HEIDENHAIN

Benutzer-Handbuch

IK 342

VMEbus-Zählerkarte

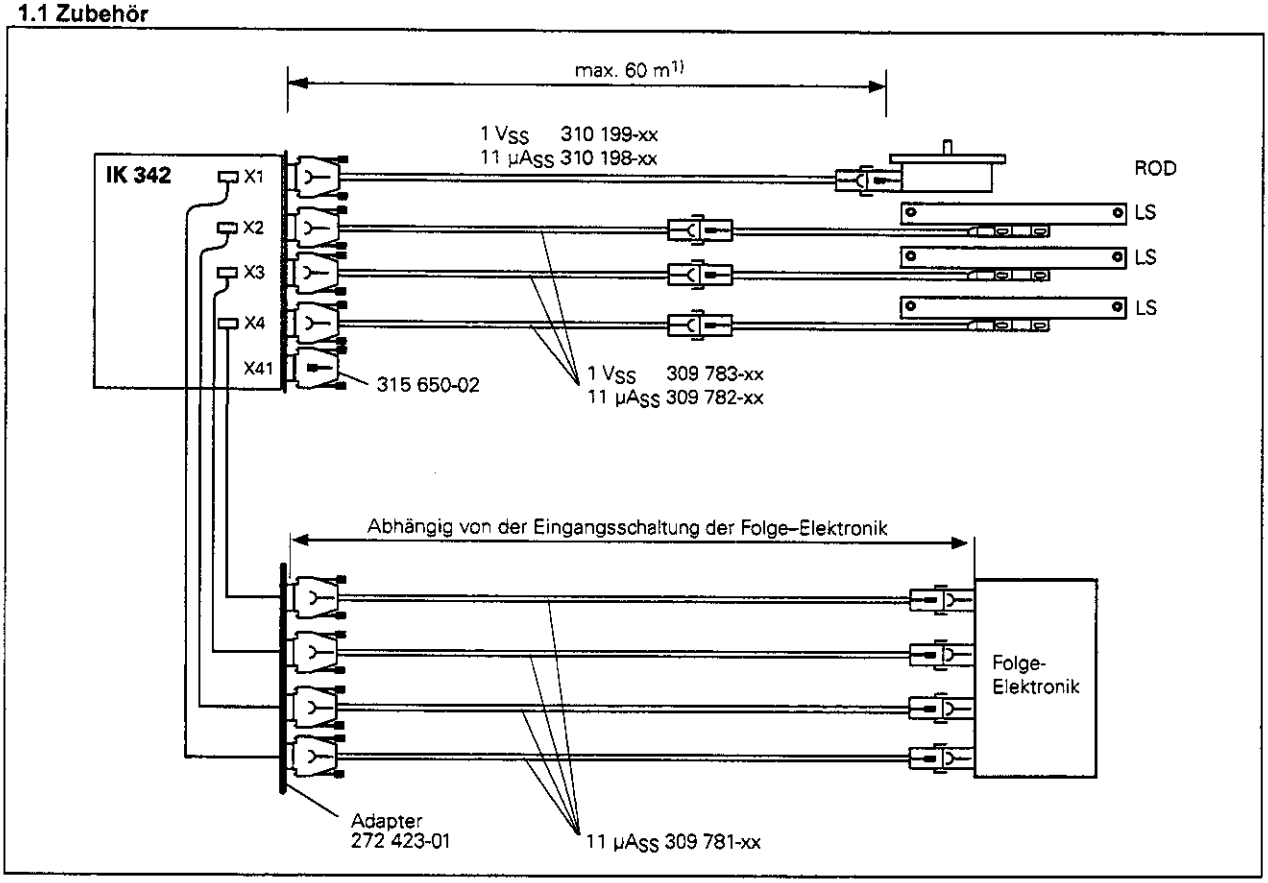
11/2000

Inhaltsübersicht

1 Lieferumfang

VMEbus-Zählerkarte IK 342 mit Meßsystem-Eingängen für sinusförmige Signale (1 V_{SS} oder 11 μA_{SS} umschaltbar), Programmierbeispiele, Treiber-Software und Benutzer-Handbuch

1.1 Zubehör



1) Kabel bis 150 sind möglich, falls durch eine externe Versorgung gewährleistet ist, daß 5 V am Meßsystem anliegen.

310 199-xx 15/12polig	Adapter-Kabel mit Stecker für HEIDENHAIN-Meßsysteme (1 V_{SS}) mit Flanschdose; Standardlänge 0,5 m
310 198-xx 15/9polig	Adapter-Kabel mit Stecker für HEIDENHAIN-Meßsysteme (11 μA_{SS}) mit Flanschdose; Standardlänge 0,5 m
309 783-xx 15/12polig	Adapter-Kabel mit Kupplung für HEIDENHAIN-Meßsysteme (1 V_{SS}); Standardlänge 0,5 m
309 782-xx 15/9polig	Adapter-Kabel mit Kupplung für HEIDENHAIN-Meßsysteme (11 μA_{SS}); Standardlänge 0,5 m
315 650-02	Stecker für die externen Funktionen am Anschluß X41
272 423-01	Adapter für Meßsystem-Ausgänge (sinusförmige Stromsignale)
309 781-xx	Verbindungskabel vom Meßsystem-Ausgang an eine weitere Anzeige oder Steuerung

2 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.

Einige Hinweise zu den verwendeten Begriffen:

- Zahlen in hexadezimaler Schreibweise werden mit $ gekennzeichnet, z.B. $FF.
- Invertierte Signale erhalten vor dem Signalnamen ein Minus-Zeichen, z.B. –IMPULS1.
- Der Begriff „**einspeichern**" bedeutet, daß der Zählerwert im Daten-Register festgehalten wird. Dieser Zählerwert muß anschließend **abgerufen** werden, d.h. per Software gelesen und im Rechner gespeichert oder am Bildschirm angezeigt werden.
- Die Bedeutung der VMEbus-Signale und -Begriffe entnehmen Sie bitte der VMEbus-Fachliteratur.

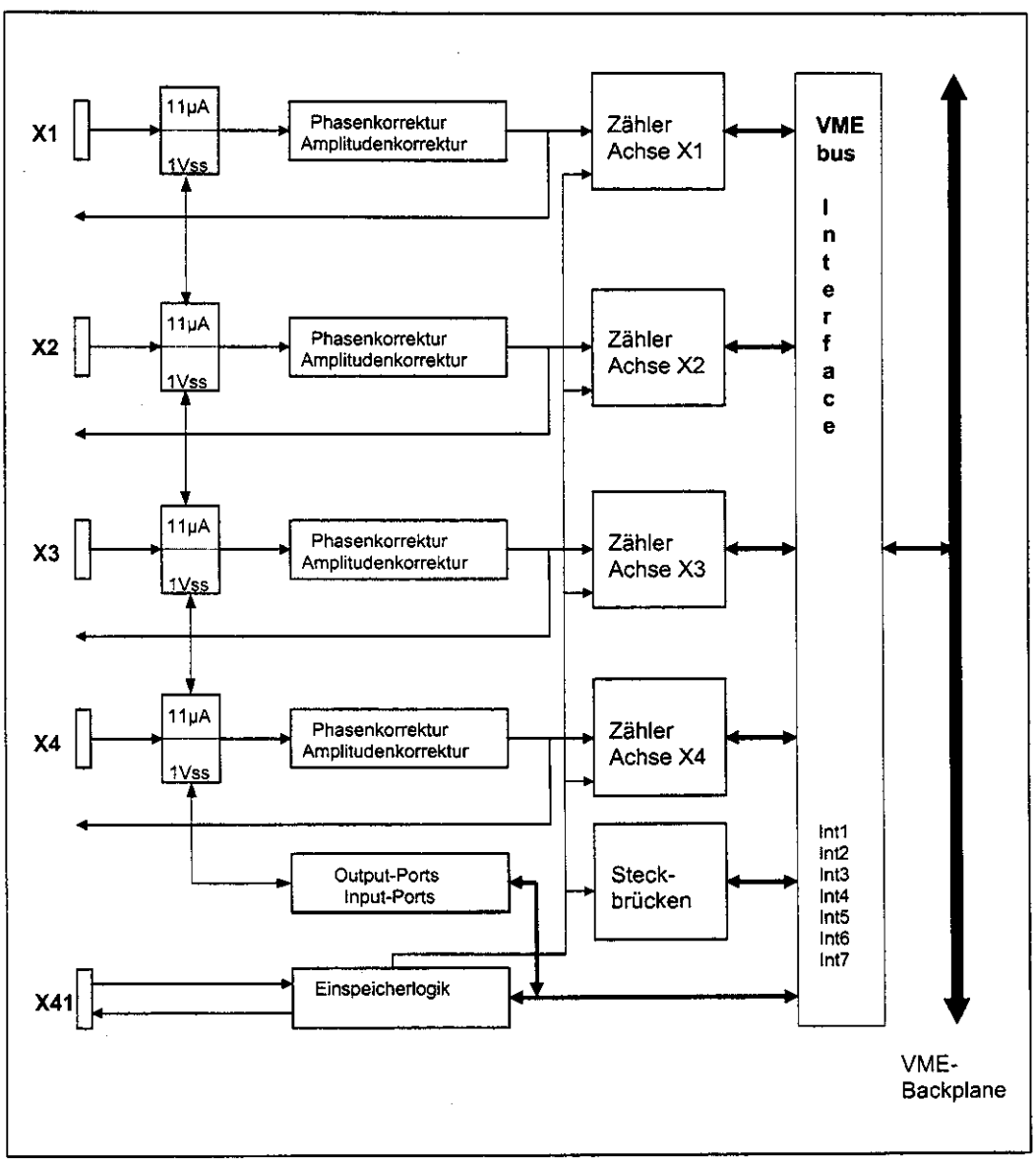
3 Technische Beschreibung der IK 342

Die IK 342 ist eine Zählerkarte zur Weg- und Winkelmessung mit Hilfe von VMEbus-Computern, an die Sie vier HEIDENHAIN-Meßsysteme mit sinusförmigen Spannungs- und Stromsignalen anschließen können. Sie wird direkt in einen freien Steckplatz des Computers gesteckt.

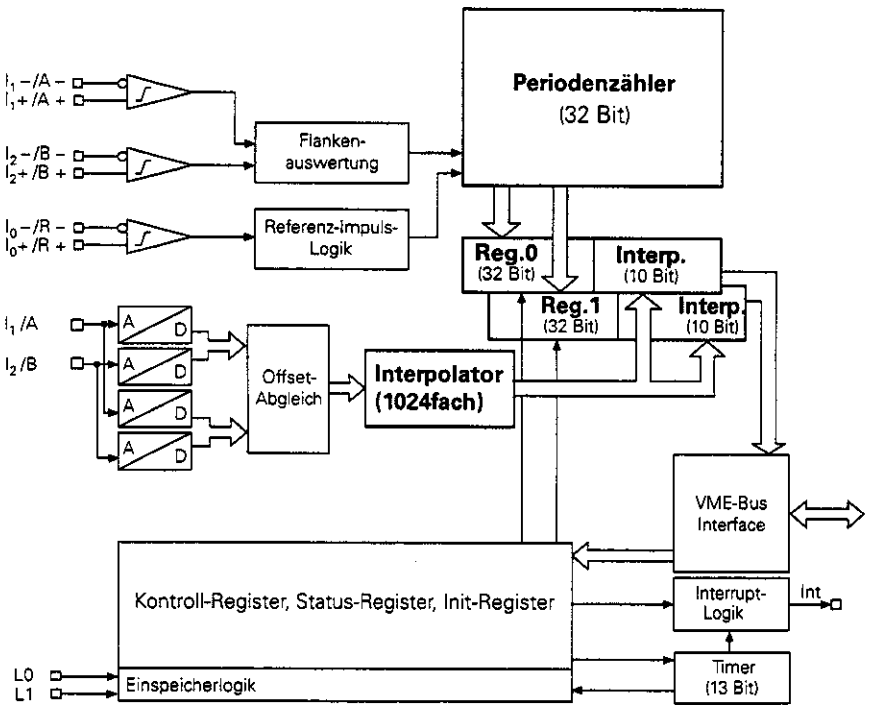
Die Positionen der vier Meßsysteme können Sie über externe Abruf-Eingänge oder per Software abrufen und im Computer weiterverarbeiten.

