programming refer to the sample program "INTERUPT.PAS".

Using reference marks

In order to provide an absolute reference in incremental encoders, one or several reference marks are located next to the periodic graduation. A reference mark consists of a random graduation pattern which - when traversed - produces one narrow signal peak.

The counter ICs can be programmed to reset the count value to zero when the reference mark has been detected When using the reference mark any further changes in the datum point can be made with a software preset value which is added to the counter value in the software.

It is good practice to reset and start the counter when the reference mark is detected. This gives a repeatable zero point for the measurement, which is useful for comparing the results of two measurements.

Using distance-coded reference marks

It is often impractical to move axes by large distances in order to re-establish the offset of workpiece datum to machine reference point. HEIDENHAIN solves this problem by offering distance-coded reference marks on both linear (LS and LID series) and rotary encoders.

The scale comprises the grating and a parallel track for the reference marks. By varying the distance between neighbouring reference marks by a defined rule, the absolute position of each reference mark is encoded.

With **encoders** featuring distance-coded reference marks, the absolute position value can be re-established by crossing over two consecutive reference marks, i.e. after a traverse of 20 mm/0.8 in. or less.

Encoders featuring distance-coded reference marks are designated with the suffix C.

To use distance-coded reference marks refer to "IK120.PAS, FUNCTION DistanceCodeREFMarks" and thhe program "CREFMARK.PAS".

Software Description

Driver software description

A driver software is supplied in Turbo Pascal and Microsoft "C", bothh using the same names and variables.

The following describes the driver software in Turbo Pascal but can also be applied to the driver software in "C" whhich is very siimilar.

Pre-defined variables and data types

The units "IK 120.PAS" and IK 120.H" define types and variables to be used as parameters when calling the procedures in the driver software.

Variable: **BOARDADR**

Description: Specifies the base address (segment address) of the first IK 120. This is set to C800h in the routine "IK 120.PAS" and must be changed if the base address of the first IK 120 is changed using the DIP switches.

Used by: All routines in the driver software.

Possible values:

C800h, C840h, C880h,... FFC0h.

Type: **CONTROL**

Description: Programmable functions of the counter ICs such as start, stop, reset, etc.

Used by: Init_Counter, Init_Latch.

Possible values:

Value	Meaning
c_reset	Set the counter to zero. The counter can only be set to zero and not to any other value.
c_start	Start counter
c_stop	Stop counter
reset_start	Zero and start counter.
reset_stop	Zero and stop counter.
RI_stopt	Stop counter when reference (REF) mark is detected.
RI_start	Start counter when the REF mark is detected.
RI_reset_stop	Zero and stop counter when REF mark is detected.
RI_reset_start	Zero and start counter when REF mark is detected.
RI_reset	Zero counter when REF mark is detected.
RI_latch_stop	When REF mark is detected store current count value to the data register specified by bit D4 of the initialising register @0Fh then stop counter
RI_latch_start	When REF mark is detected store current count value to the data register specified by bit D4 of the initialising register @0Fh then start counter
RI_latch_reset_stop	When REF mark is detected store current count value to the data register specified by bit D4 of the initialising register @0Fh then stop and zero counter
RI_latch_reset_start	When REF mark is detected store current count value to the data register specified by bit D4 of the initialising register @0Fh then start and zero counter
RI_latch_reset	When REF mark is detected store current count value to the data register specified by bit D4 of the initialising register @0Fh then zero counter

Used only with procedure "Init_Latch".

Value	Meaning
c_latch	The interval counter generates a latch signal immediately.
c_load	Set the interval counter to the value to be used as the interval value.
RI_load	When REF mark is detected set the interval counter to the value to be used as the interval value.
Strobe_Latch	Generate a latch signal when a the strobe register is written to.

Type: **EVAL**
Description: Programs the signal evaluation of the counter.
Used by: Init_Counter, Init_Latch, DistanceCodedREFMarks.
Possible values:

Value	Meaning
onefold	One fold signal evaluation. Only one edge per TTL signal period is counted.
twofold	Two fold signal evaluation. Two edges per TTL signal period are counted.
fourfold	Four fold signal evaluation. Four edges per TTL signal period are counted.

Type: **DIRECTION**
Description: Programs the count direction of the counter.
Used by: Init_Counter, Init_Latch, DistanceCodedREFMarks.
Possible values

Value	Meaning
normal	"Normal" counting direction, positive = signal I_1 leads I_2 by 90° el
inverse	inverse" counting direction positive = signal I_1 lags I_2 by 90° el

Type: **METHOD**

Description: Programs the range of the counter.
Used by: Init_Counter, Init_Latch, DistanceCodedREFMarks.
Possible values:

Value	Meaning
linear	Counter range -2^{25} to $+2^{25}-1$
arc	Counter range -180000 to +179999. This can be used for rotary axes if this range is required. "Linear" can also be used for rotary axes.

