RSF Elektronik

BENUTZERHANDBUCH
IFC 430R

PC-EINSTECKKARTE

12/2017

INHALTSVERZEICHNIS

1. ALLGEMEINES

Auf die Funktion des PCI-Bus kann hier nicht eingegangen werden.

1.1 Wichtige Hinweise

- **Gefahr für Bauteile bei Nichtbeachtung dieser Hinweise**
- **Bitte beachten Sie die Vorsichtsmaßnahmen bei der Handhabung elektrostatisch entladungsgefährdeter Bauelemente (ESD) nach DIN EN 100 015**
- **Als Transport-Verpackung nur antistatisches Material verwenden.**
- **Beim Einbau ist eine ausreichende Erdung am Arbeitsplatz sicherzustellen**
- **Keine Steckverbindungen herstellen oder lösen bei eingeschalteter Stromversorgung**
- **Vor dem Einstecken der IFC 430R Jumper-Stellungen auf richtige Betriebs-Spannungen der Encoder überprüfen**

1.2 Anwendung

Die IFC 430R ist eine PC-Einsteckkarte mit PCI-Interface und dient zur Erfassung und Auswertung von Encoder-Signalen. Sie kann aber auch für alle anderen gängigen Zählfunktionen (Ereigniszähler, Frequenzzähler usw.) eingesetzt werden.

1.3 Lieferumfang

- PCI-Interfacekarte IFC 430R
- Diskette mit Demoprogramm und Treiber-Software
- Benutzerhandbuch

2. TECHNISCHE DATEN

2.1 Mechanik und Umgebung

- Abmessungen (Leiterplattenmaße): ca. 120 x 92 mm, ein Slot breit
- maximal zulässige Umgebungstemperatur: +40°C
- eine D-Sub Buchsenleiste 25-polig für Zählereingänge
- eine D-Sub Buchsenleiste 9-polige für I/O-Signale

2.2 PCI-Bus

- PCI-Stecker 5 V 32-Bit 2 x 60 Pin
- Target Interface (Slave) nach Spezifikation Rev. 2.1 (PLX-Baustein PCI 9052)
- Bus-Taktfrequenz max. 40 MHz
- Stromaufnahme an +5 V ca. 0,5 A, ohne Messgeräte
- Spannungsversorgung der Messgeräte mit +5 V oder +12 V aus PCI-Versorgung
  (Stromaufnahme abhängig von den angeschlossenen Messgeräten)

2.3 Zähler-Interface (X1)

- neun RS 422- bzw. TTL-Eingänge für drei Encoder mit Quadratur-Signalen und Referenzmarke
- Eingangsfrequenz max.: 4 MHz bei Differenzsignalen (Line Driver RS 422 Standard)
- 2 MHz bei Single-end-Signalen
- ein TTL-Eingang für Störsignal-Überwachung
- separate Stromversorgungsleitungen für jeden Encoder (max. 0,5 A. pro Encoder)

2.4 I/O-Interface (X2)

- sechs Eingänge (3 bis 30 V) als Referenz-Impulssperren bzw. asynchrone Einspeichersignale verwendbar
- ein Eingang (3 bis 30 V) zum synchronen Einspeichern mehrerer Kanäle
- ein Ausgang (TTL) zur Kaskadierung mehrerer Karten

2.5 Zählerbetriebsarten

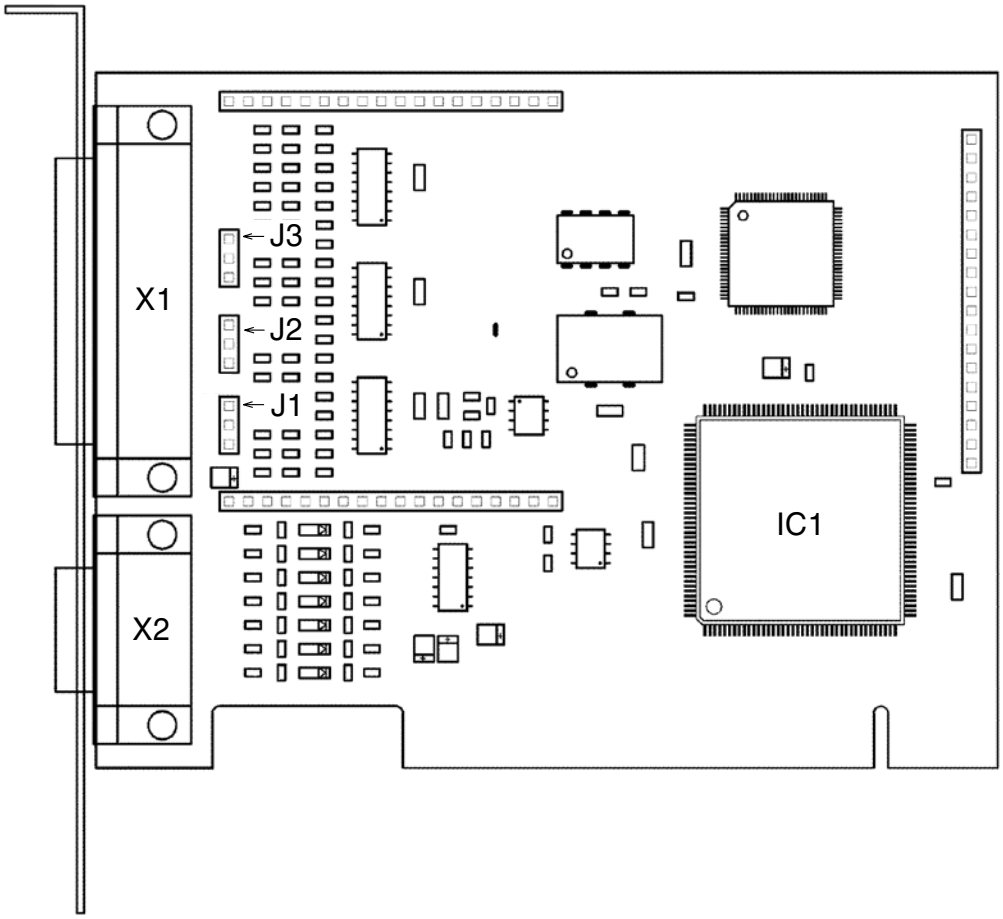
- drei Zähler-Kanäle à 32-Bit mit je einem Lade- und zwei Latch-Register
- Zählen von Encoder-Quadratursignalen mit Einfach-, Zweifach- oder Vierfach-Auswertung
- Ereigniszähler mit Richtungs- und Clear-Eingang
- Integrierter Timer für Impulsbreiten-, Frequenz- oder Geschwindigkeitsmessungen

2.6 Einspeicher-Logik

- asynchrones Einspeichern der Zählerwerte getrennt für jeden Encoder-Kanal per Software, Messgerät-Referenzmarke oder externes Hardwaresignal
- synchrones Einspeichern mehrerer Kanäle per Software, Timer oder externes Signal
- Ausgangssignal zur Kaskadierung mehrerer Karten programmierbar auf Software-, Timer- oder externe Hardware-Synchronisation
- Einspeicherzeit = zwei Bus-Takte = 60 ns bei einer Bus-Taktfrequenz von 33,3 MHz

3. HARDWARE

3.1 Bestückungsplan



X1 = D-Sub Buchsenleiste 25-polig für Zähler-Interface
X2 = D-Sub Buchsenleiste 9-polig für Schalt- und Steuersignale
J1-J3 = Jumper zur Selektierung der Encoder-Betriebspannung (5 V oder 12 V)
IC1 = PCI-Interface

3.2 Selektierung der Encoder-Betriebsspannung

Die Betriebsspannung der Encoder kann mit den Jumpern J1 bis J3 getrennt für jeden Kanal auf +5 V oder +12 V eingestellt werden. Die selektierte Spannung ist aus dem Positionsdruck der Karte ersichtlich.

J1 = Encoder-Kanal 1
J2 = Encoder-Kanal 2
J3 = Encoder-Kanal 3

Hinweis!
Bitte beachten Sie, dass bei falscher Jumper-Stellung der Encoder zerstört werden kann.
Werkseitig sind bei Auslieferung die Betriebsspannungen auf +5 V eingestellt.

