



# ***bitcontrol® IEC 60870-5-104***

***Toolkit für QNX® 4.xx***

BitCtrl Systems GmbH  
Weißenfelser Str. 67  
04229 Leipzig

**2007**

***[www.bitcontrol.com](http://www.bitcontrol.com)***

---

## Copyright und Trademark

Die Software und die Dokumentation des Produktes **bitcontrol® IEC 60870-5-104** sind Eigentum der BitCtrl Systems GmbH und unterliegen bei ihrer Nutzung der lizenzrechtlichen Vereinbarung zwischen dem Endkunden und der BitCtrl Systems GmbH. Das Kopieren in jeglicher Form sowie die Weitergabe und der Verkauf der Software und der Dokumentation vom Endkunden an Dritte sind streng untersagt.

Die Dokumentation entspricht dem jeweiligen Stand der Entwicklung des Produktes **bitcontrol® IEC 60870-5-104**. Bei Fehlern und Ungenauigkeiten in der Beschreibung wenden Sie sich bitte an:



BitCtrl Systems GmbH  
 Weißenfeller Str. 67  
 04229 Leipzig, Germany  
 Tel. +49-341-49067 0  
 Fax +49-341-49067 15  
 Email [info@bitctrl.de](mailto:info@bitctrl.de)

Aufgeführte Marken und Produktnamen sind Marken der jeweiligen Rechtsinhaber. Intel®, Pentium® sind registrierte Warenzeichen von Intel Corporation. Microsoft®, Windows®, Windows NT®, DirectX®, DirectShow®, Windows Media® sind registrierte Warenzeichen der Microsoft Corporation.

## Haftung

Die BitCtrl Systems GmbH (im folgenden BitCtrl genannt) übernimmt für die Software **bitcontrol® IEC 60870-5-104** und seine Komponenten weder ausdrückliche noch implizite Garantien. Dies schließt einschränkungslos jegliche Ansprüche hinsichtlich Anwendbarkeit und Eignung der Software für einen bestimmten Nutzungszweck ein. BitCtrl haftet in keiner Weise für zufällige, indirekte oder Folgeschäden, die bei richtiger oder falscher Anwendung der Software entstehen. Dies gilt auch, wenn BitCtrl über die Möglichkeit eines solchen Schadens informiert wurde.

Es gelten die Allgemeinen Geschäftsbedingungen der BitCtrl Systems GmbH. Änderungen der Software bzw. der Dokumentation im Sinne des technischen Fortschritts bleiben vorbehalten.

## Release Stände

Dokumentation Version 1.0	erstellt von: BitCtrl Systems GmbH erreichbar unter: <a href="mailto:info@bitctrl.de">info@bitctrl.de</a> Stand: 27.10.06 Copyright © 2005,2006 by BitCtrl Systems GmbH - Leipzig, Germany
Dokumentation Version 1.1	erstellt von: BitCtrl Systems GmbH erreichbar unter: <a href="mailto:info@bitctrl.de">info@bitctrl.de</a> Stand: August 2007 Copyright © 2005 - 2007 by BitCtrl Systems GmbH - Leipzig, Germany
Für Hinweise auf Tipp- oder Formulierungsfehler wären wir dankbar.	

## Inhaltsverzeichnis

Abbildungsverzeichnis	4
Glossar	4
1 Einleitung	5
2 Systemvoraussetzungen	5
3 Funktionsweise	5
3.1 Controlled Station	5
3.2 Controlling Station	6
4 Library-Funktionen	7
4.1 TCS104_Open	7
4.2 TCS104_Close	7
4.3 TCS104_Event	7
4.4 TCS104_ReadEvent	8
4.5 TCS104_Request	8
5 Datenstrukturen	8
6 Ereignis-Behandlung	9
7 Ausgabe von IEC-ASDUs	10
8 Verbindungszustand	10
9 Beispielprogramme	11

## Abbildungsverzeichnis

Abbildung 1: Zustandsdiagramm Verbindungsschicht Controlling Station (li) und Controlled Station (re) ..... 11

## Glossar

<b>Begriff</b>	<b>Erklärung</b>
<b>ANSI</b>	ANSI steht für „American National Standards Institute“, die US-amerikanische Stelle zur Normung industrieller Verfahrensweisen.
<b>ASDU</b>	ASDU steht für „Application Service Data Unit“, Anwendungsdienste einer Dateneinheit.
<b>IEC</b>	IEC steht für „Internationale elektrotechnische Kommission“, das ist das internationale Normierungsgremium für Normen im Bereich der Elektrotechnik und Elektronik.
<b>PID</b>	Programm ID
<b>QNX®</b>	QNX® ist ein Echtzeitbetriebssystem, welches für eingebettete Systeme sehr gut geeignet ist.
<b>TCP</b>	Transmission Control Protocol

## 1 Einleitung

Das **bitcontrol® IEC 60870-5-104** Toolkit beinhaltet Komponenten zur Entwicklung von Kommunikationsanwendungen unter den Betriebssystem QNX® 4.xx, die auf dem Standard IEC 60870-5-104 basieren.