Type: **INTERPOL**
Description: Programs the signal interpolation of the counter.
Used by: Interpolation.
Possible values:

Value	Meaning
I_25	25-fold signal interpolation
I_50	50-fold signal interpolation

Type: **LCONTROL**
Description: Specified a particular latch signal.
Used by: Latch_Enable, Latch_Disable.
Possible values:

Value	Meaning
Internal	Latch signal from interval counter otherwise known as latch0
ExternalX1	External latch signal to counter X1
ExternalX2	External latch signal to counter X2
ExternalX1X2	External latch signal to counters X1 and X2
Latchout	Latch output signal to connector X3

Type: **INTERR**
Description: Specifies a particular interrupt signal source.
Used by: Interrupt_Enable
Possible values:

Value	Meaning
Int_0	Interrupt signal generated by interval counter.
Int_1	Interrupt signal generated from external latch signals.

Init_Interface

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2

Initialises the counter. Should be called before any other functions.

Interpolation

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
m_interpol:	I_25 = Set interpolation to 25-fold
	I_50 = Set interpolation to 50-fold

Sets the interpolation of the selected input signal.

When possible use 25-fold interpolation with a maximum input frequency of 50 kHz.

Init_Counter

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
m_control:	Variable of type control to program function of the interval counter.
m_eval:	onefold, twofold, fourfold
Direction:	Counting direction = normal or inverse
m_method:	Counting method = linear or arc

This routine programs the main functions of the specified counter IC.
See section "Pre-defined types" for the description of the parameters

Soft_Count

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Register:	0 = Data register 0, 1= Data register 1
Result	
COUNT	Value from the specified data register returned in 32 bit variable COUNT.

This routine stores the counter value to the specified data register of the specified counter IC and returns it in the variable named "COUNT" as a 32 bit, 2's compliment signed integer.

This routine is a combination of "Latch_Count" and "Read_Count".

Latch_Count

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Register:	0 = Data register 0, 1= Data register 1

Stores the count value of the selected input to the specified data register in the counter IC.

Use procedure "Read_Count" to read the count value after latching it.

Read_Count

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Register:	0 = Data register 0, 1= Data register 1
Result:	
COUNT	Value from the specified data register returned in 32 bit variable COUNT.

This routine reads the counter value from the specified data register of the specified counter IC.
The value is returned in the variable named "COUNT" and is a 32 bit, 2's compliment signed integer.

Use when the interval counter or an external latch signal latches the counter.

Reset_Status

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2

Resets the status byte of the selected counter.

Reset_uas

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2

Reset the error bit in control register 1 which shows errors on the input signals.

Signal_Error

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Result	True = Signal error on this counter False = No signal error

Function returns true when an error has occurred on the input signal of the selected counter.

Read_Status

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Result	Status byte from selected counter.

Status byte gives information on whether a latch signal or signal error has occurred.

Read_ScanReg

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Result	Returns byte from scan register from selected counter.

Scan register gives the signal levels on several pins on the counter ICs.

Latch_Enable

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Lcommand:	Internal, ExternalX1, ExternalX2, ExternalX11X2, LatchOut

Enables the specified latch signal for the specified counter.

Latch_Disable

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
Lcommand:	Internal, ExternalX1, ExternalX2, ExternalX11X2, LatchOut

Disables the specified latch signal for the specified counter.

Latched

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
latch:	0 = latch from interval counter 1 = latch from external-functions connector
Result	true = Latch has occurred false = Latch not yet occurred.

Use this function to detect when a latch signal has occurred.

Init_Latch

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
m_control:	Variable of type CONTROL to reset, start, stop interval counter
m_eval:	Signal evaluation = onefold, twofold, fourfold
value:	16 bit Interval value

Initialises the interval counter, sets the interval value and signal evaluation.

Set signal evaluation to onefold to obtain a maximum interval separation.

Write_Strobe

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2

Writes a strobe signal on the selected IK 120.

Interrupt_Enable

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
m_interr:	Int_0, Int_1

Enable the selected interrupt.
Int_0 = interrupt from interval counter
Int_1 = interrupt from external signal

Clear_Int

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2

Resets the interrupt flag on the IK 120. Should be used at the end of the interrupt service routine.

DistanceCodedREFMarks

(Only in IK 120.PAS, not available in Microsoft "C".)

Parameters:	
axis:	0 = Counter X1, 1 = Counter X2
BasicSpacing:	1000 GP, 2000 GP, 500 GP
SubDiv:	I_25 or I_50
Edges:	onefold, twofold, fourfold
CountDir:	normal, inverse
Meth:	linear, arc
Result	Position of the first reference mark in signal periods.