Die IK 342 ist ideal für Anwendungen, bei denen eine hohe Auflösung der Meßsystem-Signale und eine schnelle Meßwert-Erfassung erforderlich sind.

Blockschaltbild der IK 342:



Blockschaltbild des Zählerbausteins:



Die Interpolations-Elektronik in der IK 342 unterteilt die Signalperiode des Eingangs-Signals 1 024fach.
Der Interpolationswert (10 Bit) bildet zusammen mit dem Wert des Periodenzählers (32 Bit) den 42 Bit breiten Meßwert. Die IK 342 speichert die Meßwerte in 48 Bit breiten Daten-Registern, wobei die oberen Bits entsprechend der Zweierkomplement-Darstellung vorzeichenrichtig erweitert werden.
Sie können die Meßwerte entweder per Software, über externe Eingänge, durch das Überfahren der Referenzmarke oder per Timer einspeichern und anschließend über den VMEbus zum Rechner übertragen.

Phase und Amplitude der sinusförmigen Meßsystem-Signale können Sie über elektronische Potentiometer per Software abgleichen; den Offset über Datenregister im Zählerbaustein.

3.1 Zugriffszeit auf Meßwerte

Die Zeit für die Meßwertbildung ist < 25 µs.

4 Hardware

4.1 VMEbus-Interface

Spezifikation:	ANSI/IEEE STD 1014-1987, IEC 821 und 297
Größe:	double height board (170 mm x 261 mm), 1 Slot
Slave:	A16, D16, D08(EO), D08(O) Interrupter
Interruptleitungen:	7
Stromaufnahme (max):	+12 V: 25 mA −12 V: 25 mA +5 V: 350 mA (ohne Meßsysteme)
Leistungsaufnahme (max):	2,5 W ohne Meßsysteme

4.2 Meßsystem-Eingänge IK 342

An die IK 342 können Sie HEIDENHAIN-Längenmeßsysteme oder -Winkelmeßsysteme mit sinusförmigen Spannungssignalen A und B oder mit sinusförmigen Stromsignalen I_1 und I_2 anschließen. Die Meßsystem-Eingänge sind per Software umschaltbar und die Eingangsfrequenz ist programmierbar (siehe Register BA + $180 + $08 sowie BA $180 + $0A).

Spezifikation der Meßsystemsignale 1 V_{SS}

Signalamplituden: A, B (0°, 90°) R (Referenzmarke)	0,6 V_{SS} bis 1,2 V_{SS} 0,2 V bis 0,85 V Nutzanteil
Signalpegel für Fehlermeldung A, B	≤ 0,23 V_{SS} ≥ 2 V_{SS}
Maximale Eingangsfrequenz	umschaltbar 33 kHz / 500 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.

Spezifikation der Meßsystemsignale 11 μA_{SS}

Signalamplituden: I_1, I_2 (0°, 90°) I_0 (Referenzmarke)	7 μA_{SS} bis 16 μA_{SS} 3,5 µA bis 8 µA Nutzanteil
Signalpegel für Fehlermeldung I_1, I_2	≤ 2,5 μA_{SS} ≥ 25,6 μA_{SS}
Maximale Eingangsfrequenz	umschaltbar 33 kHz / 175 kHz
Kabellänge	max. 30 m (Versorgungsspannung 5,0 V)

Anschluß X1 bis X4 für Meßsysteme
Sub-D-Anschluß mit Stifteinsatz (15polig)

Anschluß-Nr.	Belegung 1 Vss	Belegung 11 µAss
1	+5 V (U_P)	+5 V (U_P)
2	0 V (U_N)	0 V (U_N)
3	A +	I_1+
4	A –	I_1–
5	0 V	0 V
6	B +	I_2+
7	B –	I_2–
8	0 V	0 V
9	+5 V	+5 V
10	R +	I_0+
11	0 V	0 V
12	R –	I_0–
13	0 V	0 V
14	nicht belegt	nicht belegt
15	nicht belegt	nicht belegt
Gehäuse	Außenschirm	Außenschirm

4.3 Meßsystem-Ausgänge

Die IK 342 gibt die Meßsystem-Signale der Eingänge X1 bis X4 zusätzlich über vier 10polige AMP-Stecker auf der Platine **immer** als **sinusförmige Stromsignale** (11 μA_{SS}) aus. Bezeichnung der Platinenstecker: X11, X12, X13 und X14. Über eine zusätzliche Baugruppe (Id.-Nr. 272 423 01) können diese Signale nach außen zu 9poligen Sub-D-Anschlüssen geführt werden. Adapter-Kabel (Id.-Nr. 309 781 ..) zum Anschluß an HEIDENHAIN-Positionsanzeigen oder Interpolations-Elektroniken sind lieferbar (siehe „1.1 Zubehör").

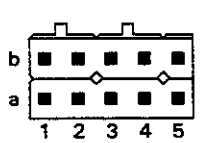
Meßsystem-Ausgänge
Sub-D-Anschluß mit Stifteinsatz (9polig)

Anschluß-Nr.	**Belegung**
1	I_1 –
2	0 V (U_N)
3	I_2 –
4	nicht angeschlossen
5	I_0 –
6	I_1 +
7	nicht angeschlossen
8	I_2 +
9	I_0 +
Gehäuse	Außenschirm

Platinenstecker für Meßsystem-Ausgänge
AMP mit Stifteinsatz (10polig)

Anschluß-Nr.[1)]	**Signal**
1a	nicht angeschlossen
1b	nicht angeschlossen
2a	nicht angeschlossen
2b	0 V (U_N)
3a	I_0 –
3b	I_0 +
4a	I_2 –
4b	I_2 +
5a	I_1 –
5b	I_1 +

1) Die Seite mit den Verriegelungs-Stiften ist mit b bezeichnet.
Anschlüsse 1a und 1 b befinden sich auf der Seite mit der Einkerbung.



4.4 Abgleich der Meßsystem-Signale

Bei hohen Anforderungen an die Genauigkeit der Meßergebnisse sollten Sie die sinusförmigen Meßsystem-Signale abgleichen. Den Abgleich führen Sie per Software durch. Folgende Größen können Sie abgleichen:

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

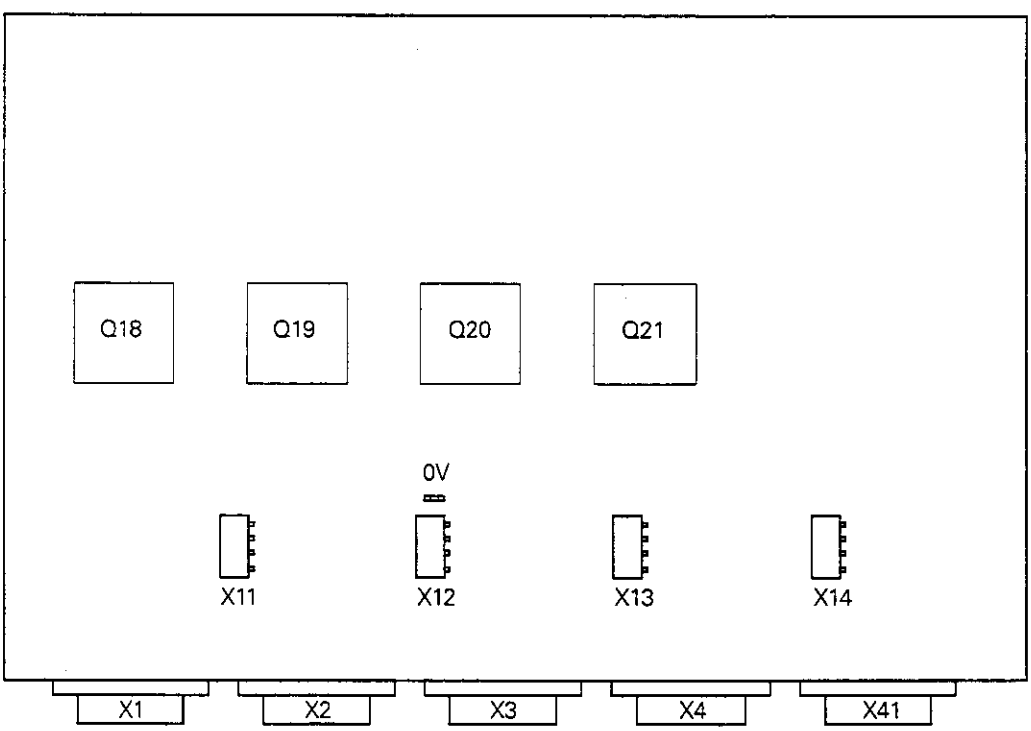
Die Ansteuerung der Potentiometer erfolgt über I²C-Bus, der über ein Register der Konfigurations- bzw. Einspeicherlogik (Adresse $0E, Bit0, Bit1, Bit2) gesteuert wird. Da die Erzeugung der Steuersequenzen aufwendig ist, empfehlen wir Ihnen, das Programm POTIS.EXE oder – falls dieses Programm mit Ihrem Rechner nicht läuft – die Funktionen und Prozeduren in „BORLAND

C++" aus den Dateien IIC.CPP, POTI_1.CPP und POTIS.CPP als Orientierungshilfe für eigene Funktionen zu nutzen.

Die Korrekturwerte für Phase und Amplitude speichert die IK 342 netzausfallsicher in den Bausteinen der elektronischen Potentiometer. Die Offset-Register in den Zählerbausteinen sind nicht netzausfallsicher. Deshalb müssen Sie die Offset-Korrekturwerte in einem EEPROM im Baustein für die elektronischen Potentiometer speichern. Nach dem Einschalten müssen Sie per Software die Offset-Korrekturwerte vom EEPROM in die Offset-Register der Zählerbausteine laden. In der Datei IIC.CPP sind zwei Prozeduren definiert, die diese Aufgaben erfüllen. Die Prozedur „StoreOffset" speichert die Offset-Korrekturwerte in das EEPROM. Die Prozedur „LoadOffset" liest die Offset-Korrekturwerte aus dem EEPROM in die Offset-Register der Zählerbausteine. „LoadOffset" wird auch von der Prozedur „InitIk342" genutzt.

Zum Abgleich benötigen Sie folgende Hilfsmittel:
- Oszilloskop für XY-Darstellung
- Digitales Multimeter
- Software zum Einstellen der elektronischen Potentiometer und zum Beschreiben der Offset-Register (z.B. POTIS.EXE)

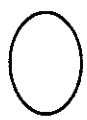
An den Platinensteckern für Meßsystem-Ausgänge mit den Bezeichnungen X11, X12, X13 und X14 können Sie die Meßsystem-Signale für die vier Meßsystem-Eingänge abgreifen. Zusätzlich befindet sich auf der Platine eine Lötfahne mit der Beschriftung 0 V (siehe Zeichnung).



Phasenverschiebung und Amplitudenverhältnis abgleichen

➤ Schließen Sie das Oszilloskop wie folgt an:
  – X-Ablenkung an Pin 4a des Platinensteckers
  – Y-Ablenkung an Pin 5a des Platinensteckers
  – Masse an der Lötfahne mit der Beschriftung 0 V

➤ Verfahren Sie den Abtastkopf, Abtastfrequenz ca. 1 kHz. Am Oszilloskop sehen Sie nun eine LISSAJOUS-Figur. Sind die beiden sinusförmigen Meßsystem-Signale optimal abgegleichen, sehen Sie einen Kreis. Sind die Amplituden der beiden Signale nicht gleich, sehen Sie eine waagerecht oder senkrecht liegende Ellipse. Wenn die Phasenverschiebung der beiden sinusförmigen Signale nicht 90° ist, sehen Sie eine schrägliegende Ellipse (siehe Bild).

