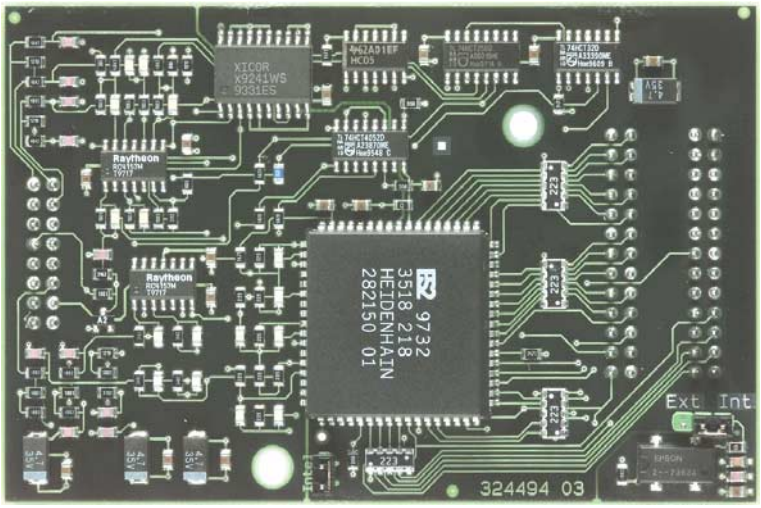


HEIDENHAIN

Benutzer-Handbuch

IK 410V

Zählerplatine mit
Bus-Schnittstelle für
inkrementale Meßsysteme 1 V_{SS}

11/97

Inhalt

Lieferumfang

- Zählerkarte IK 410 V
- Benutzer-Handbuch
- Programmierbeispiele

Hinweis zur EMV-Richtlinie 89/336/EWG

Die Bestimmungen der EMV-Richtlinie 89/336/EWG wurden mit dem COMPAQ-Rechner DESKPRO 386/20e geprüft.

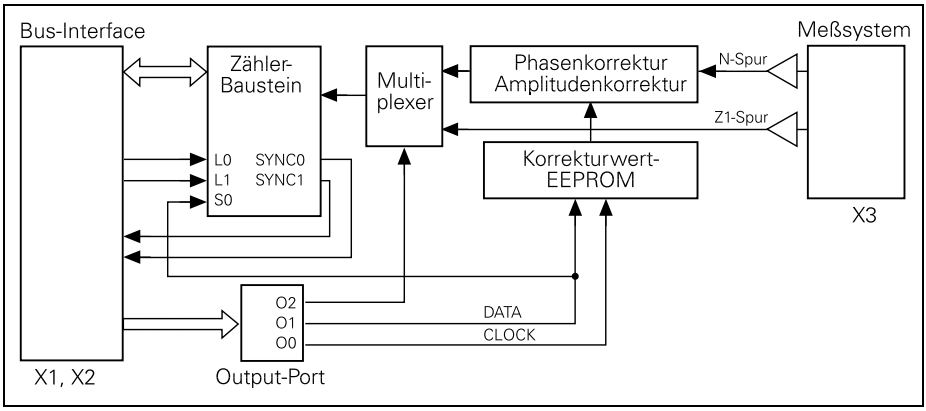
Wichtige Hinweise

Gefahr für interne Bauteile!
Die Vorsichtsmaßnahmen bei der Handhabung **elektrostatisch entladungsgefährdeter Bauelemente (ESD)** nach DIN EN 100 015 beachten. Als Transport-Verpackung nur antistatisches Material verwenden. Beim Einbau ausreichende Erdung des Arbeitsplatzes und der Person sicherstellen.

Technische Beschreibung der IK 410 V

Die **IK 410 V** ist eine Interpolations- und Zählerplatine zur Weg- und Winkelmessung mit einem HEIDENHAIN-Meßsystem mit sinusförmigen Spannungssignalen. Zusätzlich zum Eingang für die Inkrementalsignale ist ein Eingang für die Kommutierungsspur eines Motorgebers vorgesehen (ein Sinus/Cosinus pro Umdrehung). Somit eignet sie sich auch für die Motorregelung. Die **IK 410 V** läßt sich direkt auf die Platine einer Steuerung oder Folge-Elektronik stecken.
Sie kann in zwei Betriebsarten betrieben werden. Im Periodenzähler-Modus wird in einem 32-Bit-Register die Anzahl der Signalperioden gezählt. Im Interpolations-Modus wird zusätzlich das Meßsystemsignal 1 024fach unterteilt (10 Bit) und zu der Anzahl der Signalperioden (32 Bit) addiert. Die Meßwerte werden entweder über externe Abruf-Eingänge oder per Software oder Timer sowie durch das Überfahren der Referenzmarken über Port-Adressierung abgerufen und gespeichert.
Der Interpolationswert (10 Bit) bildet zusammen mit dem Wert des Periodenzählers (32 Bit) den 42 Bit breiten Meßwert. Die Meßwerte werden in 48 Bit breiten Daten-Registern gespeichert, wobei die oberen Bits entsprechend der Zweierkomplement-Darstellung vorzeichenrichtig erweitert werden. Der Meßwert kann direkt über den 16-Bit-Datenbus der Steuerung bzw. der Folge-Elektronik ausgelesen werden.

Blockschaltbild IK 410 V



Die Zählerkarte IK 410 V besitzt ein an den ISA-Bus angelehntes Interface (Stecker X1 und X2), d.h. 16-Bit-Datenbus, 6-Bit-Adreßbus, sowie die Signale –RD, –WR, –CS, –RESET und INTERRUPT.

Am Interface sind zusätzlich die Signale L0, L1, SYNC0 und SYNC1 herausgeführt. Mit den Signalen L0 und L1 kann man den Zählerwert in die Registerbänke 0 bzw. 1 einspeichern. Mit den Signalen SYNC0 und SYNC1 lassen sich mehrere Karten IK 410 V synchron betreiben.

Mit einem Output-Port kann man das Korrekturwert-EEPROM beschreiben.

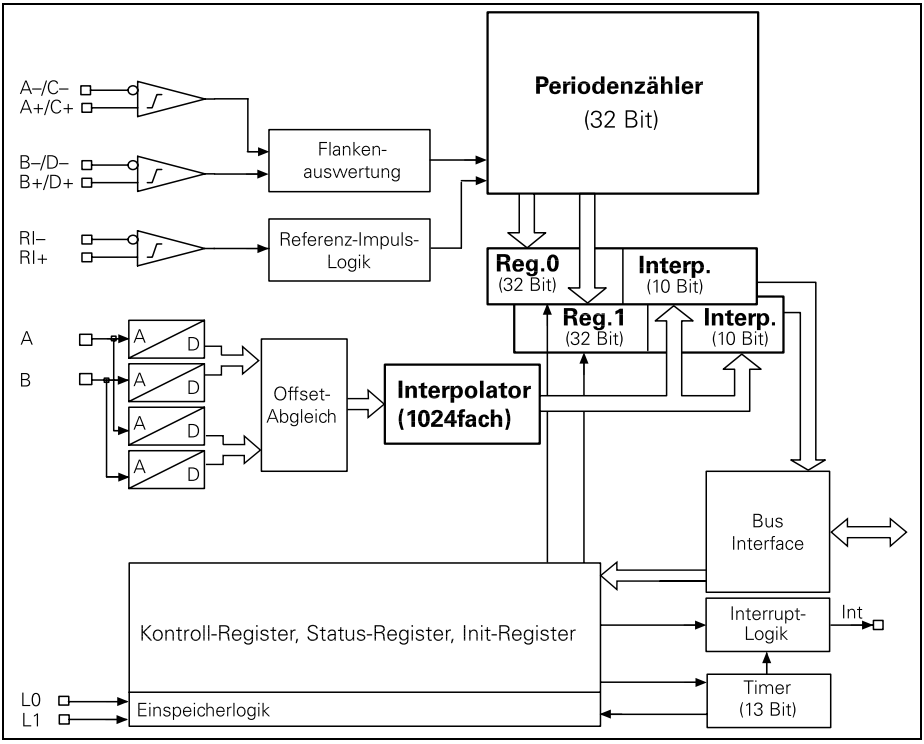
Der Offset der sinusförmigen Meßsystem-Signale läßt sich über Datenregister im Zählerbaustein per Software abgleichen; Phase und Amplitude über elektronische Potentiometer.