3.3 Steckerbelegung X1

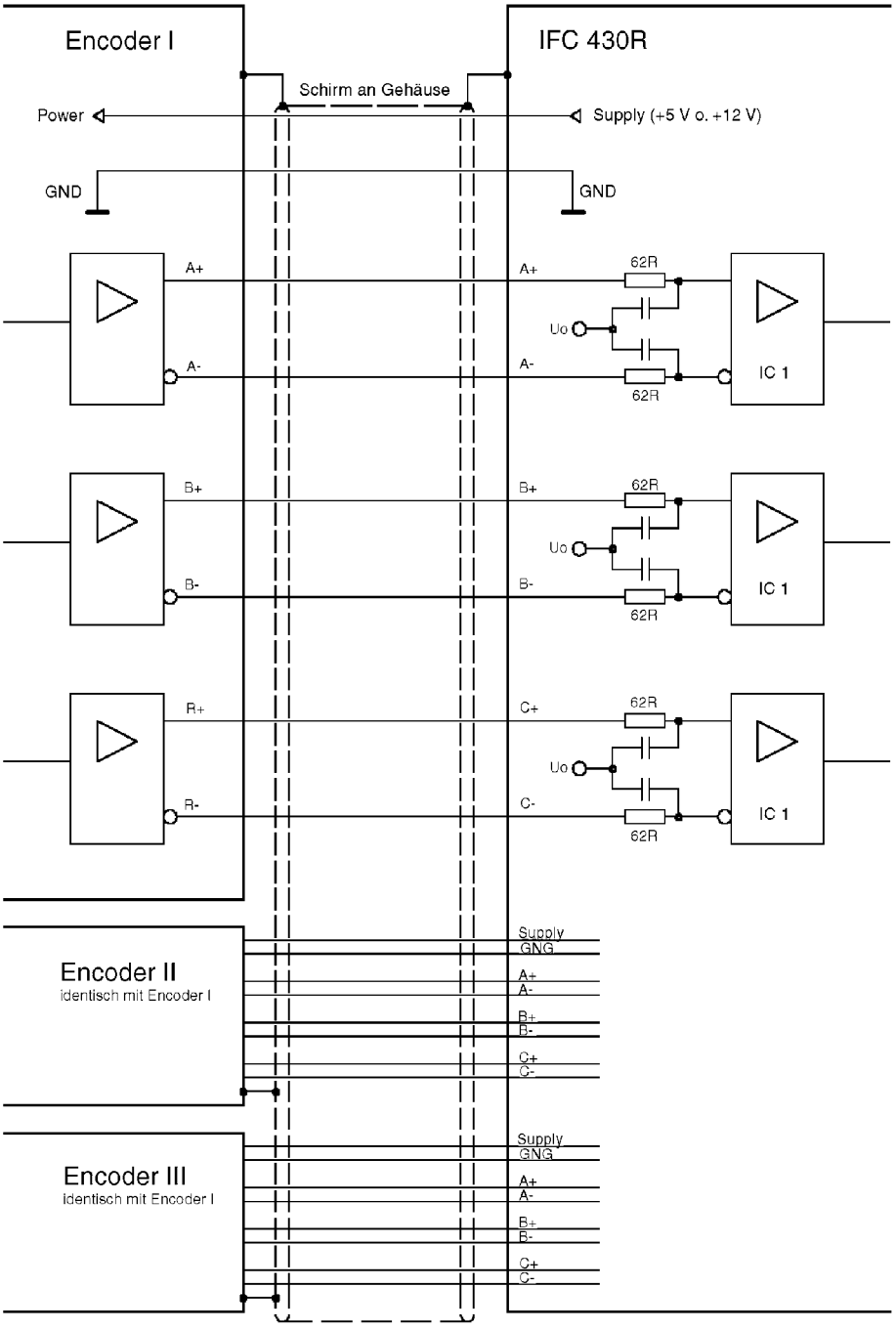
Pin	Signal
1	Kanal 1 Supply +5 V oder +12 V (Selektierung mit J1)
14	Kanal 1 GND
2	Kanal 1 Eingang A+
15	Kanal 1 Eingang A-
3	Kanal 1 Eingang B+
16	Kanal 1 Eingang B-
4	Kanal 1 Eingang C+
17	Kanal 1 Eingang C-
5	Kanal 2 Supply +5 V oder +12 V (Selektierung mit J2)
18	Kanal 2 GND
6	Kanal 2 Eingang A+
19	Kanal 2 Eingang A-
7	Kanal 2 Eingang B+
20	Kanal 2 Eingang B-
8	Kanal 2 Eingang C+
21	Kanal 2 Eingang C-
9	Kanal 3 Supply +5 V oder +12 V (Selektierung mit J3)
22	Kanal 3 GND
10	Kanal 3 Eingang A+
23	Kanal 3 Eingang A-
11	Kanal 3 Eingang B+
24	Kanal 3 Eingang B-
12	Kanal 3 Eingang C+
25	Kanal 3 Eingang C-
13	Störsignal

3.4 Steckerbelegung X2

Pin	Signal
1	GND
2	In 1
3	In 2
4	In 3
5	In 4
6	In 5
7	In 6
8	In Sync.
9	Out Casc.

3.5 Eingangsbeschaltungen

Encoder-Anschluss X1

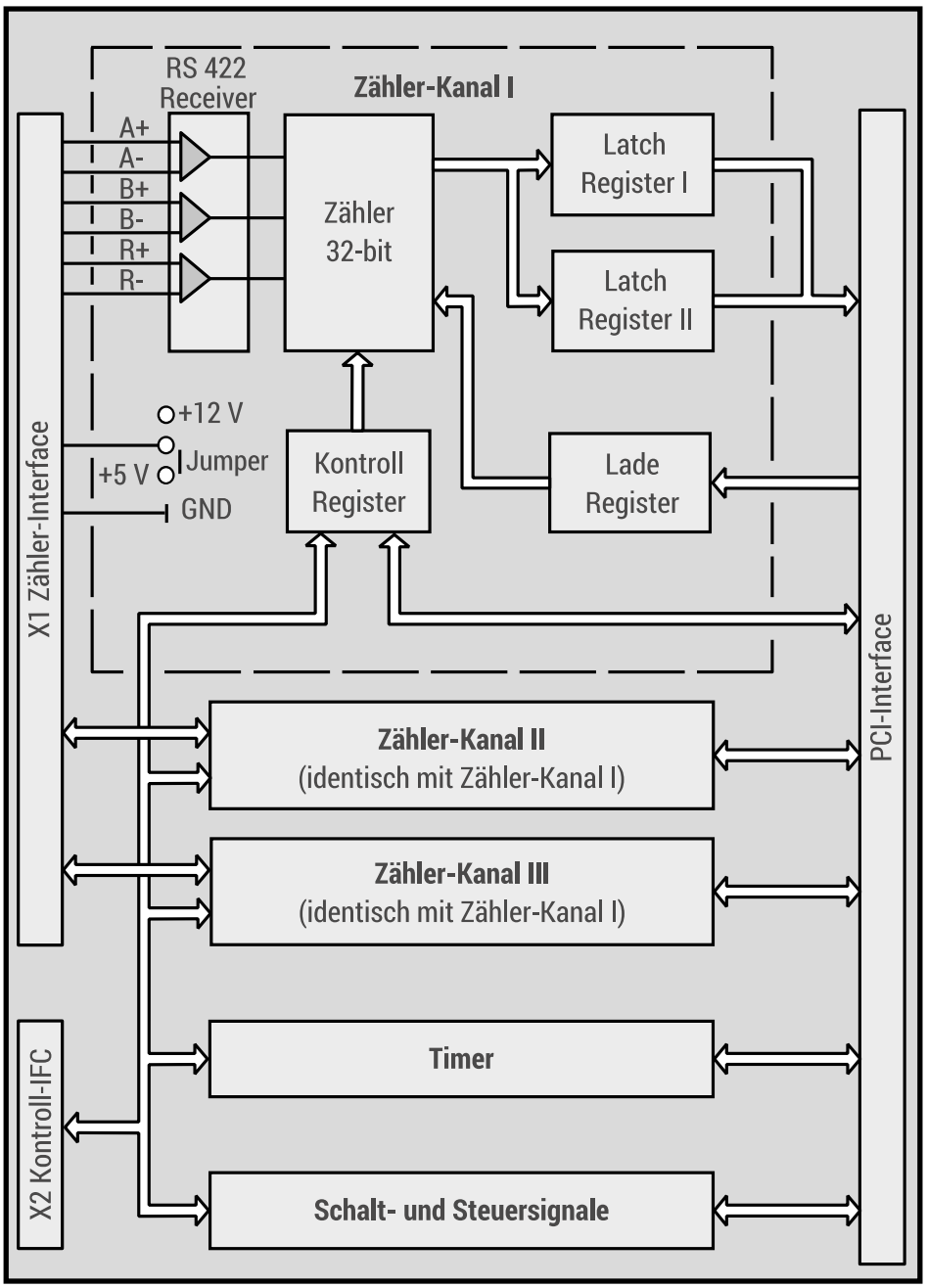


Als Leitungstreiber können alle gängigen RS 422-Driver wie z. B. MC3487, AM32LS31 usw. verwendet werden.