Schirmbild	Auswertung
	Amplituden sind gleich Phasenverschiebung 90° Meßsystem optimal abgeglichen
	Phasenverschiebung ≠ 90°
	Amplituden unterschiedlich

➤ Gleichen Sie das Amplitudenverhältnis und den Phasenwinkel per Software durch Verstellen der elektronischen Potentiometer ab.

Offset abgleichen

➤ Stellen Sie das digitale Multimeter auf **DC** (Gleichspannungsanzeige) ein und schließen Sie es wie folgt an:
- Zum Abgleichen des 0°-Signales: an Pin 5b (Bezugspotential) und Pin 5a des Platinensteckers.
- Zum Abgleichen des 90°-Signals: an Pin 4b (Bezugspotential) und Pin 4a des Platinensteckers.

➤ Den Meßwert bei gleichmäßigem Verfahren des Abtastkopfs ablesen. **Vorzeichen beachten!**

➤ Korrekturwert für das Offset-Register ermitteln: 1 Schritt = 4, 88 mV. Den zu programmierenden Wert berechnen Sie also wie folgt:

$$\text{Eingabe} = - \frac{\text{Abgelesener Mittelwert mV}}{4{,}88\ \text{mV}}$$

Beachten Sie, daß der Korrekturwert mit umgekehrtem Vorzeichen des abgelesenen Mittelwerts programmiert werden muß. Wundern Sie sich nicht, wenn Sie nach der Programmierung des Offset-Registers am Meßpunkt keine Änderung feststellen. Das Offset-Register befindet sich hinter Ihrem Meßpunkt, und deshalb können Sie die Wirkung Ihres Korrekturwerts nicht überprüfen. Die Wirkung des Korrekturwerts kann nur durch Auslesen der Register $10 für die 0°-Amplitude und $0C für die 90°-Amplitude überprüft werden.

4.5 Externe Funktionen

Zum externen Einspeichern der Meßwerte und zum synchronen Einspeichern mehrerer Karten ist ein 9poliger Sub-D-Anschluß vorhanden. Den dafür benötigten Stecker (Id.-Nr. 297 050 ZY) können Sie bei HEIDENHAIN bestellen. Weitere Beschreibung dieser Funktionen siehe „6 Positionswert einspeichern".

Pinbelegung 9polige Sub-D-Buchse

Signalbedeutung	Signalbezeichung	Belegung Anschluß-Nr.
Einspeichern X1, X2, X3, X4	E1	3
(über Einspeicher-Logik kon-	E2	4
figurierbar)	E3	5
	E4	6
Kaskadierung 0	–CASC0	7
Kaskadierung 1	–CASC1	8
0 V	0 V	1
0 V	0 V	2
+5 V	+5 V	9

5 Adressierung

5.1 Aufteilung des Adreßraums

Die VMEbus-Zählerkarte belegt 512 Bytes des vorhandenen Adreßraums. Der auf der Karte vorhandene 8polige DIP-Schalter S1 mit den Schaltebenen 1 bis 8 teilt den Adreßraum A16 des VMEbus (Address Modifier AM: $29) in 128 Teile zu je 512 Bytes auf. Die Schaltebene 1 ist für die Einstellung der Basisadresse ohne Funktion: sie wird für den Interruptvektor benutzt. Es sind also nur 128 Basisadressen möglich.
Die Basisadresse der Karte errechnen Sie wie folgt:
Basisadresse = Schalterstellung (ohne S1) * 512

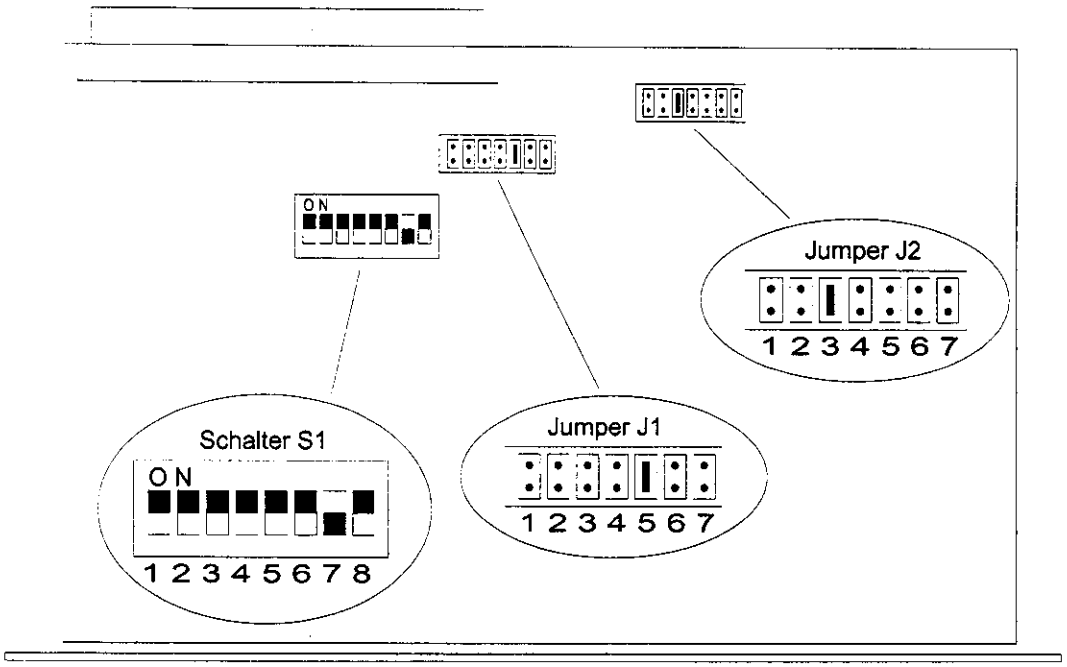
Beispiel:
DIP-Schalter S1 = $A1; Basisadresse = ($A1 shift right 1) * 512 = $A000

Aufteilung des Adreßraums:

	Funktion
BA + $000	ID-Register, 1 Register (8 Bit), nur Lesezugriff
BA + $080	Zähler X1, 32 Register (16 Bit)
BA + $0C0	Zähler X2, 32 Register (16 Bit)
BA + $100	Zähler X3, 32 Register (16 Bit)
BA + $140	Zähler X4, 32 Register (16 Bit)
BA + $180	Einspeicher-/ Konfigurations-Logik, 8 Register (8 Bit)

Für die vier Zähler (Eingang X1 bis X4) ist jeweils ein eigener Registersatz vorhanden. Die Funktionen der Zähler- und Kontrollregister sind für alle vier Zähler identisch. In dieser Beschreibung werden die Zählerregister und Kontrollregister ohne Offset angegeben: wollen Sie z.B. ein Register des 4. Zählers ansprechen, so müssen Sie den Offset (Basisadresse + $140) zur Register-Adresse addieren.

5.2 Schalter und Jumper



Belegung	**Jumper 1** Interrupt-**Request**-Leitung	**Jumper 2** Interrupt-**Acknowledge**-Leitung
1	–IRQ7	–INT1
2	–IRQ6	–INT2
3	–IRQ5	–INT3*
4	–IRQ4	–INT4
5	–IRQ3*	–INT5
6	–IRQ2	–INT6
7	–IRQ1	–INT7

*Ab Werk bestückt
Ohne Steckbrücke ist der Interrupt gesperrt.

DIP-Schalter S1: Basis-Adresse und IACK-Vektor

Switch-Nr.	Basis-Adresse	Vektor-Nr.
1	0	1
2	1	2
3	2	4
4	4	8
5	8	16
6	16	32
7	32	64
8	64	128

Schalter OFF: Wertigkeit siehe Tabelle
Schalter ON: Wertigkeit = 0

Basis-Adresse = 512 * (Summe der eingestellten Wertigkeiten)
Vektor-Nummer = Summe der eingestellten Wertigkeiten

Einstellung der Basis-Adresse ab Werk: 32 = $20
Einstellung der Vektor-Nummer ab Werk: 64 = $40

5.3 Interrupts

Die Interruptquellen E1 bis E4 (externe Einspeichereingänge) sind nach –RESET gesperrt und können durch Setzen des Bits 3 des Registers mit Adresse BA + $0E der Einspeicher-Logik (siehe „7.3 Register zur Konfiguration der Einspeicher-Logik") freigegeben werden. Mit Jumper J1 wählen Sie die Interrupt-Request-Leitung (mögliche Leitungen: –IRQ7 bis –IRQ1) und mit Jumper J2 die Interrupt-Acknowledge-Leitung (mögliche Leitungen: –INT1 bis –INT7). Ein Interrupt wird gespeichert und solange ausgegeben, bis ein erfolgreicher Interrupt-Acknowledge-Zyklus durchgeführt wurde. Erkennt die IK 342 einen gültigen Interrupt-Acknowledge-Zyklus, so gibt sie die Vektor-Nummer (Einstellung des DIP-Schalters) als Kennung aus und löscht den Interrupt.

6 Positionswert einspeichern

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

6.1 Überblick

Damit Sie den Positionswert auslesen können, müssen Sie ihn zuerst in einem der beiden Daten-Register 0 oder 1 einspeichern.
Es gibt folgende Einspeichermöglichkeiten:

- per Software: jede Achse einzeln oder alle Achsen gemeinsam per interner Kaskadierung
- per Hardware über X41 (Signale E1 bis E4)
- über mehrere Karten per externer Kaskadierung: über –CASC0 und –CASC1
- per Referenzimpuls
- per Timer

Beim Einspeichern übernimmt die IK 342 vollständig den aktuell gültigen Positionswert in ein oder mehrere Daten-Register, ohne daß der Zählvorgang beeinflußt wird. Danach sind die Daten-Register für weitere Speichervorgänge gesperrt und werden erst nach dem Auslesen wieder freigegeben.

