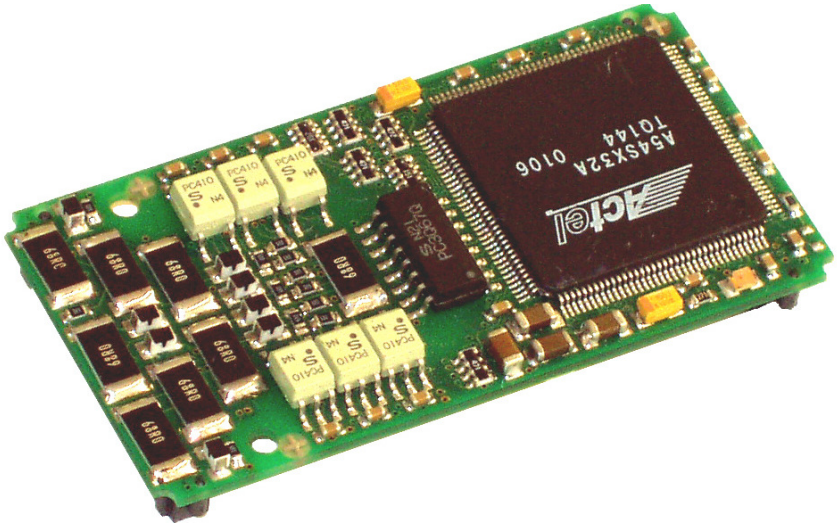
X-C16-3i

Zähler-Modul mit 3 universellen Zählerkanälen,
12 Opto-entkoppelten Ein- und 8 Ausgängen



9. X-C16-3i

Inhaltsverzeichnis

9.1. Beschreibung

Das Modul stellt 3 16-Bit-Zähler zur Verfügung. Diese sind kaskadierbar. Weiterhin hat das Modul 12 externe, einzeln opto-entkoppelte High-Speed Eingänge. Die Eingänge können per Software für unterschiedliche Funktionen konfiguriert werden. Alle 12 Eingänge sind interruptfähig und können als allgemeine Eingänge per Software abgefragt werden. Weiterhin können sie auf verschiedene Art und Weise als Zähler- und Steuereingänge für die 3 Zählerkanäle konfiguriert werden.

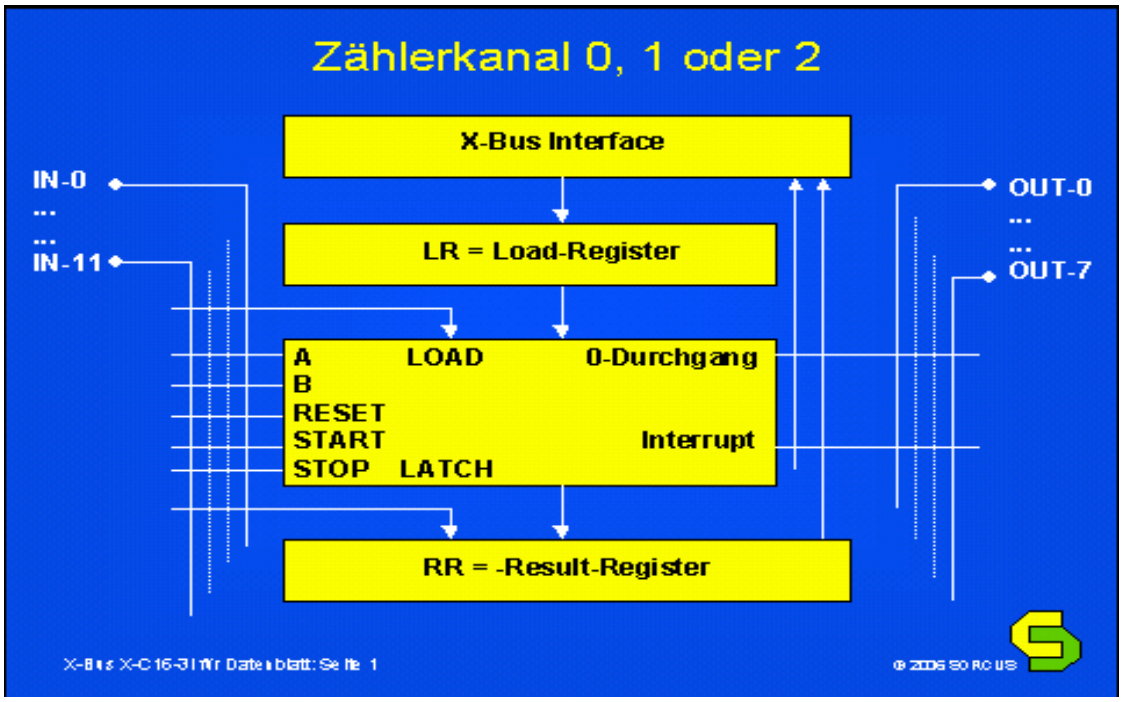
Das Modul ist in 3 Bestückungsvarianten lieferbar, die sich durch die Eingangspegel unterscheiden:

Typ	Subtyp	Modul	Pegel	Schwelle log. 0	Schwelle log. 1
54	0	X-C16-3i/L	Logik	< 2,2 Volt	> 4 Volt
54	1	X-C16-3i/P	Prozess	< 5 Volt	> 13 Volt
54	2	X-C16-3i/T	TTL	< 0,8 Volt	> 2,4 Volt

Auf Wunsch können auch gemischte Bestückungen und andere Schwellen eingestellt werden.

Zusätzlich bietet das Modul 8 einzeln opto-entkoppelte Ausgänge. Sie können per Software gesetzt werden und haben keine feste Zuordnung zu den Zählerkanälen oder zu den Eingängen.

9.2. Blockschaltbild



9.3. Modul-Device-Treiber (MDD)

9.3.1. Installation

Der Modul-Device-Treiber für das OsX hat die Programmnummer 8036h und den Dateinamen mxc163.exe. Der Modul-Device-Treiber für Windows hat den Namen mxc163.sys. Der Modul-Device-Treiber für Windows CE hat den Dateinamen mxc163.dll und der Modul-Device-Treiber für CEoX hat den Dateinamen mxc163_ceox.dll. Die Installation aus einem PC-Programm (z. B. für Steckplatz 1, Layer 0):

Error = max_load_mdd (hModul, 1, 0, 0, 0x8036, NULL, &hMDD);

Befehl in einer INS-Datei (z. B. für Steckplatz 1, Layer 0):

MAXLOADMDD slot=1 layer=0 progno=8036

9.3.2. Kanaleigenschaftsstruktur CPS_XC163_A

Die CPS für das Modul hat den Namen CPS_XC163_A. CPS_XC163_A wurde gegenüber CPS_XC163 um den Parameter *ulCallbackEvent* erweitert.

9.3.3. LED

Um auf die LED des Moduls zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_LED*	Kanal zu einer LED
.usIndexFirst	*0*	Nummer der LED
.usIndexLast	*0*	Nummer der LED
.usFlags	*0*	Keine Bedeutung
.usReadMode	*IO_MODE_DIRECT*	Direkter Lesezugriff
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
.usMode	*0*	Keine Bedeutung

Eingabe- und Ausgabedienst

Um die LED ein- bzw. auszuschalten muss eine 1 bzw. 0 in den Kanal geschrieben werden. Der Datentyp des Kanals ist DATA_UCHAR.

- **max_write_channel_uchar**
- **max_read_channel_uchar**

9.3.4. Digitale Eingänge

Das Modul hat 12 digitale Eingänge (DIN-0...DIN-11). Die Eingänge können entweder direkt oder aus einem internen Zwischenspeicher (Latch) gelesen werden.

Das Übertragen der aktuellen Eingangszustände in den Zwischenspeicher (Latchen) kann über einen externen Eingang und/oder per Software erfolgen. Das softwaremäßige Latchen erfolgt über einen DEVICE_TRIGGER-Kanal (siehe „Latchen der digitalen Eingänge per Software“) und ist auch dann möglich, wenn bereits ein externer Latcheingang verwendet wird. Die Quelle für das externe Latchen wird mit einem Steuer-Kanal (siehe „Definition des externen Latch-Eingangs für die digitalen Ein-

gänge“) festgelegt. Um auf die digitalen Eingänge zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_DIN*	Kanal auf einen digitalen Eingang
.usIndexFirst	*0 ... 11*	Nummer des ersten Eingangs
.usIndexLast	*0 ... 11*	Nummer des letzten Eingangs
.usFlags	*0*	Keine Bedeutung
	_CP_EXCLUSIVE	Falls der Kanal mit Callback Funktionalität geöffnet wird, muss diese Flag gesetzt sein. Es wird kein Ereignis signalisiert.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT*	Direkter Lesezugriff
	IO_MODE_LATCH	Werte aus internem Zwischenspeicher lesen.
.usWriteMode	*0*	Keine Bedeutung.
.usMode	*0*	Keine Bedeutung.
.ulCallbackEvent[1]	*0*	Es wird kein Ereignis signalisiert.
	XC163_EVENT_NEG_EDGE	Eine positive Flanke wird signalisiert.
	XC163_EVENT_POS_EDGE	Eine negative Flanke wird signalisiert. Da ein DIN nur entweder eine positive oder negative Flanke signalisieren kann, dürfen die beiden Werte nicht gleichtzeitig gesetzt werden.
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert.

