http://www.cgc-instruments.de/

Messkarte ISA-C32T12

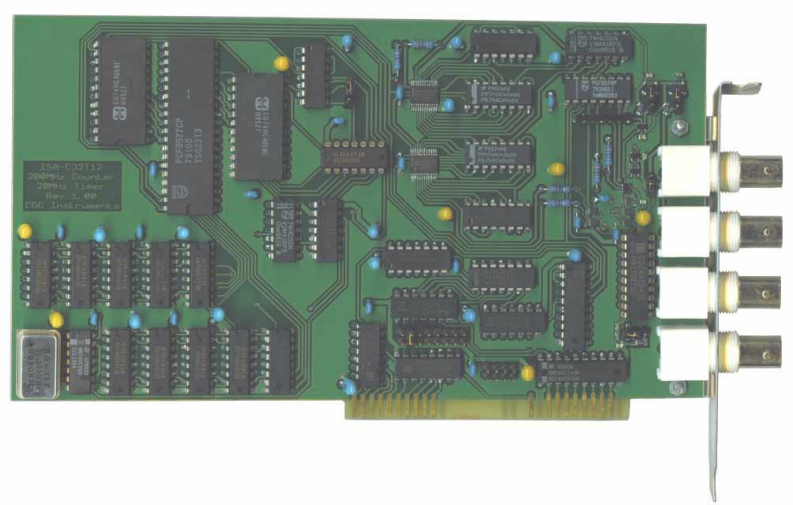
Bedienungsanleitung

Dokument-Version 1.01, erstellt am 03.02.2004

200 MHz 32-bit Zähler, 20 MHz 12-bit Zeitgeber

ISA-C32T12

Version 1.00



Beschreibung

Die Messkarte enthält einen Zähler und zwei hintereinander geschaltete Zeitgeber. Der Zähler verfügt über einen Signal- und einen Gate-Eingang. Er zählt Impulse auf dem Signaleingang, solange der Gate-Eingang aktiv ist. Der Gate-Eingang kann durch den Rechner freigegeben oder aktiviert werden. Die Messung kann einmalig (ein Puls auf dem Gate-Eingang) oder wiederholt (jeder Puls auf dem Gate-Eingang) erfolgen. Der Zähler kann zu jeder Zeit zurückgesetzt werden, oder es können mehrere Messsequenzen aufsummiert werden, wobei die Zwischenergebnisse ebenfalls zu jeder Zeit abgefragt werden können.

Die Zeitgeber bestehen aus zwei identischen unabhängigen Kanälen. Jeder Kanal kann durch seinen Trigger-Eingang oder durch die Steuersoftware getriggert werden. Das Vorhandensein einer vordefinierten Signalflanke auf dem Trigger-Eingang löst einen Ausgangspuls programmierbarer Länge aus. Der Puls startet und endet synchron mit dem Taktsignal, so dass eine hohe Reproduzierbarkeit auch bei den längsten Pulsen garantiert ist. Die Pulslänge und ihre Stabilität sind durch einen internen Quarzoszillator gegeben. Die beiden Kanäle sind intern gekoppelt, um zeitlich verschobene Pulse im Vergleich zum Trigger-Signal zu erzeugen. Der Trigger-Eingang des ersten und der Ausgang des zweiten Kanals sind an die BNC-Buchsen der Messkarte angeschlossen. Die Polarität des Ausgangssignals kann umgeschaltet werden. Weiterhin kann der Ausgang des zweiten Kanals mit dem Gate-Eingang des Zählers intern verbunden werden.

Technische Daten

PC-Interface

- PC-Bussystem: ISA
- E/A-Adressen: 2 Adressen beginnend bei 200h, 210h, 220h, 230h, 300h, 310h, 320h, oder 330h
- IRQ-Leitung: IRQ3, 4, 5, oder 7 (für die Funktion der Messkarte nicht erforderlich)
- Software-Treiber: für Windows 95/98/ME und für Windows NT/2000/XP

Zähler

- Pulsbreite auf den Eingängen: $t_W \geq 2$ ns
- Zulässige Flankensteilheit auf den Eingängen: ≤ 5 ns/V
- Polarität des Signaleingangs: positiv oder negativ (computersteuerbar) entspricht der steigenden oder fallenden aktiven Flanke
- Gate-Eingang: aktiv bei log. 1, und/oder computersteuerbar
- Messsequenz: einzelne oder wiederholte Messungen
- Zählerlänge: 32 Bit (d.h. maximal $2^{32}-1$ = 4,294,967,295 Impulse pro Messung)
- Überlauf: Signalisiert ab dem 2^{31} = 2,147,483,648-ten Puls
- Kontrollbits:
  Gate Enable (Freigabe des Gate-Eingangs)
  Gate Control (Aktivierung des Zähler-Gates durch die Software)
  Input Negative (invertierter Signaleingang)
  Measurement Run (Freigabe der Messung)
  Measurement Single (einzelne oder wiederholte Messung)
  Reset Control (Zurücksetzten des Zählers und der Kontrolllogik durch die Software)
- Statusbits:
  Gate Active (Status vom Gate)
  Measurement Active (Freigabe des Zählers)
  Overflow (Überlauf des Zählers)

Zeitgeber

(für die Definitionen der Zeitverzögerungen siehe Abb. 5)

- Zwei 20 MHz 12-bit Zeitgeber
- Auslösen: Hardware / Software
- Aktive Signalflanke: steigend / fallend (computersteuerbar)

- Zulässige Flankensteilheit auf dem Triggereingang: ≤ 100 ns/V
- Ausgangssignal: positive oder negative Pulse
- Pulsbreite auf dem Triggereingang: $t_W \geq 15$ ns
- Zeitverzögerung zwischen dem Triggereingang und dem Ausgang: $t_P \leq 110$ ns + T_{CLK}
- Taktfrequenz: $f_{CLK} = 1/T_{CLK} \leq 20$ MHz, wählbar durch Austausch des Quarzoszillators
- Einstellbare Pulslänge auf dem Ausgang: $t_D = 400$ ns … 1600 s bei $f_{CLK} = 10$ MHz
- Zeitgenauigkeit: besser als 10 ns + Genauigkeit des Quarzoszillators (±100 ppm)
- Zeitauflösung: 15 Bit (12 Bit Mantisse + 3 Bit Exponent)