Sollten keine Differenz-Signale vorhanden sein, so können die Eingänge auch als „Single End-Eingang“ beschaltet werden, hierbei bleibt einer der beiden Differenzeingänge (in der Regel –) offen.

Encoder mit der Spannungs-Schnittstelle „1 Vss“ können ohne Einschränkung wie RS 422-Encoder betrieben werden.

3.6 Blockschaltbild



4. ADRESSBELEGUNG

PCI-Interface

Interface: 32-Bit PCI-Bus mit 5-Volt-Anschluss

Zugriff: Memory-Space 40 Hex-Adressen oder
I/0-Space 40 Hex-Adressen

Basis-Adresse: Vergabe automatisch durch Betriebssystem

4.1 Header-Configuration

Adr.	Byte 3	Byte 2	Byte 1	Byte 0	Wert (Hex)
00h	Device ID		Vendor ID		9050 10B5
18h	Base Address Local Memory Space				xxxx xxxx
1Ch	Base Address Local I/O Space				xxxx xxxx
2C	Sub Device ID		Sub Vendor ID		2302 4301

Die Identifizierung einer PCI-Interfacekarte erfolgt durch die vier ID-Werte, wobei „Sub Device ID“ zusammen mit „Sub Vendor ID“ nur der IFC 430R zugeordnet sind.

4.2 Lokale Adress-Belegung

Basis-Adr. +	Funktion	Zugriff	Busbreite
00h	System-Kontrollregister	Schreib- u. Lese-	16 Bit
02h	Timer-Register	Schreib- u. Lese-	16 Bit
04h	Status-Register 1	nur Lese-	16 Bit
06h	Status-Register 2	nur Lese-	16 Bit
08h	Interrupt-Register	Schreib- u. Lese-	16 Bit
0Ah	Abschalt-Verzögerung für externen Sync-In	nur Schreib -	16 Bit
0Ch	reserviert		16 Bit
0Eh	reserviert		16 Bit
10h – 1Eh	Zählerkanal 1	Schreib- u. Lese-	16 Bit
20h – 2Eh	Zählerkanal 2	Schreib- u. Lese-	16 Bit
30h – 3Eh	Zählerkanal 3	Schreib- u. Lese-	16 Bit

5. REGISTER-BESCHREIBUNG

5.1 System-Kontrollregister

Basisadr. + 0 (Schreib- und Lesezugriff)

Bit	Funktion
0	Aktivierung Software-Sync.
1	Aktivierung Casc-Out mit Software
2	Aktivierung Casc-Out mit Timer
3	Aktivierung Casc-Out mit Sync-In
4	Aktivierung von In1 als Referenzimpulssperre für Encoder-Kanal 1
5	Aktivierung von In3 als Referenzimpulssperre für Encoder-Kanal 2
6	Aktivierung von In5 als Referenzimpulssperre für Encoder-Kanal 3
7	Aktivierung Interrupt-Sperre an PCI-Bus (erst ab Hardware-Revision 1)
8	Invertierung von In1
9	Invertierung von In2
10	Invertierung von In3
11	Invertierung von In4
12	Invertierung von In5
13	Invertierung von In6
14	Invertierung von Sync-IN
15	Invertierung von Casc-Out

5.2 Timer-Register

Basisadr. + 2 (Schreib- und Lesezugriff)

Schreibzugriff	Lesezugriff
Timerwert 0 = Timer off 1 bis FFFFh = Timer on	Restlaufzeit bis zum nächsten Impuls

F = PCI-Frequenz / 256 / (Timerwert + 1)

5.3 Status-Register 1

Basisadr. + 4 (nur Lesezugriff)

Bit	Funktion
0	Logikpegel Encoder-Kanal 1 Spur A (X1 Pin 2 u. 15)
1	Logikpegel Encoder-Kanal 1 Spur B (X1 Pin 3 u. 16)
2	Logikpegel Encoder-Kanal 1 Spur C (X1 Pin 4 u. 17)
3	Logikpegel Encoder-Kanal 2 Spur A (X1 Pin 6 u. 19)
4	Logikpegel Encoder-Kanal 2 Spur B (X1 Pin 7 u. 20)
5	Logikpegel Encoder-Kanal 2 Spur C (X1 Pin 8 u. 21)
6	Logikpegel Encoder-Kanal 3 Spur A (X1 Pin 10 u. 23)
7	Logikpegel Encoder-Kanal 3 Spur B (X1 Pin 11 u. 24)
8	Logikpegel Encoder-Kanal 3 Spur C (X1 Pin 12 u. 25)
9	Logikpegel Encoder-Störsignal (X1 Pin 13)
10	Logikpegel I1 (X2 Pin 2)
11	Logikpegel I2 (X2 Pin 3)
12	Logikpegel I3 (X2 Pin 4)
13	Logikpegel I4 (X2 Pin 5)
14	Logikpegel I5 (X2 Pin 6)
15	Logikpegel I6 (X2 Pin 7)

5.4 Status-Register 2

Basisadr. + 6 (nur Lesezugriff)

Bit	Funktion
0	1 = erste Referenzmarke am Encoder-Kanal 1 wurde überfahren
1	1 = zweite Referenzmarke am Encoder-Kanal 1 wurde überfahren
2	1 = erste Referenzmarke am Encoder-Kanal 2 wurde überfahren
3	1 = zweite Referenzmarke am Encoder-Kanal 2 wurde überfahren
4	1 = erste Referenzmarke am Encoder-Kanal 3 wurde überfahren
5	1 = zweite Referenzmarke am Encoder-Kanal 3 wurde überfahren
6	Logikpegel Sync.-In (X2 Pin 8)
7	1 = Interrupt-Anforderung aktiv (erst ab Hardware-Revision 1)
8	reserviert
9	reserviert
10	reserviert
11	reserviert
12	Hardware-Revision Bit 0
13	Hardware-Revision Bit 1
14	Hardware-Revision Bit 2
15	Hardware-Revision Bit 3

5.5 Interrupt-Register

Basisadr. + 8 (Schreib- und Lesezugriff)