Eingabedienst

Der Datentyp des Kanals ist DATA_USHORT, der Zugriff erfolgt mit:

- **max_read_channel_ushort**

Callback-Funktion

Über das Strukturelement ulCallbackEvent kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 2 ULONG-Werte übergeben. Im ersten ULONG-Wert wird die entsprechende Konstante, die in *ulCallbackEvent* angegeben wurde, zurückgegeben. Der zweite ULONG-Wert zeigt an, bei welchen DINs eine Flanke aufgetreten ist. Die unteren 2 Bytes signalisieren dabei, dass eine positive Flanke eines DINs aufgetreten ist. Die oberen 2 Bytes si-

[1] bei Benutzung von CPS-Typ CPS_XC163_A

gnalisieren dabei, dass eine negative Flanke eines DINs aufgetreten ist. Jedes Bit repräsentiert einen DIN. Die Daten sind rechtsbündig angeordnet. Bei einem Kanal zu einem DIN-5...11 würde also Bit-0...6 eine pos. Flanke und Bit-16...22 eine neg. Flanke an DIN-5...11 anzeigen.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.
- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXLOADMDD slot=0 layer=0 progno=8FFF
  ```
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

9.3.5. Definition des externen Latch-Eingangs für die digitalen

9.3.6. Eingänge

Mit diesem Device wird festgelegt, welcher digitale Eingang die digitalen Eingänge in den internen Zwischenspeicher (Latch) des Moduls übertragen soll. Um darauf zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_CTRL*	Kanal auf ein Kontroll-Device
.usIndexFirst	*XC163_DIN_LATCH_CTRL*	Nummer des Kontroll-Devices
.usIndexLast	*XC163_DIN_LATCH_CTRL*	Nummer des Kontroll-Devices
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv
.usReadMode	*IO_MODE_DIRECT*	Direkter Lesezugriff
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
.usMode	*0*	Keine Bedeutung

Eingabedienst

Der Datentyp des Kanals ist DATA_USHORT, der Zugriff erfolgt mit:

- **max_write_channel_ushort**
- **max_read_channel_ushort**

Der zu schreibende bzw. gelesene Wert resultiert aus einer Oder-Verknüpfung der folgenden Flags:

Flags	Bedeutung
_XC163_DIN_x	DIN-x (x = 0 ... 11) ist Latch-Eingang
_XC163_POS_EDGE	Positive Flanke löst Latch-Impuls aus
_XC163_NEG_EDGE	Negative Flanke löst Latch-Impuls aus
_XC163_ENABLE	Externer Latch-Eingang ist aktiviert
_XC163_DISABLE	Externer Latch-Eingang ist deaktiviert

Beispiel:

Der folgende Befehl definiert, dass eine positive Flanke an DIN-7 alle DINs latcht:

```
max_write_channel_ushort (hChan, _XC163_DIN_7 | _XC163_POS_EDGE | _XC163_ENABLE);
```

9.3.7. Latchen der digitalen Eingänge per Software

Mit diesem Device können die digitalen Eingänge softwaremäßig in den Zwischenspeicher übertragen (gelatcht) werden. Das softwaremäßige Zwischenspeichern kann auch dann erfolgen, wenn bereits ein externer Latch-Eingang aktiviert ist. Um darauf zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_TRIGGER*	Kanal auf einen Trigger
.usIndexFirst	*XC163_DIN_TRIGGER*	Nummer des Triggers
.usIndexLast	*XC163_DIN_TRIGGER*	Nummer des Triggers
.usFlags	*0*	Keine Bedeutung
.usMode	*0*	Keine Bedeutung

Ausgabedienst

Der Datentyp des Kanals ist DATA_VOID, der Zugriff erfolgt mit:

- **max_trigger_channel**

9.3.8. Filterfunktion der digitalen Eingänge

Die Opto-entkoppelten Eingänge des X-C16-3 sind als solche relativ unempfindlich gegen Störungen. Trotzdem kann es bei langsamen Flankenverläufen und entsprechend großen Störsignalen vorkommen, dass ein Optokoppler bei einer solchen "gestörten" oder "schlechten" Eingangsflanke mehrmals sperrt bzw. durchschaltet. Der nachgeschaltete Zählerbaustein erkennt dann möglicherweise statt einer einzigen Flanke mehrere Flanken und wertet diese auch aus.

Verhindern lässt sich das Problem, indem die Abtastrate der Opto-Eingänge so eingestellt wird, dass während der Flankendauer nur ein einziger Abtastzeitpunkt vorkommen kann. Damit ist sichergestellt, dass trotz schlechtem oder gestörtem Verlauf des Eingangssignals nur eine einzige Flanke vom Zähler-Baustein erkannt wird. Mit Hilfe des Kontroll-Kanals vom Typ XC163_INPUT_FILTER kann die Abtastrate der Opto-Eingänge eingestellt werden. Die Einstellung gilt für alle Opto-Eingänge, d. h. alle Opto-Eingänge werden mit der gleichen Frequenz abgetastet.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_CTRL*	Kanal auf ein Kontroll-Device
.usIndexFirst	*XC163_INPUT_FILTER*	
.usFlags	*0*	reservierter Parameter
.usMode	*0*	reservierter Parameter

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_ULONG. Um die Abtastfrequenz zu setzen bzw. zurückzulesen, können folgende Funktionen verwendet werden:

- **max_write_channel_ulong**
- **max_read_channel_ulong**

Mögliche Werte für das Schreiben bzw. Lesen sind:

XC163_20MHZ
XC163_10MHZ
XC163_5MHZ
XC163_2MHZ
XC163_1MHZ
XC163_500KHZ
XC163_200KHZ
XC163_100KHZ
XC163_50KHZ
XC163_20KHZ
XC163_10KHZ

XC163_5KHZ
XC163_2KHZ
XC163_1KHZ

9.3.9. Digitale Ausgänge

Um auf die digitalen Ausgänge des Moduls zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_DOUT*	Kanal auf einen digitalen Ausgang
.usIndexFirst	*0 ... 7*	Nummer des ersten Ausgangs
.usIndexLast	*0 ... 7*	Nummer des letzten Ausgangs
.usFlags	*0*	Keine Bedeutung
.usReadMode	*IO_MODE_DIRECT*	Direkter Lesezugriff
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
.usMode	*0*	Keine Bedeutung

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_UCHAR, der Zugriff erfolgt mit:

- **max_write_channel_uchar**
- **max_read_channel_uchar**

9.3.10. Grundlagen Zähler

Zähleingänge

Das Modul verfügt über drei Zähler (0 ... 2). Jeder Zähler besitzt zwei Zähleingänge (A, B), die mit verschiedenen digitalen Eingängen (DIN) verbunden werden können. Hierzu kann beim Öffnen eines MDD-Kanals im Strukturelement *.usCounterInput* angegeben werden, welche DINs man mit dem jeweiligen Zähler verbinden will. Wird beim Öffnen eines Zählerkanals das Flag *_XC163_NEG_EDGE_x* (x = A, B) gesetzt, wird auf eine negative Flanke am entsprechenden Zähleingang reagiert, ansonsten auf eine positive Flanke.

Die Zählereingänge können entsprechend folgender Tabelle den digitalen Eingängen (DIN) zugeordnet werden. Die Zuordnung geschieht immer paarweise für die Zählereingänge A und B eines Zählers.

Modul-Stecker A	Zähleingänge		
	Zähler 0	**Zähler 1**	**Zähler 2**
DIN-0 DIN-1	A B		
DIN-2 DIN-3	A B	A B	
DIN-4 DIN-5	A B		A B
DIN-6 DIN-7	A B		
DIN-8 DIN-9	A B	A B	
DIN-10 DIN-11	A B		A B

Die Funktion der Zählereingänge A und B hängt von der Betriebsart des Zählers (Aufwärtszähler, Frequenzmessung, Inkrementalgeber, ...) ab. Zu jeder Betriebsart gibt es ein eigenes Kapitel, in dem die jeweilige Funktion der Zählereingänge A und B beschrieben ist.

Zählerausgänge

Bei einigen Betriebsarten der Zähler ist es möglich, bei einem Zählerunterlauf einen digitalen Ausgang (DOUT) umzuschalten. Dabei ist dem Zähler 0 DOUT-0, dem Zähler 1 DOUT-1 und dem Zähler 2 DOUT-2 zugeordnet. Sind mehrere Zähler kaskadiert, wird immer der DOUT des ersten Zählers verwendet.

Kaskadierung

Jeder Zähler hat eine Breite von 16 Bit. Durch ein Zusammenschalten von zwei bzw. drei Zählern (Kaskadieren) kann man die Breite auf 32 bzw. 48 Bit erweitern. Das Kaskadieren erfolgt beim Öffnen eines MDD-Kanals dadurch, dass man für die Strukturelemente *.usIndexFirst* und *.usIndexLast* unterschiedliche Werte angibt.

Zugriff auf einen Zählerkanal

Der Datentyp ist immer DATA_UCHAR. Das Schreiben bzw. Lesen eines Zählerkanals erfolgt immer mit einem **max_read_channel_block** bzw. **max_write_channel-_block** Befehl. Die Anzahl der Datenbytes, die dabei übertragen werden, hängt dabei von der Breite des Zählers ab:

Anzahl der (kaskadierten) Zähler	Anzahl der Datenbytes
1	2
2	4
3	6

Zugriff auf den erweiterten Zähler (64 Bit)

Mittels des Befehls **max_channel_info** kann mit dem Kommando INFO_XC163_-EXT_COUNT ein per Software auf 64 Bit erweiterter Zähler ausgelesen werden. Dieser liefert aber nur korrekte Ergebnisse, wenn der Kanal nicht mit dem Flag _XC-163_AUTOLOAD_UNDERFLOW geöffnet wurde bzw. der Zähler zum Starten mit 0 geladen wurde. Eine Kaskadierung wird ebenfalls nicht berücksichtigt. Der Datenpuffer für **max_channel_info** muss mindestens die Größe „Anzahl der Zähler im Kanal“ * sizeof(__int64) haben. Beispiel:

```
__int64 allData[3];
USHORT  usSize;

usSize = sizeof(__int64) * (cps.usIndexLast - cps.usIndexFirst + 1);
Error = max_channel_info(hUpDownCounter, INFO_XC163_EXT_COUNT, &usSize,
                         (void*)&allData[0]);
```

Steuern der Zähler

Das Steuern (Starten, Stoppen, Latchen, Laden und Zurücksetzen) von Zählern kann auf zwei Arten erfolgen:

Zum einen können Flanken an den digitalen Eingängen Steuerbefehle auslösen (siehe „Definition der externen Zähler-Steuereingänge“).
Zum anderen können die Zähler per Software gesteuert werden (siehe „Steuern der Zähler per Software“).

Umschalten der Referenz-Frequenzen

Der im CPS-Element *usReferenceTime* angegebenen Wert kann dynamisch während einer Messung mit Hilfe des Kanal-Steuerkommandos *CTRL_SET_TIME* umgeschaltet werden.
Das folgende Beispiel schaltet die Referenz-Frequenz auf 1 KHz um:

```
ulTime = XC163_1KHZ;
max_channel_control(hChannel, CTRL_SET_TIME, 4, (void*)&ulTime);
```

Status der Messung

Mit Hilfe des Kanal-Steuerkommandos INFO_DATA kann der Status der aktuellen Messung ermittelt werden.
Falls neue Daten vorliegen ist das Bit *_XC163_NEW_DATA* gesetzt.

Beispiel:

```
usSize = 4;
max_channel_info(hChannel, INFO_DATA, &usSize, (void*)&ulStatus);
```

Zähler-Über- / Unterlauf

Liegt seit dem letzten Auslesen des Zählers ein Zähler-Überlauf bzw. ein Unterlauf vor, so liefert der Eingabedienst der Fehler ERR_OVERFLOW bzw. ERR_UNDERFLOW zurück. Liegen mehrere Überläufe/Unterläufe vor, so liefert der Dienst den Fehler ERR_INVALID_DATA. Diese Rückgabewerte treten nur auf, wenn keine Callback-Funktion für den Overflow bzw. Underflow definiert ist.

9.3.11. Definition der externen Zähler-Steuereingänge

Mit diesem Device kann definiert werden, welche digitalen Eingänge als Steuereingänge für die Zähler verwendet werden können. Insgesamt stehen vier Steuereingänge (0 ... 3) zur Verfügung. Diese können verwendet werden, um die Zähler zu steuern, d.h.:

- der Zähler kann mit einem Wert geladen werden, der zuvor mit einem Schreibbefehl in einen internen Zwischenspeicher des Zählerkanals geschrieben wurde (LOAD)
- der Zähler kann gestartet werden (START)
- der Zähler kann angehalten werden (STOP)
- der aktuelle Zählerstand kann in einem Zwischenspeicher abgelegt werden (LATCH)
- der aktuelle Zählerstand kann auf 0 zurückgesetzt werden (RESET)

Die Steuereingänge können entsprechend folgender Tabelle den digitalen Eingängen (DIN) zugeordnet werden. Die Zuordnung geschieht immer paarweise für die Steuereingänge 0 und 1, bzw. für die Steuereingänge 2 und 3.

Modul-Stecker A	Steuereingänge für Zähler 0, 1 und 2	
	Steuereingänge 0 und 1	Steuereingänge 2 und 3
DIN-0 DIN-1	0 1	2 3
DIN-2 DIN-3	0 1	
DIN-4 DIN-5		2 3
DIN-6 DIN-7	0 1	2 3
DIN-8 DIN-9	0 1	
DIN-10 DIN-11		2 3

Die Steuereingänge 0 bis 3 stehen allen drei Zählern (0, 1 und 2) gleichermaßen zur Verfügung. Für jeden Steuereingang kann festgelegt werden, ob die positive oder die negative Flanke einen Steuerbefehl auslösen soll. Welche Aktion (LOAD, START, STOP, LATCH, RESET) bei einer Flanke an einem der Steuereingänge ausgeführt wird, kann für jeden Zähler individuell und unabhängig von den anderen Zählern eingestellt werden.

Um einen Steuereingang für einen Zähler zu nutzen, muss beim Öffnen des entsprechenden Zählerkanals angegeben werden, welcher Steuereingang (0 ... 3) einen Befehl auslösen soll.

Um einen Steuereingang zu definieren, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_CTRL*	Kanal auf ein Kontroll-Device
.usIndexFirst	*XC163_COUNTER_HW_CTRL*	Externer Zähler-Steuereingang
.usIndexLast	*XC163_COUNTER_HW_CTRL*	Externer Zähler-Steuereingang
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv
.usReadMode	*IO_MODE_DIRECT*	Direkter Lesezugriff
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
.usMode	*0*	Keine Bedeutung

Eingabe- und Ausgabedienst

Der Datentyp des Kanals ist DATA_USHORT, der Zugriff erfolgt mit:

- **max_write_channel_ushort**
- **max_read_channel_ushort**

Mögliche Werte bei einen Schreib- bzw. Lesezugriff sind:

Wert	Bedeutung
_XC163_DIN_x_y_TO_CTRL_0_1	Verbindet die digitalen Eingänge x/y (x/y = 0/1, 2/3, 6/7, 8/9) mit den Steuereingängen 0 und 1
_XC163_DIN_x_y_TO_CTRL_2_3	Verbindet die digitalen Eingänge x/y (x/y = 0/1, 4/5, 6/7, 10/11) mit den Steuereingängen 2 und 3
_XC163_CTRL_x_NEG_EDGE	Dieses Flag legt fest, dass eine negative Flanke an dem Steuereingang x (x = 0 ... 3) einen Befehl auslöst.

Durch eine Oder-Verknüpfung der Konstanten können die Steuereingänge gleichzeitig gesetzt werden.

Beispiel:

Der folgende Befehl verbindet DIN-0 und DIN-1 mit den Steuereingängen 0 und 1 und DIN-10 und DIN-11 mit den Steuereingängen 2 und 3.

```
max_write_channel_ushort (hChan,
                  _XC163_DIN_0_1_TO_CTRL_0_1 | _XC163_DIN_10_11_TO_CTRL_2_3)
```

9.3.12. Steuern der Zähler per Software

Die drei Zähler des X-C16-3 können nicht nur über die digitalen Opto-Eingänge (DIN) gesteuert werden, sonden auch per Software. Das folgende Device erlaubt es, die Zähler per Software zu steuern, d.h.:

- der Zähler kann mit einem Wert geladen werden, der zuvor mit einem Schreibbefehl in einen internen Zwischenspeicher des Zählerkanals geschrieben wurde (LOAD)
- der Zähler kann gestartet werden (START)
- der Zähler kann angehalten werden (STOP)
- der aktuelle Zählerstand kann in einem Zwischenspeicher abgelegt werden (LATCH)
- der aktuelle Zählerstand kann auf 0 zurückgesetzt werden (RESET)

Um darauf zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_CTRL*	Kanal auf ein Kontroll-Device
.usIndexFirst	*XC163_COUNTER_SW_CTRL*	Zähler Software-Steuerung

Strukturelement	Werte	Bedeutung
.usIndexLast	*XC163_COUNTER_SW_CTRL*	Zähler Software-Steuerung
.usFlags	*0*	Keine Bedeutung
.usReadMode	*0*	Keine Bedeutung
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
.usMode	*0*	Keine Bedeutung

Ausgabedienst

Der Datentyp des Kanals ist DATA_USHORT, der Zugriff erfolgt mit:

- **max_write_channel_ushort**

Mögliche Werte für einen Schreibzugriff sind:

Wert	Bedeutung
_XC163_COUNTER_x	Zähler x (x = 0 ... 2), an den das Kommando gesendet werden soll.
_XC163_START_COUNTER	Zähler starten
_XC163_STOP_COUNTER	Zähler anhalten
_XC163_LOAD_COUNTER	Zähler laden
_XC163_LATCH_COUNTER	Die aktuellen Zählerstände in den internen Zwischenspeicher übernehmen
_XC163_RESET_COUNTER	Zählerstände auf 0 setzen

Durch eine Oder-Verknüpfung der Konstanten können auch mehrere Zähler gleichzeitig gesteuert werden.

Beispiel:

Der folgende Befehl startet den Zähler 0 und den Zähler 2:

```
max_write_channel_ushort (hChannel, _XC163_COUNTER_0 | _XC163_COUNTER_2 |
_XC16_START_COUNTER)
```

9.3.13. Aufwärtszähler

Mit diesem Device kann ein Aufwärtszähler realisiert werden. An den Zähleingang A werden die Zählimpulse gelegt. Zähleingang B wird nicht benutzt. Um auf den Zähler zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_UP_COUNTER*	Kanal zu einem Aufwärtszähler
.usIndexFirst	*0 ... 2*	Nummer des ersten Aufwärtszählers
.usIndexLast	*0 ... 2*	Nummer des letzten Aufwärtszählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE* *_CP_SYNC_CALLBACK*	Der Zugriff erfolgt immer exklusiv Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Lesezugriff Zählerwert wird aus Zwischenspeicher gelesen
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in Zwischenspeicher geschrieben
.usMode	*0* *_XC163_AUTORUN*	Keine Bedeutung Zähler startet sofort nach Öffnen des Kanals mit Startwert = 0.

Strukturelement	Werte	Bedeutung
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_POS_EDGE_A	Zählen bei positiver Flanke an Eingang A
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
.ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_OVERFLOW	Der Zähler ist übergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zähler nach dem Beschreiben des Kanals mit dem geschriebenem Wert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibzugriff der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden sollen, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Ist *.usReadMode = IO_MODE_DIRECT*, kann der aktuelle Zählerstand durch einen Lesebefehl ermittelt werden. Wurde der Lesemodus auf *IO_MODE_LATCH* eingestellt, wird durch einen Lesezugriff der im Latch zwischengespeicherte Zählerstand ausgelesen. Ein Latchbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  `MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit`
  Die Tasknummer kann beliebig gewählt werden.
- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  `MAXLOADMDD slot=0 layer=0 progno=8FFF`
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal auf einen Aufwärtszähler geöffnet. Dabei werden die Zähler 0 und 1 zu einem Zähler kaskadiert. Der Zähler startet sofort beim Öffnen des Kanals.