Außerdem ermöglicht das Qutput-Port die Umschaltung der 0°- und 90°-Signale von N- und Z1-Spur eines HEIDENHAIN-Meßsystems über einen Multiplexer.

Zugriffszeit auf Meßwerte

Die Zugriffszeit auf die Meßwerte beträgt typisch 30 µs (abhängig von der Taktfrequenz).

Blockschaltbild des Zählerbausteins



Beschreibung der Signale *)

D0 bis D15	Datenbus
A0 bis A5	Adreßbus
CLOCK	Eingang für Takt (Falls Brücke BR2 auf EXT)
–RD	Eingang Read Strobe
–WR	Eingang Write Strobe
–CS	Eingang Chip Select
–RESET	Eingang Reset
–INT	Ausgang Interrupt
–L0	Eingang Hardware-Latch für Register 0
–L1	Eingang Hardware-Latch für Register 1
–SYNC0	Ausgang für Synchronisation (L0 zugeordnet)
–SYNC1	Ausgang für Synchronisation (L1 zugeordnet)
A+	0°-Signal + (N-Spur)
A–	0°-Signal – (N-Spur)
B+	90°-Signal + (N-Spur)
B–	90°-Signal – (N-Spur)
RI+	Referenzimpuls +
RI–	Referenzimpuls –
C+	0°-Signal + (Z1-Spur)
C–	0°-Signal – (Z1-Spur)
D+	90°-Signal + (Z1-Spur)
D–	90°-Signal – (Z1-Spur)

*) – bedeutet invers

Steckerbelegung

Schnittstelle zur Folge-Elektronik

X1: 26poliger AMP-Stecker

Pin-Nummer	Signal
1a	Gnd
1b	frei
2a	Gnd
2b	frei
3a	–SYNC1
3b	frei
4a	–SYNC0
4b	frei
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

X2: 26poliger AMP-Stecker

Pin-Nummer	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	CLOCK
13b	Gnd

Meßsystem-Eingang

An die IK 410 V können Sie HEIDENHAIN-Längenmeßsysteme oder -Winkelmeßsysteme mit sinusförmigen Spannungssignalen A und B (N-Spur) und Referenzmarken-Signal R anschließen. Zusätzlich kann man bei Motorgebern die Kommutierungssignale C und D (Z1-Spur) auswerten.

Signalamplituden A, B, C, D (0°, 90°) R (Referenzmarke)	 0,6 V_{SS} bis 1,2 V_{SS} 0,2 V bis 0,85 V
Signalpegel für Fehlermeldung	≤ 0,22 V_{SS}
Maximale Eingangsfrequenz	350 kHz
Kabellänge[1)]	max. 60 m (Versorgungsspannung 5,0 V

1) Kabel bis 150 m sind möglich, falls durch eine externe Versorgung gewährleistet ist, daß 5 V am Meßsystem anliegen.
Die Eingangsfrequenz reduziert sich in diesem Fall auf max. 250 kHz.

X3: 16poliger AMP-Stecker

Pin-Nummer	**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

Abgleich der Meßsystem-Signale

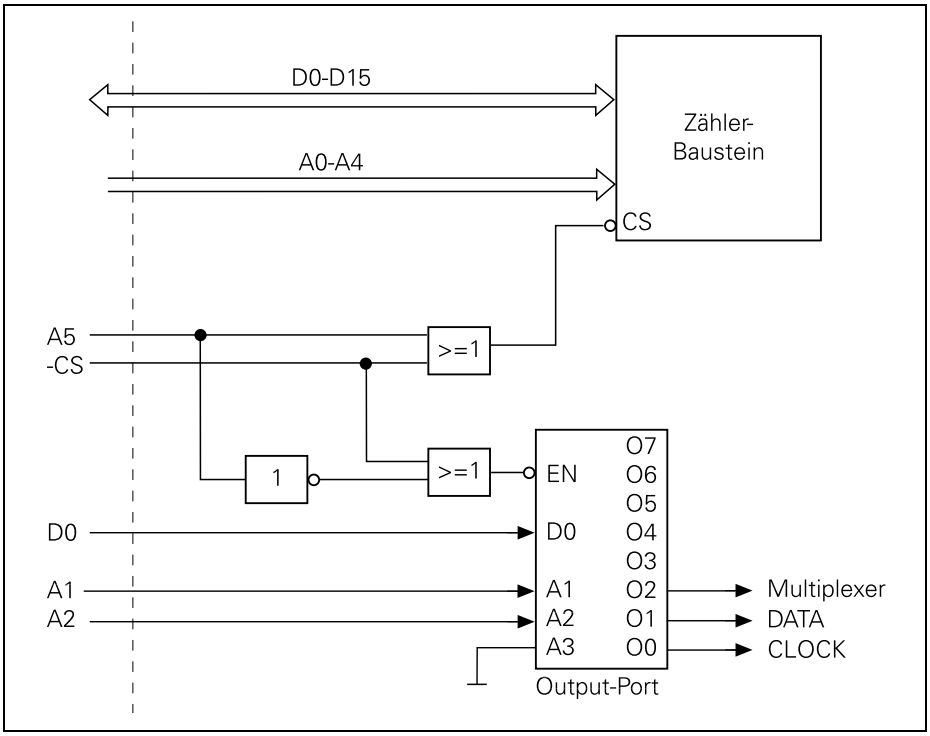
Meßsystem-Signale können wie folgt abgeglichen werden:

- Phase und Amplitude über elektronische Potentiometer
- Symmetrie (Offset) in den Zählerbausteinen mit Offset-Registern

Die Ansteuerung der Potentiometer erfolgt über I²C-Bus. Beispiele zur Erzeugung der Steuersequenzen finden Sie im Quellcode der Programme POTIS.CPP sowie POTIS.PAS und ADJUST.PAS.

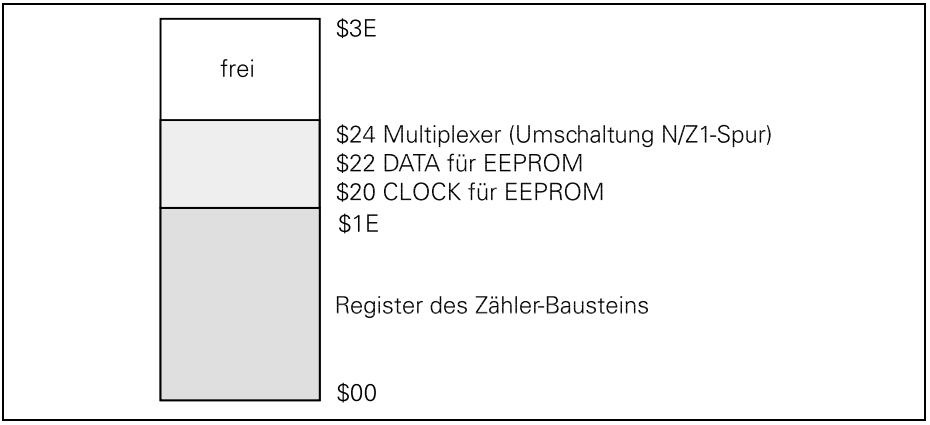
Die Korrekturwerte für Phase und Amplitude werden in den Bausteinen der elektronischen Potentiometer netzausfallsicher gespeichert. Die Offset-Register in den Zählerbausteinen sind nicht netzausfallsicher. Deshalb werden die Offset-Korrekturwerte in einem EEPROM gespeichert. Nach dem Einschalten müssen die Offset-Korrekturwerte vom EEPROM in die Offset-Register der Zählerbausteine geladen werden. In den Dateien IIC.CPP und IIC. PAS sind zwei Funktionen als Anwendungs-Beispiele enthalten. Die Funktion „Store Offset“ bzw. „store_offset" speichert die Offset-Korrekturwerte in das EEPROM. Die Funktion „Load Offset“ bzw. „load_offset" liest die Offset-Korrekturwerte aus dem EEPROM in die Offset-Register der Zählerbausteine.

Adreßraum und Datenformat



Das Zähler-Gatearray benötigt zum Lesen bzw. Schreiben seiner Register die Adressen A0 bis A4 (Adreßleitung A5=0), d.h. Adreßbereich: $0 bis $1E.