This routine zeros the specified counter as the first reference mark is detected. When the encoder traverses the next reference mark the absolute position of the first reference mark is returned. This value is a 32 bit signed integer (type "longint") which must be multiplied by the signal period.

The counter for the selected axis is zeroed on the first reference mark that was detected.

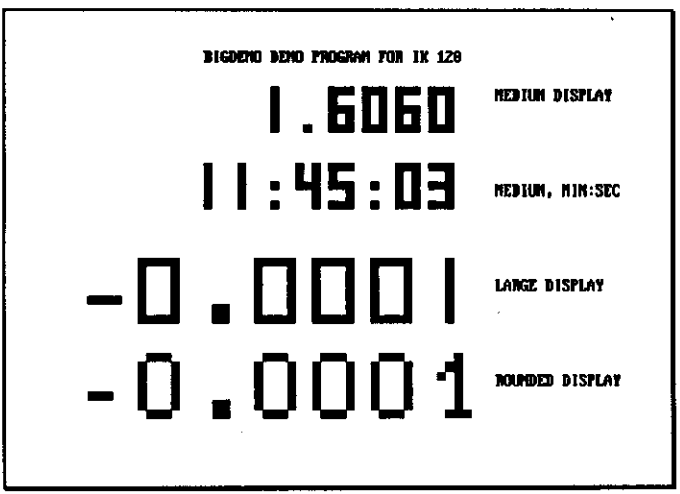
The "BasicSpacing" is the constant separation between the reference marks 0 and 2, 2 and 4, etc. This basic spacing is quoted in signal periods and is normally 1000 with standard LS systems with a signal period of 20 µm and a basic spacing of 20 mm. Other values such as 500 for LID systems and 2000 for certain angle measuring systems are also common.

The routine can be interrupted by pressing any key on the keyboard and the value 0 is returned.

Unit for large count display - BIGDISP.PAS

The file "BIGDISP.PAS" is a Turbo Pascal unit to display count values in a large, easily readable form in character mode.

The program "BIGDEMO.PAS" shows the various types of display and illustrates how to use this unit.



Pre-defined variables and data types

Type: **DisplayMode**
Description: Specifies the size of the display
Used by: LDisplay
Possible values:

Value	Meaning
Medium	Medium sized display
Large	Large display
Rounded	Large display with rounded numbers

Procedure LDisplay

Parameters:	
Value:	The value to be displayed (real).
DMode:	Type of display: Medium, Large, Rounded (see above).
posx:	Horizontal position on the screen of the top left character of the first digit or sign (byte).
posy:	Vertical position of the top left character of the first digit or sign (byte).
NoOfChars:	Total number of characters to be displayed, including sign, decimal point and spaces (byte).
DigitsAfterDecimalPoint:	Number of digits to be displayed after the decimal point, trailing zeros will be added if necessary (byte).
MinutesSeconds:	True=display in deg:minutes:seconds False=normal decimal display
Flash:	True = display flashes False = normal display Note: Use function "Signal_Error" to give a flashing display when a signal error has occurred.
LastValStr:	This variable stores the string that is displayed on the screen. The user must use a separate variable for each separate axis display and must initialise the string to the correct length with "X" characters.

PC Timer unit - TIMER.PAS

The file "TIMER.PAS" is a Turbo Pascal unit to store values from the IK 120 at constant intervals in time. This unit contains special routines which allows you to achieve a higher time resolution than that of 55 ms normally available in Turbo Pascal.

Using this routine will cause the DOS clock to run slow or fast while this routine is running.

The program "MEASURE" shows how to use this unit, see description below.

Pre-defined variables and data types

Constant: LocalMPMax
Description: Specifies the maximum number of points per axis that can be stored.
Set to: 5000 points

Variable: Buffer
Description: Two dimensional array of longint where the count values for each axes are stored.
Definition: Buffer: array [0..LocalMpmax+1,0..1] of longint;

Procedure SetTimerInterval

Parameters:	
Interval:	The time interval in milliseconds between data points (real).

Sets the DOS timer interrupt so that it no longer occurs every 55ms but at time intervals defined by the parameter "Interval". The DOS clock now registers 55 ms for the time interval that has been set by this routine.

Procedure StartDataAquisition

Parameters:	
No_of_points:	The number of count values to be stored per axis.

Starts to store the specified number of count values to RAM in the array "BUFFER" . Procedure "SetTimerInterval" must be called before this procedure.

Procedure RemoveTimer

Parameters:	none

Resets the timer interrupt to the standard DOS interrupt. From this point on the DOS clock will run normally again but may need resetting to the correct time.