Allgemein

- Anschlüsse: 50 Ω BNC-Buchsen
- Eingangs- und Ausgangspegel: TTL / 5V-CMOS
  Eingang: logische 0: $V_L = 0…1.5$ V, logische 1: $V_H = 3.5…5$ V
- Eingangsimpedanz: 47 kΩ + schaltbare 50 Ω *pull-up* Widerstände (aktiver Abschluss)
- Überspannungsschutz der Eingänge: Spannungen < 0 V oder > 5 V, maximal erlaubter Eingangsstrom durch die Schutzdioden: 20 mA
- Maximaler Ausgangsstrom: 100 mA
- Abmessungen (Leiterplatte ohne Buchsen, Stecker und PC-Kartenhalter): ca. 115 × 200 mm^2
- Gewicht: ca. 210 g

Anschlüsse

(siehe Abb. 1)

Übersicht der Buchsen und Stecker

Buchse, Stecker	Funktion
CN1	Zähler-Signaleingang
CN2	Zähler-Gateeingang
CN3	Zeitgeber-Ausgang
CN4	Zeitgeber-Eingang
CN5	ISA Busstecker

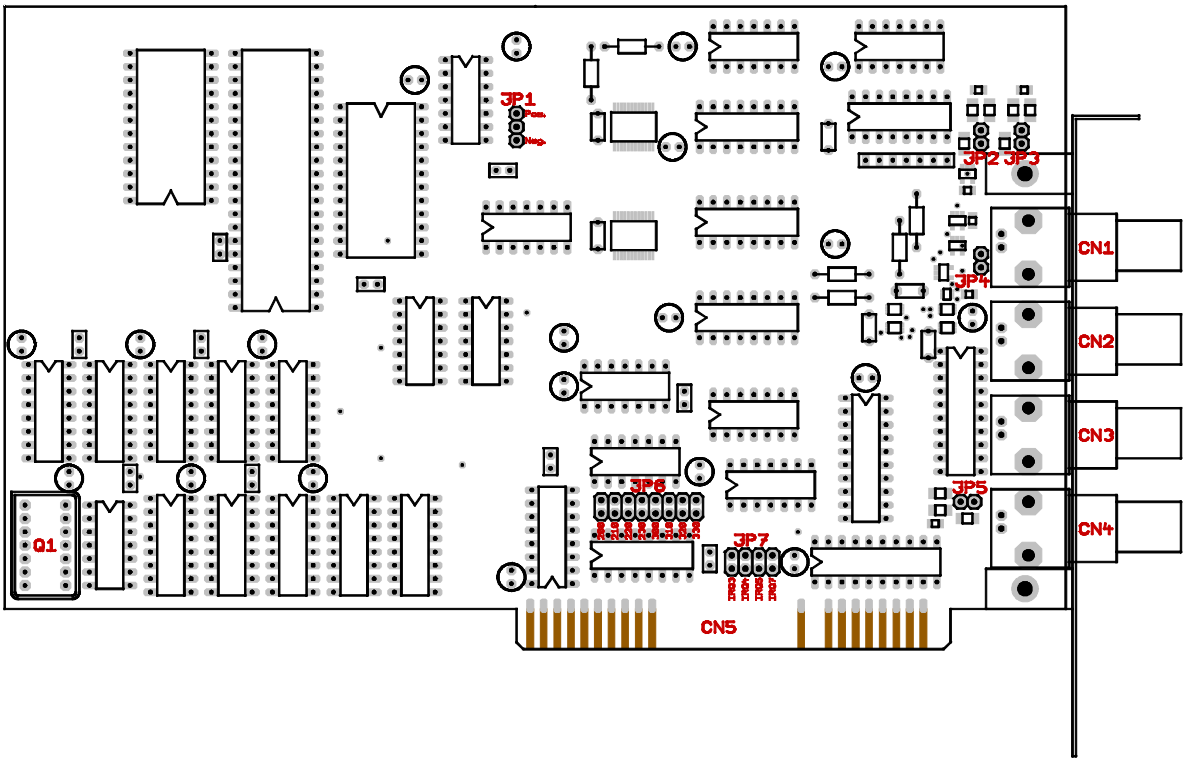


Abb. 1. Anschlüsse und Konfigurationselemente der Messkarte ISA-C32T12.

Konfiguration

(siehe Abb. 1)

Übersicht der Konfigurationsschalter (Jumpers)

Jumper	Funktion
JP1	Polarität des Zeitgeber-Ausgangs (CN3)
JP2	50 Ω Abschluss vom Zähler-Signaleingang (CN1)
JP3	50 Ω Abschluss vom Zähler-Gateeingang (CN2)
JP4	Anschluss vom Zähler-Gateeingang an den Zeitgeber-Ausgang
JP5	50 Ω Abschluss vom Zeitgeber-Eingang (CN4)
JP6	E/A-Adresse
JP7	IRQ-Leitung

50 Ω-Abschluss der Eingänge

JP2, JP3, JP5	Abschluss
offen (Voreinstellung)	47 kΩ *pull-up* Widerstand
geschlossen	50 Ω *pull-up* Widerstand (aktiver Abschluss)

Polarität des Zeitgeber-Ausgangs

JP1	Polarität
Mitte + oben geschlossen (Voreinstellung)	Positiv (Ausgang ist aktiv bei log. 1)
Mitte + unten geschlossen	Negativ(Ausgang ist aktiv bei log. 0)

Kopplung des Zählers mit dem Zeitgeber

JP4	Zähler-Gateeingang
offen (Voreinstellung)	extern steuerbar
geschlossen	angeschlossen an den Zeitgeber-Ausgang

Achtung: Ist der Jumper JP4 geschlossen, wird der Zähler-Gateeingang intern versorgt. An die Buchse CN2 darf in diesem Falle keine externe Leitung angeschlossen werden. Das Signal zur Steuerung des Zähler-Gateeingangs ist auf dem Zeitgeber-Ausgang CN3 verfügbar.