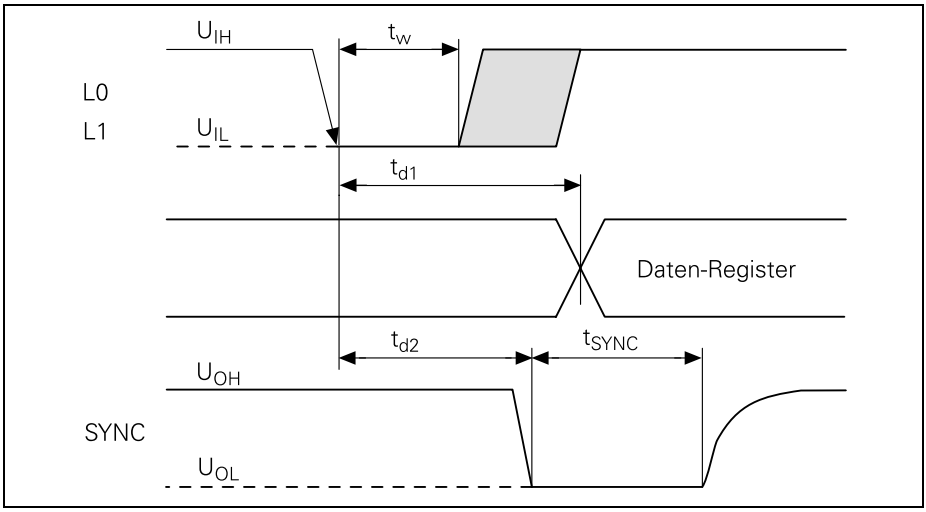
Der Output-Port wird mit den Adressen A1 und A2 und der Datenleitung D0 angesprochen (Adreßleitung A5=1), d.h. Adreßbereich: $20 bis $3E.



Datenformat
Sie können per Brücke BR2 wählen, ob Sie mit dem INTEL- oder MOTOROLA-Datenformat arbeiten (siehe Anschlußmaße).

Takt, Zeitablauf und Spannungspegel

Sie können per Brücke BR1 wählen, ob Sie die IK 410 V mit dem internen Takt (20 MHz) oder mit einem externen Takt (von 3 MHz) bis 20 MHz) versorgen (siehe Anschlußmaße).



Bezeichnung	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: Takt in (1/s)
für CLK > 16 MHz: t_{DIFF} = 38 *1/CLK
für CLK ≤ 6 MHz: t_{DIFF} = 32*1/CLK
Beispiel: CLK = 20 MHz t_{max} = 0,4 µs + 20,2 µs + 1,9 µs = 24,3 µs

Register

Für die folgende Beschreibung ist das Prinzip-Schaltbild der Zählerbausteine auf der letzten Seite hilfreich.

Wichtiger Hinweis zur Programmierung:
Der Zugriff auf die IK 410 V erfolgt durch Lesen von Datenworten, die in Registern abgelegt sind, und durch Schreiben von Datenworten in die Register. Deshalb dürfen nur gerade Port-Adressen mit Wort-Schreib- und Wort-Lesebefehlen adressiert werden.

Register-Übersicht

Register des Zählerbausteins

Adresse (Hex) A0 bis A5	Schreibzugriff	Lesezugriff
00		Daten-Register 0, LS-Word
02	ohne Funktion	Daten-Register 0
04		Daten-Register 0, MS-Word
06		Daten-Register 1, LS-Word
08	ohne Funktion	Daten-Register 1
0A		Daten-Register 1, MS-Word
0C Low Byte	Initialisierungs-Register 1	Initialisierungs-Register 1
High Byte	Initialisierungs-Register 2	Initialisierungs-Register 2
0E Low Byte	Kontroll-Register 1	Status-Register 1
High Byte	ohne Funktion	Status-Register 2
10 Low Byte	Referenzmarken-Register	ohne Funktion
High Byte	ohne Funktion	Amplitudenwert-Register
12 Low Byte	Freigabe-Register für Meßwert-Abruf	ohne Funktion
High Byte	Achs-Kaskadierung	ohne Funktion
14 Low Byte	Interrupt-Freigabe-Register	Interrupt-Status-Register 1
High Byte	ohne Funktion	Interrupt-Status-Register 2
16 Low Byte	Offset-Register für 0°-Signal	Amplitude für 0°-Signal
High Byte	ohne Funktion	Amplitude für 0°-Signal
18 Low Byte	Offset-Register für 90°-Signal	Amplitude für 90°-Signal
High Byte	ohne Funktion	Amplitude für 90°-Signal
1A Low Byte	Timer-Register, LS-Byte	ohne Funktion
High Byte	Timer-Register, MS-Byte	ohne Funktion
1C Low Byte	Kontroll-Register 2	Status-Register 3
High Byte	ohne Funktion	Kennungs-Register
1E Low Byte	Kontroll-Register 3	Status-Register 4
High Byte	ohne Funktion	ohne Funktion

Register des Output-Ports

Adresse (Hex) A0 bis A5	Schreibzugriff	Lesezugriff
20	I²C-Bus-Leitung SCL: Takt für EEPROM (Signale müssen invertiert programmiert werden)	–
22	I²C-Bus-Leitung SDA: Daten für EEPROM (Signale müssen invertiert programmiert werden)	–
24	Umschaltung des Multiplexers für die Meßsystem-Signale	–

Daten-Register für die Zähler

Die Meßwerte speichert die IK 410 V in 48 Bit breiten Registern. Pro Achse stehen zwei Daten-Register zur Verfügung: Daten-Register 0 (00h bis 04h) und Daten-Register 1 (06h bis 0Ah). Die Meßwerte setzt die IK 410 V aus dem 10-Bit-Interpolationswert und dem 32 Bit breiten Wert des Periodenzählers zusammen. Von den 48 Bit breiten Registern lassen sich also nur 42 Bit für den Meßwert nutzen. Die oberen 6 Bits erweitert die IK 410 V entsprechend der Zweier-Komplement-Darstellung vorzeichenrichtig.

Die Datenbreite von 48 Bit können Sie über das Initialisierungs-Register 1 (0Ch), Bit D6 auf 32 Bit verkürzen.

Über das Initialisierungs-Register 1 (0Ch), Bit D7 können Sie außerdem festlegen, ob der Meßwert nur aus dem Wert des Periodenzählers (Datenbits D0 bis D9 sind nicht definiert) oder aus dem Wert des Periodenzählers und aus dem Interpolationswert gebildet wird.
Das Speichern der Zählerwerte in die Daten-Register kann erfolgen über:

- Software-Abruf
- Einspeicher-Eingänge
- Timer
- Referenzmarken

Die Wirkungsweise der verschiedenen Abruf-Signale zeigt das Prinzip-Schaltbild der Zählerbausteine auf der letzten Seite.

Über das Status-Register 1 (0Eh), Bit D0 oder D1 können Sie abfragen, ob der Meßwert in den Daten-Registern gespeichert wurde. Solange Bit D0 oder D1 gesetzt sind, können Sie keinen weiteren Meßwert speichern, bis das oberste Wort des Meßwerts gelesen wurde. (Ausnahme: über Kontroll-Register 2, Bit D6 oder D7, wird das Abrufen ohne vorheriges Auslesen des Meßwerts freigegeben.) Im 48-Bit-Modus sind dies die Daten-Register 04h oder 0Ah, im 32-Bit-Modus die Daten-Register 02h oder 08h. Nach dem Lesen des Meßwerts wird in Status-Register 1 (0Eh), Bit D0 oder D1 wieder zurückgesetzt.

Wird der Zähler gestoppt oder durch Überfahren der Referenzmarke gespeichert, steht in D0 bis D9 der feste Wert 256.

0Ch: Initialisierungs-Register 1 (Schreibzugriff)

Bit	Funktion
D0	**Betrieb mit/ohne Interpolation** 0 = Betrieb als Periodenzähler (ohne Interpolation – Datenbits D0 bis D9 sind nicht definiert) 1 = Meßwert wird aus dem Wert des Perioden-zählers und aus dem Interpolationswert gebildet.
D1	0
D2	**Timer** 0 = Timer nullen und stoppen 1 = Timer starten
D3	0
D4	0
D5	0
D6	**Einspeicher-Freigabe** 0 = Betriebsart: 32-Bit-Register Lesen der Bits D24 bis D31 setzt das Status-Bit D0 bzw. D1 im Status-Register 1 (0Eh) zurück. 1 = Betriebsart: 48-Bit-Register Lesen der Bits D40 bis D47 setzt das Status-Bit D0 bzw. D1 im Status--Register 1 (0Eh) zurück.
D7	**Zählrichtung** Die Zählrichtung legt fest, ob die Zähler bei positiver Verfahrrichtung positiv (normal) oder negativ (invers) zählen. 0 = Zählrichtung normal 1 = Zählrichtung invers Die Zählrichtung darf nur im Periodenzähler-Betrieb invers sein! Im Betrieb mit Interpolation ergibt die invertierte Zählrichtung eine fehlerhafte Verknüpfung von Interpolationswert und Periodenzählerwert.