Sample programs in Turbo Pascal

Both source code ".PAS" and executable files ".EXE" for all sample programs are stored in directory "\TP" on the driver software disk.

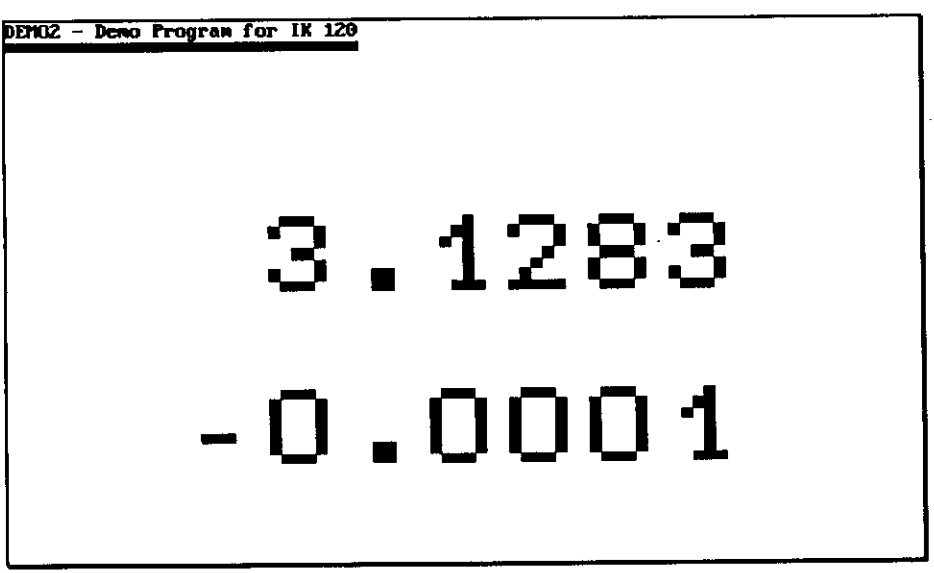
Sample programs show how to use the driver software "IK 120.PAS" and can be modified for special applications.

DEMO2

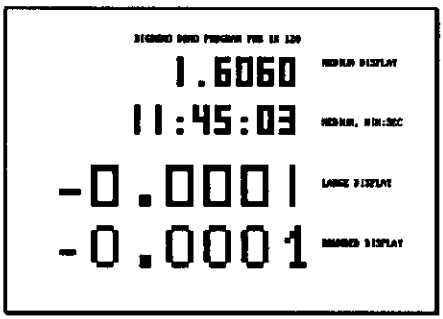
A simple program to display the count from inputs X1 and X2 on the screen. Below is a listing of the source code:

```
(*****************************************************************)
(* Dr JOHANNES HEIDENHAIN, Traunreut, Germany.                   *)
(* Demonstration program to illustrate the programming of the interface  *)
(* card IK120 using the IK120.PAS unit supplied.                 *)
(* Illustrates the use of Soft_Count                             *)
(* Signal checking is included in this program                   *)
(*****************************************************************)
program DEMO2;

uses IK120, crt,BigDisp;

var
  Period:      real;
  SubDivision : integer;
  s,sz:        string;
  lastval :    array [0..1] of string;

procedure Display_count( axis,x,y: integer);
var Count:longint;
   Status: byte;
   Val:real;
begin
  Count := 0;
  Status := 0;
  Soft_Count(axis,Status,Count);
  Val:= Period*Count/Subdivision;
  Ldisplay(Val, Rounded,x,y,8,4,false,Signal_error(axis),lastval[axis]);
end;

(********************** main program **********************)
BEGIN
  clrscr;
```

```
  Writeln('DEMO2 - Demo Program for IK 120');
  Writeln('_____');
  Period := 0.010;  { Grating period in mm }
  Subdivision := 200;
  m_interpol:=I_50;
  Init_Interface(0);
  Init_Interface(1);
  Interpolation(0,m_interpol);
  Interpolation(1,m_interpol);
  Init_Counter (0,reset_start,fourfold,normal,linear);
  Init_Counter (1,reset_start,fourfold,normal,linear);
  Reset_uas(0);
  Reset_uas(1);
  Reset_Status(0);
  Reset_Status(1);

  lastval[0]:='XXXXXXXXXX';
  lastval[1]:='XXXXXXXXXX';

  repeat
    Display_count(0,10,10);
    Display_count(1,10,18);
  until keypressed;
END.
```



BIGDEMO
Shows how to use the unit BIGDISP.PAS (see description above) to display count values as large digits on the screen.