```
CPS_XC163 rcUpCounter;
MAXCHLHND hUpCounter;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[4];

// CPS ausfuellen
rcUpCounter.usDevice = DEVICE_UP_COUNTER;
rcUpCounter.usIndexFirst = 0;
rcUpCounter.usIndexLast = 1;
rcUpCounter.usFlags = _CP_EXCLUSIVE;
rcUpCounter.usReadMode = IO_MODE_DIRECT;
rcUpCounter.usWriteMode = IO_MODE_DIRECT;
rcUpCounter.usMode = _XC163_AUTORUN;
rcUpCounter.usCounterInput = _XC163_DIN_0_1;
rcUpCounter.ulControlInput = 0;
// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcUpCounter), &rcUpCounter,
                                                NULL, NULL, &hUpCounter);
// Kanal lesen
ulSize = 4;
Error = max_read_channel_block(hUpCounter, &ulSize, (void*)aucData);
```

9.3.14. Abwärtszähler

Mit diesem Device kann ein Abwärtszähler realisiert werden. An den Zähleingang A werden die Zählimpulse gelegt. Zähleingang B wird nicht benutzt. Um auf den Zähler zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_DOWN_COUNTER*	Kanal auf einen Abwärtszähler
.usIndexFirst	*0 ... 2*	Nummer des ersten Zählers
.usIndexLast	*0 ... 2*	Nummer des letzten Zählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE* *_CP_SYNC_CALLBACK*	Der Zugriff erfolgt immer exklusiv Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Lesezugriff Der Zählerwert wird aus dem Zwischenspeicher gelesen
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in den Zwischenspeicher geschrieben
.usMode	*0* *_XC163_AUTORUN*	Keine Bedeutung Zähler startet sofort nach Öffnen des Kanals mit Startwert = 0.
	_XC163_OUTPUT	Der zum Zähler gehörige DOUT schaltet bei jedem Zählerunterlauf.
	_XC163_AUTOLOAD_ UNDERFLOW	Nach einem Zählerunterlauf wird der Zähler wieder mit dem Startwert geladen. Ansonsten wird er mit dem Maximalwert geladen.

Strukturelement	Werte	Bedeutung
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_POS_EDGE_A	Zählen bei positiver Flanke an Eingang A
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_UNDERFLOW	Der Zähler ist untergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist der Mode *_XC163_AUTORUN* aktiviert, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zähler nach dem Beschreiben des Kanals mit dem geschriebenem Wert.

- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibzugriff der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Ist *.usReadMode = IO_MODE_DIRECT*, kann der aktuelle Zählerstand durch einen Lesebefehl ermittelt werden. Wurde der Lesemodus auf *IO_MODE_LATCH* eingestellt, wird durch einen Lesezugriff der im Latch zwischengespeicherte Zählerstand ausgelesen. Ein Latchbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:

```
MAXLOADMDD slot=0 layer=0 progno=8FFF
```

Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Siehe Aufwärtszähler.

9.3.15. Auf- und Abwärtszähler

Mit diesem Device kann ein Auf- und Abwärtszähler realisiert werden. Um darauf zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE _UP_DOWN_COUNTER*	Kanal auf einen Auf- und Abwärtszähler
.usIndexFirst	*0 ... 2*	Nummer des ersten Zählers
.usIndexLast	*0 ... 2*	Nummer des letzten Zählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE* *_CP_SYNC_CALLBACK*	Der Zugriff erfolgt immer exklusiv Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werde die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Lesezugriff Der Zählerwert wird aus dem Zwischenspeicher gelesen.
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in den Zwischenspeicher geschrieben.

Strukturelement	Werte	Bedeutung
.usMode	*_XC163_MODE_A*	Impulse am Zähleingang A werden in Aufwärtsrichtung gezählt, Impulse am Zähleingang B in Abwärtsrichtung.
	_XC163_MODE_B	Die Zählimpulse werden am Zähleingang A angelegt. Über den Zähleingang B wird die Zählrichtung festgelegt (0 = aufwärts; 1 = abwärts).
	_XC163_AUTORUN	Der Zähler startet nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
	_XC163_OUTPUT	Der zum Zähler gehörige DOUT schaltet bei jedem Zählerunterlauf.
	_XC163_AUTOLOAD_ UNDERFLOW	Nach einem Zählerunterlauf wird der Zähler wieder mit dem Startwert geladen. Ansonsten wird er mit dem Maximalwert geladen.
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_POS_EDGE_A	Zählen bei positive Flanke an Eingang A
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_OVERFLOW	Der Zähler ist übergelaufen
	XC163_EVENT_UNDERFLOW	Der Zähler ist untergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OV ERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist der Mode *_XC163_AUTORUN* aktiviert, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zähler nach dem Beschreiben des Kanals mit dem geschriebenem Wert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibzugriff der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Ist *.usReadMode = IO_MODE_DIRECT*, kann der aktuelle Zählerstand durch einen Lesebefehl ermittelt werden. Ist *.usReadMode = IO_MODE_LATCH*, wird durch einen Lesezugriff der im Latch zwischengespeicherte Zählerstand ausgelesen. Ein Latchbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:

```
MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
```

Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXLOADMDD slot=0 layer=0 progno=8FFF
  ```
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Beispiel:

Nachfolgend wird ein Kanal auf einen Auf- / Abwärtszähler im Mode A geöffnet. Dabei werden die Zähler 0 bis 2 zu einem Zähler kaskadiert. Der Zähler startet sofort beim Öffnen des Kanals.

```
CPS_XC163 rcUpDownCounter;
MAXCHLHND hUpDownCounter;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[6];

// CPS ausfuellen
rcUpDownCounter.usDevice = DEVICE_UP_DOWN_COUNTER;
rcUpDownCounter.usIndexFirst = 0;
rcUpDownCounter.usIndexLast = 2;
rcUpDownCounter.usFlags = _CP_EXCLUSIVE;
rcUpDownCounter.usReadMode = IO_MODE_DIRECT;
rcUpDownCounter.usWriteMode = IO_MODE_DIRECT;
rcUpDownCounter.usMode = _XC163_MODE_A | _XC163_AUTORUN;
rcUpDownCounter.usCounterInput = _XC163_DIN_0_1;
rcUpDownCounter.ulControlInput = 0;

// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcUpDownCounter), &rcUpDownCounter,
                                                  NULL, NULL, &hUpDownCounter);

// Kanal lesen
ulSize = 6;
Error = max_read_channel_block(hUpDownCounter, &ulSize, (void*)aucData);
```

Beispiel:

Das nachfolgende Beispiel zeigt einen Auf- Abwärtszahler, der die Über- und Unterläufe signalisiert.
In der Callback-Funktion werden diese Ereignisse unterschieden.

```
MAX_CALLBACK MyCallbackFunction(MAXCHLHND handle, ULONG param, ULONG size, void*
pData)
{
    ULONG ulData;

    ulData = *(ULONG*)pData;

    if ( (ulData & XC163_EVENT_OVERFLOW) != 0)
    {
        ... es ist ein Überlauf aufgetreten
    }

    if ( (ulData & XC163_EVENT_UNDERFLOW) != 0)
    {
        ... es ist ein Unterlauf aufgetreten
    }
}

void OpenChannel(void)
{
    CPS_XC163_A  cps;

    cps.usDevice = DEVICE_UP_DOWN_COUNTER;
    cps.usIndexFirst = 0;
    cps.usIndexLast = 0;
    cps.usFlags = _CP_EXCLUSIVE;
    cps.usReadMode = IO_MODE_DIRECT;
    cps.usWriteMode = IO_MODE_DIRECT;
    cps.usMode = _XC163_MODE_A | _XC163_AUTORUN;
    cps.usCounterInput = XC163_DIN_0_1;
    cps.ulControlInput = 0;
    cps.ulCallbackEvent = XC163_EVENT_OVERFLOW | XC163_EVENT_UNDERFLOW;

    // öffnen des Kanals mit Callback-Funktionalität
    max_open_channel(g_hMdd, sizeof (cps), &cps, MyCallbackFunction, NULL,
&hUpCounter);

}
```

9.3.16. Timer

Mit diesem Device kann ein Timer realisiert werden. Der Zähler arbeitet als Abwärtszähler. Die Zählsignale kommen entweder von einem modulinternen Taktgeber oder von einem externen Signal, das an Zähleingang A anliegen muss. Zähleingang B wird nicht benutzt. Es muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_TIMER*	Kanal auf einen Timer
.usIndexFirst	*0 ... 2*	Nummer des ersten Timers
.usIndexLast	*0 ... 2*	Nummer des letzten Timers (Ist dieser Wert größer als *.usIndexFirst*, werden die

Strukturelement	Werte	Bedeutung
		Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE* *_CP_SYNC_CALLBACK*	Der Zugriff erfolgt immer exklusiv. Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Lesezugriff Der Zählerwert wird aus dem Zwischenspeicher gelesen.
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in den Zwischenspeicher geschrieben.
.usMode	*0* *_XC163_CONTINUOUS* *_XC163_OUTPUT*	Keine Bedeutung Der Timer startet nach dem Nulldurchlauf automatisch. Der zum Zähler gehörige DOUT schaltet bei jedem Zählerunterlauf.
.ulReferenceTime	*XC163_EXTERNAL* *XC163_20MHZ* *XC163_10MHZ* *XC163_5MHZ* *XC163_2MHZ* *XC163_1MHZ* *XC163_500KHZ* *XC163_200KHZ* *XC163_100KHZ* *XC163_50KHZ* *XC163_20KHZ* *XC163_10KHZ* *XC163_5KHZ* *XC163_2KHZ* *XC163_1KHZ* *XC163_500HZ* *XC163_200HZ* *XC163_100HZ* *XC163_50HZ* *XC163_20HZ* *XC163_10HZ*	Referenz-Frequenz

Strukturelement	Werte	Bedeutung
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_NEG_EDGE_B	Zählen bei negativer Flanke an Eingang B
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_UNDERFLOW	Der Zähler ist untergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zähler nach dem Beschreiben des Kanals mit dem geschriebenem Wert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibzugriff der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch

nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.usControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Ist *usReadMode = IO_MODE_DIRECT*, kann der aktuelle Zählerstand durch einen Lesebefehl ermittelt werden. Wurde der Lesemodus auf *IO_MODE_LATCH* eingestellt, wird durch einen Lesezugriff der im Latch zwischengespeicherte Zählerstand ausgelesen. Ein Latchbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXLOADMDD slot=0 layer=0 progno=8FFF
  ```
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal auf einen Timer mit einer Zählfrequenz von 1 kHz geöffnet. Durch das Beschreiben des Kanals startet der Timer und schaltet bei jedem Unterlauf den digitalen Ausgang DOUT-0 um.