0Ch: Initialisierungs-Register 2 (Schreibzugriff)

Bit	Funktion
D8 D9	**Nur im Betrieb als Periodenzähler:** **Flankenauswertung** Durch die beiden inkrementalen Meßsystem-Signale (0°el. und 90° el.) stehen pro Signalperiode maximal vier Flanken zur Auswertung zur Verfügung. Die Zähler können so programmiert werden, daß sie entweder eine, zwei oder vier Flanken pro Signalperiode zählen. 00 = 1fach 01 = 2fach 11 = 4fach Im Betrieb mit Interpolationswert ist automatisch die 1fache Auswertung eingestellt.
D10	**Nur im Betrieb als Periodenzähler:** **Zählweise** 0 = Linear-Zählweise -2^{41} bis $+2^{41}-1$ 1 = Winkel-Zählweise wie in D11 festgelegt Für Winkelmeßsysteme mit 36 000 oder 360 000 Strichen pro Umdrehung.
D11	**Nur bei Winkel-Anzeige:** **Zählweise** 0 = 17 999 bis –18 000 1 = 179 999 bis –180 000
D12	0
D13	0
D14 D15	**Meßwert-Abruf mit Referenzimpuls** 0 = 1. Referenzmarke speichert in Daten-Register 0 1 = 1. Referenzmarke speichert in Daten-Register 1 0 = 2. Referenzmarke speichert in Daten-Register 0 1 = 2. Referenzmarke speichert in Daten-Register 1

0Ch: Lesezugriff: Bits D0 bis D15: Rücklesen der Initialisierungs-Register 1 und 2

0Eh: Kontroll-Register 1 (Schreibzugriff)

Bit	Funktion
D0	1 = Software-Abruf: Meßwert in Daten-Register 0
D1	1 = Software-Abruf: Meßwert in Daten-Register 1
D2	1 = Software-Abruf in alle Daten-Register (muß im Abruf-Freigabe-Register freigegeben werden)
D3	1 = Zähler starten
D4	1 = Zähler stoppen
D5	1 = Zähler löschen
D6	1 = Meßsystemfehler löschen (Frequenzüberschreitung)
D7	Amplitudenwert-Register löschen
D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

0Eh: Status-Register 1 (Lesezugriff)

Bit	Funktion
D0	Status für Software-Abruf in Register 0 1 = Meßwert ist bereit
D1	Status für Software-Abruf in Register 1 1 = Meßwert ist bereit
D2	ohne Funktion
D3	ohne Funktion
D4	1 = Zähler ist gestoppt
D5	I²C-Bus-Leitung SDA (Eingang)
D6	1 = Meßsystemfehler (Frequenzüberschreitung)
D7	ohne Funktion

0Eh: Status-Register 2 (Lesezugriff)

Bit	Funktion
D8	1 = Anfahren der Referenzmarke ist aktiv
D9 D10 D11 D12	ohne Funktion
D13	Logik-Pegel für das 0°-Signal
D14	Logik-Pegel für das 90°-Signal
D15	Logik-Pegel der Referenzmarke

10h: Referenzmarken-Register (Schreibzugriff)
HEIDENHAIN-Längen- und Winkelmeßsysteme können eine oder mehrere Referenzmarken haben. HEIDENHAIN empfiehlt insbesondere Meßsysteme mit abstandscodierten Referenzmarken.

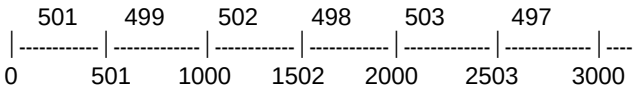
Bei einer Stromunterbrechung geht die Zuordnung zwischen der Position des Meßsystems und dem angezeigten Positionswert verloren. Mit den Referenzmarken der Meßsysteme kann man die Zuordnung nach dem Einschalten wieder herstellen.

Beim Überfahren der Referenzmarken wird ein Signal erzeugt, das diese Maßstabs-Position als Referenzpunkt kennzeichnet. Mit diesem Referenzpunkt lassen sich die Zuordnungen zwischen den Achspositionen und Anzeigewerten, die zuletzt festgelegt wurden, wieder herstellen. Bei Längenmeßsystemen mit abstandscodierten Referenzmarken braucht man dazu nur maximal um 20 mm zu verfahren.

Bit	Funktion beim Überfahren der Referenzmarke
D0	1 = Zähler starten
D1	1 = Zähler stoppen
D2	1 = Zähler löschen
D3	1 = Meßwert abrufen
D4	1 = Meßwert mit dem Überfahren der 2. Referenzmarke abrufen
D5	1 = Zähler mit dem Überfahren jeder Referenzmarke löschen
D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

Auswerten abstandscodierter Referenzmarken
Bei Meßsystemen mit abstandscodierten Referenzmarken befinden sich über den ganzen Meßweg Referenzmarken in einem festen Abstand. Zwischen zwei dieser Referenzmarken befindet sich eine dritte, deren Abstand zu den anderen beiden so variiert, daß jeder Abstand ein Vielfaches der Teilungsperiode beträgt und nur einmal über den ganzen Meßweg vorkommt. So können nach einer Stromunterbrechung nur durch Überfahren von zwei Referenzmarken die Zuordnungen zwischen Achspositionen und Anzeigewerten wieder her-gestellt werden.

Die Auswertung dieser abstandscodierten Referenzmarken erläutert das folgende Beispiel:
Bei einem Grundabstand von 1 000 Signalperioden ergibt sich die folgende Verteilung in Inkrementen für die Referenzmarken:



Zunächst müssen Sie das Referenzmarken-Register 10h wie folgt initialisieren:

- Zähler mit dem Überfahren der 1. Referenzmarke löschen und starten (Bit D0 = 1 und D2 =1).
- Zähler mit dem Überfahren der 2. Referenzmarke abrufen (Bit D4 = 1).

Zum Berechnen der absoluten Position wird nur der Abstand zwischen den Referenzmarken in Signalperioden benötigt. Diesen Abstand in Inkrementen, in der folgenden Beschreibung mit DIFF bezeichnet, müssen Sie durch 1 024 teilen.

DIFF = DISTANCE_IN_INCR : 1 024

Zum Berechnen der absoluten Position muß man den Abstand (OFFSET) zwischen der 1. Referenzmarke auf dem Maßstab (Position „0“ auf der Zeichnung) und der 1. überfahrenen Referenzmarke ermitteln.

Es gibt vier mögliche Fälle:

1. Positive Verfahrrichtung und DIFF > 500
   OFFSET = (DIFF – 501) • 1 000
2. Positive Verfahrrichtung und DIFF < 500
   OFFSET = (500 – DIFF) • 1 000 – DIFF
3. Negative Verfahrrichtung und $|DIFF| > 500$
   OFFSET = (DIFF – 501) • 1 000 + DIFF
4. Negative Verfahrrichtung und $|DIFF| < 500$
   OFFSET = (500 – DIFF) • 1 000

Die absolute Position in Inkrementen berechnet sich wie folgt:

ABS_POS_INCR = ACT_POS + PRESET + DISTANCE

ACT_POS: Abstand zwischen der aktuellen Position und der 1. überfahrenen Referenzmarke (in Inkrementen).

PRESET: Position, die beim Bezugspunkt-Setzen auf die 1. Referenzmarke des Maßstabs (Position „0“ auf der Zeichnung) gesetzt wird (in Inkrementen).

DISTANCE: Abstand zwischen der 1. Referenzmarke auf dem Maßstab und der 1. überfahrenen Referenzmarke (in Inkrementen).

DISTANCE = OFFSET • 1 024

Die absolute Position wird – z.B. bei einem Maßstab mit Signalperiode von 0,02 mm – wie folgt berechnet:

$$ABS\_POS\_MM = \frac{ABS\_POS\_INCR \cdot 0{,}02}{1024}$$

Bei Winkelmeßsystemen:

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

Ein Anwendungsbeispiel in „TURBO PASCAL“ zum Auswerten von Referenzmarken finden Sie im Quellcode der Datei CNT_2.PAS unter (*Distance-coded ref.marks*).