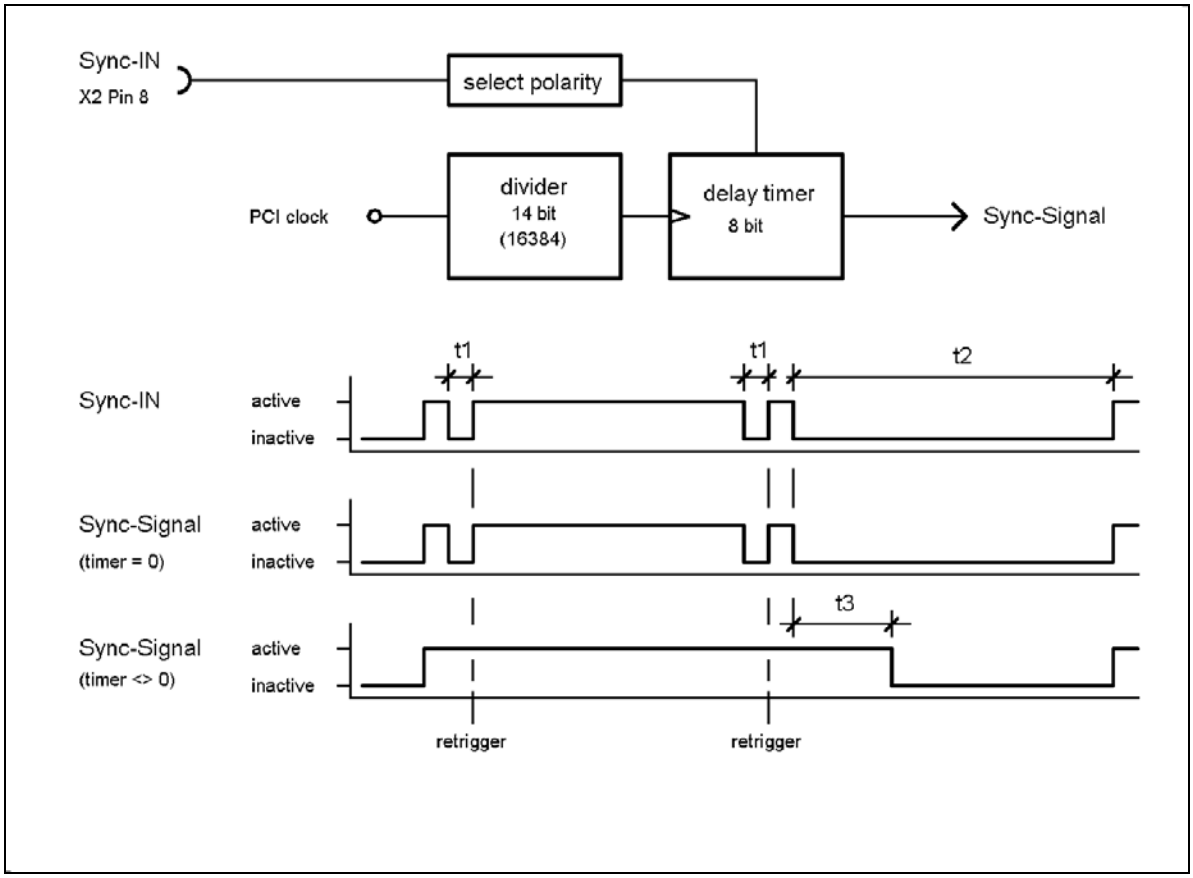
Bit	Funktion
0	0 = disable Timer-Interrupt
	1 = enable Timer-Interrupt
1	0 = disable Sync-Interrupt
	1 = enable Sync-Interrupt (X2 Pin 8)
2	0 = disable In1-Interrupt
	1 = enable In1-Interrupt (X2 Pin 2)
3	0 = disable In2-Interrupt
	1 = enable In2-Interrupt (X2 Pin 3)
4	0 = disable In3-Interrupt
	1 = enable In3-Interrupt (X2 Pin 4)
5	0 = disable In4-Interrupt
	1 = enable In4-Interrupt (X2 Pin 5)
6	0 = disable In5-Interrupt
	1 = enable In5-Interrupt (X2 Pin 6)
7	0 = disable In6-Interrupt
	1 = enable In6-Interrupt (X2 Pin 7)
8	0 = disable Encoderkanal 1 Ref1-Interrupt
	1 = enable Encoderkanal 1 Ref1-Interrupt
9	0 = disable Encoderkanal 1 Ref2-Interrupt
	1 = enable Encoderkanal 1 Ref2-Interrupt
10	0 = disable Encoderkanal 2 Ref1-Interrupt
	1 = enable Encoderkanal 2 Ref1-Interrupt
11	0 = disable Encoderkanal 2 Ref2-Interrupt
	1 = enable Encoderkanal 2 Ref2-Interrupt
12	0 = disable Encoderkanal 3 Ref1-Interrupt
	1 = enable Encoderkanal 3 Ref1-Interrupt
13	0 = disable Encoderkanal 3 Ref2-Interrupt
	1 = enable Encoderkanal 3 Ref2-Interrupt
14	0 = disable Error-Interrupt
	1 = enable Error-Interrupt (X1 Pin 25)
15	reserviert

Die Interrupts sind flankengetriggert und werden beim Übergang vom inaktiven in den aktiven Zustand ausgelöst, sofern diese durch Setzen der einzelnen Bits freigegeben wurden. Ein Lesezugriff auf das Interrupt-Register gibt die aktive(n) Interrupt-Quelle(n) zurück. Nach einem Leszugriff wird der Interrupt-Status automatisch gelöscht.

5.6 Delay-Timer für externen Sync-In

Basisadr. + 10 (nur Schreibzugriff)

Schreibzugriff
Timerwert (8-BIT)
0 = Timer off, 1 bis 255 = Timer on



t1 < t3 < t2

t3 = Timerwert * T_PCI * 16384

(T_PCI = 30 ns bei 33,3 MHz PCI-Frequenz)
(T_PCI * 16384 = 491,52 µs bei 33,3 MHz PCI-Frequenz)

Mit dem Delay-Timer kann eine Abfallverzögerung und somit ein Entprellen des Einganges "Sync-IN" aktiviert werden. Ist der Timerwert "0", so ist der Delay-Timer ausgeschaltet und das Sync-Signal folgt direkt dem Eingang.
Ist der Timerwert "<> 0", so wird der Timer mit jeder Aktivierung vom Eingangssignal getriggert. Bleibt das Eingangs-signal (t2) länger als der Timer (t3) inaktiv, so wird nach Ablauf des Timers auch das Sync-Signal inaktiv.

Hinweise!
Die Zeit für t3 muss größer programmiert sein, als die Zeit für die Signaleinbrüche (t1).
Wird der Ausgang "Casc-Out" (X2 Pin 9) so programmiert, dass dieser mit dem Sync-Signal schaltet
(siehe Kapitel 5.1 System-Kontrollregister), so kann dieses am "Casc-Out" gemessen werden).

5.7 Zähler-Laderegister (Schreibzugriff)

Basis-Adr. +	Lade-Register
10h	Encoder-Kanal 1 Lo-Word
12h	Encoder-Kanal 1 Hi-Word
...	
20h	Encoder-Kanal 2 Lo-Word
22h	Encoder-Kanal 2 Hi-Word
..	
30h	Encoder-Kanal 2 Lo-Word
32h	Encoder-Kanal 2 Hi-Word

Hinweis!

Mit dem Schreibzugriff wird nur das Laderegister gesetzt. Die Übernahme vom Laderegister in den Zähler erfolgt erst durch eine weitere Hard- oder Software-Aktion (siehe Kapitel 5.9 Zähler-Kontrollregister und Kapitel 5.10 Zähler-Moderegister Bit 4 bis 6).

5.8 Zähler-Latchregister (Lesezugriff)

Basis-Adr. +	Latch-Register
10h	Encoder-Kanal 1 Latch-Register 0 Lo-Word
12h	Encoder-Kanal 1 Latch-Register 0 HI-Word
14h	Encoder-Kanal 1 Latch-Register 1 Lo-Word
16h	Encoder-Kanal 1 Latch-Register 1 HI-Word
..	
20h	Encoder-Kanal 2 Latch-Register 0 Lo-Word
22h	Encoder-Kanal 2 Latch-Register 0 HI-Word
24h	Encoder-Kanal 2 Latch-Register 1 Lo-Word
26h	Encoder-Kanal 2 Latch-Register 1 HI-Word
..	
30h	Encoder-Kanal 3 Latch-Register 0 Lo-Word
32h	Encoder-Kanal 3 Latch-Register 0 HI-Word
34h	Encoder-Kanal 3 Latch-Register 1 Lo-Word
36h	Encoder-Kanal 3 Latch-Register 1 HI-Word

Hinweis!

Jeder Encoder-Kanal besitzt zwei Latch-Register. Um den aktuellen Zählerstand zu erhalten, muss dieser erst durch eine vorangegangene Hard- oder Software-Aktion in eines der beiden Latch-Register zwischengespeichert werden

(siehe Kapitel 5.9 Zähler-Kontrollregister und Kapitel 5.10 Zähler-Moderegister Bit 8 bis 10 bzw. 12 bis 14).