E/A-Adresse der Messkarte

(vgl. mit Abb. 1)

JP6	E/A-Adresse
links	200h
(Voreinstellung)	210h
	220h
	230h
	300h
	310h
	320h
rechts	330h

Achtung: Die Einstellung der E/A-Adresse muss mit der Konfiguration des Software-Treibers übereinstimmen (siehe Abschnitt "Installation").

IRQ-Leitung der Messkarte

(vgl. mit Abb. 1)

JP7	IRQ-Leitung
links	IRQ3
	IRQ4
	IRQ5
rechts	IRQ7
offen (Voreinstellung)	nicht angeschlossen

Bemerkung: Der Software-Treiber verwendet zur Zeit keine Unterbrechungen (*Interrupts*). Der Jumper ist für künftige Erweiterungen vorgesehen.

Weitere Konfigurationselemente

Q1	interner Quarzoszillator des Zeitgebers

Die Messkarte ist standardmäßig mit einem 10 MHz-Quarzoszillator ausgestattet. Werden andere Taktfrequenzen erwünscht, kann dieser mit einem entsprechenden 14-pin 5V Quarzoszillator ersetzt werden.

Funktion

Zähler

Der Zähler zählt Impulse auf seinem Signaleingang. Er hat eine Länge von 32 Bit, wobei das höchste Bit (MSB) während des Zählvorgangs nur gesetzt wird und somit als Anzeige des Zähler-Überlaufes verwendet wird. Das Bit wird bei dem 2^{31}-ten Puls gesetzt und kann nur zusammen mit dem gesamten Zähler zurückgesetzt werden. Der Zählvorgang erfolgt in den restlichen 31 Bits, d.h. gleicht der Inhalt des Zähler-Datenregisters 2^{32}–1, wird er nach dem Registrieren eines weiteren Pulses erneut auf 2^{31} gesetzt. Das Datenregister kann zu jeder Zeit vom Rechner ausgelesen werden und somit der Zählvorgang genau verfolgt werden. Da das Füllen der 31 Bits des Datenregisters auch bei der maximalen Signalfrequenz von 200 MHz etwa (2^{31}–1)/200 MHz ≈ 10 s dauert, ist durch ein wiederholtes Abtasten des Datenregisters mit einer Periode von nur einigen Sekunden auch beim Zähler-Überlauf eine genaue Rekonstruktion des Zählerzustandes möglich. Dadurch können mit Hilfe eines einfachen Algorithmus durch den Rechner mehr als 2^{32}–1 Pulse während einer Messung registriert werden.

Der Zählvorgang kann nur dann erfolgen, wenn der Zähler freigegeben wurde (Statusbit "Measurement Active", bzw. `CNT32_MeasAct`, siehe Abschnitt "Bits vom Zählerstatus") und wenn das Gate aktiv ist (Statusbit "Gate Active", bzw. `CNT32_GateAct`, siehe Abb. 2). Das Gate kann per Software aktiviert (Kontrollbit "Gate Control", bzw. `CNT32_GateCtrl`) oder direkt durch den Gate-Eingang des Zählers gesteuert werden (siehe Abb. 3). Dazu muss die Software-Steuerung vom Gate ausgeschaltet werden (Kontrollbit "Gate Control" deaktiviert) und der Gate-Eingang muss freigegeben werden (Kontrollbit "Gate Enable", bzw. `CNT32_GateEn`). In beiden Fällen wiederspiegelt das Statusbit "Gate Active" den Zustand des Zähler-Gates. Wird zur Steuerung des Zähler-Gates der Gate-Eingang benutzt, kann dieser indirekt durch Auslesen die-

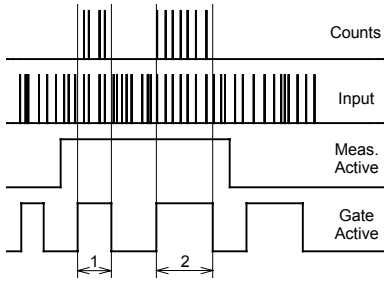


Abb. 2. Schematische Darstellung der Funktion des Zählers. Der Zählvorgang erfolgt dann, wenn der Zähler freigegeben und das Gate aktiv ist (Zeitintervalle 1 und 2).

ses Statusbits durch den Rechner abgefragt werden (vgl. mit Abb. 3). Die Funktion der Gate-Logik kann durch folgende Boolesche Funktion beschrieben werden:

Gate Active = (Gate Input AND Gate Enable) OR Gate Control.

Der Zähler wird nach seinem Rücksetzen durch die Aktivierung des Kontrollbits "Measurement Run" entweder für eine Einzelmessung oder für eine Messsequenz freigegeben (siehe Abb. 4). Dies wird anhand des Kontrollbits "Measurement Single" (`CNT32_MeasSngl`) zum Zeitpunkt der Zählerfreigabe (Zeitpunkte 1 und 4 in Abb. 4) entschieden. Eine weitere Zustandsänderung dieses Kontrollbits während der Messung spielt keine Rolle. Wird eine Einzelmessung gestartet, wird der Zähler nach einer Aktivierung des Gates wieder gesperrt (Zeitpunkt 2 in Abb. 4). Weitere Zustandsänderungen sowohl des Gates als auch des Kontrollbits "Measurement Run" bleiben danach unberücksichtigt. Die Sperrung des Zählers wird erst durch das Zurücksetzen des Zählers und seiner Kontrolllogik durch eine kurze Aktivierung des Kontrollbits "Reset Control" aufgehoben (Zeitpunkt 3 in Abb. 4). Eine Messsequenz kann im Gegensatz zu einer Einzelmessung aus mehreren Aktivierungen des Gates bestehen. Die Messung kann weder durch eine Zustandsänderung des Gates noch durch eine Änderung des Kontrollbits "Measurement Run" beendet werden. Dies geschieht erst durch das Zurücksetzen des Zählers (Zeitpunkt 5 in Abb. 4).