10h: Amplitudenwert-Register (Lesezugriff)

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6 D7	ohne Funktion
D8 D9	**Aktuelle Amplitude** Mit jedem Meßwert-Abruf wird ein neuer Amplitudenwert ermittelt. D9 D8 0 0 normale Amplitude 0,47 V_{SS} < U_e < 1,41 V_{SS} 0 1 kleine Amplitude 0,23 V_{SS} < U_e < 0,47 V_{SS} 1 0 hohe Amplitude U_e > 1,41 V_{SS} 1 1 fehlerhaft kleine Amplitude U_e < 0,23 V_{SS} Der Amplitudenwert sollte vor dem Lesen durch Bit D4 im Kontroll-Register 2 eingefroren werden. Das Amplitudenwert-Register wird durch Bit D7 im Kontroll-Register 1 zurückgesetzt.
D10 D11	**Minimalwert der Amplitude** Kodierung und Lesezugriff siehe Bits D8 und D9. Wenn der aktuelle Amplitudenwert mehr als viermal hintereinander den gespeicherten Minimalwert unterschreitet, ersetzt die IK den alten Minimalwert durch den aktuellen Amplitudenwert.
D12 D13 D14 D15	ohne Funktion

12h: Freigabe-Register für Meßwert-Abruf (Schreibzugriff)

Bit	Funktion
D0	1 = Freigabe L0 für Daten-Register 0
D1	1 = Freigabe L0 über Verzögerungsglied (125 ns) für Daten-Register 0
D2	1 = Freigabe des „Software-Abrufs in alle Daten-Register" für Daten-Register 0
D3	1 = Freigabe des „Software-Abrufs über Timer" für Daten-Register 0
D4	1 = Freigabe L1 für Daten-Register 1
D5	1 = Freigabe L1 über Verzögerungsglied (125 ns) für Daten-Register 1
D6	1 = Freigabe des „Software-Abrufs in alle Daten-Register" für Daten-Register 1
D7	1 = Freigabe des „Software-Abrufs über Timer" für Daten-Register 1

Das Prinzip-Schaltbild der Zählerbausteine auf der Seite 46 verdeutlicht die Funktion der einzelnen Bits.

12h: Register für Achs-Kaskadierung (Schreibzugriff)

Bit	Funktion
D8	1 = Freigabe von Pin L 0 zu SYNC 1
D9	1 = Freigabe des Software-Abrufs zu SYNC 1
D10	1 = Freigabe des Timerstrobes zu SYNC 1
D11	0
D12	1 = Freigabe von Pin L1 zu SYNC 0
D13	1 = Freigabe des Software-Abrufs zu SYNC 0
D14	1 = Freigabe des Timerstrobes zu SYNC 0
D15	0

12h: Lesezugriff ohne Funktion

14h: Interrupt-Freigabe-Register (Schreibzugriff)

Die Interrupt-Logik programmieren Sie über das Interrupt-Freigabe-Register. Die Interrupt-Ursache kann der Meßwert-Abruf in Register 0, Register 1 oder der Timer-Strobe sein. Alle drei Interrupt-Quellen können Sie unabhängig voneinander programmieren. Bei gleichzeitigem Eintreffen mehrerer Interrupts besteht folgende Priorität:

- höchste Priorität: Meßwert-Abruf über Register 0
- zweithöchste Priorität: Meßwert-Abruf über Register 1
- niedrigste Priorität: Meßwert-Abruf über Timer-Strobe

Nach einem Interrupt kann kein weiterer Interrupt erfolgen, bis Sie das Interrupt-Status-Register 2 gelesen haben.

Bit	Funktion
D0	1 = Freigabe des Interrupts 0 beim Meßwert-Abruf über Register 0
D1	1 = Freigabe des Interrupts 1 beim Meßwert-Abruf über Register 1
D2	1 = Freigabe des Interrupts 2 für Timer-Strobe
D3	1 = Interrupt wird erzeugt (D0 = D1 = D2 = 0)
D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

Das Prinzip-Schaltbild der Zählerbausteine auf der Seite 60 verdeutlicht die Funktion der einzelnen Bits.

14h: Interrupt-Status-Register 1 (Lesezugriff)

In diesem Register wird der gerade aktive Interrupt angezeigt (es kann immer nur 1 Bit gesetzt sein). Durch Lesen dieses Registers setzt die IK 410 V den gerade aktiven Interrupt zurück und löscht das zugehörige Status-Bit, so daß Sie den nächsten Interrupt durch eine negative Flanke auslösen können.

Bit	Funktion
D0	1 = Interrupt 0 ist aktiv, es erfolgte ein Meßwert-Abruf über Daten-Register 0
D1	1 = Interrupt 1 ist aktiv, es erfolgte ein Meßwert-Abruf über Daten-Register 1
D2	1 = Interrupt 2 ist aktiv, es erfolgte ein Meßwert-Abruf über den Timer-Strobe
D3 D4 D5 D6 D7	ohne Funktion

14h: Interrupt-Status-Register 2 (Lesezugriff)

In diesem Register zeigt die IK 410 V alle Interrupts an, die anstehen, d.h. der aktive Interrupt und die Interrupts, die noch ausgeführt werden müssen. Es können also mehrere Bits gleichzeitig gesetzt sein.

Bit	Funktion
D8	1 = Interrupt 0 steht an, wird aber noch nicht ausgeführt
D9	1 = Interrupt 1 steht an, wird aber noch nicht ausgeführt
D10	1 = Interrupt 2 steht an, wird aber noch nicht ausgeführt
D11 D12 D13 D14 D15	ohne Funktion

16h: Offset-Register für 0°-Signal (Schreibzugriff)
Dieses Register enthält den 7-Bit-Offset-Korrekturwert für das 0°-Signal in Zweier-Komplement-Darstellung. Daraus folgt eine maximale Korrektur von ± 63.
Die Korrekturwerte können nur geschrieben werden, falls eines der Status-Bits D5 oder D6 im Status-Register 3 den Wert 0 hat.

Funktionsweise:
Zu den digitalen Werten des 0°-Signals (0 bis 1023) und 90°-Signals addiert die IK 410 V die Offset-Korrekturwerte. Bei einem Überlauf wird auf 1023 oder 0 begrenzt.

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6	Offset-Korrekturwert für das 0°-Signal in Zweier-Komplement-Darstellung
D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

Das Offset-Register in den Zählerbausteinen ist **nicht** netzausfallsicher. Deshalb werden die Offset-Korrekturwerte in einem EEPROM gespeichert. Nach dem Einschalten müssen die Offset-Korrekturwerte vom EEPROM in die Offset-Register der Zählerbausteine geladen werden.

16h: Amplitude für das 0°-Signal (Lesezugriff)
Bei jedem Analog-Digital-Wandelvorgang speichert die IK 410 V das Ergebnis der Wandlung. Vor dem Auslesen müssen Sie durch Bit D4 im Kontroll-Register 2 die Werte einfrieren.

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9	Amplitude für das 0°-Signal
D10 D11 D12 D13 D14 D15	ohne Funktion

18h: Offset-Register für das 90°-Signal (Schreibzugriff)
Dieses Register enthält den 7-Bit-Offset-Korrekturwert für das 90°-Signal.
Die Beschreibung der Funktionsweise siehe „16h: Offset-Register für 0°-Signal"

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6	Offset-Korrekturwert für das 90°-Signal in Zweier-Komplement-Darstellung
D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

18h: Amplitude für das 90°-Signal (Lesezugriff)
Bei jedem Analog-Digital-Wandelvorgang speichert die IK 410 V das Ergebnis der Wandlung. Vor dem Auslesen müssen Sie durch Bit D4 im Kontroll-Register 2 die Werte einfrieren.

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9	Amplitude für das 90°-Signal
D10 D11 D12 D13 D14 D15	ohne Funktion

1Ah: Timer-Register (Schreibzugriff)
In den Registern 1Ah und 1Bh legen Sie den 13 Bit breiten Timerwert ab. Im Timer-Register können Sie also Werte von 0 bis 8191 speichern. Die Zykluszeit wird in µs angegeben, wobei Sie von der gewünschten Zykluszeit 1 µs abziehen müssen.

Beispiel:
Gewünschte Zykluszeit = 1 ms = 1 000 µs
Zu programmierender Wert = 1 000 – 1 = 999
Mit dem Beschreiben dieses Registers ist der Timer noch nicht gestartet. Dies erfolgt durch Setzen von Bit D2 im Initialisierungs-Register 1 (0Ch). Außerdem müssen Sie die entsprechenden Bits in den Freigabe-Registern 12h, 13h oder 14h setzen.