5.9 Zähler-Kontrollregister (Schreibzugriff)

Basisadr. + 14 (Encoder-Kanal 1)
Basisadr. + 24 (Encoder-Kanal 2)
Basisadr. + 34 (Encoder-Kanal 3)

Bit	Funktion
0	1 = Zähler löschen
1	1 = Zähler laden
2	1 = Zähler Speichern nach Latchregister 0
3	1 = Zähler Speichern nach Latchregister 1
4	1 = Freigabe für Encoder-Referenzimpuls
5 - 15	reserviert

Mit dem Zähler-Kontrollregister kann per Software der Zähler gelöscht oder mit dem Inhalt vom Lade-Register geladen werden, sowie die Zählerstände in eines der beiden Lacht-Register abgespeichert werden.

5.10 Zähler-Moderegister (Schreib- / Lesezugriff)

Basisadr. + 18 (Encoder-Kanal 1)
Basisadr. + 28 (Encoder-Kanal 2)
Basisadr. + 38 (Encoder-Kanal 3)

Bit	Funktion
0 1	**Phasendiskriminator** Bit 1 Bit 0 0 0 Zähler ohne Phasendiskriminator Spur A = Zählrichtungssignal Spur B = Zählertaktsignal Spur C = Zählerlade- oder Latchsignal 0 1 Zähler mit Phasendiskriminator und Einfach-Auswertung 1 0 Zähler mit Phasendiskriminator und Zweifach-Auswertung 1 1 Zähler mit Phasendiskriminator und Vierfach-Auswertung Betriebsart 0 ist für Zählfunktionen ohne Encoder geeignet Die Betriebsarten 1 bis 3 sind für Encoderanwendungen geeignet
2	**Zählrichtung** 0 = Zählrichtung normal 1 = Zählrichtung invers

3	**Invertierung Encoderspur A u. B** 0 = Encoder-Spur A u. B werden nicht invertiert Encoder-Referenzsignal liegt im 1. Quadranten (Spur A, B und C = 1) 1 = Encoder-Spur A u. B werden invertiert Encoder-Referenzsignal liegt im 3. Quadranten (Spur A und B = 0, C = 1)
4 5 6	**Selektierung Zähler Lade- / Clear-Signal** Bit 6 Bit 5 Bit 4 0 0 0 Hardware-Signale gesperrt 0 0 1 Zähler löschen mit nächsten Encoder-Referenzimpuls 0 1 0 Zähler löschen mit allen Encoder-Referenzimpulsen 0 1 1 Zähler löschen mit Timer 1 0 0 Zähler laden mit nächsten Encoder-Referenzimpuls 1 0 1 Zähler laden mit allen Encoder-Referenzimpulsen 1 1 0 Zähler löschen mit allen Encoder-Referenzimpulsen und zusätzlich Zähler laden bei negativem Nulldurchgang 1 0 1 Zähler laden mit In 1 an X2 für Encoder-Kanal 1 In 3 an X2 für Encoder-Kanal 2 In 5 an X2 für Encoder-Kanal 3
7	reserviert
8 9 10	**Selektierung Hardware-Signal für Latch-Register 0** Bit 10 Bit 9 Bit 8 0 0 0 Hardware-Signale gesperrt 0 0 1 Zähler einspeichern mit Software-Sync (Adr. 0 Bit 0) 0 1 0 Zähler einspeichern mit Timer 0 1 1 Zähler einspeichern mit Sync-In an X2 1 0 0 Zähler einspeichern mit In 2 an X2 für Encoder-Kanal 1 In 4 an X2 für Encoder-Kanal 2 In 6 an X2 für Encoder-Kanal 3 1 0 1 Zähler einspeichern mit nächsten Encoder-Referenzimpuls 1 1 0 Zähler einspeichern mit zweiten Encoder-Referenzimpuls 1 1 1 Zähler einspeichern mit allen Encoder-Referenzimpulsen
11	reserviert
12 13 14	**Selektierung Hardware-Signal für Latch-Register 1** Identisch mit Bit 8 bis 10
15	reserviert

6. INSTALLATIONS-ANLEITUNG

6.1 Hardware-Installation

- **Bitte beachten Sie bei der Installation die Vorsichtsmaßnahmen zur Handhabung elektrostatisch entladungsgefährdeter Bauelemente (ESD) nach DIN EN 100 015**
- Trennen Sie den PC durch Ziehen des Netzsteckers vom Versorgungsnetz
- Öffnen Sie den PC
- Setzen Sie die Interface-Karte IFC 430R in einen freien PCI-Steckplatz ein
- Schliessen Sie den PC
- Verbinden Sie den PC wieder mit dem Versorgungsnetz
- Schalten Sie den PC ein

6.2 Treiber-Installation

WINDOWS XP / VISTA / 7 / 8
Nach dem Boot-Vorgang wird vom Betriebssystem die neue Hardware automatisch erkannt,
hierfür müssen nun die dazugehörigen Treiber installiert werden.

- Legen Sie bitte die Diskette mit der Aufschrift "IFC430R Driver" in das Laufwerk ein.
- Folgen Sie den Dialogen am Bildschirm.

Mit Durchführung der Treiber-Installation erfolgte der Eintrag in die Registry und es wurde folgende Dateie in das Systemverzeichnis kopiert:

z. B. C:\Windows\System32\Drivers\IFC430R.SYS

Auf der mitgelieferten CD befinden sich auch die DLLs für 32 und 64 Bit Systeme.
Die entsprechende DLL sollte direkt in die Applikation eingebunden werden.

6.3 Test-Programm

Das Test-Programm verwendet die vorher Installierten Treiber, eine Installation ist nicht notwendig. Das Test-Programm kann direkt aus dem CD-Laufwerk oder nach einem Kopiervorgang auf die Festplatte von dort aus gestartet werden.

6.4 Programmier Beispiele

Auf der mitgelieferten CD befinden sich Programmier Beispiele für:

- Borland C-Builder 4
- Borland Delphi 5
- Microsoft Visual Basic 6
- Microsoft Visual C 6

7. DLL-FUNKTIONEN

IFC_OpenDrv

Öffnet den Device-Driver

Prototyp: **Data = IFC_OpenDrv();**
Data: 0 = Treiber wurde nicht gefunden
1 = Treiber wurde geöffnet

IFC_CloseDrv
Schliesst den Device-Driver

Prototyp: **IFC_CloseDrv();**

IFC_ScanBUS
Sucht den PCI-Bus nach vorhandenen IFC430R ab und gibt die Anzahl der gefundenen Karten sowie deren Basis-Adressen zurück. Es werden bis zu 8 Karten unterstützt und somit 8 Langworte zurückgegeben. Für jede gefundene IFC430R ist der Wert für deren Basis-Adresse ungleich Null.

Prototyp: **Card = IFC_ScanBus(long *pBaseAddr);**
Card: Anzahl der gefundenen Karten (0 bis 8)
Data: Zeiger auf 8 Langworte
Langwort n = 0 - es wurde keine IFC430R an dieser Position gefunden
Langwort n <> 0 - Basis-Adresse einer vorhandenen IFC430R

IFC_GetHwRev
Liefert die Hardware-Revision der IFC430 R

Prototyp: **Revision = IFC_GetHwRev(UCHAR Card);**
Revision: Hardware-Revision (0 bis 255)
Card: Nummer der Karte (0 bis 7)

IFC_Init
Initialisiert die IFC430R mit den Werten aus einer Initialisierungs-Datei

Prototyp: **Data = IFC_Init(char *pFileName);**
Data: 0 = kein Initialisierungs-Datei gefunden
1 = Initialisierungs-Datei wurde übertragen
*pFileName: Zeiger auf Dateiname für die Initialisierungs-Datei. Wird kein Dateiname angegeben, so gilt die Standard-Initialisierungsdatei „IFC430R.INI“

IFC_ExtInit
Initialisierung der Polarität für die externen Ein- und Ausgänge an X2.