Diese Komponenten sind:

- Kommunikationsserver Slave-Station (Controlled-Station)
- Kommunikationsserver Master-Station (Controlling-Station)
- ANSI-C-Bibliothek und Headerdateien zur Kommunikation des Anwenderprogramms (Schicht 7) mit dem Kommunikationsservern
- Beispielprogramme

## 2 Systemvoraussetzungen

Die Benutzung des **bitcontrol® IEC 60870-5-104** Toolkits setzt folgende Komponenten voraus:

- Betriebssystem QNX® 4.xx
- Watcom C/C++ Compiler 10.6
- TCP-Runtime für QNX® 4.xx

## 3 Funktionsweise

Die Kommunikationsserver sind selbständig laufende Programme. Sie realisieren die unteren Schichten des Kommunikationsmodells bis zum Transport-Interface (User to TCP interface) innerhalb der Schicht 7.

Die Kommunikationsserver untereinander sind als Client-Server implementiert. Dabei agiert die 'Controlled Station' als Server und die 'Controlling Station' als Client.

### 3.1 Controlled Station

Der Kommunikationsserver für die Controlled Station ist als Daemon implementiert. Der Daemon wartet auf Kommunikationsanforderungen auf dem IEC 60870-5-104 Port (2404) und erzeugt für jede Verbindungsanforderung einen Kindprozess. Dieser Verbindungsprozess meldet sich beim zugehörigen Anwenderprozess (Application layer) an, der dann über diese Verbindung kommunizieren kann. Im Folgenden agiert der Anwendungsprozess als Client gegenüber dem Verbindungsprozess.

Die Kommunikation ist dann wie folgt organisiert:

- Anwendungsprozess initialisiert die Kommunikation mit dem Verbindungsprozess,
- Anwendungsprozess wartet auf Ereignisnotifikationen vom Verbindungsprozess oder sendet Aufträge,
- wird eine Ereignisnotifikation empfangen, fragt der Anwendungsprozess die Ereignisse ab und behandelt sie

Der Daemon ist ein selbständig laufender Prozess, der im Normalfall vom Anwendungsprozess gestartet wird. Der Start erfolgt mit:

**tcs104d -f<infile> -t<servertask> [options]\* &**mit:

infile: Initialisierungsdatei mit IEC-Parametern  
servertask: registrierter Name des Anwenderprozesses

Optionen:

-v: Verbose-Ausgaben  
-T: schreibt alle Telegramme in Tracefile, das File befindet sich im selben Verzeichnis wie das Deamon-Programm und heißt "l2\_<pid>.trc" mit  
<pid> = PID des Verbindungsprozesses

### 3.2 Controlling Station

Der Kommunikationsserver für die Controlling Station wird vom Anwendungsprozess mit dem gewünschten Kommunikationspartner (Host) als Parameter gestartet. Im Folgenden agiert der Anwendungsprozess als Client gegenüber dem Kommunikationsserver. Dieser versucht eine Verbindung aufzubauen und wartet auf Anfragen vom Anwendungsprozess.

Die Kommunikation ist dann wie folgt organisiert:

- Anwendungsprozess initialisiert die Kommunikation mit dem Verbindungsprozess,
- Anwendungsprozess steuert die Übertragung (START/STOP),
- Anwendungsprozess wartet auf Ereignisnotifikationen vom Verbindungsprozess oder sendet Aufträge,
- wird eine Ereignisnotifikation empfangen, fragt die Anwendungsprozess die Ereignisse ab und behandelt sie.

Der Kommunikationsserver der Controlling Station ist ein selbständig laufender Prozess, der im Normalfall vom Anwendungsprozess gestartet wird. Der Start erfolgt mit:

**tcs104client -f<infile> -h<host> [options]\* &**mit:

infile: Initialisierungsdatei mit IEC-Parametern  
host: Hostname oder IP-Adresse des Kommunikationspartners (Controlled-Station)

Optionen:

-v: Verbose-Ausgaben  
-T: schreibt alle Telegramme in Tracefile, das File befindet sich im selben Verzeichnis wie das Deamon-Programm und heißt "l2\_<pid>.trc" mit  
<pid> = PID des Verbindungsprozesses

## 4 Library-Funktionen

### 4.1 TCS104\_Open

#### Syntax:

**int TCS104\_Open (pid\_t pid)**

Die Funktion initialisiert die Kommunikation zum Kommunikationsserver.