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12	Timerwert
D13 D14 D15	ohne Funktion

1Ah: Lesezugriff ohne Funktion

1Ch: Kontroll-Register 2 (Schreibzugriff)

Bit	Funktion
D0 D1 D2 D3	Festen Wert ≥ 8 programmieren
D4	1 = Amplitude für 0°-Signal (16h) und 90°-Signal (18h) sowie Amplitudenwert-Register (10h High Byte) einfrieren. Um eine Änderung der Registerwerte während des Auslesens zu vermeiden, müssen Sie dieses Bit setzen. Ob die Registerwerte eingefroren sind, können Sie über Bit D4 im Status-Register 3 abfragen.
D5	0
D6	1 = Erneuter Abruf über Daten-Register 0 möglich, ohne den Meßwert vorher abzuholen. In dieser Betriebsart ist eine Änderung des Meßwerts während des Lesens möglich.
D7	1 = Erneuter Abruf über Daten-Register 1 möglich, ohne den Meßwert vorher abzuholen. In dieser Betriebsart ist eine Änderung des Meßwerts während des Lesens möglich.
D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

1Ch: Status-Register 3 (Lesezugriff)

Bit	Funktion
D0 D1 D2 D3	nicht lesbar
D4	1 = Amplitude für 0°-Signal (16h) und 90°-Signal (18h) sowie Amplitudenwert-Register (10h High Byte) sind eingefroren und können gelesen werden.
D5	0 = Das Offset-Register für das 0°-Signal ist geschrieben.
D6	0 = Das Offset-Registers für das 90°-Signal ist geschrieben.
D7	ohne Funktion

1Ch: Kennungs-Register (Lesezugriff)

Bit	Funktion
D8	
D9	
D10	
D11	Bausteinkennung: fester Wert 8
D12	
D13	
D14	
D15	

1Eh: Kontroll-Register 3 (Schreibzugriff)

Bit	Funktion
D0	fester Wert 0
D1	
D2	ohne Funktion
D3	
D4	
D5	
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	
D14	
D15	

1Eh: Status-Register 4 (Lesezugriff)

Bit	Funktion
D0	ohne Funktion
D1	logischer Pegel am Pin L0
D2	logischer Pegel am Pin L1
D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

Registerbeschreibung des Output-Ports

20h: I²C-Bus-Leitung SCL:
Takt für EEPROM (Schreibzugriff)

Bit	Funktion
D0	0 = High-Pegel am CLOCK-Eingang des EEPROM 1 = Low-Pegel am CLOCK-Eingang des EEPROM

22h: I²C-Bus-Leitung SDA:
DATEN für EEPROM (Schreibzugriff)

Bit	Funktion
D0	0 = High-Pegel am DATEN-Eingang des EEPROM 1 = Low-Pegel am DATEN-Eingang des EEPROM

24h: Umschaltung des Multiplexers für die Meßsystem-Signale (Schreibzugriff)

Bit	Funktion
D0	0 = N-Spur 1 = Z1-Spur

Programmierung

Im Lieferumfang der IK 410 V finden Sie eine Diskette mit Programmbeispielen. Auf dieser Diskette befinden sich drei Verzeichnisse:

BC
Dieses Verzeichnis enthält einfache Beispiele in „BORLAND C“. Als Hardware für diese Beispiele wurde eine IBM-PC - kompatible AT-Bus-Platine mit einer IK 410 V bestückt.

BCPP
Dieses Verzeichnis enthält Beispiele mit einem RAM-Speichermodell der Register in „BORLAND C++“. Als Hardware für diese Beispiele wurde ein IMB-PC-komaptible AT-Bus-Platine mit zwei IK 410 V bestückt. Das Programm ist so aufgebaut, daß bis zu 16 Platinen mit je zwei IK 410 V angesteuert werden können.

TP
Dieses Verzeichnis enthält Beispiele mit einem RAM-Speichermodell der Register in „TURBO PASCAL“. Als Hardware für diese Beispiele wurde eine IBM-PC-kompatible AT-Bus-Platine mit zwei IK 410 V bestückt.

Einfache Beispiele in „BORLAND C“

Grundfunktionen zum Schreibenund Lesen der Register
Die im folgenden beschriebenen Funktionen finden Sie auf der mitgelieferten Diskette unter dem Verzeichnis **BC** in der Datei **IK410_0.C** und der dazugehörenden Header-Datei **IK_0.H**.
Die beiden Funktionen **write_g26**[1)] und **read_g26**[1)] schreiben und lesen die Daten-Register; sie sind die Grundfunktionen für das Arbeiten mit der IK 410 V.

1) g26 ist die HEIDENHAIN-Bezeichnung des Zählerbausteins

Funktion zum Schreiben in die Register
Die folgende Funktion schreibt einen Wert in ein 16 Bit breites Register eines Zählerbausteins.
Diese Funktion ist an die für diese Beispiele verwendete AT-Bus-Platine angepaßt. Für Ihre Hardware müssen Sie eine eigene Funktion schreiben, die auf die Register der IK 410 V zugreift.

Funktion: **write_g26**
Parameter
baseadr: Grundadresse
address: Adresse des Registers des Zählerbausteins
datum: Wert, der auf die Adresse geschrieben werden soll

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

Funktion zum Lesen der Register
Die folgende Funktion liest einen Wert von einem 16 Bit breiten Register eines Zählerbausteins. Diese Funktion ist an die für diese Beispiele verwendete AT-Bus-Platine angepaßt. Für Ihre Hardware müssen Sie eine eigene Funktion schreiben, die auf die Register der IK 410 V zugreift.

Funktion: **read_g26**[1)]
Parameter
baseadr: Grundadresse
address: Adresse des Registers des Zählerbausteins
datum: Wert aus dem unter „address“ angegebenen Daten-Register als 16-Bit-Variabel

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

Einfache Funktionen für den Meßwert-Abruf über Software

Funktion zum Speichern eines Meßwerts
die folgenden Funktionen speichern den Zählerstand der gewünschten Achse im Daten-Register 0 (soft_l0) oder Daten-Register 1 (soft_l1).

Funktion: **soft_l0**
Parameter
baseadr: Grundadresse

```
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);
}
```

Funktion zum Prüfen, ob der Meßwert gespeichert wurde
Die folgende Funktion prüft, ob ein Meßwert gespeichert wurde.

Funktion:	**latched**
Parameter	
baseadr:	Grundadresse
reg:	0 = Daten-Register 0, 1 = Daten-Register 1
Ergebnis:	false = Es wurde kein Meßwert gespeichert true = Es wurde ein Meßwert gespeichert

```
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);
}
```

Funktion zum wiederholten Prüfen, ob der Meßwert gespeichert wurde
Die folgende Funktion wiederholt solange die Abfrage nach einem Meßwert, bis ein Meßwert gespeichert wurde.

Funktion: **poll_latch**
Parameter
baseadr: Grundadresse
reg: 0 = Daten-Register 0, 1 = Daten-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;
       }
}
```

Funktion zum Lesen eines 32-Bit-Meßwerts
Die folgende Funktion liest einen 32 Bit breiten Meßwert von einem Zählerbaustein.

Funktion: **read_count_value32**
Parameter
baseadr: Grundadresse
reg: 0 = Daten-Register 0, 1 = Daten-Register 1
Ergebnis: Integer-Variable des Typs **long**

```
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);
}
```

Funktion zum Lesen eines 48-Bit-Meßwerts
Die folgende Funktion liest einen 48 Bit breiten Meßwert von einem Zählerbaustein.

Funktion: **read_count_value48**
Parameter
baseadr: Grundadresse
reg: 0 = Daten-Register 0, 1 = Daten-Register 1
Ergebnis: Floating-Point-Variable des Typs **double**