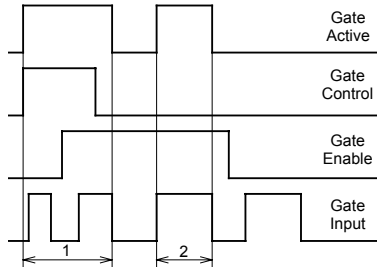


Abb. 3. Schematische Darstellung der Funktion des Zähler-Gates. Das Gate wird abwechselnd per Software und durch den Gate-Eingang aktiviert. Den aktiven Zustand (Zeitintervalle 1 und 2) spiegelt das Statusbit "Gate Active" wider.

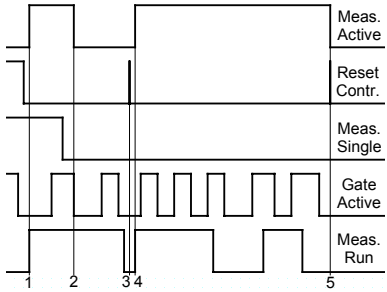


Abb. 4. Schematische Darstellung der Funktion der Kontroll-Logik des Zählers. Der Zähler wird zum Zeitpunkt 1 für eine Einzelmessung freigegeben. Nach der ersten Aktivierung des Gates wird er im Zeitpunkt 2 wieder gesperrt. Die Kontrolllogik des Zählers wird zum Zeitpunkt 3 zurückgesetzt, so dass eine neue Messung möglich ist. Diese wird zum Zeitpunkt 4 als eine Messsequenz gestartet und endet im Zeitpunkt 5 mit dem Zurücksetzen des Zählers.

Unabhängig von dem Zustand einer Messung kann diese jederzeit durch das Zurücksetzen des Zählers unterbrochen werden. Ebenfalls kann während einer Messung die Signalpolarität geändert werden (Kontrollbit "Input Negative", bzw. `CNT32_InputNeg`). Die Einstellung der Signalpolarität soll den Ruhezustand des Eingangssignals widerspiegeln. Liegt der Ruhezustand beispielsweise bei der log. 1 und wurde das Kontrollbit "Input Negative" fälschlicherweise deaktiviert, wird beim inaktiven Eingangssignal bei jedem Öffnen des Gates ein parasitärer Puls gezählt.

Um eine Messung zuverlässig zu starten, soll eine folgende Kontroll-Sequenz ausgeführt werden.

- Zurücksetzen des Zählers und seiner Kontrolllogik durch eine kurze Aktivierung und Deaktivierung des Kontrollbits "Reset Control". Dabei sollen ebenfalls die Kontrollbits "Gate Enable", "Gate Control" und "Measurement Run" deaktiviert werden. Gleichzeitig kann der Zustand der Kontrollbits "Measurement Single " und "Input Negative" definiert werden.
- Freigabe des Zählers durch die Aktivierung des Kontrollbits "Measurement Run".
- Freigabe des Zähler-Gateeingangs durch die Aktivierung des Kontrollbits "Gate Enable" oder Öffnung des Zähler-Gates durch die Aktivierung des Kontrollbits "Gate Control".

Zu jedem Zeitpunkt kann der Zustand der Messung durch das Abfragen der Statusbits "Measurement Active" und "Gate Active" und der Zählerstand durch das Auslesen des Datenregisters ermittelt werden. Somit kann eine Akquisitionssoftware beispielsweise feststellen, ob das Gate aktiv ist und ob bei einer Einzelmessung bereits eine Aktivierung des Gates vorlag.

Um eine Messung zu beenden und den Zählerstand zuverlässig durch den Rechner auszulesen, soll die folgende Kontroll-Sequenz ausgeführt werden.

- Sperren des Zähler-Gates durch die Deaktivierung der Kontrollbits "Gate Enable" und "Gate Control".
- Ermitteln des Zählerstandes durch das Auslesen des Datenregisters des Zählers

Danach kann wie oben beschrieben eine neue Messung gestartet werden.

Zeitgeber

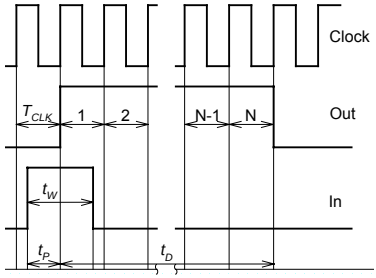


Abb. 5. Signalverläufe und Definitionen der Zeitverzögerungen des Zeitgebers. Die dargestellten Signalformen entsprechen einem positiven Ausgangssignal und dem Auslösen durch eine steigende Flanke.

Jeder Kanal des Zeitgebers wird durch seinen Trigger-Eingang aktiviert. Als Antwort auf eine vordefinierte Signalflanke auf dem Trigger-Eingang wird auf dem Kanalausgang ein Puls erzeugt (siehe Abb. 5). Der Puls auf dem Triggereingang muss länger sein als ein minimal erlaubter Wert (siehe Abschnitt "Technische Daten"). Die maximale Länge des Eingangspulses ist dagegen nicht limitiert und, die Pulslänge hat keinen Einfluss auf die Funktion des Zeitgebers. Der Zeitgeber wird durch den Eingangspuls aktiviert und der Kanalausgang wird nach einer Verzögerung t_P synchron mit dem internen Quarzoszillator aktiviert. Die Verzögerung t_P ist durch die interne Verzögerung in den Logikbausteinen $t_{P,Int}$ des Zeitgebers und durch die Periode des Taktsignals T_{CLK} gegeben. Die interne Verzögerung in den Logikbausteinen hat einen typischen Wert von $t_{P,Int}$ = 60 ns (maximal 110 ns, siehe auch Abschnitt "Technische Daten"). Der tatsächliche Wert der Verzögerung t_P hängt von der zeitlichen Verschiebung des Eingangspulses relativ zu dem Taktsignal und wenn diese nicht kohärent sind, liegt der statistische Mittelwert bei $t_{P,Int} + T_{CLK}/2$.