```
CPS_XC163  rcTimer;
MAXCHLHND hTimer;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[2];

// CPS ausfuellen
rcTimer.usDevice = DEVICE_TIMER;
rcTimer.usIndexFirst = 0;
rcTimer.usIndexLast = 0;
rcTimer.usFlags = _CP_EXCLUSIVE;
rcTimer.usReadMode = IO_MODE_DIRECT;
rcTimer.usWriteMode = IO_MODE_DIRECT;
rcTimer.usMode = _XC163_CONTINUOUS | _XC163_OUTPUT;
rcTimer.ulReferenceTime = XC163_1KHZ;
rcTimer.usCounterInput = _XC163_DIN_0_1;
rcTimer.ulControlInput = 0;
// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcTimer), &rcTimer,
                                                 NULL, NULL, &hTimer);
// Kanal mit Wert 1000 (=3E8hex) beschreiben und starten
ulSize = 2;
aucData[0] = 0xE8;
aucData[1] = 0x03;
Error = max_write_channel_block(hTimer, &ulSize, (void*)aucData);
```

9.3.17. Pulsbreitenmessung

Das Signal, dessen Pulsbreite gemessen werden soll, wird an den Zähleingang B angelegt. Die Referenz-Frequenz, die während der Pulsdauer des Messsignals gezählt wird, kann intern generiert werden oder über den Zähleingang A vorgegeben werden.

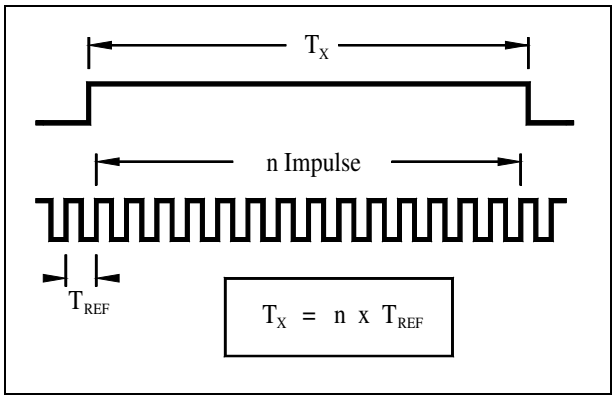


Abb. 9-1: Pulsbreitenmessung

Um eine Pulsbreitenmessung durchführen zu können, muss nachfolgende CPS verwendet werden.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE _PULSEWIDTH_COUNTER*	Kanal auf einen Zähler für Pulsbreitenmessung
.usIndexFirst	*0 ... 2*	Nummer des ersten Zählers
.usIndexLast	*0 ... 2*	Nummer des letzten Zählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv
	_XC163_NEW_DATA_ONLY	Ist dieses Flag gesetzt, liefert der Lesedienst den Fehler ERR_DATA_NOT_READY zurück falls noch keine neuen Daten bereitstehen. Andernfalls wird das alte Ergebnis zurückgeliefert.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_LATCH*	Der Zählerwert wird aus Zwischenspeicher gelesen.
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
	IO_MODE_LATCH	Der Zählerwert wird in den Zwischenspeicher geschrieben.

Strukturelement	Werte	Bedeutung
.usMode	*0*	Keine Bedeutung
	_XC163_AUTORUN	Der Zählvorgang startet sofort nach dem Öffnen des Kanals (Zähler wird mit 0 vorinitialisiert).
	_XC163_CONTINUOUS	Am Ende der Messung wird sofort eine neue Messung gestartet. Wurde der Zähler mit einem Schreibzugriff auf einen bestimmten Wert voreingestellt, wird dieser Wert jeweils übernommen.
	_XC163_OVERRIDE	Im Zwischenspeicher liegt jeweils das Ergebnis der letzten Messung. Ist dieser Mode nicht aktiviert, wird nur ein neuer Wert in den Zwischenspeicher geschrieben, wenn der alte Wert ausgelesen wurde.
.ulReferenceTime	*XC163_EXTERNAL* *XC163_20MHZ* *XC163_10MHZ* *XC163_5MHZ* *XC163_2MHZ* *XC163_1MHZ* *XC163_500KHZ* *XC163_200KHZ* *XC163_100KHZ* *XC163_50KHZ* *XC163_20KHZ* *XC163_10KHZ* *XC163_5KHZ* *XC163_2KHZ* *XC163_1KHZ* *XC163_500HZ* *XC163_200HZ* *XC163_100HZ* *XC163_50HZ* *XC163_20HZ* *XC163_10HZ*	Referenz-Frequenz
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_NEG_EDGE_B	Zählen bei negativer Flanke an Eingang B
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)

Strukturelement	Werte	Bedeutung
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_END_OF_MEASUREMENT	Eine Messung ist beendet worden
	XC163_EVENT_OVERFLOW	Der Zähler ist übergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zählvorgang nach dem Beschreiben des Kanals. Der Zähler wird dabei mit dem geschriebenem Wert vorinitialisiert.
- Ist *.usWriteMode = IO_MODE_LATCH,* wird mit einem Schreibbefehl der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Gelesen wird immer aus einem internen Zwischenspeicher.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXLOADMDD slot=0 layer=0 progno=8FFF
  ```
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal für eine Pulsbreitenmessung mit einer Zählfrequenz von 1 kHz geöffnet.

```
CPS_XC163 rcPulseWidth;
MAXCHLHND hPulseWidth;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[2];

// CPS ausfuellen
rcPulseWidth.usDevice = DEVICE_PULSEWIDTH_COUNTER;
rcPulseWidth.usIndexFirst = 0;
rcPulseWidth.usIndexLast = 0;
rcPulseWidth.usFlags = _CP_EXCLUSIVE;
rcPulseWidth.usReadMode = IO_MODE_LATCH;
rcPulseWidth.usWriteMode = IO_MODE_DIRECT;
rcPulseWidth.usMode = _XC163_AUTORUN | _XC163_OVERRIDE | _XC163_CONTINUOUS;
rcPulseWidth.ulReferenceTime = XC163_1KHZ;
rcPulseWidth.usCounterInput = _XC163_DIN_0_1;
rcPulseWidth.ulControlInput = 0;

// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcPulseWidth), &rcPulseWidth,
                                                NULL, NULL, &hPulseWidth);

// Messwerte lesen
ulSize = 2;
Error = max_read_channel_block(hPulseWidth, &ulSize, (void*)aucData);
```

9.3.18. Frequenzmessung

Die zu messende Frequenz wird an den Zähleingang A angelegt. Die Torzeit, während der die Messfrequenz gezählt wird, kann intern generiert werden oder über den Zähleingang B vorgegeben werden.

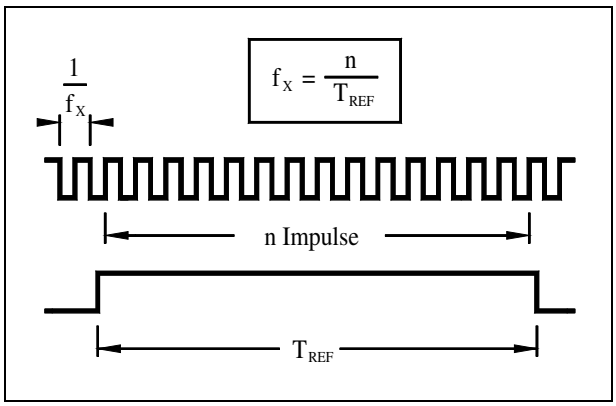


Abb. 9-2: Frequenzmessung

Um eine Frequenzmessung durchführen zu können, muss nachfolgende CPS verwendet werden.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE _FREQUENCY_COUNTER*	Kanal auf einen Zähler für Frequenzmessung
.usIndexFirst	*0 ... 2*	Nummer des ersten Zählers
.usIndexLast	*0 ... 2*	Nummer des letzten Zählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv.
	_XC163_NEW_DATA_ONLY	Ist dieses Flag gesetzt, liefert der Lesedienst den Fehler ERR_DATA_NOT_READY zurück falls noch keine neuen Daten bereitstehen. Andernfalls wird das alte Ergebnis zurückgeliefert.
.usReadMode	*IO_MODE_LATCH*	Der Zählerwert wird aus dem Zwischenspeicher gelesen
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
	IO_MODE_LATCH	Der Zählerwert wird in den Zwischenspeicher geschrieben.
.usMode	*0*	Keine Bedeutung
	_XC163_AUTORUN	Der Zählvorgang startet sofort nach dem Öffnen des Kanals (Zähler wird mit 0 vorinitialisiert).
	_XC163_CONTINUOUS	Am Ende der Messung wird sofort eine neue Messung gestartet. Wurde der Zähler mit einem Schreibzugriff auf einen bestimmten Wert voreingestellt, wird dieser Wert jeweils übernommen.
	_XC163_OVERRIDE	Im Zwischenspeicher liegt jeweils das Ergebnis der letzten Messung. Ist dieser Mode nicht aktiviert, wird nur ein neuer Wert in den Zwischenspeicher geschrieben, wenn der alte Wert ausgelesen wurde.