6.2 Einspeichern per Software

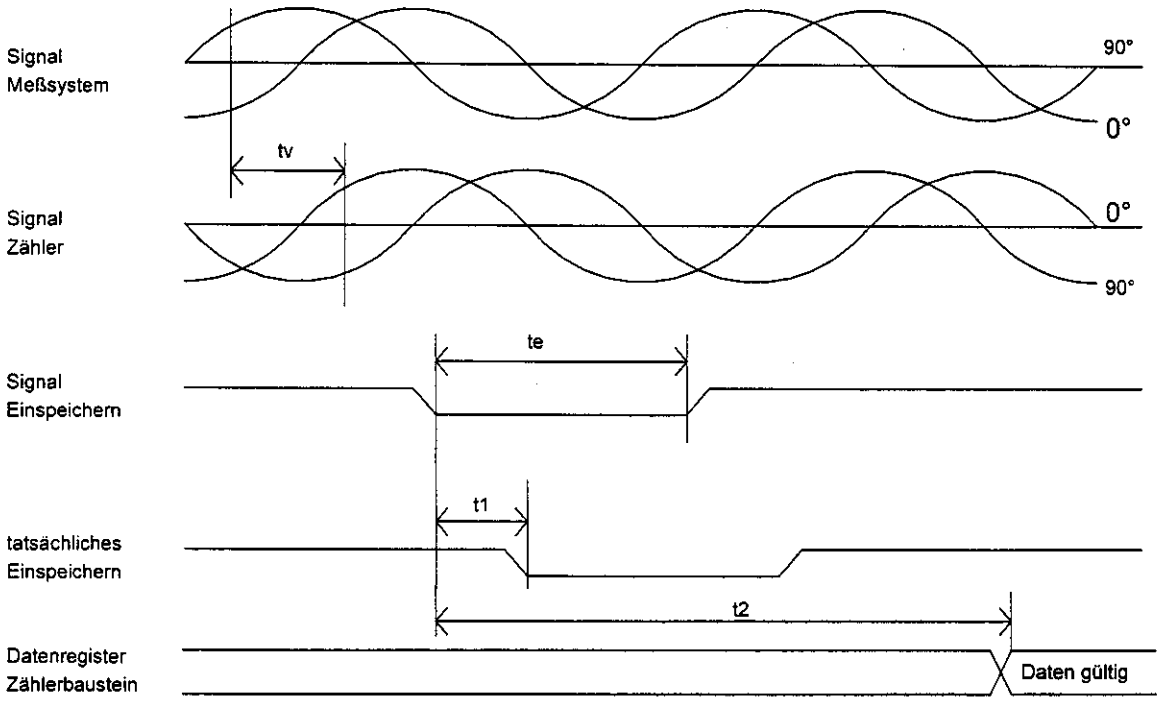
Durch Beschreiben des entsprechenden Bits des Kontroll-Registers 1 ($20) des Zählerbausteins lösen Sie ein Einspeichern per Software für die Daten-Register aus.
Das **gemeinsame Einspeichern in mehreren Zählerbausteinen** geschieht über die Ausgänge SYNC0 (Daten-Register 0) und SYNC1 (Daten-Register 1) des Zählerbausteins der Achse X1. Die Freigabe des Software-Abrufs über die SYNC-Leitungen erfolgt über das Register zur Achs-Kaskadierung ($18). Die Einspeicher-Logik gibt diese Signale an die Eingänge weiterer Zählerbausteine über X1.L0 bis X4.L0 (SYNC0) und X1.L1 bis X4.L1 (SYNC1) und die Freigabe-Register ($18) weiter. So kann die IK 342 alle Achsen synchron einspeichern.

Zum Ausgleich der Signallaufzeit zwischen den Achsen sollen Sie bei der Achse X1 einen Signalweg **mit** und für alle weiteren Achsen **ohne** Verzögerungsglied wählen.

6.3 Einspeichern per Hardware über X41

Durch Anlegen des aktiven Pegels an E1 bis E4 wird ein Einspeichersignal über X1.L0 bis X4.L0 oder X1.L1 bis X4.L1 und die Freigabe-Register $18 auf den entsprechenden Daten-Registern der Zähler erzeugt (siehe „7.3 Register zur Konfiguration der Einspeicher-Logik").
Die externen Einspeichersignale können einen Interrupt erzeugen. Dazu muß in der Einspeicher-Logik auf Adresse $0A das Bit 1 gesetzt werden und die Interrupt-Nummer durch die Jumper J1 und J2 gesetzt sein.

Zeitdiagramm



Signal	Bezeichnung	min.	max.
Verzögerung Meßsystemeingang – Zähler	t_v	–	< 2 µs
Breite der Einspeichersignale E1 bis E4	t_e	1,2 µs	–
Verzögerung der Einspeichersignale	t_1	–	< 0,8 µs
Verzögerung bis zum Bereitstellen des Meßwerts in den Datenregistern	t_2	–	< 24 µs

Signalpegel für X41 (E1 – E4)

Bezeichnung	min.	max.
U_{eH}	3,5 V	32 V
U_{eL}	–20 V	1 V
I_I	0,5 mA	8 mA

Die Eingänge sind low-aktiv oder high-aktiv (über die Einspeicher-Logik konfigurierbar) und werden durch interne Pull-Up-/Pull-Down-Widerstände auf dem entsprechenden inaktiven Pegel gehalten. Sie können die Eingänge durch TTL-Standard, LS,- ALS- oder CMOS-Bausteine sowie mit 24-V-Signalen ansteuern.

6.4 Einspeichern über mehrere Karten per externe Kaskadierung

Der Zählerbaustein der Achse X1 steuert über die Ausgänge SYNC0 und SYNC1 die beiden Schaltausgänge –CASC0 und –CASC1, die über die 9polige Sub-D-Steckverbindung X41 nach außen geführt werden. Signal –CASC0 entspricht Daten-Register 0, –CASC1 Daten-Register 1. Mit einer externen Verbindung der –CASCx-Ausgänge mit den externen Einspeicher-Eingängen weiterer Karten über X41 ist ein synchrones Einspeichern über mehrere IK 342 möglich.
Signalpegel für X41 (CASC0, CASC1): TTL

6.5 Einspeichern per Referenzmarke

Durch Setzen der entsprechenden Bits im Initialisierungs- ($24) und Referenzmarken-Register ($1C) wird beim Überfahren von Referenzmarken die programmierte Funktion ausgeführt. Dadurch können Sie sowohl einzelne als auch abstandscodierte Referenzmarken auswerten.

6.6 Einspeichern per Timer

Im Timer-Register ($08) können Sie eine Zeit festlegen, nach der ein Abruf über das Freigabe-Register ($18) erfolgt. Der Timer wird im Initailisierungs-Register ($24) gestartet.

7 Register

7.1 BA + $0000: ID-Register (nur Lesezugriff)

Das ID-Register gibt eine Hardwarekennung für die Karte aus.

D8 – D4: Leiterplattenversion	D3 – D0: Bestückungsvariante
0: Version 01	0: IK 342

7.2 Register der Zählerbausteine

Bei der IK 342 wird heute der Zählerbaustein G38 eingesetzt, der den Zählerbaustein G26 ablöste, wobei G38 zu G26 aufwärts kompatibel und um einige Lesefunktionen auf einzelne Register erweitert ist. Die Bausteine (Q18 bis Q21, siehe Grafik Seite 11) unterscheiden sich nicht nur optisch durch ihre Nummern (G26 = 282 150 01, G38 = 315 860 01), sondern auch elektrisch durch die Bausteinkennung (siehe Kennungs-Register des Zählerbausteins).
Für die folgende Beschreibung ist das Prinzip-Schaltbild der Zählerbausteine auf der letzten Seite hilfreich.

Register-Adressierung

Sie können auf die Zählerbausteine 16 Bit oder 8 Bit breit zugreifen (Wort- oder Byte-Zugriff). Die Art des Zugriffs wird im Kontroll-Register 3 ($00) Bit 0 programmiert. Da die IK 342 einen 16-Bit-Datenbus hat, sollte natürlich auch der Zugriff auf die Zählerbausteine in diesem Modus erfolgen. Deshalb geht die folgende Registerbeschreibung von 16-Bit-Registern aus.
Nach dem Einschalten sind die Zählerbausteine defaultmäßig im 8-Bit-Modus. Zum Umstellen auf 16 Bit muß man entweder über einen Wort-Zugriff auf Adresse $00 das Bit 0 auf 0 zurücksetzen oder über einen Byte-Zugriff auf Adresse $03.

Wort-Zugriff

Die Adresse von einem Wort (16 Bit) zum nächsten erhöht sich um vier, weil die Leitung A1 des VMEbus mit A0 des Zählerbausteins verbunden ist. Bei der Adreßangabe in der folgenden Register-Übersicht sind die Wort-Adressen angegeben. Somit errechnet sich die Adresse (z.B. für das Initialisierungs-Register 1 der Achse X2) wie folgt (Annahme: DIP-Schalter S1 = $40):

Register-Adresse = (($40 shift right 1) * $200) + $0C0 + $024 = $40E4

Byte-Zugriff

Bei einem Byte-Zugriff auf das High Byte gilt die Wortadresse der folgenden Register-Übersicht. Die Adresse des Low Bytes berechnet sich wie folgt:

Low-Byte-Adresse = Adresse aus der Register-Übersicht + $03

Register-Übersicht

Adreß-Offset zur Zählerbasisadr.	Schreibzugriff	Lesezugriff
$3C $38 $34	ohne Funktion	Daten-Register 0, LS-Word Daten-Register 0 Daten-Register 0, MS-Word
$30 $2C $28	ohne Funktion	Daten-Register 1, LS-Word Daten-Register 1 Daten-Register 1, MS-Word
$24 Low Byte High Byte	Initialisierungs-Register 1 Initialisierungs-Register 2	Initialisierungs-Register 1 Initialisierungs-Register 2
$20 Low Byte High Byte	Kontroll-Register 1 ohne Funktion	Status-Register 1 Status-Register 2
$1C Low Byte High Byte	Referenzmarken-Register ohne Funktion	ohne Funktion Amplitudenwert-Register
$18 Low Byte High Byte	Freigabe-Register für Meßwertabruf Achs-Kaskadierung	ohne Funktion ohne Funktion
$14 Low Byte High Byte	ohne Funktion ohne Funktion	ohne Funktion ohne Funktion
$10 Low Byte High Byte	Offset-Register für 0°-Signal ohne Funktion	Amplitude für 0°-Signal Amplitude für 0°-Signal
$0C Low Byte High Byte	Offset-Register für 90°-Signal ohne Funktion	Amplitude für 90°-Signal Amplitude für 90°-Signal
$08 Low Byte High Byte	Timer-Register, LS-Byte Timer-Register, MS-Byte	ohne Funktion ohne Funktion
$04 Low Byte High Byte	Kontroll-Register 2 ohne Funktion	Status-Register 3 Kennungs-Register
$00 Low Byte High Byte	Kontroll-Register 3 ohne Funktion	Status-Register 4 ohne Funktion

$28 bis $3C: Daten-Register für die Zähler

Die Meßwerte werden in 48 Bit breiten Registern gespeichert. Pro Achse stehen zwei Daten-Register zur Verfügung: Daten-Register 0 ($3C bis $34) und Daten-Register 1 ($30 bis $28). Die Meßwerte werden aus dem 10-Bit-Interpolationswert und dem 32 Bit breiten Wert des Periodenzählers zusammengesetzt. Von den 48 Bit breiten Registern werden also nur 42 Bit für den Meßwert genutzt. Die oberen 6 Bits werden entsprechend der Zweierkomplement-Darstellung vorzeichenrichtig erweitert.

Die Datenbreite von 48 Bit kann über das Initialisierungs-Register 1 ($24), Bit D6 auf 32 Bit verkürzt werden.

Über das Initialisierungs-Register 1 ($24), Bit D7 kann außerdem festgelegt werden, 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.

Die Zählerwerte speichern Sie in die Daten-Register (siehe auch „6 Positionswert einspeichern") über:

- Software-Abruf
- externe Eingänge
- Referenzmarken
- Timer

Über das Status-Register 1 ($20), Bit D0 oder D1, fragen Sie ab, ob der Meßwert in den Daten-Registern gespeichert wurde. Solange Bit D0 oder D1 gesetzt sind, kann kein weiterer Meßwert gespeichert werden, 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 $34h oder $28, im 32-Bit-Modus die Daten-Register $38 oder $2C. Nach dem Lesen des Meßwerts wird in Status-Register 1 ($20), Bit D0 oder D1, wieder zurückgesetzt.

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

$24: 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 Periodenzählers und aus den 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 ($20) zurück. 1 = Betriebsart: 48-Bit-Register Lesen der Bits **D40 bis D47** setzt das Status-Bit D0 bzw. D1 im Status--Register 1 ($20) 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.