Prototyp: **IFC_ExtInit(UCHAR Card, USHORT Data);**
Card: Nummer der Karte (0 bis 7)
Data: Bit 0 = 0 - Eingang 1 low aktiv (X2 Pin 2)
1 - " " high "
Bit 1 = 0 - Eingang 2 low aktiv (X2 Pin 3)
1 - " " high "
Bit 2 = 0 - Eingang 3 low aktiv (X2 Pin 4)
1 - " " high "
Bit 3 = 0 - Eingang 4 low aktiv (X2 Pin 5)
1 - " " high "
Bit 4 = 0 - Eingang 5 low aktiv (X2 Pin 6)
1 - " " high "
Bit 5 = 0 - Eingang 6 low aktiv (X2 Pin 7)
1 - " " high "
Bit 6 = 0 - Eingang SyncIn low aktiv (X2 Pin 8)
1 - " " high "
Bit 7 = 0 - Ausgang CascOut low aktiv (X2 Pin 9)
1 - " " high "

IFC_ExtIn
Liefert die Zustände der externen Eingänge einer IFC430R

Prototyp: **Data = IFC_ExtIn(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Data: Bit 0 = 0 - Eingang 1 inaktiv (X2 Pin 2)
1 - " " aktiv
Bit 1 = 0 - Eingang 2 inaktiv (X2 Pin 3)
1 - " " aktiv
Bit 2 = 0 - Eingang 3 inaktiv (X2 Pin 4)
1 - " " high "
Bit 3 = 0 - Eingang 4 inaktiv (X2 Pin 5)
1 - " " aktiv
Bit 4 = 0 - Eingang 5 inaktiv (X2 Pin 6)
1 - " " high "
Bit 5 = 0 - Eingang 6 inaktiv (X2 Pin 7)
1 - " " aktiv

IFC_CascInit
Initialisierung für den Ausgang „CascOut" (X2 Pin 9).

Prototyp: **IFC_CascInit(UCHAR Card USHORT Value);**
Card: Nummer der Karte (0 bis 7)
Value: 0 = CascOut inaktiv
1 = CascOut aktiv
2 = CascOut schaltet mit Timer
3 = CascOut folgt dem Eingangssignal SyncIn

IFC_ExtSync
Liefert den Zustand vom Eingang SyncIn (X2 Pin 8)

Prototyp: **SyncIn = IFC_ExtSync(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
SyncIn: 0 = SyncIn inaktiv
1 = " aktiv

IFC_SetSyncDelay
Liefert den Zustand vom Eingang SyncIn (X2 Pin 8)

Prototyp: **SyncIn = IFC_ExtSync(UCHAR Card);**
Bool: 0 = kein Delay-Timer bei dieser Hardware-Revision vorhanden
1 = Delay-Timer wurde gesetzt
Card: Nummer der Karte (0 bis 7)
TimerValue: 0 bis 255

IFC_IntInit
Interrupt-Initialisierung. Die Interrupt-Quellen werden mit „Oder" verknüpft, somit können gleichzeitig mehrere Interrupt-Ereignisse bearbeitet werden.

Prototyp: **void IFC_IntInit(UCHAR Card, USHORT Data);**
Card: Nummer der Karte (0 bis 7)
Data: Bit 0 = 1 - enable Interrupt bei Timer-Nulldurchgang
Bit 1 = 1 - enable Interrupt bei Aktivierung SyncIn (X2 Pin 8)
Bit 2 = 1 - enable Interrupt bei Aktivierung In1 (X2 Pin 2)
Bit 3 = 1 - enable Interrupt bei Aktivierung In2 (X2 Pin 3)
Bit 4 = 1 - enable Interrupt bei Aktivierung In3 (X2 Pin 4)
Bit 5 = 1 - enable Interrupt bei Aktivierung In4 (X2 Pin 5)
Bit 6 = 1 - enable Interrupt bei Aktivierung In5 (X2 Pin 6)
Bit 7 = 1 - enable Interrupt bei Aktivierung In6 (X2 Pin 7)
Bit 8 = 1 - enable Interrupt mit 1. Referenzmarke Axis 0
Bit 9 = 1 - enable Interrupt mit 2. Referenzmarke Axis 0
Bit 10 = 1 - enable Interrupt mit 1. Referenzmarke Axis 1
Bit 11 = 1 - enable Interrupt mit 2. Referenzmarke Axis 1
Bit 12 = 1 - enable Interrupt mit 1. Referenzmarke Axis 2
Bit 13 = 1 - enable Interrupt mit 2. Referenzmarke Axis 2
Bit 14 = 1 - enable Interrupt bei Aktivierung Encoder–Error (X1 Pin 25)
Bit 15 = reserviert

IFC_IntStatus
Liefert den Interrupt-Status zurück. Nach dem Auslesen vom Interrupt-Status werden alle Bits gelöscht und erst mit dem nächsten Interrupt-Ereignis wieder gesetzt (Flankentriggerung).

Prototyp: **Status = USHORT IFC_IntStatus(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Status: Bit 0 = 1 - Interrupt Aktivierung durch Timer-Nulldurchgang
Bit 1 = 1 - Interrupt-Aktivierung durch SyncIn (X2 Pin 8)
Bit 2 = 1 - Interrupt-Aktivierung durch In1 (X2 Pin 2)
Bit 3 = 1 - Interrupt-Aktivierung durch In2 (X2 Pin 3)
Bit 4 = 1 - Interrupt-Aktivierung durch In3 (X2 Pin 4)
Bit 5 = 1 - Interrupt-Aktivierung durch In4 (X2 Pin 5)
Bit 6 = 1 - Interrupt-Aktivierung durch In5 (X2 Pin 6)
Bit 7 = 1 - Interrupt-Aktivierung durch In6 (X2 Pin 7)
Bit 8 = 1 - Interrupt-Aktivierung durch 1. Referenzmarke Axis 0
Bit 9 = 1 - Interrupt-Aktivierung durch 2. Referenzmarke Axis 0
Bit 10 = 1 - Interrupt-Aktivierung durch 1. Referenzmarke Axis 1
Bit 11 = 1 - Interrupt-Aktivierung durch 2. Referenzmarke Axis 1
Bit 12 = 1 - Interrupt-Aktivierung durch 1. Referenzmarke Axis 2
Bit 13 = 1 - Interrupt-Aktivierung durch 2. Referenzmarke Axis 2
Bit 14 = 1 - Interrupt-Aktivierung durch Encoder–Error (X1 Pin 25)
Bit 15 = reserviert

IFC_IntUnMask
Aktivierung Interrupt–Handler. Mit Aktivierung des Interrupt–Handlers wird die als Zeiger übergebene Interrupt-Serviceroutine vom System-Treiber mit jedem Hardware-Interrupt abgearbeitet.
Es können verschieden Interrupt-Quellen initialisiert werden (siehe IFC_IntInit)

Prototyp: **void IFC_IntUnMask(UCHAR Card, BOOL IrqShared, pIrqHandler Long);**
Card: Nummer der Karte (0 bis 7)
IrqShared: True = Interrupt ist Shared
False = Interrupt ist nicht Shared
pIrqHandler: Zeiger auf Interrupt-Serviceroutine

Beispiel für Interrupt-Serviceroutine

```
void __stdcall IRQ_Handler(USHORT IRQ_Number, ULONG IRQ_Source)
        {
          /*        Interrupt-Serviceroutine
                    Der Interrupt–Handler liest mit jeder Generierung den Interrup-Status von der IFC430R
                    und übergibt diesen als Parameter (IRQ-Source) an die Interrupt-Serviceroutine.
                    Ebenfalls wird die Interrupt-Nummer als Parameter (IRQ-Nummer 1-15) übergeben.
           */
        }
```

Hinweis:

zur Zeit nicht Lauffähig unter „Windows 2000“ und „Windows XP“)

IFC_IntMask
Deaktivierung Interrupt–Handler.

Prototyp: **Data = IFC_IntMask(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Data: 0 = Interrupt–Handler war bereits inaktiv
1 = Interrupt–Handler wurde vom aktiven in den inaktiven Zustand geschaltet

IFC_IntIsMask
Liefert den Zustand vom Interrupt–Handler.

Prototyp: **Data = IFC_IntIsMask(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Data: True = Interrupt-Handler ist inaktiv
False = Interrupt-Handler ist aktiv

IFC_IntCnt
Interrupt–Counter auslesen.