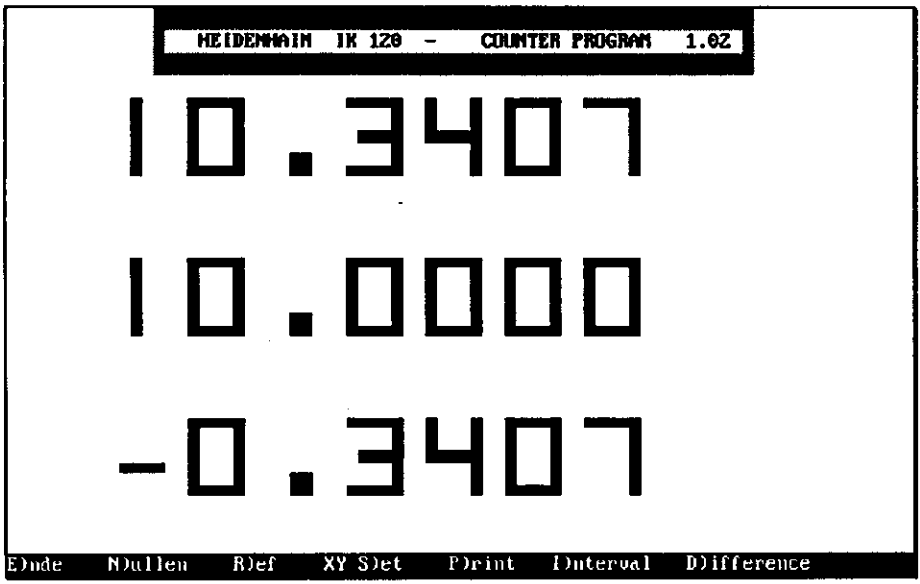


COUNTER
This program offers the functions of a standard counter with up to eight inputs, with 4 IK 120s:

- Set to zero,
- Preset,
- Large display,
- Any signal period or line number,
- Evaluation of single or distance-coded reference marks,
- Difference display with two inputs,
- Printout of display values,
- Interval counter can store count values from both axes at constant distances.

This program can also be easily modified for specific applications.

Below is a copy of the screen with the difference display activated.



The program counter reads its set-up information from the file **'COUNTPAR.DAT'** shown below.

```
Number of Axes
2
Address of Interface card IK 120 in decimal format not HEX ( C800= 51200 )
51200
Places after the decimal point
4
Name for the axes e.g. XYZ
XYZ45678
Signal period in mm for each axes, Reference marks 0=Standard, 500, 1000, 2000
0.010 0
0.010 0
0.010 0
0.010 0
0.010 0
0.010 0
0.010 0
0.010 0
```

Edit this file to change the set-up.

Number of axes: 1 to 8

Address of the IK 120:

Use this parameter to try the IK 120 at several address locations without having to recompile the source code.

Base address (Segment address)	Decimal value
C000	49152
C7C0	51136
C800	51200
C840	51264
C880	51328
D000	53248
E000	57344
F000	61440

Places after the decimal point:
This only changes the display. The IK 120 is automatically set to 50-fold interpolation and four fold signal evaluation.

Name for the axes:
Each character on the line is used as the name for the input on the display e.g. "XYZ45678", "ABCDEFGH", "12345678".

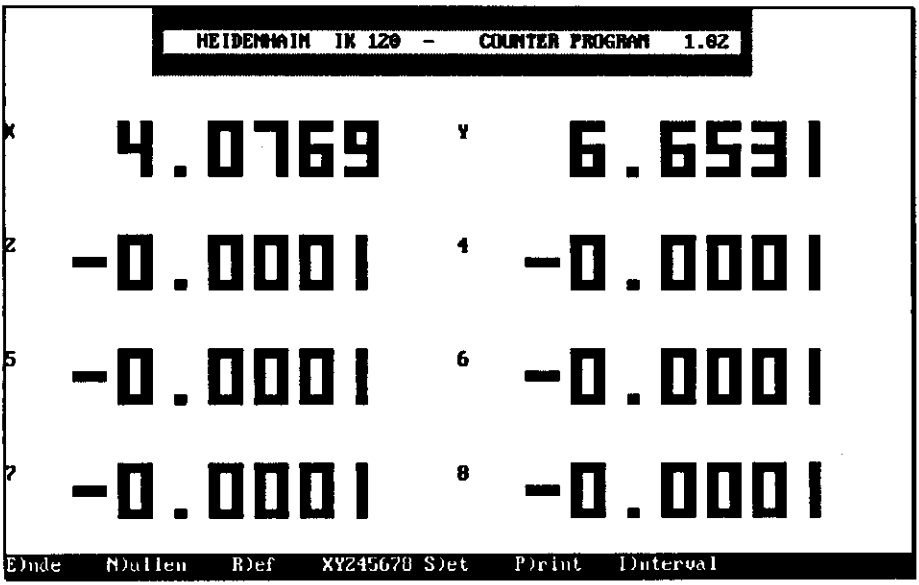
Signal period and reference mark:
Each line is the signal period in mm and the reference mark definition for each input.