$24: 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 10 = 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

$24: Lesezugriff: Bits D0 bis D15: Zurücklesen der Initialisierungs-Register 1 und 2

$20: 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	1 = Amplitudenwert-Register löschen
D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

$20: Status-Register 1 (Lesezugriff)

Bit	Funktion
D0	Status für Software-Abruf in Daten-Register 0; 1 = Meßwert ist bereit
D1	Status für Software-Abruf in Daten-Register 1; 1 = Meßwert ist bereit
D2 D3	ohne Funktion
D4	1 = Zähler ist gestoppt
D5	Signalunterschreitung: 1 = Signalpegel ok 0 = Signalpegel zu klein (Defekt, Verschmutzung)
D6	1 = Meßsystemfehler (Frequenzüberschreitung)
D7	ohne Funktion

$20: 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

$1C: Referenzmarken-Register (Schreibzugriff)

Bit	Funktion
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

$1C: Referenzmarken/Amplitudenwert-Register (Lesezugriff)

Bit	Funktion
D0	Status „Zählerstarten mit REF" (nur bei G38)
D1	Status „Zähler stoppen mit REF" (nur bei G38)
D2	Status „Zähler löschen mit REF" (nur bei G38)
D3	Status „Meßwert abrufen mit REF" (nur bei G38)
D4	Status „Meßwert mit Überfahren der 2. Referenzmarke abrufen" (nur bei G38)
D5	Status „Zähler mit dem Überfahren jeder Referenzmarke löschen" (nur bei G38)
D6 – D7	ohne Funktion
D8 D9	**Aktuelle Amplitude** Mit jedem Meßwert-Abruf wird ein neuer Amplitudenwert ermittelt. D9 D8 **IK 342** 0 0 normale Amplitude 0,47 Vss < U_e < 1,41 Vss 0 1 kleine Amplitude 0,22 Vss < U_e < 0,47 Vss 1 0 hohe Amplitude U_e > 1,41 Vss 1 1 fehlerhaft kleine Amplitude U_e < 0,22 Vss 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 342 den alten Minimalwert durch den aktuellen Amplitudenwert.
D12 D13	G26: ohne Funktion G38:Maximalwert der Amplitude (analog Minimalwert Amplitude)
D14 D15	ohne Funktion

$18: 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 hinteren Umschlagseite verdeutlicht die Funktion der einzelnen Bits.

$18: Register für Achs-Kaskadierung (Schreibzugriff)

Bit	Funktion
D8	1 = Freigabe L0 für alle Achsen (SYNC0)
D9	1 = Freigabe des „Software-Abrufs in alle Daten-Register" für alle Achsen (SYNC0)
D10	1 = Freigabe des Timerstrobes für alle Achsen (SYNC0)
D11	ohne Funktion
D12	1 = Freigabe L1 für alle Achsen (SYNC1)
D13	1 = Freigabe des „Software-Abrufs in alle Daten-Register" für alle Achsen (SYNC1)
D14	1 = Freigabe des Timerstrobes für alle Achsen (SYNC1)
D15	ohne Funktion

Das Prinzip-Schaltbild der Zählerbausteine auf der hinteren Umschlagseite verdeutlicht die Funktion der einzelnen Bits.

$18: **Lesezugriff:**
G26: Register nicht zurücklesbar
G38: Gesamtes Register zurücklesbar

$10: 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 1 023) und 90°-Signals werden Offset-Korrekturwerte addiert. Bei einem Überlauf wird auf 1 023 oder 0 begrenzt.

Bit	Funktion
D0 D1 D2 D3 D4 D5 D6	Offset-Korrekturwert für das 0°-Signal in Zweierkomplement-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 im Baustein für die elektronischen Potentiometer gespeichert. Nach dem Einschalten müssen die Offset-Korrekturwerte vom EEPROM in die Offset-Register der Zählerbausteine geladen werden (siehe Prozeduren „StoreOffset" und „LoadOffset").

$10: Amplitude für das 0°-Signal (Lesezugriff)
Bei jedem Analog-Digital-Wandelvorgang wird das Ergebnis der Wandlung gespeichert. Vor dem Auslesen sollten durch Bit D4 im Kontroll-Register 2 die Werte eingefroren werden.

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

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

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

$0C: Amplitude für das 90°-Signal (Lesezugriff)
Bei jedem Analog-Digital-Wandelvorgang wird das Ergebnis der Wandlung gespeichert. Vor dem Auslesen sollten durch Bit D4 im Kontroll-Register 2 die Werte eingefroren werden.

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

$08: Timer-Register (Schreibzugriff)
Im Timerregister $08 wird der 13 Bit breite Timerwert (0 bis 8 191) abgelegt. Die Zykluszeit wird in µs angegeben, wobei von der gewünschten Zykluszeit 1 µs abgezogen werden muß. Die Taktfrequenz des Timers ist 16MHz (VMEbus) / 16 = 1 Mhz, d.h. ein Inkrement des Timerregisters entspricht 1 µs.

Beispiel:
Gewünschte Zykluszeit = 1 ms = 1 000 µs
Zu programmierender Wert = 1 000 – 1 = 999
Mit dem Beschreiben dieses Registers ist der Timer jedoch noch nicht gestartet, sondern erst durch Setzen von Bit D2 im Initialisierungs-Register 1 ($24). Außerdem müssen Sie Bit D3 im Freigabe-Register $18 setzen.

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

$08: **Lesezugriff**
G26: Register nicht zurücklesbar
G38: Timerwert zurücklesbar

$04: Kontroll-Register 2 (Schreibzugriff)

Bit	Funktion
D0 D1 D2 D3	Festen Wert ≥ 8 programmieren
D4	1 = Amplitude für 0°-Signal ($10) und 90°-Signal ($0C) sowie Amplitudenwert-Register ($1C High Byte) einfrieren. Um eine Änderung der Registerwerte während des Auslesens zu vermeiden, muß dieses Bit gesetzt werden. Ob die Registerwerte eingefroren sind, kann über Bit D4 im Status-Register 3 abgefragt werden.
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

$04: Status-Register 3 (Lesezugriff)

Bit	Funktion
D0 D1 D2 D3	G26: Wert nicht zurücklesbar G38: Wert zurücklesbar
D4	1 = Amplitude für 0°-Signal ($10) und 90°-Signal ($0C) sowie Amplitudenwert-Register ($1C High Byte) sind eingefroren und können gelesen werden.
D5	0 = Das Offset-Register für das 0°-Signal ist geschrieben.
D6	0 = Das Offset-Register für das 90°-Signal ist geschrieben.
D7	ohne Funktion

$04: Kennungs-Register (Lesezugriff)

Bit	Funktion
D8 D9 D10 D11 D12 D13 D14 D15	Bausteinkennung: G26: fester Wert 8 G38: fester Wert 9

$00: Kontroll-Register 3 (Schreibzugriff)

Bit	Funktion
D0	0 = 16-Bit-Bus 1 = 8-Bit-Bus
D1	fester Wert 0
D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	ohne Funktion

$00: Status-Register 4 (Lesezugriff)

Bit	Funktion
D0	Zurücklesen von D0 (Kontroll-Register 3)
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

7.3 BA + $180: Register zur Konfiguration der Einspeicher-Logik

Die Konfigurations-Logik besteht aus acht 8Bit-Registern, die alle schreib- und lesbar sind.
Nach –RESET sind alle Register 0.
Nach jeder Konfiguration der Einspeicher-Logik müssen die Datenregister der Zählerbausteine für Daten-Register 0 und Daten-Register 1 einmal ausgelesen werden, denn durch die Konfiguration können Einspeicherimpulse ausgelöst worden sein, die die Zählerbausteine für weitere Einspeichervorgänge sperren.

Beispiel
Adresse für das Register zur Konfiguration der Eingänge E1 bis E4
(Annahme: DIP-Schalter = $40):
Register-Adresse = (($40 shift right 1) * $200) + $180 + $08 = $4188
Dies ist die Wort-Adresse, in deren Low Byte sich das 8-Bit-Register befindet.

Übersicht
Basisadresse Konfigurations-Logik (BAK): BA + $180

Adresse-Offset	Funktion
BAK + $00	Einspeicher-Konfiguration X1
BAK + $02	Einspeicher-Konfiguration X2
BAK + $04	Einspeicher-Konfiguration X3
BAK + $06	Einspeicher-Konfiguration X4
BAK + $08	Konfiguration der Eingangs-Pegel E1 bis E4 Konfiguration Meßsystem Eingang X1,X2
BAK + $0A	Konfiguration Meßsystem Eingang X2,X3,X4
BAK + $0C	Einspeicherquelle
BAK + $0E	I²C-Bus-Ansteuerung, Interrupt enable/ disable (1 Bit), Überschreitung Eingangssignalpegel Meßsysteme

$00 bis $06: Konfigurations-Register für die externen Eingänge E1 bis E4 (Schreib- und Lesezugriff)

Defaultwert nach RESET: 0

Adresse $00	Konfigurations-Register X1
Adresse $02	Konfigurations-Register X2
Adresse $04	Konfigurations-Register X3
Adresse $06	Konfigurations-Register X4

Die Bedeutung der einzelnen Bits ist für jede Achse gleich:

Bit	Funktion
D0	0 = E1 ist gesperrt für diese Achse 1 = E1 speichert diese Achse ein
D1	0 = E1 speichert Daten-Register 0 dieser Achse ein 1 = E1 speichert Daten-Register 1 dieser Achse ein
D2	0 = E2 ist gesperrt für diese Achse 1 = E2 speichert diese Achse ein
D3	0 = E2 speichert Daten-Register 0 dieser Achse ein 1 = E2 speichert Daten-Register 1 dieser Achse ein
D4	0 = E3 ist gesperrt für diese Achse 1 = E3 speichert diese Achse ein
D5	0 = E3 speichert Daten-Regiser 0 dieser Achse ein 1 = E3 speichert Daten-Register 1 dieser Achse ein
D6	0 = E4 ist gesperrt für diese Achse 1 = E4 speichert diese Achse ein
D7	0 = E4 speichert Daten-Register 0 dieser Achse ein 1 = E4 speichert Daten-Register 1 dieser Achse ein

$08: Konfigurations-Register für die Eingangs-Pegel E1 bis E4 sowie Konfiguration Meßsystem-Eingang X1, X2 (Schreib- und Lesezugriff)

Defaultwert nach RESET: 0

Bit	Funktion
D0	0 = E1 ist low-aktiv 1 = E1 ist high-aktiv
D1	0 = E2 ist low-aktiv 1 = E2 ist high-aktiv
D2	0 = E3 ist low-aktiv 1 = E3 ist high-aktiv
D3	0 = E4 ist low-aktiv 1 = E4 ist high-aktiv
D4	0 = X1: 1Vss-Eingang 1 = X1: 11µA-Eingang
D5	**Auf 1 setzen (zwingend!)**
D6	0= X1: Eingangsverstärker schnell 1 Vss: 500 kHz; 11 µAss: 175 kHz 1= X1: Eingangsverstärker langsam (33 kHz)
D7	0 = X2: 1Vss-Eingang 1 = X2: 11µA-Eingang

$0A: Konfigurations-Register für Meßsystem-Eingänge X2, X3, X4 (Schreib- und Lesezugriff)

Defaultwert nach RESET: 0

Bit	Funktion
D0	**auf 1 setzen (zwingend!)**
D1	0 = X2: Eingangsverstärker schnell 1 Vss: 500 kHz; 11 µAss: 175 kHz 1 = X2: Eingangsverstärker langsam (33 kHz)
D2	0 = X3: 1Vss-Eingang 1 = X3: 11µA-Eingang
D3	**auf 1 setzen (zwingend!)**
D4	0 = X3: Eingangsverstärker schnell 1 Vss: 500 kHz; 11 µAss: 175 kHz 1 = X3: Eingangsverstärker langsam (33 kHz)
D5	0 = X4: 1Vss-Eingang 1 = X4: 11µA-Eingang
D6	**auf 1 setzen (zwingend!)**
D7	0 = X4: Eingangsverstärker schnell 1 Vss: 500 kHz; 11 µAss: 175 kHz 1 = X4: Eingangsverstärker langsam (33 kHz)

$0C: Register zum Anzeigen der Einspeicherquelle (Lesezugriff)
Defaultwert nach RESET: 0

Nach einem Einspeichervorgang wird in diesem Register angezeigt, von welcher Quelle der Einspeicherimpuls gekommen ist. Mögliche Quellen sind E1-E4, SYNC0 und SYNC1.