Prototyp: **Data = IFC_IntCnt(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Data: Anzahl (ULONG) der ausgeführten Interrupt-Routinen seit der letzten Handler-Aktivierung

IFC_SetTimer
Preload-Wert für Timer setzen. Ist der Preload-Wert auf „0" gesetzt, so wird ein laufender Timer mit dem nächsten Null-Durchgang angehalten. Ist der Preload-Wert <> 0, so wird ein stehender Timer sofort mit dem vorgegebenen Wert gestartet oder ein laufender Timer im nächsten Null-Durchgang mit dem neuen Wert geladen.
Time = (TimerValue + 1) * 256 * T_PCI
TimerValue = Preload-Wert
T_PCI = PCI-Clock-Zeit (= 30ns bei 33,3MHz)
Prototyp: **IFC_SetTimer(UCHAR Card, USHORT TimerValue)**;
Card: Nummer der Karte (0 bis 7)
TimerValue: 0 bis 65535

IFC_GetTimer
Liefert die Restlaufzeit bis zum nächsten Timer Nulldurchgang zurück. Während des Timer-Nulldurchgangs wird ein Signal erzeugt, dass je nach Initialisierung für verschiedene Funktionen wie z. B. Zählerwerte synchron einspeichern, Zähler Laden oder Erzeugung eines Impulses am Ausgang „CascOut" verwendet werden kann.
Prototyp: **Time = IFC_GetTimer(UCHAR Card);**
Time: 0 bis 65535
Card: Nummer der Karte (0 bis 7)

IFC_SetAdr
Schreibt Daten direkt in die angegebene Adresse
Prototyp: **IFC_SetAdr(UCHAR Card, USHORT Offset, USHORT Data);**
Card: Nummer der Karte (0 bis 7)
Offset: 0 bis 64 (Adresse = Basis-Adresse der Karte + Offset)
Data: Daten

IFC_GetAdr
Liest Daten aus der angegebene Adresse
Prototyp: **Data = IFC_GetAdr(UCHAR Card, USHORT Offset);**
Data: 0 bis 65535
Card: Nummer der Karte (0 bis 7)
Offset: 0 bis 64 (Zeiger = Basis-Adresse + Offset)

IFC_EncStatus
Liefert den Status der Encoder-Eingänge einer IFC430R
Prototyp: **Data = IFC_EncStatus(UCHAR Axis);**
Axis: Nummer der Achse (0 bis 23)
Data: 0 bis 7
Bit 0 = 0 - Spur A inaktiv
1 - " " aktiv
Bit 1 = 0 - Spur B inaktiv
1 - " " aktiv
Bit 2 = 0 - Spur C inaktiv
1 - " " aktiv

IFC_EncErr
Liefert den Zustand vom Encoder-Störsignal (Eingange X1 Pin 13)
Prototyp: **Data = IFC_EncErr(UCHAR Card);**
Card: Nummer der Karte (0 bis 7)
Data: 0 = Encoder- Störsignal inaktiv
1 = " " aktiv

IFC_RefInit
Initialisierung externer Eingänge als Referenz-Impulssperre
Prototyp: **IFC_RefInit(UCHAR Card, USHORT Data)**
Card: Nummer der Karte (0 bis 7)
Data: 0 bis 7
Bit 0 = 0 - Eingang 1 (X2 Pin 2) hat keinen Einfluss auf Referenz-Impuls
1 - " " wirkt als Referenz-Impulssperre für Encoder-Kanal 1
Bit 1 = 0 - Eingang 3 (X2 Pin 4) hat keinen Einfluss auf Referenzmarke
1 - " " wirkt als Referenz-Impulssperre für Encoder-Kanal 2
Bit 1 = 0 - Eingang 5 (X2 Pin 6) hat keinen Einfluss auf Referenzmarke
1 - " " wirkt als Referenz-Impulssperre für Encoder-Kanal 3

IFC_RefClear
Feigabe der Referenz-Impulse (löscht den Referenz-Status)
Prototyp: **IFC_RefClear(UCHAR Axis)**
Axis: Nummer der Achse (0 bis 23)

IFC_RefStatus
Liefert den Status der überfahrenen Referenzmarken
Prototyp: **Data = IFC_RefStatus(UCHAR Axis)**
Axis: Nummer der Achse (0 bis 23)
Data: 0 bis 3
Bit 0 = 0 - 1. Referenzmarke wurde noch nicht überfahren
1 - 1. " " überfahren
Bit 1 = 0 - 2. Referenzmarke wurde noch nicht überfahren
1 - 2. " " überfahren

IFC_SetLoadReg
Wert für Zähler-Laderegister übergeben.
Prototyp: IFC_SetLoadReg(UCHAR Axis, ULONG LoadValue);
Axis: Nummer der Achse (0 bis 23)
LoadValue: 32-Bit Wert

IFC_Load
Der Zähler einer Achse wird mit dem Inhalt des Laderegisters per Software geladen. Ein Zähler kann auch durch verschiedene Hardware-Quellen geladen werden (siehe IFC_LdClr).
Prototyp: IFC_Load(UCHAR Axis);
Axis: Nummer der Achse (0 bis 23)

IFC_Clear
Der Zähler einer Achse wird gelöscht. Ein Zähler kann auch durch verschiedene Hardware-Quellen gelöscht werden (siehe IFC_LdClr).
Prototyp: IFC_Clear(UCHAR Axis);
Axis: Nummer der Achse (0 bis 23)

IFC_ CntLdClr
Mode Zähler laden oder löschen mit Hardware-Signal
Prototyp: IFC_ CntLdClr(UCHAR Axis, UCHAR Value);
Axis: Nummer der Achse (0 bis 23)
Value: 0 bis 7

- 0 = Hardware-Signale gesperrt
- 1 = Zähler löschen mit nächstem Encoder-Referenzimpuls
- 2 = Zähler löschen mit allen Encoder-Referenzimpulsen
- 3 = Zähler löschen mit Timer
- 4 = Zähler laden mit nächstem Encoder-Referenzimpuls
- 5 = Zähler laden mit allen Encoder-Referenzimpulsen
- 6 = Zähler löschen mit allen Encoder-Referenzimpulsen und zusätzlich Zähler laden bei negativem Nulldurchgang
- 7 = Zähler laden mit externem Signal
  In 1 an X2 für Encoder-Kanal 1
  In 3 an X2 für Encoder-Kanal 2
  In 5 an X2 für Encoder-Kanal 3

IFC_GetLatchReg
Liefert den 32-Bit Zählerwert der vorher in ein Latchregister abgespeichert wurde.
Prototyp: Data = IFC_GetLatchReg(UCHAR Axis, UCHAR Reg);
Data: 32-Bit Zählerwert
Axis: Nummer der Achse (0 bis 23)
Reg: Nummer des Latchregisters (0 oder 1)

IFC_Latch
Speichert den 32-Bit Zählerwert einer Achse in eines der beiden Latchregister per Software. Der Zählerwert kann auch durch verschiedene Hardware-Quellen in die Latchregister eingespeichert werden (siehe IFC_CntLatch).
Prototyp: IFC_Latch(UCHAR Axis, UCHAR Reg);
Axis: Nummer der Achse (0 bis 23)
Reg: Nummer des Latchregisters (0 oder 1)

IFC_LatchImp
Erzeugt einen Hardware-Impuls der gleichzeitig an alle Latchregister einer Karte geführt wird, somit können mehrere Zählerwerte einer Karte synchron per Software eingespeichert werden. Es ist auch möglich den Impuls gleichzeitig auf den Ausgang „CascOut" (X2 Pin 9) zu schalten. Ist der „CascOut" dieser Karte mit dem „SyncIn" der nächsten Karten verbunden, ist es möglich, mehrere Zählerwerte mehrerer Karten synchron per Software einzuspeichern.
Prototyp: IFC_LatchImp(UCHAR Card, UCHAR Casc);
Card: Nummer der Karte (0 bis 7)
Casc: 0 = Latch-Impuls wird nicht auf „CascOut" (X2 Pin 9) geschaltet
1 = Latch-Impuls wird gleichzeitig auf „CascOut" (X2 Pin 9) geschaltet