Strukturelement	Werte	Bedeutung
.ulReferenceTime	*XC163_EXTERNAL* *XC163_50NS* *XC163_100NS* *XC163_200NS* *XC163_500NS* *XC163_1US* *XC163_2US* *XC163_5US* *XC163_10US* *XC163_20US* *XC163_50US* *XC163_100US* *XC163_200US* *XC163_500US* *XC163_1MS* *XC163_2MS* *XC163_5MS* *XC163_10MS* *XC163_20MS* *XC163_50MS* *XC163_100MS*	Torzeit
.usCounterInput	*_XC163_DIN_x_y* *_XC163_NEG_EDGE_A* *_XC163_NEG_EDGE_B*	Zähleingang (siehe Tabelle) Zählen bei negativer Flanke an Eingang A Zählen bei negativer Flanke an Eingang B
.ulControlInput	*0* *_XC163_START_CTRL_x* *_XC163_STOP_CTRL_x* *_XC163_LOAD_CTRL_x* *_XC163_LATCH_CTRL_x* *_XC163_RESET_CTRL_x*	Keine Bedeutung Start, Stop, Laden, Zwischenspeichern und Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)

Strukturelement	Werte	Bedeutung
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_END_OF_ MEASUREMENT	Eine Messung ist beendet worden
	XC163_EVENT_OVERFLOW	Der Zähler ist übergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT _OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zählvorgang nach dem Beschreiben des Kanals. Der Zähler wird dabei mit dem geschriebenem Wert vorinitialisiert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibbefehl der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Gelesen wird immer aus einem internen Zwischenspeicher.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  `MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit`
  Die Tasknummer kann beliebig gewählt werden.
- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  `MAXLOADMDD slot=0 layer=0 progno=8FFF`
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal auf eine Frequenzmessung mit einer Torzeit von 1 ms geöffnet.

```
CPS_XC163  rcFrequency;
MAXCHLHND hFrequency;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[2];

// CPS ausfuellen
rcFrequency.usDevice = DEVICE_FREQUENCY_COUNTER;
rcFrequency.usIndexFirst = 0;
rcFrequency.usIndexLast = 0;
rcFrequency.usFlags = _CP_EXCLUSIVE;
```

```
rcFrequency.usReadMode = IO_MODE_LATCH;
rcFrequency.usWriteMode = IO_MODE_DIRECT;
rcFrequency.usMode = _XC163_AUTORUN | _XC163_OVERRIDE | _XC163_CONTINUOUS;
rcFrequency.ulReferenceTime = XC163_1MS;
rcFrequency.usCounterInput = _XC163_DIN_0_1;
rcFrequency.ulControlInput = 0;

// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcFrequency), &rcFrequency,
                                                  NULL, NULL, &hFrequency);
// Messwerte lesen
ulSize = 2;
Error = max_read_channel_block(hFrequency, &ulSize, (void*)aucData);
```

9.3.19. Periodendauermessung

Das Signal, dessen Periodendauer bestimmt werden soll, wird an Zähleingang B gelegt. Die Referenz-Frequenz, die während der Periodendauer des Messsignals gezählt wird, kann intern generiert werden oder über Zähleingang A zugeführt werden.

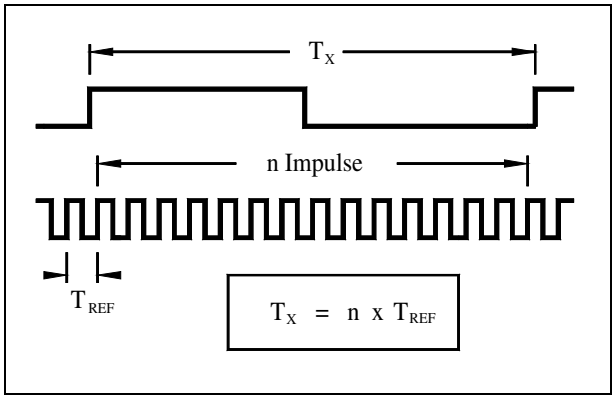


Abb. 10-3: Periodendauermessung

Um eine Periodendauermessung durchführen zu können, muss nachfolgende CPS verwendet werden.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE _PERIOD_COUNTER*	Kanal auf einen Zähler für Periodendauermessung
.usIndexFirst	*0 ... 2*	Nummer des ersten Zählers
.usIndexLast	*0 ... 2*	Nummer des letzten Zählers (Ist dieser Wert größer als *.usIndexFirst*, werden die Zähler von *.usIndexFirst* bis *.usIndexLast* kaskadiert.)
.usFlags	*_CP_EXCLUSIVE _XC163_NEW_DATA_ONLY*	Der Zugriff erfolgt immer exklusiv Ist dieses Flag gesetzt, liefert der Lesedienst den Fehler ERR_DATA_NOT_READY zu-

Strukturelement	Werte	Bedeutung
		rück falls noch keine neuen Daten bereitstehen. Andernfalls wird das alte Ergebnis zurückgeliefert.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_LATCH*	Der Zählerwert wird aus dem Zwischenspeicher gelesen
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in den Zwischenspeicher geschrieben
.usMode	*0* *_XC163_AUTORUN*	Keine Bedeutung Der Zählvorgang startet sofort nach dem Öffnen des Kanals (Zähler wird mit 0 vorinitialisiert).
	_XC163_CONTINUOUS	Am Ende der Messung wird sofort eine neue Messung gestartet. Wurde der Zähler mit einem Schreibzugriff auf einen bestimmten Wert voreingestellt, wird dieser Wert jeweils übernommen.
	_XC163_OVERRIDE	Im Zwischenspeicher liegt jeweils das Ergebnis der letzten Messung. Ist dieser Mode nicht aktiviert, wird nur ein neuer Wert in den Zwischenspeicher geschrieben, wenn der alte Wert ausgelesen wurde.
.ulReferenceTime	*XC163_EXTERNAL* *XC163_20MHZ* *XC163_10MHZ* *XC163_5MHZ* *XC163_2MHZ* *XC163_1MHZ* *XC163_500KHZ* *XC163_200KHZ* *XC163_100KHZ* *XC163_50KHZ* *XC163_20KHZ* *XC163_10KHZ* *XC163_5KHZ* *XC163_2KHZ* *XC163_1KHZ* *XC163_500HZ*	Referenz-Frequenz

Strukturelement	Werte	Bedeutung
	XC163_200HZ *XC163_100HZ* *XC163_50HZ* *XC163_20HZ* *XC163_10HZ*	
.usMultiple	*0*	Reserviert
.usCounterInput	*_XC163_DIN_x_y* *_XC163_NEG_EDGE_A* *_XC163_NEG_EDGE_B*	Zähleingang (siehe Tabelle)
.ulControlInput	*0* *_XC163_START_CTRL_x* *_XC163_STOP_CTRL_x* *_XC163_LOAD_CTRL_x* *_XC163_LATCH_CTRL_x* *_XC163_RESET_CTRL_x*	Keine Bedeutung Start, Stop, Laden, Zwischenspeichern und Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0* *XC163_EVENT_END_OF_MEASUREMENT* *XC163_EVENT_OVERFLOW* *XC163_EVENT_START* *XC163_EVENT_STOP* *XC163_EVENT_LOAD* *XC163_EVENT_LATCH* *XC163_EVENT_RESET* *XC163_EVENT_INTERRUPT_OVERRUN*	Es wird kein Ereignis signalisiert Eine Messung ist beendet worden Der Zähler ist übergelaufen Der Zähler wurde durch einem Steuerkanal gestartet Der Zähler wurde durch einem Steuerkanal gestoppt Der Zähler wurde durch einem Steuerkanal geladen Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert Der Zähler wurde durch einem Steuerkanal zurückgesetzt Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zählvorgang nach dem Beschreiben des Kanals. Der Zähler wird dabei mit dem geschriebenem Wert vorinitialisiert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibbefehl der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

2. Lesen der Zählerwerte

Gelesen wird immer aus einem internen Zwischenspeicher.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  `MAXLOADMDD slot=0 layer=0 progno=8FFF`
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal auf eine Periodendauermessung mit einer Zählfrequenz von 1 kHz geöffnet.

```
CPS_XC163  rcPeriod;
MAXCHLHND hPeriod;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[2];

// CPS ausfuellen
rcPeriod.usDevice = DEVICE_PERIOD_COUNTER;
rcPeriod.usIndexFirst = 0;
rcPeriod.usIndexLast = 0;
rcPeriod.usFlags = _CP_EXCLUSIVE;
rcPeriod.usReadMode = IO_MODE_LATCH;
rcPeriod.usWriteMode = IO_MODE_DIRECT;
rcPeriod.usMode = _XC163_AUTORUN | _XC163_OVERRIDE | _XC163_CONTINUOUS;
rcPeriod.ulReferenceTime = XC163_1KHZ;
rcPeriod.usMultiple = 0;
rcPeriod.usCounterInput = _XC163_DIN_0_1;
rcPeriod.ulControlInput = 0;

// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcPeriod), &rcPeriod,
                                                  NULL, NULL, &hPeriod);

// Messwerte lesen
ulSize = 2;
Error = max_read_channel_block(hPeriod, &ulSize, (void*)aucData);
```

9.3.20. Periodendauermessung über mehrere Perioden

Das Signal, dessen Periodendauer bestimmt werden soll, wird an den Zähleingang B gelegt. Die Referenz-Frequenz, die während der Periodendauer des Messsignals gezählt wird, kann intern generiert werden oder über den Zähleingang A zugeführt werden.

Um eine Periodendauermessung über mehrere Perioden durchführen zu können, muss nachfolgende CPS verwendet werden.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE _PERIOD_COUNTER*	Kanal auf einen Zähler für Periodenmessung über mehrere Perioden
.usIndexFirst	*0*	Nummer des ersten Zählers
.usIndexLast	*1*	Nummer des letzten Zählers
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv
	_XC163_NEW_DATA_ONLY	Ist dieses Flag gesetzt, liefert der Lesedienst den Fehler ERR_DATA_NOT_READY zurück falls noch keine neuen Daten bereitstehen. Andernfalls wird das alte Ergebnis zurückgeliefert.
	_CP_SYNC_CALLBACK	Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_LATCH*	Der Zählerwert wird aus dem Zwischenspeicher gelesen.
.usWriteMode	*IO_MODE_DIRECT*	Direkter Schreibzugriff
	IO_MODE_LATCH	Der Zählerwert wird in den Zwischenspeicher geschrieben.
.usMode	*_XC163_AUTORUN*	Der Zählvorgang startet sofort nach dem Öffnen des Kanals (Zähler wird mit 0 vorinitialisiert).
	_XC163_OVERRIDE	Im Zwischenspeicher liegt jeweils das Ergebnis der letzten Messung. Ist dieser Mode nicht aktiviert, wird nur ein neuer Wert in den Zwischenspeicher geschrieben, wenn der alte Wert ausgelesen wurde.
	_XC163_MULTIPLE	Dieses Flag muss gesetzt sein.