To reverse the count direction enter a negative signal period.

Reference marks:

0 =	Standard single reference marks
1000=	Standard distance-coded reference marks on LS encoders, 20mm spacing, 20µm signal period
2000 =	Distance-coded reference marks on LID 311/351 20mm spacing, 10µm signal period.
500 =	Distance-coded reference marks with RON systems, 36 REF marks, 18000 lines

This program can be used with up to four IK 120 with the base address (segment address) of each card set in increments of 40h.



The file "COUNTPRE.DAT" contains the absolute position of the reference mark with respect to the last used datum point for each axis and is used by the program "COUNTER" when operating in reference mode.

EXTLATCH

Program detects external latch signals and displays the count value on the screen. Connect a switch between pins 1 and 4, and 2 and 4 of the input X3 to generate an external latch signal for inputs X1 and X2 respectively.

```
EXTLATCH - External latching of IK 120 using input X3

X3/1        X3/2        X3/3        X3/4
Blue        Green       Red         White
LatchInX1   LatchInX2   LatchOut    0 Volt
Axis X1 =    -0.0001
Axis X1 =     0.0000
Axis X2 =    -0.0005
Axis X2 =    -0.0005
Axis X2 =    -0.0005
```

FREEZE2

Program shows how to use the interval counter and the strobe function to generate an internal latch signal to freeze count values simultaneously from inputs X1 and X2.
This method should be used where counts are taken on the run to avoid the small time delay between storage of values of each input X1 and X2.

INTERVAL

```
(* Programming the interval counter using the driver software IK 120.PAS*)
Program INTERVAL;
uses IK120, crt,BigDisp;
var
  Period:      real;
  SubDivision : integer;
  s,sz:        string;
  lastval :    array [0..1] of string;

procedure Display_count( axis,x,y: integer);
var Count:longint;
   Status: byte;
   Val:real;
begin
  Count := 0;
  Status := 0;
  Read_Count(axis,Status,Count);
  Val:= Period*Count/Subdivision;
  Ldisplay(Val, Rounded,x,y,8,4,false,Signal_error(axis),lastval[axis]);
end;
(********************** main program ************************)
BEGIN
   clrscr;
   Writeln('INTERVAL - Demo Program for IK 120');
   Writeln('_____');
   Period := 0.010;  { Grating period in mm }
   Subdivision := 200;
   m_interpol:=I_50;
   Init_Interface(0);
   Init_Interface(1);
   Interpolation(0,m_interpol);
   Interpolation(1,m_interpol);
   Init_Counter (0,RI_reset_start,fourfold,normal,linear);
   Init_Counter (1,RI_reset_start,fourfold,normal,linear);
   Reset_uas(0);
```

```
  Reset_uas(1);
  Reset_Status(0);
  Reset_Status(1);

  Latch_Enable(0,Internal);
  Init_Latch(0,C_load,fourfold,20000);
  Init_Latch(0,C_Reset,fourfold,20000);
  Init_Latch(0,RI_Start,fourfold,20000);

  lastval[0]:='XXXXXXXXXX';
  lastval[1]:='XXXXXXXXXX';
  repeat
    if latched(0,0) then
    begin
        Display_count(0,10,10);
        Display_count(1,10,18);
    end;
  until keypressed;
END.
```

The interval counter is programmed to generate a latch signal every one millimetre and siplay on the screen.
For othher examples see COUNTER.PAS and EXAMPLE2.PAS and EXAMPLE2.BAS in section "Programming the IK 120".

CREFMARK
Shows how to use the procedure "DistanceCodedREFMarks" to determine the absolute position when traversing distance coded reference marks (encoders with suffix "C").

MEASURE
Program uses the unit "TIMER.PAS" to store a specified number of counts from both axes to a specified file. The counts are stored in constant time intervals which must be specified in milliseconds.