IFC_ CntLatch0
Selektierung Hardware-Signal für Zähler-Latchregister 0
Prototyp: IFC_ CntLatch0(UCHAR Axis, UCHAR Value);
Axis: Nummer der Achse (0 bis 23)
Value: 0 bis 7
0 = Hardware-Signale gesperrt
1 = Zähler einspeichern mit Funktion „IFC_LatchImp"
2 = Zähler einspeichern mit Timer
3 = Zähler einspeichern mit Sync-In an X2
4 = Zähler einspeichern mit
In 2 an X2 für Encoder-Kanal 1
In 4 an X2 für Encoder-Kanal 2
In 6 an X2 für Encoder-Kanal 3
5 = Zähler einspeichern mit nächstem Encoder-Referenzimpuls
6 = Zähler einspeichern mit zweitem Encoder-Referenzimpuls
7 = Zähler einspeichern mit allen Encoder-Referenzimpulsen

IFC_ CntLatch1

Selektierung Hardware-Signal für Zähler-Latchregister 1

Prototyp: **IFC_ CntLatch1(UCHAR Axis, UCHAR Value);**

Axis: Nummer der Achse (0 bis 23)

Value: 0 bis 7

0 = Hardware-Signale gesperrt

1 = Zähler einspeichern mit Funktion „IFC_LatchImp"

2 = Zähler einspeichern mit Timer

3 = Zähler einspeichern mit Sync-In an X2

4 = Zähler einspeichern mit

In 2 an X2 für Encoder-Kanal 1

In 4 an X2 für Encoder-Kanal 2

In 6 an X2 für Encoder-Kanal 3

5 = Zähler einspeichern mit nächstem Encoder-Referenzimpuls

6 = Zähler einspeichern mit zweitem Encoder-Referenzimpuls

7 = Zähler einspeichern mit allen Encoder-Referenzimpulsen

IFC_ CntMode

Einstellung für Phasendiskriminator, Zählrichtung und Encoder-Quaqdranten.

Prototyp: **IFC_ CntMode(UCHAR Axis, UCHAR Value);**

Axis: Nummer der Achse (0 bis 23)

Value: 0 bis 15

Bit 1	**Bit 0**	**Phasendiskriminator**
0	0	Zähler ohne Phasendiskriminator
Spur A = Zählrichtungssignal		
Spur B = Zählertaktsignal		
Spur C = Zählerlade- oder Latchsignal		
0	1	Zähler mit Phasendiskriminator und Einfach-Auswertung
1	0	Zähler mit Phasendiskriminator und Zweifach-Auswertung
1	1	Zähler mit Phasendiskriminator und Vierfach-Auswertung
		Betriebsart 0 ist für Zählfunktionen ohne Encoder geeignet Die Betriebsarten 1 bis 3 sind für Encoder-Anwendungen geeignet

Bit 2	**Zählrichtung**
0	Zählrichtung normal
1	Zählrichtung invers

Bit 3	**Invertierung Encoderspur A u. B**
0	Encoder-Spur A u. B werden nicht invertiert
	Encoder-Referenzsignal liegt im 1. Quadranten (Spur A, B und C = 1)
1	Encoder-Spur A u. B werden invertiert Encoder-Referenzsignal liegt im 3. Quadranten (Spur A und B = 0, C = 1)

VERTRIEBSKONTAKTE

AUSTRIA *Stammsitz*	**RSF Elektronik Ges.m.b.H.**	A-5121 Tarsdorf 93	☎ +43 62 78 81 92-0 FAX +43 62 78 81 92-79	e-mail: info@rsf.at internet: www.rsf.at
BELGIEN	**HEIDENHAIN NV/SA**	Pamelse Klei 47 1760 Roosdaal	☎ +32 (54) 34 3158 FAX +32 (54) 34 3173	e-mail: sales@heidenhain.be internet: www.heidenhain.be
FRANKREICH	**HEIDENHAIN FRANCE sarl**	2 Avenue de la Christallerie 92310 Sèvres	☎ +33 1 41 14 30 00 FAX +33 1 41 14 30 30	e-mail: info@heidenhain.fr internet: www.heidenhain.fr
GROßBRITANNIEN	**HEIDENHAIN (GB) Ltd.**	200 London Road Burgess Hill West Sussex RH15 9RD	☎ +44 1444 247711 FAX +44 1444 870024	e-mail: sales@heidenhain.co.uk internet: www.heidenhain.co.uk
ITALIEN	**HEIDENHAIN ITALIANA S.r.l.**	Via Asiago, 14 20128 Milano	☎ +39 02 27075-1 FAX +39 02 27075-210	e-mail: info@heidenhain.it internet: www.heidenhain.it
NIEDERLANDE	**HEIDENHAIN NEDERLAND B.V.**	Copernicuslaan 34 6716 BM EDE	☎ +31 318-581800 FAX +31 318-581870	e-mail: verkoop@heidenhain.nl internet: www.heidenhain.nl
SPANIEN	**FARRESA ELECTRONICA S.A**	Les Corts 36-38 08028 Barcelona	☎ +34 93 4 092 491 FAX + 34 93 3 395 117	e-mail: farresa@farresa.es internet: www.farresa.es
SCHWEDEN	**HEIDENHAIN Scandinavia AB**	Storsätragränd 5 SE-12739 Skärholmen	☎ +46 8 531 933 50 FAX +46 8 531 933 77	e-mail: sales@heidenhain.se internet: www.heidenhain.se
SCHWEIZ	**HEIDENHAIN (SCHWEIZ) AG**	Vieristrasse 14 8603 Schwerzenbach	☎ +41 44 806 27 27 FAX +41 44 806 27 28	e-mail: verkauf@heidenhain.ch internet: www.heidenhain.ch
CHINA	**DR. JOHANNES HEIDENHAIN (CHINA) Co., Ltd**	Tian Wei San Jie, Area A, Beijing Tianzhu Airport Industrial Zone Shunyi District, Peking 101312	☎ +86 10 80 42-0000	e-mail: sales@heidenhain.com.cn internet: www.heidenhain.com.cn
HONG KONG S.A.R.	**HEIDENHAIN LIMITED**	Unit 2007-2010 Apec Plaza 49 Hoi Yuen Road, Kwun Tong Kowloon, Hong Kong	☎ +852 27 59 19 20 FAX +852 27 59 19 61	e-mail: service@heidenhain.com.hk
ISRAEL	**MEDITAL Hi-Tech**	7 Leshem Str. 47170 Petach Tikva	☎ +972 0 3 923 33 23 FAX +972 0 3 923 16 66	e-mail: avi@medital.co.il internet: www.medital.co.il
JAPAN	**HEIDENHAIN K.K.**	Hulic Kojimachi Bldg., 9F 3-2 Kojimachi, Chiyoda-ku Tokio, 102-0083	☎ +81 3 3234 7781 FAX +81 3 3262 2539	e-mail: sales@heidenhain.co.jp internet: www.heidenhain.co.jp
KOREA	**HEIDENHAIN LTD.**	202 Namsung Plaza, 9th Ace Techno Tower, 130, Digital-Ro, Geumcheon-Gu, Seoul, Korea 153-782	☎ +82 2 20 28 74 30	e-mail: info@heidenhain.co.kr internet: www.rsf.co.kr
RUSSLAND	**OOO «HEIDENHAIN»**	ul. Goncharnaya, d. 21 115172 Moskau	☎ +7 (495) 777 34 66 FAX +7 (499) 702 33 31	e-mail: info@heidenhain.ru internet: www.heidenhain.ru
SINGAPUR	**HEIDENHAIN PACIFIC PTE LTD.**	51, Ubi Crescent 408593 Singapur	☎ +65 67 49 32 38 FAX +65 67 49 39 22	e-mail: info@heidenhain.com.sg internet: www.heidenhain.com.sg
TAIWAN	**HEIDENHAIN CO., LTD.**	No. 29, 33rd Road; Taichung Industrial Park Taichung 40768	☎ +886 4 2358 89 77 FAX +886 4 2358 89 78	e-mail: info@heidenhain.tw internet: www.heidenhain.com.tw
USA	**HEIDENHAIN CORPORATION**	333 East State Parkway Schaumburg, IL 60173-5337	☎ +1 847 490 11 91	e-mail: info@heidenhain.com internet: www.rsf.net

Ausgabe 12/2017 ■ Art.Nr.824548-01 ■ Dok.Nr. D824548-01-A-01 ■ Technische Änderungen vorbehalten!