Der Ausgangspuls jedes Kanals des Zeitgebers wird synchron mit dem internen Quarzoszillator gestartet und gestoppt. Durch das Synchronisieren mit dem Quarzoszillator wird eine exzellente Reproduzierbarkeit der Pulslänge erreicht, die lediglich durch die Stabilität des Quarzoszillators und durch den Jitter der Logikbausteine gegeben ist. Die Pulslänge t_D beträgt immer ein ganzzahliges Vielfaches der Periode des Taktsignals T_{CLK}:

$$t_D = N \cdot T_{CLK},$$

wobei die natürliche Zahl N beim Programmieren des Kanals festgelegt wird. Die Zahl N wird an den Zeitgeber als ein Steuerwort im speziellen 15-Bit Fliesskomma-Format übergeben:

$$N = M \cdot 10^E,$$

wobei M für eine 12-Bit Mantisse und E einen 3-Bit Exponent stehen. Die Mantisse M enthält eine BCD (*Binary-Coded Decimal*) Zahl und darf

Werte von 4 bis 1600 annehmen. Bei dem Exponenten sind alle zulässigen Werte von 0 bis 7 erlaubt. Somit kann die Anzahl *N* der Perioden des Taktsignals über einen weiten Bereich von $4 \cdot 10^0 = 4$ bis $1600 \cdot 10^7 = 1.6 \cdot 10^{10}$ variiert werden. Die genaue Kenntnis des internen Fliesskomma-Formates ist zum Programmieren des Zeitgebers nicht erforderlich, denn die Konversion ist in der Benutzerfunktion `TMR12_ConvertDelay` implementiert, die aus der gewünschten Zeitverzögerung das Steuerwort rechnet.

Beispiel:

Taktfrequenz $f_{CLK} = 10$ MHz
(bei anderen Taktfrequenzen ändern sich die Angaben proportional)

Pulslänge	Zeitauflösung (minimaler Zeitschritt)
≤160 µs	100 ns
160 µs…1.6 ms	1 µs
1.6 ms…16 ms	10 µs
16 ms…160 ms	100 µs
160 ms…1.6 s	1 ms
1.6 s…16 s	10 ms
16 s…160 s	100 ms
160 s…1600 s	1 s

Die in Abb. 5 dargestellten Signalformen beziehen sich auf den Fall des Auslösens durch eine steigende Flanke und der Erzeugung eines positiven Ausgangssignals. Beim Umschalten des jeweiligen Signalparameters werden die Signalformen entsprechend invertiert (logische Negation), die Zeitabfolge der Signale ändert sich dagegen durch das Umschalten nicht.

Die beiden Kanäle des Zeigebers sind intern gekoppelt: der Trigger-Eingang des zweiten ist mit dem Ausgang des ersten Kanals verbunden. Dadurch werden auf dem Gesamtausgang (d.h. auf dem Ausgang des zweiten Kanals) des Zeitgebers zeitlich verschobene Pulse im Vergleich zum Trigger-Signal erzeugt. Die Pulslänge des ersten Kanals definiert die Verschiebung des Ausgangspulses, die des zweiten Kanals die tatsächliche Pulslänge. Aufgrund der Zeitverzögerung t_P zwischen den Eingangs- und den Ausgangspulsen ist jedoch die Verschiebung des Ausgangspulses um diese Zeitverzögerung höher. Bei der Taktfrequenz $f_{CLK} = 10$ MHz beträgt diese Differenz zwei Perioden des Taktsignals, d.h. $2T_{CLK} = 200$ ns.

Installation

Hardware

Für die Messkarte ISA-C32T12 benötigen Sie einen freien ISA-Steckplatz in einem IBM-kompatiblen PC. Vor dem Einbau ist die Konfiguration der Messkarte zu überprüfen. Insbesondere die E/A-Adresse muss eingestellt werden, damit Konflikte mit anderer, bereits eingebauter Hardware vermieden werden. Die Kenntnis der E/A-Adresse ist für die Installation der Software erforderlich.

Schalten Sie den Rechner aus, öffnen Sie das Gehäuse, stecken Sie die Messkarte ein und befestigen Sie diese mit der dafür im Rechner-Gehäuse vorgesehenen Schraube. Schließen Sie das Gehäuse und schalten den Rechner wieder an.

Software

Je nach der Version des Windows-Betriebsystems stehen zwei Varianten der Software-Treiber zur Verfügung. Die restliche Software ist Plattform-unabhängig. Die Software wird in Form eines ZIP-Archivs bereitgestellt. Extrahieren Sie für die Durchführung der Softwareinstallation alle Dateien aus diesem Archiv in ein temporäres Installationsverzeichnis.

Windows 95, 98, ME

Unter diesen früheren Windows-Versionen ist kein Systemtreiber erforderlich. Kopieren Sie die Treiber-Bibliothek `I2C.dll` und die Konfigurationsdatei `I2C-IFC.ini` aus dem Unterverzeichnis `Win95` in das Verzeichnis, das Sie für die Steuersoftware vorgesehen haben. Wenn erforderlich, ändern Sie in der Konfigurationsdatei die Einstellung der E/A-Adresse der Messkarte:

```
BaseAddress=210h
```

Kopieren Sie die restliche Software aus dem Installationsverzeichnis in das von Ihnen ausgewählte Verzeichnis und starten Sie das Testprogramm `C32T12Control.exe`.

Windows NT, 2000, XP

Diese moderneren Windows-Versionen benötigen einen Systemtreiber. Alle erforderlichen Dateien befinden sich im Unterverzeichnis `WinNT`

des Installationsverzeichnisses. Je nach der Konfiguration Ihres Rechners benötigen Sie möglicherweise Administrationsrechte, um die Installation durchzuführen. Der Systemtreiber wird anhand der Anleitung `Install.txt` wie folgt installiert:

- Kopieren Sie den Treiber in das Windows-Treiberverzeichnis:

```
copy I2CDriver.sys
   %SystemRoot%\system32\drivers\
```

- Editieren sie die Datei `I2CDriver.reg`, um die E/A-Adresse der Messkarte einzustellen:

```
"IoBaseAddress"=dword:00000210
```

- Fügen Sie diese Einträge in die Windows-Registrierung entweder durch Doppelklicken auf die Datei `I2CDriver.reg` oder durch folgende Anweisung ein:

```
REGEDIT I2CDriver.ini
```

- Starten Sie den Rechner neu.