Bit	Funktion
D0	1 = Einspeicherquelle war E1
D1	1 = Einspeicherquelle war E2
D2	1 = Einspeicherquelle war E3
D3	1 = Einspeicherquelle war E4
D4	1 = Einspeicherquelle war SYNC0
D5	1 = Einspeicherquelle war SYNC1
D6 D7	ohne Funktion

$0C: Register zum Anzeigen der Einspeicherquelle (Schreibzugriff)
Defaultwert nach RESET: 0

Das entsprechende gesetzte Bit kann mit einem Schreibzugriff mit Wert "1" wieder gelöscht werden.

Bit	Funktion
D0	1 = Löschen Bit0
D1	1 = Löschen Bit1
D2	1 = Löschen Bit2
D3	1 = Löschen Bit3
D4	1 = Löschen Bit4
D5	1 = Löschen Bit5
D6 D7	ohne Funktion

$0E: Konfigurations-Register für I²C-Bus, Interrupt enable/ disable, Überschreitung Eingangssignalpegel Meßsysteme (Lesezugriff)
Defaultwert nach RESET: 0

Bit	Funktion
D0	Pin Scan serieller Clock Output I²C-Bus
D1	Pin Scan seriellen Data Output I²C-Bus
D2	Pin Scan seriellen Data Input I²C-Bus
D3	Pin Scan Interrupt disable
D4	0 = Überschreitung Eingangspegel Meßsystem X1 (Pin Scan)
D5	0 = Überschreitung Eingangspegel Meßsystem X2 (Pin Scan)
D6	0 = Überschreitung Eingangspegel Meßsystem X3 (Pin Scan)
D7	0 = Überschreitung Eingangspegel Meßsystem X4 (Pin Scan)

$0E: Konfigurations-Register für I²C-Bus, Interrupt enable/ disable, Überschreitung Eingangssignalpegel Meßsysteme (Schreibzugriff)
Defaultwert nach RESET: 0

Bit	Funktion
D0	0 = Setzt seriellen Clock Output I²C-Bus 1 = Löscht seriellen Clock Output I²C-Bus
D1	0 = Setzt seriellen Data Output I²C-Bus 1 = Löscht seriellen Data Output I²C-Bus
D2	keine Funktion
D3	0 = Interrupt disable 1 = Interrupt enable
D4 D5 D6 D7	ohne Funktion

8. Programmierung

Die Programmierung einer IK 342 wird in dieser Beschreibung in „BORLAND C++"-Beispielen gezeigt. Die Programme wurden auf einem Industrie-Rechner (von Firma ROTEC, D-76411 Rastatt) mit einer INTEL 486 CPU (DOS-Version 6.0), VMEbus-Interface und BORLAND C++-Compiler (Version 4.5) erstellt und getestet.
Folgende Dateien auf der mitgelieferten Diskette dienen zur Anpassung des ISA-Bus an den VMEbus:

- VMEROTEC.H und
- VMEINIT.C

Die Daten- und Funktionsdefinitionen in diesen Dateien werden nicht weiter erläutert, da sie keine Funktionen der IK 342 beschreiben.

8.1 Grundfunktionen

Die im folgenden beschriebenen Programme finden Sie auf der mitgelieferten Diskette unter dem Verzeichnis **SAMPLE1**.

Die Dateien

- IK342_0.H und
- IK342_0.C

enthalten die wichtigsten Daten- und Funktionsdefinitionen, die Sie bei der Arbeit mit der IK 342 benötigen.

Die wichtigsten Funktionen in den folgenden Beispielen:

VmeInit()
Initialisiert den VMEbus. Diese Funktion ist angepaßt an den Industrie-Rechner von Firma ROTEC. Für die Hardware, die Sie verwenden, müssen Sie eine eigene Funktion zum Initialisieren schreiben.

WriteRegister
Schreibt einen Wert in ein 16 Bit breites Register eines Zählerbausteins.

ReadRegister
Liest einen Wert aus einem 16 Bit breiten Register eines Zählerbausteins.

SoftLatch_0 und **SoftLatch_1**
Speichern einen Zählerstand im Daten-Register 0 oder Daten-Register 1.

CountValueLatched
Prüft, ob der Meßwert gespeichert wurde.

PollForLatched
Wiederholt die Abfrage nach einem Meßwert, bis ein Meßwert gespeichert wurde.

ReadCountValue_32
Liest einen 32 Bit breiten Meßwert aus einem Zählerbaustein.

ReadCountValue_48
Liest einen 48 Bit breiten Meßwert aus einem Zählerbaustein.

SAMPLE32.EXE
Das Programm SAMPLE32.EXE zeigt eine einfache Anwendung zum Lesen eines 32 Bit breiten Meßwerts.

Quellcode: SAMPLE32.C
IK342_0.C
VMEINIT.C

Header-Dateien: SAMPLE.H
IK342_0.H
VMEROTEC.H

SAMPLE48.EXE
Das Programm SAMPLE48.EXEzeigt eine einfache Anwendung zum Lesen eines 48 Bit breiten Meßwerts.

Quellcode: SAMPLE48.C
IK342_0.C
VMEINIT.C

Header-Dateien: SAMPLE.H
IK342_0.H
VMEROTEC.H

Die Header-Datei IK342_0.H

```
/*----------------------------IK342_0.H-------------------------------

  DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

  Header File for the Basic Functions of the IK342

  V 1.00
  November 1998
  ----------------------------------------------------------------------*/


#define CLS printf("\x1B[2J")


/*----------------------------------------------------------------------
  Defines for register addresses.
  ----------------------------------------------------------------------*/
#define INITIALIZING_REGISTER   0x24
#define CONTROL_REGISTER_1      0x20
#define CONTROL_REGISTER_2      0x04

//Configuration- and Id-Register
#define ENLATCHX1  0x00
#define ENLATCHX2  0x02
#define ENLATCHX3  0x04
#define ENLATCHX4  0x06
#define LATCHPOL   0x08
#define MSINPUT    0x0A
#define LATCHSTAT  0x0C
#define INTREG     0x0E


/*----------------------------------------------------------------------
  Macro to calculate the IK address.
  ----------------------------------------------------------------------*/
#define CALCULATE_IK_BASE_ADDRESS(switch) \
 ((unsigned short)(switch >> 1)*0x0200) | 0x8000

/*----------------------------------------------------------------------
  Macro to switch VME to A16 memory space.
  ROTEC specific code.
  ----------------------------------------------------------------------*/
#define SWITCH_VME_TO_A16_ADDRESS_SPACE(switch)     \
 outport (ADR_REG,\
 ((((unsigned short)(switch >> 1)*0x0200)\
 & 0x8000) >> 8) | 0xFC00)


/*Definitions of functions*/
```

```
void WriteRegister (unsigned short, unsigned char,
   unsigned short, unsigned short);

unsigned short ReadRegister (unsigned short, unsigned char,
   unsigned short);

void SoftLatch_0 (unsigned short, unsigned char);

void SoftLatch_1 (unsigned short, unsigned char);

unsigned char CountValueLatched (unsigned short, unsigned char,
   unsigned char);

void PollForLatched (unsigned short,unsigned char,
   unsigned char);

long ReadCountValue_32 (unsigned short, unsigned char,
   unsigned char);

double ReadCountValue_48 (unsigned short, unsigned char,
   unsigned char);

unsigned char ReadConfReg (unsigned short usBaseAddress,
   unsigned short usRegisterAddress);

void WriteConfReg (unsigned short usBaseAddress, unsigned short
usRegisterAddress,
   unsigned char usDatum);
```

Die Funktionen in IK342_0.C

```
/*-----------------------------IK342_0.C-------------------------------

  DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

  Driver Unit for IK342 (Basic Functions)

  V 1.00
  November 1998
 ---------------------------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342_0.h"
#include "vmerotec.h"

/*----------------------Functions-----------------------------------*/
/*---------------------------------------------------------------------
                          WriteRegister
 ---------------------------------------------------------------------
 This function writes a value in a 16-bit
 register of a counter.
 -------------------------Parameters----------------------------------
 usBaseAddress       base address of the IK 342
 ucAxis                axis select - axis 1 to axis 4
 usRegisterAddress   address of the register
 usDatum             value to write to the register address
 usPortAddress       port address in which <usDatum> is written
 ---------------------------------------------------------------------*/
void WriteRegister (unsigned short usBaseAddress, unsigned char ucAxis,
               unsigned short usRegisterAddress,
               unsigned short usDatum)
{
unsigned short usPortAddress;

switch (ucAxis)
       {
       case 1:
       usPortAddress = usBaseAddress + 0x0080 + usRegisterAddress;
       break;
       case 2:
       usPortAddress = usBaseAddress + 0x00C0 + usRegisterAddress;
```

```
        break;
        case 3:
        usPortAddress = usBaseAddress + 0x0100 + usRegisterAddress;
        break;
        case 4:
        usPortAddress = usBaseAddress + 0x0140 + usRegisterAddress;
        break;
        default:
        printf ("Wrong axis in function <WriteRegister>");
        }

   /* Write <usDatum> to the counter */
outpw (usPortAddress, usDatum);
}

/*---------------------------------------------------------------------
                        ReadRegister
 ---------------------------------------------------------------------
 This function reads a value from a 16-bit
 register of a counter.
 -----------------------Parameters-----------------------------------
 usBaseAddress        basic address A0 to A9 of the IK 342
 ucAxis               axis select - axis 1 or axis 2
 usRegisterAddress    address of the register
 usPortAddress        port address in which <usDatum> is written
 ---------------------------------------------------------------------*/
unsigned short ReadRegister (unsigned short usBaseAddress,
               unsigned char ucAxis, unsigned short usRegisterAddress)
{
unsigned short usPortAddress;

/* Calculate port usRegisterAddress */
switch (ucAxis)
        {
        case 1:
        usPortAddress = usBaseAddress + 0x0080 + usRegisterAddress;
        break;
        case 2:
        usPortAddress = usBaseAddress + 0x00C0 + usRegisterAddress;
        break;
        case 3:
        usPortAddress = usBaseAddress + 0x0100 + usRegisterAddress;
        break;
        case 4:
        usPortAddress = usBaseAddress + 0x0140 + usRegisterAddress;
        break;
        default:
        printf ("Wrong axis in function <WriteRegister>");
        }
   /* Read <usDatum> from the counter */
return(inpw(usPortAddress));
}
/*---------------------------------------------------------------------
                          SoftLatch_0
 ---------------------------------------------------------------------
 This function reads the measured value and stores
 it in data register 0.
 ---------------------------------------------------------------------*/
void SoftLatch_0 (unsigned short usBaseAddress, unsigned char ucAxis)
{
WriteRegister (usBaseAddress, ucAxis, 0x20, 0x0001);
}


/*---------------------------------------------------------------------
                          SoftLatch_1
 ---------------------------------------------------------------------
 This function reads the measured value and stores
 it in data register 1.
 ---------------------------------------------------------------------*/
void SoftLatch_1 (unsigned short usBaseAddress, unsigned char ucAxis)
{
WriteRegister (usBaseAddress, ucAxis, 0x20, 0x0002);
}
```

```
/*-------------------------------------------------------------------
                        CountValueLatched
 -------------------------------------------------------------------
 This function checks whether a measured value is
 latched in data register 0 or 1.
 -------------------------------------------------------------------*/

unsigned char CountValueLatched (unsigned short usBaseAddress,
                  unsigned char ucAxis, unsigned char ucRegister)
{
unsigned char result;
switch (ucRegister)
       {
       case 0:
       result = (unsigned char)
                (ReadRegister (usBaseAddress, ucAxis, 0x020) & 0x0001);
       break;
       case 1:
       result = (unsigned char)
                (ReadRegister (usBaseAddress, ucAxis, 0x020) & 0x0002);
       break;
       }
return (result);
}


/*-------------------------------------------------------------------
                        PollForLatched
 -------------------------------------------------------------------
 This function polls until a measured value is
 latched in data register 0 or 1.
 -------------------------------------------------------------------*/

void PollForLatched (unsigned short usBaseAddress,
                     unsigned char ucAxis,
                     unsigned char ucRegister)
{
switch (ucRegister)
       {
       case 0:
       while (CountValueLatched (usBaseAddress, ucAxis, 0) == 0)
       ;
       break;

       case 1:
       while (CountValueLatched (usBaseAddress, ucAxis, 1) == 0)
       ;
       break;
       }
}

/*-------------------------------------------------------------------
                      ReadCountValue_32
 -------------------------------------------------------------------
 This function reads 32 bits of a measured value.
 -------------------------------------------------------------------*/
long ReadCountValue_32 (unsigned short usBaseAddress,
                  unsigned char ucAxis, unsigned char ucRegister)
{
union     mapper
        {
        long field0;
        unsigned short field1[2];
        }buffer;

switch (ucRegister)
       {
       case 0:
       buffer.field1[0] = ReadRegister (usBaseAddress, ucAxis, 0x3c);
       buffer.field1[1] = ReadRegister (usBaseAddress, ucAxis, 0x38);
       break;
       case 1:
       buffer.field1[0] = ReadRegister (usBaseAddress, ucAxis, 0x30);
```

```
        buffer.field1[1] = ReadRegister (usBaseAddress, ucAxis, 0x2c);
        break;
        }
return (buffer.field0);
}

/*---------------------------------------------------------------------
                    ReadCountValue_48
 ---------------------------------------------------------------------
 This function reads 48 bits of a measured value.
 ---------------------------------------------------------------------*/
double ReadCountValue_48 (unsigned short usBaseAddress,
                  unsigned char ucAxis, unsigned char ucRegister)
{
unsigned short usField[3];
double count_value48;

switch (ucRegister)
        {
        case 0:
        usField[0] = ReadRegister (usBaseAddress, ucAxis, 0x3C);
        usField[1] = ReadRegister (usBaseAddress, ucAxis, 0x38);
        usField[2] = ReadRegister (usBaseAddress, ucAxis, 0x34);
        break;
        case 1:
        usField[0] = ReadRegister (usBaseAddress, ucAxis, 0x30);
        usField[1] = ReadRegister (usBaseAddress, ucAxis, 0x2C);
        usField[2] = ReadRegister (usBaseAddress, ucAxis, 0x28);
        break;
        }

if (usField[2] & 0x8000)
        count_value48 = (double)((usField[0] - 65535.0) +
        65536.0*(usField[1]-65535.0)+
        4294967296.0*(usField[2]-65535.0)-1);
        else
        count_value48 = (double)(usField[0] +
        65536.0*usField[1] +
        4294967296.0*usField[2]);


return (count_value48);
}


/*---------------------------------------------------------------------
-----
                        ReadConfReg
 ---------------------------------------------------------------------
-----
 This function reads one configuration-Register from IK342
 ----------------------Parameters-------------------------------------
-----
 ---------------------------------------------------------------------
----*/
unsigned char ReadConfReg(unsigned short usBaseAddress,
                                                    unsigned short
usRegisterAddress)
{
   unsigned short usPortAddress;

   usPortAddress = usBaseAddress + 0x0180 + usRegisterAddress;
   return(inp(usPortAddress+1));
}

/*---------------------------------------------------------------------
-----
                        WriteConfReg
 ---------------------------------------------------------------------
-----
 This function writes the config-register from IK342
 ----------------------Parameters-------------------------------------
-----
```

```
 ------------------------------------------------------------------------
----*/
void WriteConfReg(unsigned short usBaseAddress, unsigned short
usRegisterAddress,
                                    unsigned char usDatum)
{
   unsigned short usPortAddress;

   usPortAddress = usBaseAddress + 0x0180 + usRegisterAddress;
   outp(usPortAddress+1, usDatum);         // Write <usDatum> to the
IK342
}
```