```
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);
}
```

Einfaches Programm für den Meßwert-Abruf über Software

In den folgenden Programm-Beispielen werden die vorher definierten Funktionen eingesetzt. Diese sind in den Dateien IK410_0.H und IK410_0.C deklariert und definiert. Die Beispiele finden Sie auf der mitgelieferten Diskette unter dem Verzeichnis **BC** in der Datei **SAMPLE32.C** und **SAMPLE48.C**.
Der Zählerstand wird in diesen Beispielen in Millimetern am Bildschirm angezeigt. Selbstverständlich kann die IK 410 V auch Winkelgrade anzeigen. Die Umrechnung zeigen die beiden folgenden Formeln.

Zählerstand in Millimeter umrechnen

$$\text{Wert [mm]} = \text{Zählerstand} \bullet \frac{\text{Teilungsperiode [mm]}}{1024}$$

Beispiel: Teilungsperiode = 20 µm

$$\text{Wert [mm]} = \text{Zählerstand} \bullet \frac{0{,}020\ [\text{mm}]}{1024}$$

Zählerstand in Grad umrechnen

$$\text{Wert}\ [°] = \frac{\text{Zählerstand} \bullet 360\ [°]}{1024 \bullet \text{Striche/Umdr.}}$$

Beispiel: Striche/Umdr. = 36 000

$$\text{Wert}\ [°] = \frac{\text{Zählerstand} \bullet 360\ [°]}{1024 \bullet 36\,000}$$