Kopieren Sie nach dem Neustart die Treiber-Bibliothek `I2C.dll` aus dem Unterverzeichnis `WinNT` und ebenfalls die restliche Software aus dem Installationsverzeichnis in das von Ihnen ausgewählte Verzeichnis und starten Sie das Testprogramm `C32T12Control.exe`.

Sollten bei der Installation Probleme aufgetreten sein, ist zu überprüfen, ob der Treiber erfolgreich in das Windows-System installiert wurde. Kontrollieren Sie dazu, ob ein Gerät mit dem Namen `I2C Port Driver` in der Liste der Windows-Treiber vorhanden ist und ob es als gestartet gekennzeichnet ist. Die Liste der Windows-Treiber kann unter Windows NT in der Systemsteuerung (*Control Panel*) in der Dialogbox "Geräte" (*Devices*) eingesehen werden. Unter Windows 2000 und XP öffnen Sie dazu die Dialogbox "System" (*System*), wählen die Lasche "Hardware" (*Hardware*) und klicken auf die Schaltfläche "Geräte-Manager" (*Device Manager*). Im Geräte-Manager blenden Sie schließlich über das Menü alle Geräte ein (Anzeige – Ausgeblendete Geräte anzeigen, *View – Show hidden devices*).

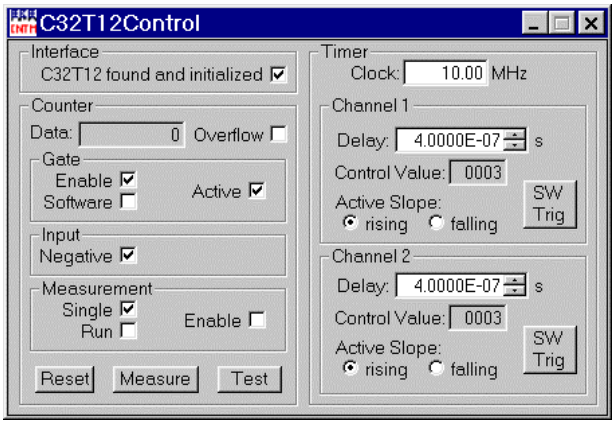
Steuersoftware

Mit dem Beispielprogramm `C32T12Control.exe` kann die Funktion des Software-Treibers und der Messkarte ISA-C32T12 selbst getestet

werden. Das Testprogramm benötigt zur Ausführung folgende dynamische Linkbibliotheken:

Treiber-Bibliothek `I2C.dll`
Software-Schnittstelle `C32T12.dll`
C-Bibliothek `cw3220.dll`
Object Windows Bibliotheken `owl50f.dll` und `bids50f.dll`

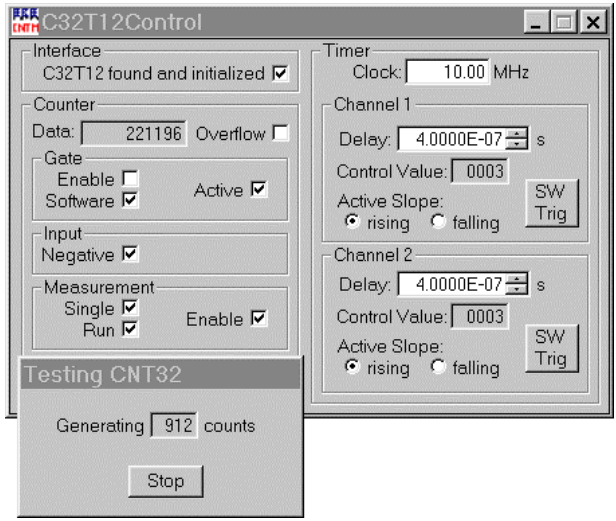
War die Installation fehlerfrei und können alle Linkbibliotheken vom Testprogramm gefunden werden, startet das Programm ohne eine Fehlermeldung, die Schaltfläche *C32T12 found and initialized* ist markiert und alle Kontrollelemente auf dem Programmfenster sind funktionsfähig (d.h. nicht grau gefärbt):



Sowohl der Zähler als auch der Zeitgeber können durch die Kontrollelemente auf dem Programmfenster manuell gesteuert werden. Die Kontrollelemente der Gruppe *Counter* entsprechen den Kontroll- und Statusbits des Zählers und seinem Datenregister. Alle Daten des Zählers werden etwa jede 50 ms abgefragt und aktualisiert. Die Kontrollelemente der Gruppe *Timer* entsprechen den Variablen zur Steuerung des Zeitgebers. Mit der Editierfläche *Clock* kann die Frequenz des internen Quarzoszillators eingestellt werden, die Editierfläche *Delay* dient der Einstellung und der Anzeige der aktuellen Zeitverzögerung. Die Anzeigefläche *Control Value* zeigt den hexadezimalen Wert des Steuerwortes für jeden Kanal an. Ferner kann die aktive Signalflanke eingestellt werden (*Active Slope: rising, falling*) und der Zeitgeber manuell gestartet werden (Schaltfläche *SW Trig*).