Die Header-Datei SAMPLE.H

```
/*---------------------------SAMPLE.H-------------------------------

  DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

  Header File for SAMPLE32.C and SAMPLE48.C of the IK342 examples

  V 1.00
  xxxx 199x
 ------------------------------------------------------------------*/

/*------------------------------------------------------------------
 Setting of the DIP switch on board of the IK 342
 ------------------------------------------------------------------*/

 #define DIP_SWITCH    0x40
```

Das Programm-Beispiel SAMPLE32.C

```
/*---------------------------SAMPLE32.C-----------------------------

 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

 A simple program for the IK 342 to display
 four axes. Measured value with 32 bits.

 V 1.00
 November 1998

 Project files:        IK342_0.C, SAMPLE32.C, VMEINIT.C
 Include files:        IK342_0.H, SAMPLE.H, VMEROTEC.H
 ------------------------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342_0.h"
#include "sample.h"
#include "vmerotec.h"

int main()
{
double dCountValue1, dCountValue2, dCountValue3, dCountValue4;
unsigned short usBaseAddress;

CLS;

  /*Initialize VME interface (ROTEC specific functions)*/
VmeInit();
gotoxy(1, 20);
puts("VMEbus initialized. Press any key!");
getch();
CLS;

usBaseAddress = CALCULATE_IK_BASE_ADDRESS(DIP_SWITCH);
SWITCH_VME_TO_A16_ADDRESS_SPACE(DIP_SWITCH);
```

```
   /* Set 1Vss 500kHZ */
WriteConfReg (usBaseAddress, LATCHPOL, 0x20);
WriteConfReg (usBaseAddress, MSINPUT,  0x49);


   /* Initialize the board in interpolation mode,
   axis 1 */
WriteRegister (usBaseAddress, 1, INITIALIZING_REGISTER, 0x0001);
   /* Initialize the board in interpolation mode,
   axis 2 */
WriteRegister (usBaseAddress, 2, INITIALIZING_REGISTER, 0x0001);
   /* Initialize the board in interpolation mode,
   axis 3 */
WriteRegister (usBaseAddress, 3, INITIALIZING_REGISTER, 0x0001);
   /* Initialize the board in interpolation mode,
   axis 4 */
WriteRegister (usBaseAddress, 4, INITIALIZING_REGISTER, 0x0001);

   /* Reset error bit, start counter, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_1, 0x0048);

   /* Write to control register 2, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_2, 0x0008);
   /* Write to control register 2, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_2, 0x0008);
   /* Write to control register 2, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_2, 0x0008);
   /* Write to control register 2, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_2, 0x0008);

   /*Cursor off*/
_setcursortype(_NOCURSOR);

while(!kbhit())
   {
                  /* Software latch in register 0, axis 1 */
   SoftLatch_0 (usBaseAddress, 1);
                  /* Software latch in register 0, axis 2 */
   SoftLatch_0 (usBaseAddress, 2);
                  /* Software latch in register 0, axis 3 */
   SoftLatch_0 (usBaseAddress, 3);
                  /* Software latch in register 0, axis 4 */
   SoftLatch_0 (usBaseAddress, 4);
                  /* Poll whether latched in axis 1 */
   PollForLatched (usBaseAddress, 1, 0);
         /* Read axis 1 */
   dCountValue1 = (double)ReadCountValue_32 (usBaseAddress, 1, 0);
                  /* Poll whether latched in axis 2 */
   PollForLatched (usBaseAddress, 2, 0);
         /* Read axis 2 */
   dCountValue2 = (double)ReadCountValue_32 (usBaseAddress, 2, 0);
                  /* Poll whether latched in axis 3 */
   PollForLatched (usBaseAddress, 3, 0);
         /* Read axis 3 */
   dCountValue3 = (double)ReadCountValue_32 (usBaseAddress, 3, 0);
                  /* Poll whether latched in axis 4 */
   PollForLatched (usBaseAddress, 4, 0);
         /* Read axis 4 */
   dCountValue4 = (double)ReadCountValue_32 (usBaseAddress, 4, 0);
          /* Display measured values */
  gotoxy(5,5);
   printf("%16.4f\t%16.4f", dCountValue1*0.02/1024,
                                    dCountValue2*0.02/1024);
  gotoxy(5,10);
   printf("%16.4f\t%16.4f", dCountValue3*0.02/1024,
                                    dCountValue4*0.02/1024);
   }
```

```
   /*Cursor on*/
_setcursortype (_NORMALCURSOR);
return (0);
}
```

Das Programm-Beispiel SAMPLE48.C

```
/*--------------------------SAMPLE48.C------------------------------

 DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

 A simple program for the IK 342 to display
 four axes. Measured value with 48 bits.

 V 1.00
 November 1998

 Project files:        IK342_0.C, SAMPLE48.C, VMEINIT.C
 Include files:        IK342_0.H, SAMPLE.H, VMEROTEC.H
 ------------------------------------------------------------------*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342_0.h"
#include "sample.h"
#include "vmerotec.h"

int main()
{
double dCountValue1, dCountValue2, dCountValue3, dCountValue4;
unsigned short usBaseAddress;

CLS;

  /*Initialize VME interface (ROTEC specific functions)*/
VmeInit();
gotoxy(1, 20);
puts("VMEbus initialized. Press any key!");
getch();
CLS;

usBaseAddress = CALCULATE_IK_BASE_ADDRESS(DIP_SWITCH);
SWITCH_VME_TO_A16_ADDRESS_SPACE(DIP_SWITCH);


   /* Set 1Vss 500kHZ, all axes */
WriteConfReg (usBaseAddress, LATCHPOL, 0x20);
WriteConfReg (usBaseAddress, MSINPUT,  0x49);


   /* Initialise the board in interpolation mode,
   axis 1 */
WriteRegister (usBaseAddress, 1, INITIALIZING_REGISTER, 0x0041);
   /* Initialise the board in interpolation mode,
   axis 2 */
WriteRegister (usBaseAddress, 2, INITIALIZING_REGISTER, 0x0041);
   /* Initialise the board in interpolation mode,
   axis 3 */
WriteRegister (usBaseAddress, 3, INITIALIZING_REGISTER, 0x0041);
   /* Initialise the board in interpolation mode,
   axis 4 */
WriteRegister (usBaseAddress, 4, INITIALIZING_REGISTER, 0x0041);

   /* Reset error bit, start counter, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_1, 0x0048);
   /* Reset error bit, start counter, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_1, 0x0048);

   /* Write to control register 2, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_2, 0x0008);
```

```
   /* Write to control register 2, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_2, 0x0008);
   /* Write to control register 2, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_2, 0x0008);
   /* Write to control register 2, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_2, 0x0008);

   /*Cursor off*/
_setcursortype(_NOCURSOR);

while(!kbhit())
   {
          /* Software latch in register 0, axis 1 */
   SoftLatch_0 (usBaseAddress, 1);
          /* Software latch in register 0, axis 2 */
   SoftLatch_0 (usBaseAddress, 2);
          /* Software latch in register 0, axis 3 */
   SoftLatch_0 (usBaseAddress, 3);
          /* Software latch in register 0, axis 4 */
   SoftLatch_0 (usBaseAddress, 4);

          /* Poll whether latched in axis 1 */
   PollForLatched (usBaseAddress, 1, 0);
      /* Read axis 1 */
   dCountValue1 = ReadCountValue_48 (usBaseAddress, 1, 0);
          /* Poll whether latched in axis 2 */
   PollForLatched (usBaseAddress, 2, 0);
      /* Read axis 2 */
   dCountValue2 = ReadCountValue_48 (usBaseAddress, 2, 0);
          /* Poll whether latched in axis 3 */
   PollForLatched (usBaseAddress, 3, 0);
      /* Read axis 3 */
   dCountValue3 = ReadCountValue_48 (usBaseAddress, 3, 0);
          /* Poll whether latched in axis 4 */
   PollForLatched (usBaseAddress, 4, 0);
      /* Read axis 4 */
   dCountValue4 = ReadCountValue_48 (usBaseAddress, 4, 0);

        /* Display measured values */
   gotoxy(5,5);
   printf("%16.4f\t%16.4f", dCountValue1*0.02/1024,
                            dCountValue2*0.02/1024);
   gotoxy(5,10);
   printf("%16.4f\t%16.4f", dCountValue3*0.02/1024,
                            dCountValue4*0.02/1024);
   }

   /*Cursor on*/
_setcursortype (_NORMALCURSOR);

return (0);
}
```