Strukturelement	Werte	Bedeutung
.ulReferenceTime	*XC163_EXTERNAL* *XC163_20MHZ* *XC163_10MHZ* *XC163_5MHZ* *XC163_2MHZ* *XC163_1MHZ* *XC163_500KHZ* *XC163_200KHZ* *XC163_100KHZ* *XC163_50KHZ* *XC163_20KHZ* *XC163_10KHZ* *XC163_5KHZ* *XC163_2KHZ* *XC163_1KHZ* *XC163_500HZ* *XC163_200HZ* *XC163_100HZ* *XC163_50HZ* *XC163_20HZ* *XC163_10HZ*	Referenz-Frequenz
.usMultiple	*1 ... 65535*	Anzahl der Perioden, die gemessen werden sollen
.usCounterInput	*_XC163_DIN_x_y* *_XC163_NEG_EDGE_A* *_XC163_NEG_EDGE_B*	Zähleingang (siehe Tabelle) Zählen bei negativer Flanke an Eingang A Zählen bei negativer Flanke an Eingang B
.ulControlInput	*0* *_XC163_START_CTRL_x* *_XC163_STOP_CTRL_x* *_XC163_LOAD_CTRL_x* *_XC163_LATCH_CTRL_x* *_XC163_RESET_CTRL_x*	Keine Bedeutung Start, Stop, Laden, Zwischenspeichern und Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0* *XC163_EVENT_END_OF_ MEASUREMENT* *XC163_EVENT_OVERFLOW* *XC163_EVENT_START* *XC163_EVENT_STOP* *XC163_EVENT_LOAD* *XC163_EVENT_LATCH*	Es wird kein Ereignis signalisiert Eine Messung ist beendet worden Der Zähler ist übergelaufen Der Zähler wurde durch einem Steuerkanal gestartet Der Zähler wurde durch einem Steuerkanal gestoppt Der Zähler wurde durch einem Steuerkanal geladen Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert

Strukturelement	Werte	Bedeutung
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße umfaßt 2 Byte.

Anmerkung

1. Periodendauermessung über mehrere Perioden

Ist der Mode *_XC163_MULTIPLE* aktiviert, wird eine Periodendauermessung über mehrere Perioden durchgeführt. Hierzu muss *usIndexFirst = 0* und *usIndexLast = 1* eingestellt sein. Im Element *usMultiple* kann dann die Anzahl der Perioden eingetragen werden, die gemessen werden sollen. Die Steuerkommandos zum Starten, Stoppen, Laden, Latchen und Zurücksetzen müssen sowohl an Zähler 0 als auch an Zähler 1 gesendet werden!

2. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).
- Ist *.usWriteMode = IO_MODE_DIRECT,* startet der Zählvorgang nach dem Beschreiben des Kanals. Der Zähler wird dabei mit dem geschriebenem Wert vorinitialisiert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibbefehl der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

3. Lesen der Zählerwerte

Gelesen wird immer aus einem internen Zwischenspeicher.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y***
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11
1	x/y = 2/3, 8/9
2	x/y = 4/5, 10/11

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  `MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit`
  Die Tasknummer kann beliebig gewählt werden.
- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  `MAXLOADMDD slot=0 layer=0 progno=8FFF`
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

Beispiel:

Nachfolgend wird ein Kanal auf eine Periodendauermessung über 2 Perioden mit einer Zählfrequenz von 1 kHz geöffnet.

```
CPS_XC163  rcPeriod;
MAXCHLHND hPeriod;
MAX_ERROR Error;
ULONG ulSize;
UCHAR aucData[2];

// CPS ausfuellen
rcPeriod.usDevice = DEVICE_PERIOD_COUNTER;
rcPeriod.usIndexFirst = 0;
rcPeriod.usIndexLast = 1;
```

```
rcPeriod.usFlags = _CP_EXCLUSIVE;
rcPeriod.usReadMode = IO_MODE_LATCH;
rcPeriod.usWriteMode = IO_MODE_DIRECT;
rcPeriod.usMode = _XC163_AUTORUN | _XC163_OVERRIDE | _XC163_MULTIPLE;
rcPeriod.ulReferenceTime = XC163_1KHZ;
rcPeriod.usMultiple = 2;
rcPeriod.usCounterInput = _XC163_DIN_0_1;
rcPeriod.ulControlInput = 0;

// Kanal oeffnen
Error = max_open_channel(hMdd, sizeof (rcPeriod), &rcPeriod,
                                                  NULL, NULL, &hPeriod);

// Messwerte lesen
ulSize = 2;
Error = max_read_channel_block(hPeriod, &ulSize, (void*)aucData);
```

9.3.21. Inkrementalgeber

Mit diesem Device kann eine Inkrementalgebererfassung realisiert werden. Es stehen zwei unterschiedliche Betriebsarten (Mode A und Mode B) zur Verfügung:

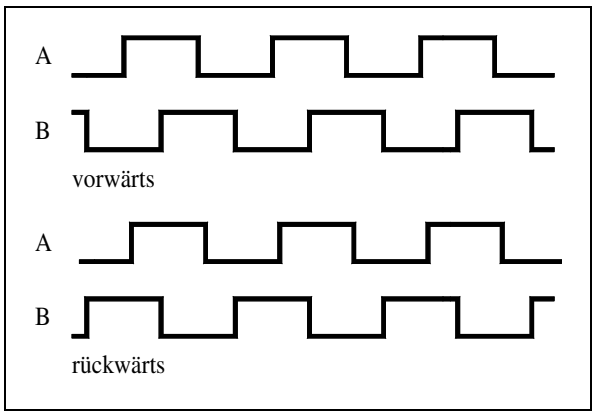


Abb. 9-4: Inkrementalgebersignale

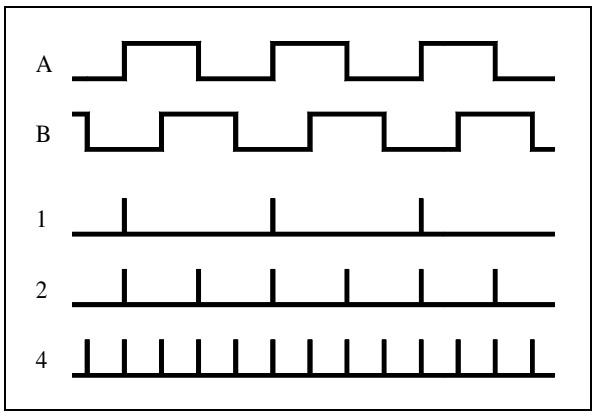


Abb. 9-5: Auswertung für Vorwärtsrichtung (ein-, zwei- oder vierfach)

Um auf einen Inkrementalgeber-Kanal zugreifen zu können, muss folgende CPS verwendet werden:

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_INCREMENTAL_COUNTER*	Kanal auf einen Zähler für Inkrementalgeber
.usIndexFirst	*0 ... 2* (Mode B: 0)	Nummer des ersten Zählers
.usIndexLast	*0 ... 2* (Mode B: 1)	Nummer des letzten Zählers
.usFlags	*_CP_EXCLUSIVE* *_CP_SYNC_CALLBACK*	Der Zugriff erfolgt immer exklusiv Nur in Echtzeitprogrammen verwendbar! Der Aufruf der Callback-Funktion erfolgt direkt. Ansonsten werden die Aufrufe gepuffert und im „Hintergrund“ (NI-Task) abgearbeitet.
.usReadMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Lesezugriff Der Zählerwert wird aus dem Zwischenspeicher gelesen
.usWriteMode	*IO_MODE_DIRECT* *IO_MODE_LATCH*	Direkter Schreibzugriff Der Zählerwert wird in den Zwischenspeicher geschrieben.

Strukturelement	Werte	Bedeutung
.usMode	*_XC163_AUTORUN*	Der Zählvorgang startet sofort nach dem Öffnen des Kanals (Zähler wird mit 0 vorinitialisiert).

	_XC163_MODE_A	Mode Flags: Es muss genau ein Flag gesetzt sein.
	_XC163_MODE_B	In dieser Betriebsart arbeitet der Zähler als Auf-/Abwärtszähler. Die Zählimpulse werden von beiden Phasen des Inkrementalgebers abgeleitet. Phase A des Inkrementalgebers wird an den Zähleingang A des Zählers und Phase B des Inkrementalgebers an Phase B des Zählers angeschlossen.
	----------	In dieser Betriebsart werden zwei Zähler des Moduls verwendet (0 und 1). Zähler 0 zählt die Impulse in Vorwärtsrichtung, Zähler 1 die Impulse in Rückwärtsrichtung. Die Position ergibt sich aus der Differenz beider Zählerstände. Die Phasen A und B werden an den Zähleingang A und B der Zähler angeschlossen (es muss ein Zähleingang verwendet werden, der an beiden Zählern verfügbar ist).
	_XC163_SINGLE_A	Flanken Flags: Es muss genau ein Flag gesetzt sein. Zähler zählt pos. bzw. neg. Flanke an Eingang A
	_XC163_SINGLE_B	Zähler zählt pos. bzw. neg. Flanke an Eingang B
	_XC163_DOUBLE_A	Zähler zählt pos. und neg. Flanke an Eingang A
	_XC163_DOUBLE_B	Zähler zählt pos. und neg. Flanke an Eingang B
	_XC163_QUADRUPLE	Zähler zählt pos. und neg. Flanke an Eingang A und B
.usCounterInput	*_XC163_DIN_x_y*	Zähleingang (siehe Tabelle)
	_XC163_NEG_EDGE_A	Zählen bei negativer Flanke an Eingang A
	_XC163_NEG_EDGE_B	Zählen bei negativer Flanke an Eingang B

Strukturelement	Werte	Bedeutung
.ulControlInput	*0*	Keine Bedeutung
	_XC163_START_CTRL_x	Start,
	_XC163_STOP_CTRL_x	Stop,
	_XC163_LOAD_CTRL_x	Laden,
	_XC163_LATCH_CTRL_x	Zwischenspeichern und
	_XC163_RESET_CTRL_x	Reset des Zählers erfolgt jeweils durch Steuerkanal x (x = 0 ... 3)
ulCallbackEvent	*0*	Es wird kein Ereignis signalisiert
	XC163_EVENT_ OVERFLOW	Der Zähler ist übergelaufen
	XC163_EVENT_ UNDERFLOW	Der Zähler ist untergelaufen
	XC163_EVENT_START	Der Zähler wurde durch einem Steuerkanal gestartet
	XC163_EVENT_STOP	Der Zähler wurde durch einem Steuerkanal gestoppt
	XC163_EVENT_LOAD	Der Zähler wurde durch einem Steuerkanal geladen
	XC163_EVENT_LATCH	Der Zählerstand wurde durch einem Steuerkanal zwischengespeichert
	XC163_EVENT_RESET	Der Zähler wurde durch einem Steuerkanal zurückgesetzt
	XC163_EVENT_INTERRUPT_OVERRUN	Interruptüberlauf wird signalisiert

Eingabe- und Ausgabedienst

Der Zugriff auf den Kanal erfolgt mit:

- **max_write_channel_block**
- **max_read_channel_block**

Die Blockgröße hängt dabei von der Anzahl der Zähler ab. Je Zähler werden 2 Byte benötigt.