```
/*----------------------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, 0x0028);
   /* 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, 0x0028);
   /* 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);
}
```

Anwendungsbeispiele mit einem RAM-Speichermodell in „BORLAND C++“

Beispiele mit einem RAM-Speichermodell in „BORLAND C++“ befinden sich im Verzeichnis BCPP. Die verwendeten Datenstrukturen und Funktionen sind in folgenden Dateien deklariert und definiert:

IK410.H:	In dieser Header-Datei können Sie die Adresse von bis zu 16 Platinen mit zwei IK 410 V festlegen.
IK410_1.H:	Datenstrukturen für ein RAM-Speichermodell der Register der IK 410 V und Funktionsdeklarationen für die Funktionen in den Dateien IK410_1.CPP, IIC.CPP und POTI_1.CPP
IK410_1.CPP:	Basis-Funktionen für die IK 410 V
IIC.CPP:	Funktionen zum Datentransfer über den I²C-Bus.
POTI_1.CPP:	Funktionen zum Einstellen der elektronischen Potentiometer.

In der Datei IK410_1.H wird mit Hilfe von Datenstrukturen ein RAM-Speichermodell der Register der IK 410 V aufgebaut. Die Daten des RAM-Speichermodells werden mit Hilfe der Prozeduren **InitHandler** und **CommHandler** in die Register der IK 410 geschrieben.

DISPLAY.CPP

Das Programm DISPLAY.CPP zeigt die Inhalte aller Register der IK 410 V.

Quellcode: DISPLAY.CPP, IK410_1.CPP, IIC.CPP
Header-Dateien: IK410.H, IK410_1.H

POTIS.CPP

Das Programm POTIS.CPP zeigt, wie Sie per Software die elektronischen Potentiometer der IK 410 V über den I²C-Bus einstellen können.

Quellcode: POTIS.CPP, IK410_1.CPP
IIC.CPP, POTI_1.CPP
Header-Dateien: IK410.H, IK410_1.H

Anwendungsbeispiele mit einem RAM-Speichermodell in „TURBO PASCAL“

Beispiele mit einem RAM-Speichermodell in „TURBO PASCAL“ befinden sich im Verzeichnis **TP**. Die verwendeten Datenstrukturen und Funktionen sind in folgenden Dateien deklariert und definiert:

IK410_0.TPU	Grundfunktionen
IK410_1.TPU	Datenstrukturen und Funktionen für ein RAM-Speichermudell der Register der IK 410 V.
IK410_2.TPU	Funktionen für ADJUST.PAS, POTIS.PAS und SCOPE.PAS
IIC.TPU	Funktionen zum Datentransfer über I^2C-Bus
SCOPE_0.TPU	Funktionen für SCOPE.PAS
POTI_0.TPU	Funktionen für POTIS.PAS
ADJ_0.TPU	Funktionen für ADJUST.PAS

Diese Dateien werden als **units** mit Hilfe des Befehls **uses** in weitere Anwendungsprogramme eingebunden. Auf der mitgelieferten Diskette finden Sie diese Dateien mit dem Nachnamen *.PAS. Die Dateien *.PAS müssen zu u**nits** *.TPU kompiliert werden.

In der Datei IK410_1.TPU wird mit Hilfe von Datenstrukturen ein RAM-Speichermodell der Register der IK 410 V aufgebaut. Die Daten des RAM-Speichermodells werden mit Hilfe der Prozeduren **init_handler** und **comm_handler** in die Adressen der IK 410 V geschrieben.
Die Programme benötigen einen BORLAND-Grafik-Treiber (BOLAND Graphics Interface = *.BGI). Im Lieferumfang der IK 410 V ist der Grafik-Treiber EGAVGA.BGI enthalten. Dieser muß in dem gleichen Verzeichnis wie die Beispielprogramme gespeichert sein.

SAMPLE32.PAS

Das Beispiel SAMPLE32.PAS ist ein einfaches Anwendungsprogramm, das den Meßwertabruf über Software zeigt.

Quellcode: SAMPLE32.PAS
Units: IK410_0.TPU Grundfunktionen

SCOPE.PAS

Das Beispiel SCOPE.PAS zeigt die sinusförmigen Signale der angeschlossenen Meßsysteme entweder als Amplituden-Zeit-Diagramm oder in XY-Darstellung. Über Tastenbefehle, die am Bildschirm erklärt sind, können die Potentiometer eingestellt werden.

Quellcode:	SCOPE.PAS	
Units:	IK410_0.TPU	Grundfunktionen
	IK410_1.TPU	Funktionen für ein RAM-Speichermodell
	IK410_2.TPU	Funktionen für ADJUST.PAS, POTIS.PAS und SCOPE.PAS
	IIC.TPU	Funktionen zum Datentransfer über I^2C-Bus
	SCOPE_0.TPU	Funktionen für SCOPE.PAS
	CNT_0.TPU	Fenster-Funktionen
	LOGO.TPU	Einstiegsbild nach Programm-Start

POTIS.PAS

Das Beispiel POTIS.PAS zeigt die Stellung der elektronischen Potentiometer für Phasenlage und Amplitude sowie die Werte der Offset-Register an. Über Tastenbefehle, die am Bildschirm erklärt sind, können die Potentiometer eingestellt werden.

Quellcode:	POTIS.PAS	
Units:	IK410_0.TPU	Grundfunktionen
	IK410_1.TPU	Funktionen für ein RAM-Speichermodell
	IK410_2.TPU	Funktionen für ADJUST.PAS
	IIC.TPU	Funktionen zum Datentransfer über I^2C-Bus
	POTI_0.TPU	Funktionen für POTIS.PAS
	CNT_0.TPU	Fenster-Funktionen
	LOGO.TPU	Einstiegsbild nach Programm-Start

ADJUST.PAS

Das Beispiel ADJUST.PAS führt einen automatischen Abgleich der Achse 1 (Anwahl:1) oder Achse 2 (Anwahl:2) für Phasenlage (Anwahl:p), Amplitude (Anwahl:a) und Offset (Anwahl:o) der sinusförmigen Meßsystem-Signale durch. Die Kompensationswerte werden durch langsames Bewegen des Meßsystems gebildet. Nach 30 Signalperioden wird durch einen Ton angezeigt, daß ein Kompensationswert gebildet wurde und durch Drücken der Taste S gespeichert werden kann.

Quellcode:	ADJUST.PAS	
Units:	IK410_0.TPU	Grundfunktionen
	IK410_1.TPU	Funktionen für ein RAM-Speichermodell
	IK410_2.TPU	Funktionen für ADJUST.PAS, POTIS.PAS und SCOPE.PAS
	ADJ_0.TPU	Funktionen für ADJUST.PAS
	CNT_0.TPU	Fenster-Funktionen
	LOGO.TPU	Einstiegsbild nach Programm-Start

IK410.PAS

Das Programm IK410.PAS ist ein Anwendungsbeispiel für ein Positionsanzeigen-Programm in „TURBO PASCAL".

Quellcode:	IK410.PAS	
Units:	IK410_0.TPU	Grundfunktionen
	IK410_1.TPU	Funktionen für ein RAM-Speichermodell
	IK410_2.TPU	Funktionen für ADJUST.PAS, POTIS.PAS und SCOPE.PAS
	CNT_0.TPU	Fenster-Funktionen
	CNT_1.TPU	Positionsanzeige
	CNT_2.TPU	Zähler-Programm
	SCOPE_0.TPU	Funktionen für SCOPE.PAS
	ADJ_0.TPU	Funktionen für ADJUST.PAS
	LOGO.TPU	Einstiegsbild nach Programm-Start
Include-	IK410.WIN	Fensterdefinitionen
Dateien:	IK410.CNT	Initialisierungs-Datei
Hilfetexte:	IK410.HLP	Datei mit Texten zur Hilfestellung

Technische Daten

Mechanische Kennwerte	
Abmessungen	100 mm x 65 mm
Arbeitstemperatur	0°C bis 55°C
Lagertemperatur	–30°C bis 70°C

Elektrische Kennwerte

Meßsystem-Eingang
X3: 16polige AMP-Stiftleiste
– Eingang für inkrementale Längen- oder Winkelmeßsysteme mit sinusförmigen Spannungssignalen (1 V_{SS})
– zusätzlicher Eingang für die Kommutierungsspur eines Motorgebers (ein Sinus/Cosinus pro Umdrehung)
Maximale Eingangsfrequenz: 350 kHz bei $f_{CLK\ (min)}$ 3 MHz
Kabellänge: maximal 60 m (Versorgungsspannung 5,0 V)
Kabel bis 150 m sind möglich, falls durch eine externe Versorgung gewährleistet ist, daß 5 V am Meßsystem anliegen. Die Eingangsfrequenz reduziert sich in diesem Fall auf max. 250 kHz.

Schnittstelle zur Folge-Elektronik
X1 und X2:26polige AMP-Stiftleiste
16-Bit-Mikrocomputer-Schnittstelle entsprechend eines statischen RAMs

Steuerbus:	–RD, –WR, –CS, –Res, Clk (max. 20 MHz)
Datenbus:	D0 bis D15
Adreßbus:	A0 bis A5
Versorgung:	Gnd und +5 V / ±15 V (±5%) ca. 20 mA (ohne Meßsystem)
Einspeicher-Eingänge:	–L0, –L1
Einspeicher-Synchronisation:	–SYNC0, –SYNC1 (zum gleichzeitigen Einspeichern über mehrere Karten)
Interruptausgang:	–INT

Signal-Unterteilung	1 024fach (10 Bit)

Abgleich der Meßsystemsignale
Offset über Register in den Zählerbausteinen
Phase und Amplitude über elektronische Potentiometer

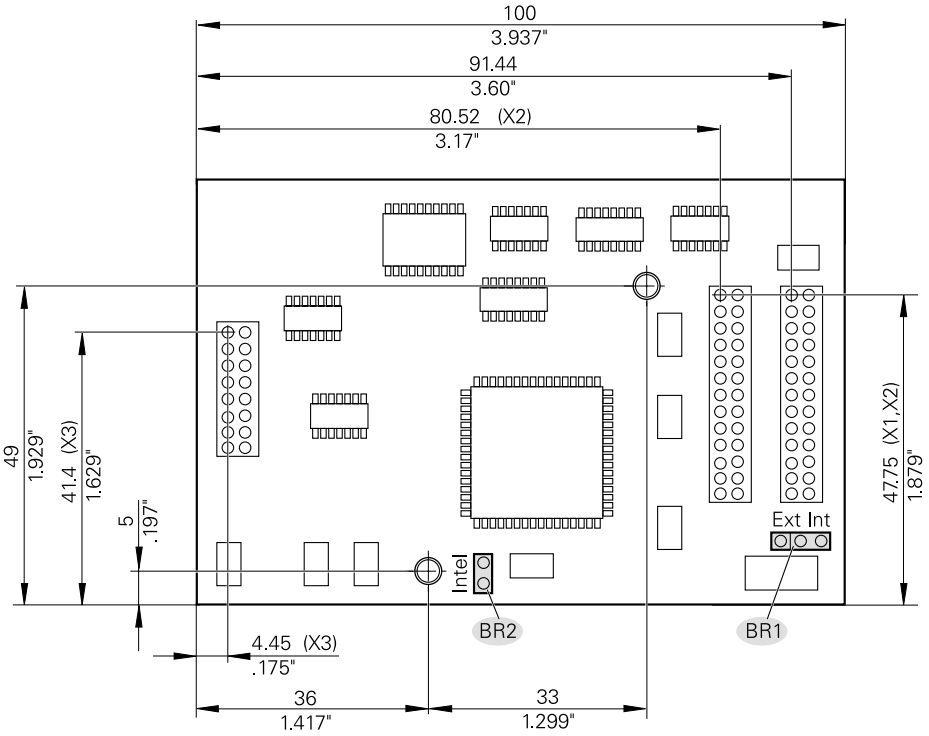
Zählerfunktionen	Zwei Betriebsarten:
Periodenzähler-Modus:	32-Bit-Zählwert
Interpolations-Modus:	32-Bit-Zählwert mit 10-Bit-Interpolationswert
Datenregister:	48 Bit, für den Meßwert werden nur 42 Bit genutzt

Meßwert-Einspeichern
Der Einspeichervorgang kann durch asynchrone Einspeicher-Signale ausgelöst werden (–L0 oder –L1, flankensensitiv), durch einen Softwarebefehl oder durch das Überfahren der Referenzmarke. Die Zeit zwischen Einspeicheranforderung und Einspeichervorgang beträgt 200 ns*) mit einer zeitlichen Unsicherheit von 100 ns*).
Nachdem der Wert gebildet wurde (max. 25 µs), wird ein Statusbit gesetzt und/oder ein Interrupt ausgelöst.

Datenformat	MOTOROLA- oder INTEL-Format

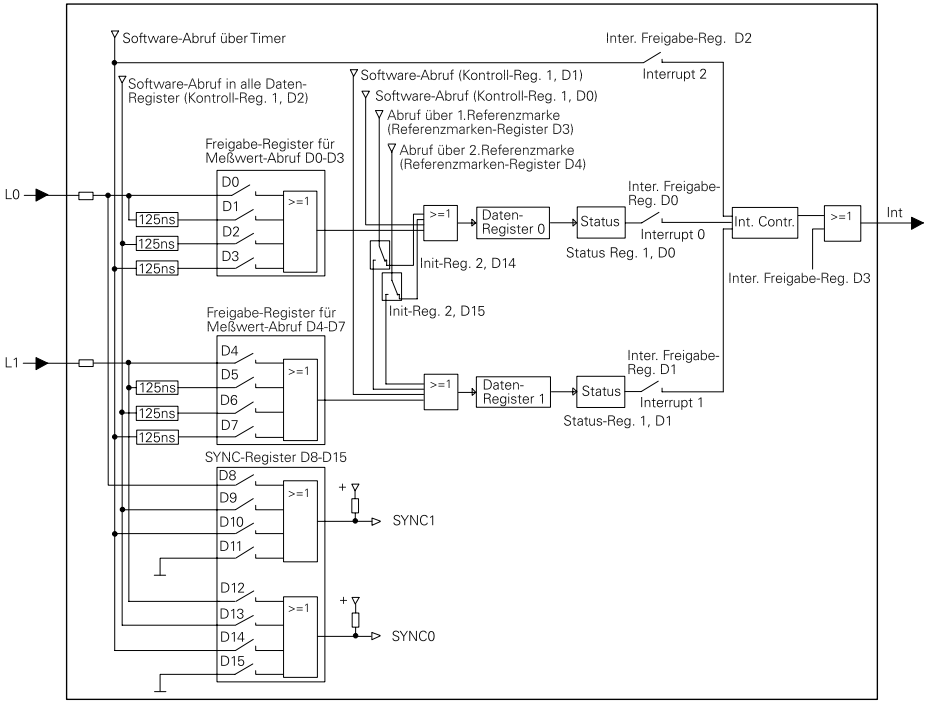
*) Die angegebenen Zeiten beziehen sich auf die Maximale Taktfrequenz von 20 MHz.

Anschlußmaße



BR1: Externer oder interner Takt
BR2: INTEL-Datenformat: Brücke gesteckt
MOTOROLA-Datenformat: Brücke offen

Prinzip-Schaltbild der Abruf-Wege in den Zählerbausteinen



HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH
Dr.-Johannes-Heidenhain-Straße 5
83301 Traunreut, Germany
☎ +49/86 69/31-0
FAX +49/86 69/50 61
E-Mail: info@heidenhain.de

☎ **Service** +49/86 69/31-12 72
☎ TNC-Service +49/86 69/31-14 46
FAX +49/86 69/98 99
E-Mail: service@heidenhain.de

www.heidenhain.de

326 959-01 · 1 · 7/2002 · S · Printed in Germany · Änderungen vorbehalten