Parameters:	
Data file:	The file to which data should be stored
Number of Points:	Number of points per input, max. 5000.
Interval:	Time interval in milliseconds.
[Period:]	Signal period in mm, standard value is 0.010mm

The program can be called as follows:

```
> MEASURE DATA.DAT 1000  2  0.010  0.020
```

The measurement can be started with an external latch signal or by pressing any key.

```
MEASURE - Data aquisition for IK120 at constant time interval

To start measurement :
                        - press any key or
                        - use external latch input
Measurement started.
Storing point :    1000

Measurment complete.

Saving data point  1000   to  data.dat
```

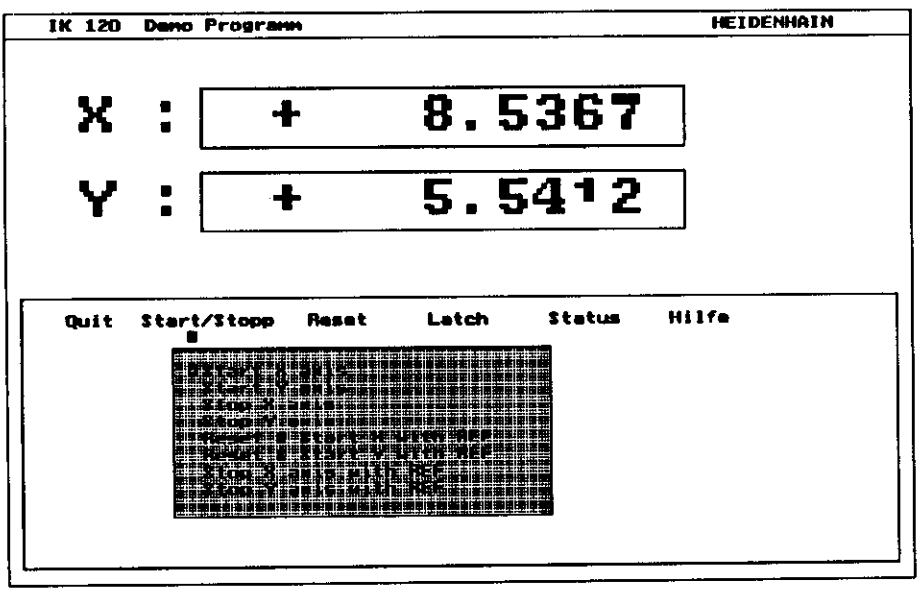
Results are written to an ASCII file as shown below:

X1	X2	X2-X1
0.1115	0.0007	-0.1108
0.1122	0.0007	-0.1115
0.1127	0.0008	-0.1119
0.1133	0.0007	-0.1126
0.1140	0.0007	-0.1133
0.1146	0.0007	-0.1139
0.1150	0.0007	-0.1143
0.1156	0.0007	-0.1149
0.1163	0.0007	-0.1156

INTERUPT

Demonstrates how to initialise the IK and program an interrupt service routine. An external latch signal generates an interrupt which causes the program to stop doing what it was currently doing and jump to the interrupt service routine. The interrupt service routine reads the count values and displays them on the screen.

DEMO
This program allows the functions of the IK to be selected to show their effect on the count values. Do not try to understand the source code in this program. Use the other programs supplied which have been kept as simple as possible for reasons of clarity.



Sample programs in Microsoft "C"
Both source code ".C" and executable files ".EXE" for all sample programs are stored in directory "\C600" on the driver software disk.

Sample programs show how to use the driver software "IK 120.C" and "IK 120.H" and can be modified for special applications.

DEMO2.C
This program displays count values from both inputs on the screen.
Below is a listing of the source code illustrating the use of the driver software.

```
/*################################################################
          demo2.c      Date: 19.03.92
  Function parameter and variables defined in header ik120.h
  Functions are defined in module ik120.c
  Link and compile:  cl /AL /Oaxz /G2 /Fc demo2.c ik120.obj
################################################################ */
#include <stdio.h>
#include <float.h>
#include <ik120.h>
unsigned int  boardseg = 0xc800;      /*  Basesadress IK120    */
unsigned int  boardoff = 0;           /*  Offset boardadress   */
unsigned int  interval = 0;           /*    Latchvalue         */

extern void Init_Interface(counter);
extern void Init_Counter(counter,control,eval,direction,method);
extern unsigned long Soft_Count(counter,latch);
/* ================================================
       Simultaneous latching and reading of both axis
  ================================================ */
void main ()
{
float c_value0,c_value1;
Init_Interface(C_0);
Init_Interface(C_1);
printf ("\n\n\t Counter1\t Counter2\n");
Init_Counter(C_0,C_start,fourfold,normal,linear);
Init_Counter(C_1,C_start,fourfold,normal,linear);

while (!kbhit())
  {
    c_value0 = (float) (signed long) Soft_Count(C_0,L_0) * 0.01 / 200;
    c_value1 = (float) (signed long) Soft_Count(C_1,L_0) * 0.01 / 200;
    printf ("\r\t%12.4f\t%12.4f",c_value0, c_value1);
  }
}
```

DEMOHEX.C
This program displays count values from both inputs on the screen in hexadecimal form.