8.2 Funktionen für ein RAM-Speicher-Modell

Beispiele mit dem RAM-Speicher-Modell befinden sich im Verzeichnis **SAMPLE2**.
Die verwendeten Datenstrukturen und Funktionen sind in folgenden Dateien deklariert und definiert:

IK342.H:	In dieser Header-Datei können Sie die Adressen von bis zu 16 IK 342 festlegen.
IK342_1.H:	Datenstrukturen für ein RAM-Speicher-Modell der Register der IK 342 und Funktionsdeklarationen für die Funktionen in den Dateien IK342_1.CPP, IIC.CPP und POTI_1.CPP
IK342_1.CPP:	Basis-Funktionen für die IK 342
IIC.CPP:	Funktionen zum Datentransfer über den I²C-Bus
POTI_1.CPP:	Funktionen zum Einstellen der elektronischen Potentiometer

In der Datei IK342_1.H wird mit Hilfe von Datenstrukturen ein RAM-Speicher-Modell der Register der IK 342 aufgebaut. Die Daten des RAM-Speicher-Modells werden mit Hilfe der Prozeduren **InitHandler** und **CommHandler** in die Register der IK 342 geschrieben.

POTIS.EXE
Das Programm POTIS.EXE zeigt, wie Sie per Software die elektronischen Potentiometer der IK 342 über den I²C-Bus einstellen können.

Quellcode: POTIS.CPP
IK342_1.CPP
IIC.CPP
POTI_1.CPP
VMEINIT.C

Header-Dateien: IK342.H
IK342_1.H
VMEROTEC.H

DISPLAY.EXE
Das Programm DISPLAY.EXE zeigt die Inhalte der Daten-Register der IK 342.

Quellcode: DISPLAY.CPP
IK342_1.CPP
IIC.CPP
VMEINIT.C

Header-Dateien: IK342.H
IK342_1.H
VMEROTEC.H

9 Technische Daten IK 342

Mechanische Kennwerte

Abmessungen	Doppel-Europakarten-Format, Größe B Außenabmessungen: 262 mm x 187 mm x 20 mm
Arbeitstemperatur **Lagertemperatur**	0 °C bis 55 °C –30 °C bis 70 °C

Elektrische Kennwerte

VMEbus-Spezifikation	ANSI/IEEE STD1014–1987, IEC 821 und 297 Double Height Board mit J1-Stecker 1 Slot Adreßraum A16: Slave D08 (EO) Slave D16 Interrupter: D08 (O) ROAK
Adressen	512 Bytes im Adreßraum A16 (128 Basisadressen) Adreßraum über DIP-Schalter einstellbar
Eingänge/Ausgänge	
Meßsystem-Eingänge	X1 bis X4: Achse 1 bis Achse 4, Sub-D-Anschluß 15polig Sinus-Signale: 1 V_{SS}, Eingangs-Frequenz: umschaltbar 33/500 kHz 11µA, Eingangs Frequenz: umschaltbar 33/175 kHz
Meßsystem-Ausgänge	Option: Adapter mit vier Ausgängen, Sub-D-Anschluß 9polig, Sinus-Signale 11 μA_{SS}
externe Abruf-Signale	X41: Sub-D-Anschluß 9polig vier Eingänge E1 bis E4: wahlweise low- oder high-aktiv U_{high}: 3,5 V bis 32 V U_{low}: –20 V bis 1 V zwei Ausgänge –CASC0, –CASC1: Signalpegel: TTL
Signal-Interpolation	1 024fach
Abgleich der Meßsystem-Signale	Phase und Amplitude über elektronische Potentiometer Offset über Register in den Zählerbausteinen
Datenregister für Meßwerte	48 Bit, wobei für den Meßwert nur 42 Bit genutzt werden
Interrupts	–INT1 bis –INT7
Leistungsaufnahme	ca. 2,5 W (ohne Meßgeräte)

10 Prinzip-Schaltbild der Abruf-Wege in den Zählerbausteinen

Das folgende Prinzip-Schaltbild zeigt:
- die Wirkungsweise der Abruf-Signale auf die Daten-Register
- die Funktion der einzelnen Bits des Freigabe-Registers für den Meßwert-Abruf
- die Achs-Kaskadierung mit den zugehörigen Bits des gleichnamigen Registers
- das Register zur I²C-Bus-Ansteuerung

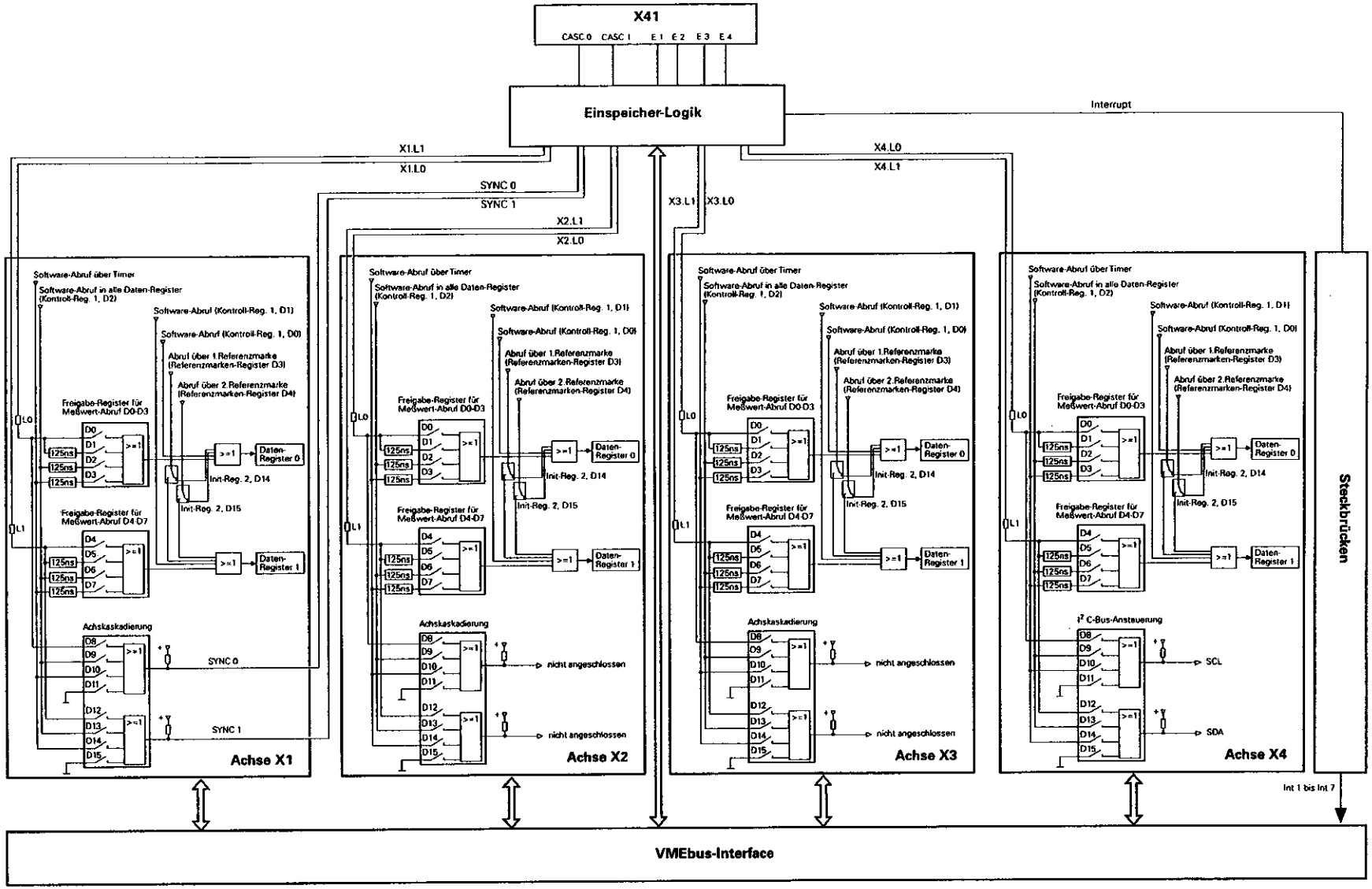
Die Verzögerungsglieder mit 125 ns Verzögerungszeit werden beim synchronen Einspeichern von mehreren Achsen zum Ausgleich der Signallaufzeit zwischen den Achsen genutzt. Sie sollten beim synchronen Einspeichern für Achse 1 einen Signalweg über ein Verzögerungsglied und für alle weitere Achsen ohne Verzögerungsglied wählen. Da für alle Achsen gleiche Zählerbausteine verwendet werden, sind in allen Achsen Verzögerungsglieder vorhanden, d.h. nicht alle Signalweg-Kombinationen ergeben einen sinnvollen Abruf!

Funktionen der Einspeicher-Logik

Gleichzeitiges Einspeichern von Daten-Registern über mehrere Achsen
Der Ausgang SYNC0 des Zählerbausteins der Achse 1 wird über die Einspeicher-Logik mit den Eingängen X1.L0 bis X4.L0 und der Ausgang SYNC1 mit den Eingängen X1.L1 bis X4.L1 verbunden (intern verdrahtet). Damit können alle Achsen gleichzeitig eingespeichert werden.

Gleichzeitiges Einspeichern über mehrere Karten
Der Ausgang SYNC0 des Zählerbausteins der Achse 1 ist über die Einspeicher-Logik zum Stecker X41 mit dem Ausgang CASC0 und der Ausgang SYNC1 mit dem Ausgang CASC1 verbunden (intern verdrahtet). Damit können die Zählerbausteine mehrerer Karten IK 342 gleichzeitig eingespeichert werden.

Konfigurierbare Eingänge
Die Eingänge E1 bis E4 werden über die Einspeicher-Logik den Einspeicher-Eingängen X1.L0 bis X4.L0 und X1.L1 bis X4.L1 zugeordnet (programmierbar über Register). Außerdem kann festgelegt werden, ob ein Signal an den Eingängen E1 bis E4 einen Interrupt erzeugen soll.


X41
CASC 0
CASC 1
E 1
E 2
E 3
E 4
Einspeicher-Logik
Interrupt
X1.L1
X1.L0
SYNC 0
SYNC 1
X2.L1
X2.L0
X3.L1
X3.L0
X4.L0
X4.L1
Steckbrücken
Int 1 bis Int 7
VMEbus-Interface
Achse X1
Achse X2
Achse X3
Achse X4
Software-Abruf über Timer
Software-Abruf in alle Daten-Register (Kontroll-Reg. 1, D2)
Software-Abruf (Kontroll-Reg. 1, D1)
Software-Abruf (Kontroll-Reg. 1, D0)
Abruf über 1.Referenzmarke (Referenzmarken-Register D3)
Abruf über 2.Referenzmarke (Referenzmarken-Register D4)
Freigabe-Register für Meßwert-Abruf D0-D3
Freigabe-Register für Meßwert-Abruf D4-D7
Daten-Register 0
Daten-Register 1
Init-Reg. 2, D14
Init-Reg. 2, D15
Achskaskadierung
I²C-Bus-Ansteuerung
nicht angeschlossen
SCL
SDA
125ns
>=1