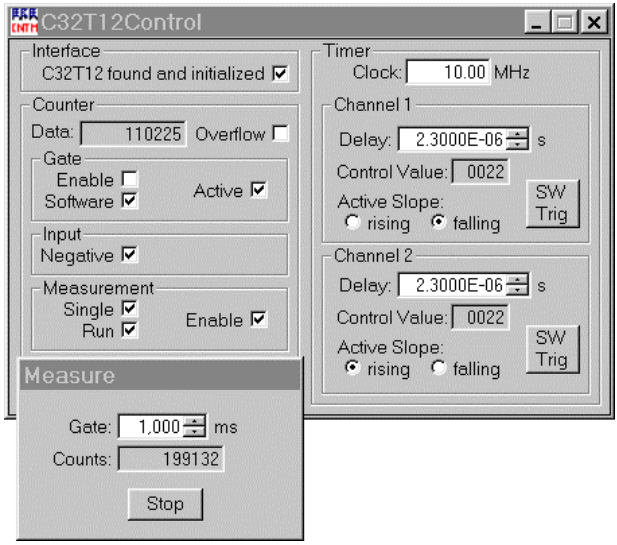
Sind die Eingänge des Zählers nicht angeschlossen (d.h. sowohl die Buchsen CN1 und CN2 als auch der Jumper JP4 sind geöffnet), kann ein

Software-Test des Zählers durchgeführt werden. Klicken Sie auf die Schaltfläche *Test*, um den Test zu starten:



Durch Umschalten der Signalpolarität generiert das Programm eine zufällige Anzahl künstlicher Pulse auf dem Zähler-Signaleingang und kontrolliert die Übereinstimmung deren Anzahl mit der durch den Zähler gezählten Pulse. Wird eine Diskrepanz festegestellt, erfolgt eine Fehlermeldung.

Mit einem weiteren Test kann die Funktion des Zeitgebers überprüft werden. Verbinden Sie mit zwei BNC-Kabeln und einem BNC-T-Stück den Ausgang des Zeitgebers mit seinem Eingang und mit dem Zähler-Signaleingang. Ändern Sie die Einstellung der aktiven Signalflanke des ersten Kanals des Zeitgebers auf fallend (*falling*) und betätigen Sie die Schaltfläche *SW Trig*, um den Zeitgeber zu starten. Klicken Sie danach auf die Schaltfläche *Measure*, um den Test durchzuführen:



Mit der Editierfläche *Gate* kann die Messdauer eingestellt werden, die Anzeigefläche *Counts* zeigt die Anzahl der durch den Zähler während der eingestellten Messdauer gezählten Pulse an. In diesem Modus triggert sich der Zeitgeber selbst und funktioniert wie ein Frequenzgenerator. Die Frequenz kann durch die Änderung der Verzögerungen beider Kanäle eingestellt werden. Die beiden Halbperioden sind bei der Taktfrequenz $f_{CLK} = 10$ MHz jeweils um zwei Perioden des Taktsignals, d.h. um $2T_{CLK} = 200$ ns länger als die angezeigten Werte (für Details siehe Abschnitt "Funktion: Zeitgeber"). Aufgrund der endlichen Prozessor-Geschwindigkeit und anderen Aktivitäten des Systems, sowie der Eigenschaften von dem jeweiligen Betriebsystem kann die tatsächliche Messdauer von der eingestellten abweichen. Dies äußert sich in Schwankungen der Anzeige *Counts*.

Programmieren

Dem Benutzer steht der Software-Treiber in Form einer DLL zur Verfügung. Alle Benutzerfunktionen für die Messkarte ISA-C32T12 sind in der dynamischen Linkbibliothek `C32T12.dll` enthalten. Für ihre Funktion sind noch die Treiber-Bibliothek `I2C.dll` und die C-Bibliothek `cw3220.dll` erforderlich.

Die Benutzerfunktionen in der dynamischen Linkbibliothek `C32T12.dll` können aus allen gängigen Programmiersprachen aufgerufen werden. Die Details entnehmen Sie dem Benutzerhandbuch Ihres Compilers. Die folgenden Beispiele sind für den Compiler Borland C++ Version 5.0 vorgesehen.

Die Definitionen der Benutzerfunktionen sind in der Datei `C32T12.H` enthalten.

Fehlerbehandlung

Funktion C32T12_ErrorMessage

```
const char * C32T12_ErrorMessage (int
   ErrorCode);
```

Liefert für den `ErrorCode` die entsprechende Fehlermeldung. Der `ErrorCode` ist der Rückgabewert der meisten Benutzerfunktionen. Ist er gleich 0, ist die Benutzerfunktion fehlerfrei ausgeführt worden. Andere Rückgabewerte deuten einen Schnittstellen-Fehler.

Initialisierung

Funktion C32T12_Open

```
int C32T12_Open();
```

Öffnet die Schnittstelle und liefert einen `ErrorCode`. Sie muss als die erste Funktion während der Kommunikation mit der Messkarte ausgeführt werden.

Funktion C32T12_Close

```
int C32T12_Close();
```

Schließt die Schnittstelle und liefert einen `ErrorCode`. Sie muss als die letzte Funktion während der Kommunikation mit der Messkarte ausgeführt werden.

Funktion C32T12_CheckPresence

```
int C32T12_CheckPresence();
```

Stellt fest, ob die Messkarte vorhanden ist und ob ihre Computer-Schnittstelle einwandfrei funktioniert. Liefert 0, wenn die Messkarte fehlerfrei antwortet, -1, wenn ein Teil der Schnittstelle nicht vorhanden ist, oder einen `ErrorCode`, wenn ein Schnittstellen-Fehler aufgetreten ist.

Bits vom Zählerstatus

```
CNT32_GateCtrl /* Gate Control */
```

Kontrollbit, aktiviert das Zähler-Gate unabhängig vom Zustand des Zähler-Gateeingangs.

```
CNT32_MeasRun /* Measurement Run */
```

Kontrollbit, aktiviert die Messung.

```
CNT32_InputNeg /* Input Negative */
```

Kontrollbit, invertiert den Zähler-Signaleingang, geeignet zum Zählen negativer Pulse.

```
CNT32_MeasSngl /* Measurement Single */
```

Kontrollbit, wenn gesetzt, führt der Zähler einzelne Messungen durch; die Messung wird nach der ersten Aktivierung vom Gate beendet.

```
CNT32_ResetCtrl /* Reset Control */
```

Kontrollbit, setzt sowohl das Datenregister des Zählers als auch die Kontrolllogik zurück.

```
CNT32_GateEn /* Gate Enable */
```

Kontrollbit, gibt den Zähler-Gateeingang frei.

```
CNT32_MeasAct /* Measurement Active */
```

Statusbit, zeigt eine laufende Messung an.

```
CNT32_GateAct /* Gate Active */
```

Statusbit, zeigt ein aktives Gate an.