DEMO.C
This program displays count values from both inputs on the screen and includes a check for signal errors.

What to do if it doesn't work!

Below is a list of things to try if the IK 120 doesn't work when it has been installed in your PC.

QEMM.SYS, EMM386.SYS

If the file "CONFIG.SYS" contains a command which loads a memory management programs such as "QEMM.SYS", "CEMM.SYS", "EMM386.SYS" you should add the following parameter to the command to exclude the memory range used by the IK 120 from the upper memory area.

```
DEVICE = C:\DOS\EMM386.SYS  RAM  X=C800-C840
or
DEVICE = C:\QEMM\QEMM386.SYS  RAM  X=C800-C840
```

The address range depends on the DIP switch setting on the IK 120.

WINDOWS

To run programs for the IK 120 from windows you must enter the following command in the file "C:\WINDOWS\SYSTEM.INI":

```
[boot.description]
emmexclude=C800-C840
```

The address range depends on the DIP switch setting on the IK 120.

Changing the base address of the IK 120

When the IK 120 does not function correctly try changing the base address using the DIP switches on the IK 120 (see chapter "HARDWARE").
When the base address has been changed with the DIP switches the software must also be informed of the change.

Use the program **"COUNTER"** and enter the change to the DIP switches in the file:

COUNTPAR.DAT

Address of Interface card IK 120 in decimal format not HEX (C800= 51200) 57344

Sample DIP switch settings with the decimal address which should be entered in file "COUNTPAR.DAT".

Base address (Hex)	Base address (Decimal)	DIP Switches		
		S1 S2	S3 S4 S5 S6	S7 S8
C8000	51200	ON ON	OFF ON ON ON	ON ON
C8400	51264	ON ON	OFF ON ON ON	ON OFF
C8800	51328	ON ON	OFF ON ON ON	OFF ON
CF000	52992	ON ON	OFF OFF OFF OFF	ON ON
D0000	53248	ON OFF	ON ON ON ON	ON ON
E0000	57344	OFF ON	ON ON ON ON	ON ON

Large hard disk or other storage device (SCSI interface)

PCs equipped with large hard disks or other storage devices often use the SCSI interface. This interface often uses the address space C8000. Try a different address setting for the IK 120.

CONFIG.SYS, AUTOEXEC.BAT, RAMDRIVE, VDISK

Remove all unnecessary command from the files "AUTOEXEC.BAT" and "CONFIG.SYS" especially commands containing "RAMDRIVE" or "VDISK".

To do this use an ASCII editor e.g. EDIT or EDLIN and start each line with the word **"REM "**, this stops the command from being carried out and allows you to restore the commands once you know which one caused the problem.

The computer must be rebooted once any changes have been made to "CONFIG.SYS" or "AUTOEXEC.BAT", before the changes take effect.

e.g. "CONFIG.SYS" or "AUTOEXEC.BAT"

```
REM DEVICE= C:\DOS\VDISK.SYS
REM xxxxxxxxxxxx
```

Remove all non-essential interface hardware

All non-essential interface hardware such as **network adapters**, scanner interface boards etc., should be removed to determine which board is interfering with the IK 120.

Network adapters

Network adapters often use the address space CC000-CCFFF and D800-DBFF. Check which address area your network adapter is using and make sure the IK 120 uses a different address space.

Technical Specifications

Mechanical Data	
Dimensions	340 mm x 107 mm, (13.39 x 4.21 in.), requires full length slot
Operating temp. Storage temp	0 to 45° C (32 to 113 °F) -30 to 70 °C (-22 to 158 °F)
Electrical Data	
Max. input frequency	60 kHz with 25-fold interpolation, 25 kHz with 50-fold interpolation
Signal interpolation	25-fold and 50-fold selectable with software
Signal evaluation	1-, 2-, 4-fold
Axes counters	26 bit, 2^{-25} to 0 to 2^{+25} -1 or -180000 to 179999
Interval counter	16 bit
Address range	C0000h to FFFFFh, 1Kbyte (400h) address space occupied
Connectors	X1,X2: 9-pin sub D, sine wave 7 to 16 μA_{PP} X3: 4-pin Lemosa connector, external functions
Latch input signal Latch output signal	U_H: 3.15 to 30 V U_L: -3.0 to 0.9 V U_H: 4.0 to 32 V U_L: 0 to 1.0 V
Interrupts	IRQ3, IRQ4, IRQ5, IRQ7
Power requirements	approx. 1 W, max. 1.5 W, without measuring system
Software	
Driver software	Turbo Pascal, Microsoft "C" supplied on disk. Programming examples in Microsoft QBASIC included in user manual.
Demo. programs	Demonstrates the functions of the PC counter board. Displays, stores and prints position values.

Index