Anmerkungen

1. Betriebsart

Ist Mode B aktiviert, muss *.usIndexFirst = 0* und *.usIndexLast = 1* sein.

2. Starten des Zählers

Das Starten des Zählers kann unterschiedlich erfolgen:

- Ist *.usMode = _XC163_AUTORUN*, startet der Zähler nach dem Öffnen des Kanals sofort (mit dem Anfangswert 0).

- Ist *.usWriteMode = IO_MODE_DIRECT*, startet der Zählvorgang nach dem Beschreiben des Kanals. Der Zähler wird dabei mit dem geschriebenem Wert vorinitialisiert.
- Ist *.usWriteMode = IO_MODE_LATCH*, wird mit einem Schreibbefehl der Zählerwert in einen Zwischenspeicher geschrieben. Zum Aktivieren des geschriebenen Wertes muss noch ein Lade- und evtl. Startbefehl (falls der Zähler noch nicht läuft) gesendet werden. Der Lade- und Startbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.usControlInput* mit der entsprechenden Konstanten belegt werden.

3. Lesen der Zählerwerte

Ist *usReadMode = IO_MODE_DIRECT*, kann der aktuelle Zählerstand durch einen Lesebefehl ermittelt werden. Wurde der Lesemodus auf *IO_MODE_LATCH* eingestellt, wird durch einen Lesezugriff der im Latch zwischengespeicherte Zählerstand ausgelesen. Ein Latchbefehl kann entweder per Software erfolgen und/oder über eine externe Steuerleitung. Wenn eine externe Steuerleitung verwendet werden soll, muss das Strukturelement *.ulControlInput* mit der entsprechenden Konstanten belegt werden.

Tabelle Zähleingang *_XC163_DIN_x_y:*

.usIndexFirst	***_XC163_DIN_x_y* für Mode A**	***_XC163_DIN_x_y* für Mode B**
0	x/y = 0/1, 2/3, 4/5, 6/7, 8/9, 10/11	x/y = 2/3, 8/9
1	x/y = 2/3, 8/9	
2	x/y = 4/5, 10/11	

Callback-Funktion

Über das Strukturelement *ulCallbackEvent* kann festgelegt werden, über welches Ereignis man informiert werden möchte. Die Anwenderfunktion bekommt 4 Byte Nutzdaten übergeben, die angeben, welche Ereignisse aufgetreten sind. Dazu werden die entsprechenden Konstanten, die in *ulCallbackEvent* anzugeben sind, zurückgegeben.

Voraussetzungen für den Einsatz unter OsX

Damit unter OsX ein Kanal mit Callback-Funktionalität geöffnet werden kann, müssen folgende Voraussetzungen erfüllt sein:

- Der Interrupt-Manager "irqmngr.exe" muss installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXINST file="irqmngr.exe" no=a00f task=65 tasktype=MAX_NI_TASKautoinit
  ```
  Die Tasknummer kann beliebig gewählt werden.

- Ist das Flag _CP_SYNC_CALLBACK im Strukturelement usFlags beim Öffnen des Kanals nicht gesetzt, so muss zusätzlich der Message-Modul-Device-Treiber installiert sein.
  Befehl für die INS-Datei:
  ```
  MAXLOADMDD slot=0 layer=0 progno=8FFF
  ```
  Ist das Flag gesetzt, muss dieser nicht installiert werden.

9.3.22. Pulsbreitenmodulation

Mit Hilfe dieses Kanals kann ein Pulsbreitenmoduliertes Signal mit den digitalen Ausgängen DOUT-1 bzw. DOUT-2 erzeugt werden.

Strukturelement	Werte	Bedeutung
.usDevice	*DEVICE_PWM*	Kanal auf für pulsbreitenmoduliertes Signal
.usIndexFirst	*1, 2*	DOUT auf dem das Signal ausgegeben werden soll
.usIndexLast	*.usIndexFirst*	
.usFlags	*_CP_EXCLUSIVE*	Der Zugriff erfolgt immer exklusiv
.usMode	*0*	reservierter Parameter
.usWriteMode	*0*	reservierter Parameter
.usReadMode	*0*	reservierter Parameter
.usControlInput	*0*	reservierter Parameter

Der Kanal belegt jeweils den Zähler 0 und 1 (für usIndexFirst = 0)
oder den Zähler 0 und 2 (für usIndexFirst = 1).

Eingabe- und Ausgabedienst

Der Kanal verfügt über keinen Ein- bzw. Ausgabedienst

Sonderdienste

Mit Hilfe von Sonderdiensten kann die Periodendauer und das Puls-/Pausenverhältnis verändert werden.

- **max_channel_control,** Steuerbefehl XC163_SET_PERIOD_TIME: dieser Befehl ändert die Periodendauer des Signals.

Dem Sonderdienst muss die Länge der Periodendauer in Form der folgenden Struktur übergeben werden:

```
Struct
{
    USHORT usSec;     // 0 ... 6500
    USHORT usMSec;    // 0 ... 999
    USHORT usUSec;    // 0 ... 999
    USHORT usNSec;    // 50 ... 999
} XC163_TIME;
```

- **max_channel_control,** Steuerbefehl XC163_SET_PULSE_PAUSE_RATIO: mit diesem Befehl kann das Puls-/Pausenverhältnis in % geändert werden. Dem Dienst wird ein USHORT-Wert mit dem Prozentwert übergeben.
- **max_channel_control,** Steuerbefehl CMD_START: mit diesem Befehl kann die Ausgabe des Signals gestartet werden (nachdem zuvor die Periodendauer und das Puls-/Pausenverhältnis eingestellt wurden).

9.3.23. Zuordnung der Zähler- und Steuereingänge zu den digitalen Eingängen

Modul-Stecker A	**Zähleingänge**			**Steuereingänge für Zähler 0, 1 und 2**	
	Zähler 0	**Zähler 1**	**Zähler 2**	**Steuereingänge 0 und 1**	**Steuereingänge 2 und 3**
DIN-0 DIN-1	A B			0 1	2 3
DIN-2 DIN-3	A B	A B		0 1	
DIN-4 DIN-5	A B		A B		2 3
DIN-6 DIN-7	A B			0 1	2 3
DIN-8 DIN-9	A B	A B		0 1	
DIN-10 DIN-11	A B		A B		2 3

9.4. Anschlusspins des Moduls (bezogen auf den Modul-Stecker A)

Eingang	+Eingang (= Anode)	-Eingang (= Kathode)	+Ausgang (= Kollektor)	-Ausgang (= Emitter)
DIN-0	1	2	-	-
DIN-1	3	4	-	-
DIN-2	5	6	-	-
DIN-3	7	8	-	-
DIN-4	9	10	-	-
DIN-5	11	12	-	-
DOUT-0	-	-	13	14
DOUT-1	-	-	15	16
DOUT-2	-	-	17	18
DOUT-3	-	-	19	20
DIN-6	21	22	-	-
DIN-7	23	24	-	-
DIN-8	25	26	-	-
DIN-9	27	28	-	-
DIN-10	29	30	-	-
DIN-11	31	32	-	-
DOUT-4	-	-	33	34
DOUT-5	-	-	35	36
DOUT-6	-	-	37	38
DOUT-7	-	-	39	40

9.5. Besondere Eigenschaften

(Angaben gelten für Version X-C16-3i/L und X-C16-3i/P)

Parameter	**Wert**	**Einheit**
Zählerkanäle, kaskadierbar, per Software verlängerbar bis 64 Bit	3	-
Auflösung	16	Bit
Eingangsfrequenz, max.	10	MHz
Zählfrequenz, max.	20	MHz
Externe Eingänge (X-C16-3i/L), Anzahl	12	-
Eingangsspannung, max.	±18	V
Schwelle max., Erkennung einer log. 0	2,2	V
Schwelle min., Erkennung einer log. 1	4	V
Schwelle Eingangsstrom, typ.	2	mA
Externe Eingänge (X-C16-3i/P), Anzahl	12	-
Eingangsspannung, max.	±30	V
Schwelle max., Erkennung einer log. 0	5	V
Schwelle min., Erkennung einer log. 1	13	V
Schwelle Eingangsstrom, typ.	2	mA
Externe Ausgänge, Anzahl	8	-
Ausgangsspannung, max.	80	V
Ausgangsspannung, max. bei log. 0 (Iout = 10 mA)	< 0,4	V
Ausgangsleistung, max. je Kanal	150	mW
Temperatur-Bereich, Betrieb	–40 .. +85	°C
Abmessungen	29x58x10	mm
Gewicht	10,6	g
Stromaufnahme (min./max.)		
3,3V (die X-Bus Spannungen ±12V werden nicht benötigt)	310/405	mA

Historie dieses Dokuments:

Datum	**Autor**	**Version**	**Änderung**
06.07.09	cb	_2	Verlängerten Zähler eingefügt
18.07.08	hb	_1	Aus 3. Auflage übernommen