Zähler-Funktionen

Funktion CNT32_GetData

```
int CNT32_GetData (unsigned long * Data,
   bool * Overflow);
```

Liest die 32-Bit Zählerdaten (Anzahl der Pulse) in die Variable `Data` und den Überlauf-Zustand in die Variable `Overflow` ein und liefert einen `ErrorCode`. Der Überlauf `Overflow` zeigt an, dass mehr als 2^{31} Pulse gezählt wurden und dass daher ein Überlauf vorliegen kann.

Funktion CNT32_Reset

```
int CNT32_Reset();
```

Setzt den Zähler zurück und liefert einen `ErrorCode`. Die Funktion setzt sowohl das Datenregister des Zählers als auch die Kontrolllogik zurück. Diese Funktion soll als erste während einer Messsequenz aufgerufen werden. Nach dem Einschalten des Rechners wird das Bit `CNT32_ResetCtrl` im Zählerstatus gesetzt. Somit befindet sich der Zähler auf der Messkarte im gesperrten Zustand und kann keine Messung durchführen, solange sie durch die Funktion `CNT32_Reset` nicht zurückgesetzt wird.

Funktion CNT32_GetStatus

```
int CNT32_GetStatus (BYTE * Status);
```

Liest den 8-Bit Zählerstatus in die Variable `Status` ein und liefert einen `ErrorCode`.

Funktion CNT32_SetStatus

```
int CNT32_SetStatus (BYTE Status);
```

Setzt den 8-Bit Zählerstatus anhand der Variable `Status` und liefert einen `ErrorCode`.

Funktion CNT32_ChangeStatus

```
int CNT32_ChangeStatus (BYTE *
   StatusBytes, unsigned Count);
```

Setzt wiederholt den 8-Bit Zählerstatus anhand des Feldes `StatusBytes` und liefert einen `ErrorCode`. Die Variable `Count` muss der Länge des Feldes `StatusBytes` gleichen und gibt gleichzeitig die Anzahl der Änderungen vom Zählerstatus an.

Beispiele:

Die Funktion `CNT32_Reset` entspricht beispielsweise etwa dem folgenden Code:

```
BYTE StatusBytes [2];
StatusBytes [0] = CNT32_ResetCtrl;
StatusBytes [1] = 0;
CNT32_ChangeStatus (StatusBytes, 2);
```

Eine Messung kann durch manuelle die Aktivierung des Gates durch folgenden Code gestartet werden (vgl. mit Abschnitt "Funktion: Zähler"):

```
BYTE StatusBytes [4];
StatusBytes [0] = CNT32_ResetCtrl;
StatusBytes [1] = 0;
StatusBytes [2] = CNT32_MeasRun;
StatusBytes [3] = CNT32_MeasRun |
  CNT32_GateCtrl;
ChangeStatus (StatusBytes, 4);
```

Funktion CNT32_FlipStatusBit

```
int CNT32_FlipStatusBit (BYTE BitMask);
```

Ändert den Zählerstatus anhand der Bitmaske `BitMask` und liefert einen `ErrorCode`.

Beispiele:

Folgender Code
```
CNT32_FlipStatusBit (CNT32_InputNeg);
```
ändert die Polarität des Eingangssignals.

Der Code
```
CNT32_FlipStatusBit (CNT32_InputNeg |
   CNT32_MeasSngl);
```
ändert die Polarität des Eingangssignals und schaltet gleichzeitig zwischen einer Einzelmessung und einer Messsequenz um.

Zeitgeber-Funktionen

Funktion TMR12_ConvertDelay

```
WORD TMR12_ConvertDelay (double * Delay,
   double ClockFrequency, int Delta);
```

Liefert für die gegebene Zeitverzögerung `Delay` (in Sekunden) und die Frequenz des internen Quarzoszillators `ClockFrequency` (in Hertz) das Steuerwort für die Steuerung des Zeitgebers (Aufruf der Funktion `TMR12_SetDelay`, siehe auch Abschnitt "Funktion: Zeitgeber"). Die Variable `Delay` wird durch die Funktion so verändert, dass sie der nächsten möglichen einstellbaren Zeitverzögerung des Zeitgebers entspricht. Für diesen Wert der Zeitverzögerung wird das Steuerwort berechnet. Als letzter Parameter wird die Änderung der Zeitverzögerung `Delta` angegeben. Ist dieser Parameter ungleich Null, ändert die Funktion die Zeitverzögerung `Delay` um `Delta` Schritte (vgl. mit der Tabelle im Abschnitt "Funktion: Zeitgeber").

Beispiel:

Ist `ClockFrequency=1E7` und `Delay=1E-6`, gibt die Funktion `TMR12_SetDelay` folgende Werte zurück:

Delta	Delay	Rückgabewert (Steuerwort)
0	1E-6	9=09h
1	1.1E-6	16=10h
2	1.2E-6	17=11h
3	1.3E-6	18=12h
-1	9E-7	8=08h
-2	8E-7	7=07h
-3	7E-7	6=06h

Ist `ClockFrequency=1E7`, `Delay=1.01E-6` und `Delta=0` ändert die Funktion die Zeitverzögerung (`Delay=1E-6`) und gibt ebenfalls das Steuerwort von `9=09h` zurück.

Funktion TMR12_SetDelay

```
int TMR12_SetDelay (WORD ControlValue,
   bool FallingSlope, int Channel);
```

Stellt für den Kanal `Channel` die Zeitverzögerung anhand des Wertes `ControlValue` und der Flankenpolarität (`FallingSlope`) ein und liefert einen `ErrorCode`. Das Steuerwort `ControlValue` kann als der Rückgabewert der Funktion `TMR12_ConvertDelay` gewonnnen werden.