#### Parameter:

pid: Process-Id des Kommunikationsservers.

#### Return:

Die Funktion liefert bei Erfolg den Wert 0. Im Fehlerfall ist **errno** auf einen der folgenden Werte gesetzt:

- **errno**-Werte der Funktion 'qnx\_proxy\_attach'
- **errno**-Werte der Funktion 'Send'
- EBUSY: der Kommunikationsserver hat bereits eine initialisierte Kommunikationsbeziehung

### 4.2 TCS104\_Close

#### Syntax:

**int TCS104\_Close (void)**

Die Funktion schließt die Kommunikation zum Kommunikationsserver. Der Kommunikationsserver schließt die Verbindung.

#### Parameter:

#### Return:

Die Funktion liefert bei Erfolg den Wert 0. Im Fehlerfall ist **errno** auf einen der folgenden Werte gesetzt:

- **errno**-Werte der Funktion 'Send'
- EBUSY: der Kommunikationsserver hat bereits eine initialisierte Kommunikationsbeziehung mit einem anderen Anwendungsprozess

### 4.3 TCS104\_Event

#### Syntax:

**int TCS104\_Event (pid\_t pid)**

Die Funktion testet, ob *pid* eine Notifikation des Kommunikationsservers ist.

#### Parameter:

pid: Process-Id

**Return:**

Die Funktion liefert einen Wert  $\neq 0$  falls *pid* eine Notifikation eines zugehörigen Kommunikationsservers ist, ansonsten 0.

## 4.4 TCS104\_ReadEvent

**Syntax:**

```
int TCS104_ReadEvent (iecappmsg *msg)
```

Die Funktion liest ein Ereignis vom Kommunikationsserver.

**Parameter:**

msg: Zeiger auf iecappmsg-Struktur.

**Return:**

Die Funktion liefert bei Erfolg den Wert 0, die Struktur *msg* ist mit einem Ereignis gefüllt. Im Fehlerfall ist **errno** auf einen der folgenden Werte gesetzt:

- **errno**-Werte der Funktion 'Send'

## 4.5 TCS104\_Request

**Syntax:**

```
int TCS104_Request (iecappmsg *msg)
```

Die Funktion sendet einen Auftrag zum Kommunikationsserver.

**Parameter:**

msg: Zeiger auf iecappmsg-Struktur.

**Return:**

Die Funktion liefert bei Erfolg den Wert 0. Im Fehlerfall ist **errno** auf einen der folgenden Werte gesetzt:

- **errno**-Werte der Funktion 'Send'

## 5 Datenstrukturen

Die Kommunikation zwischen dem Anwendungsprozess und dem Kommunikationsserver erfolgt über die Union 'iecappmsg', die in der Headerdatei 'tcs104tk.h' definiert ist. Die Unterscheidung der jeweiligen Daten erfolgt über das 'type' Element.

Folgende Typen sind in 'IECApp\_msg\_t' definiert:

'type'	Kommunikationsrichtung	Funktion
IECAPP_OPEN	Anwendung→Verbindung	wird von der Funktion TCS104_Open benutzt
IECAPP_CLOSE	Anwendung→Verbindung	wird von der Funktion TCS104_Close benutzt
IECAPP_READEVENT	Anwendung→Verbindung	wird von der Funktion TCS104_ReadEvent benutzt
IECAPP_REQUEST	Anwendung→Verbindung	Ausgabe von IEC Telegrammen
IECAPP_GETSTATE	Anwendung→Verbindung	Abfrage des Zustandes der Verbindung
IECAPP_START	Anwendung→Verbindung	IEC Verbindung START_DATA_TRANSFER (nur für Controlling Station)
IECAPP_STOP	Anwendung→Verbindung	IEC Verbindung STOP_DATA_TRANSFER (nur für Controlling Station)
IECAPP_CON	Verbindung→Anwendung	Confirmation-Ereignis
IECAPP_IND	Verbindung→Anwendung	IEC Indikation
IECAPP_STATE	Verbindung→Anwendung	Verbindungszustand
IECAPP_RESP	intern	intern
IECAPP_CAPTURE	intern	intern

## 6 Ereignis-Behandlung

Ein erfolgreicher Aufruf der Funktion TCS104\_ReadEvent liefert eine iecappmsg-Struktur mit einem der folgenden Typen:

'type'	Ereignis	Daten
IECAPP_IND	IEC-Indikation (siehe IEC 60870-5-5)	iecappmsg.ind.data enthält eine ASDU der Länge iecappmsg.ind.dlen
IECAPP_CON	IEC-Confirmation (siehe	iecappmsg.con.code enthält die Sequen-

	IEC 60870-5-5)	ce-Nummer der bestätigten ASDU
IECAPP_STATE	Zustand der Verbindung	iecapmsg.state.code enthält den Zustand der Verbindung

## 7 Ausgabe von IEC-ASDUs

Die Übergabe von zu sendenden Daten an den Kommunikationsserver erfolgt als ASDU. Die Verbindungsschicht liefert die zugehörige Sequenznummer zurück, die zur Überprüfung der Confirmation benutzt werden kann. Die Auswertung der Bestätigung der Sequenznummern muss im Anwendungsprozess entsprechend der IEC 60870-5-104 Norm erfolgen:

Die Bestätigung mit einer Sequenznummer <snr> bestätigt alle unbestätigten ASDUs mit einer Sequenznummer <= <snr>.

## 8 Verbindungszustand

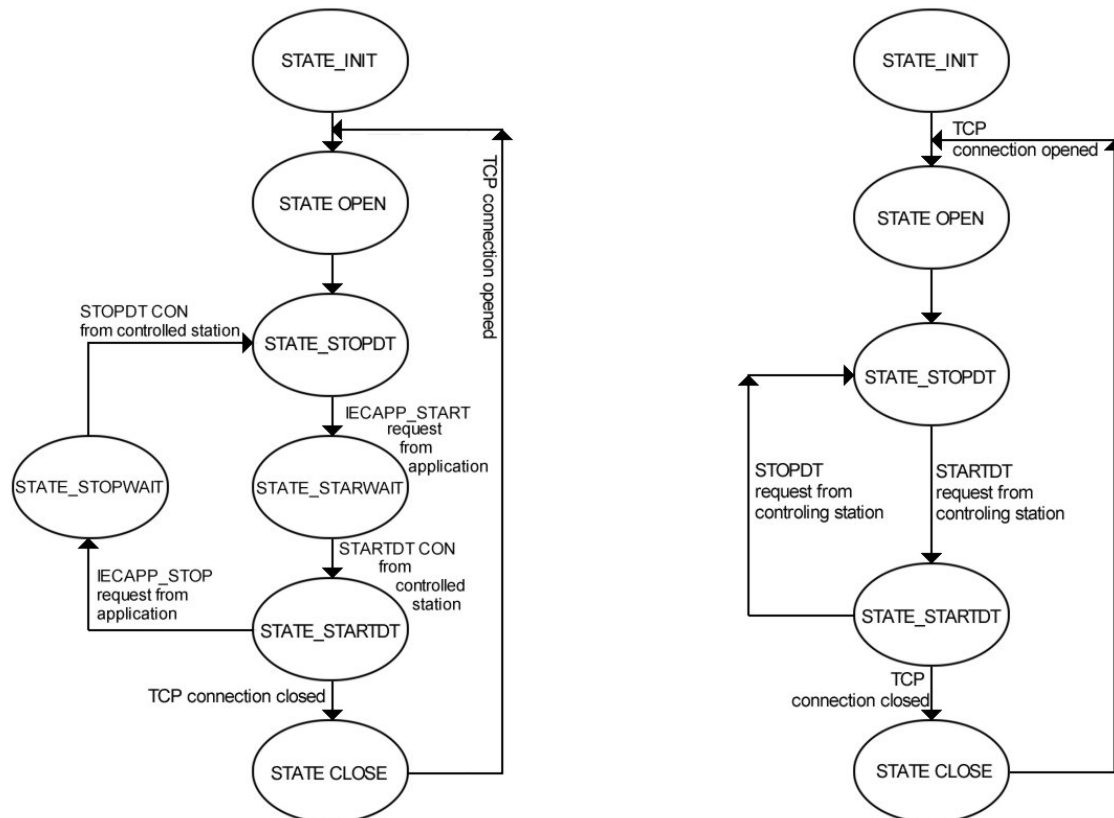
Der Zustand der Verbindung nimmt einen der in der Datei 'tcs104.h' als 'tcs104states' definierten Werte an. Diese Werte haben folgende Bedeutung:

Zustand	Bedeutung
STATE_INIT	Initialzustand der Verbindungsschicht
STATE_OPEN	die TCP-Verbindung ist hergestellt
STATE_CLOSE	die TCP-Verbindung ist unterbrochen
STATE_STARTDT	die TCS104-Verbindung befindet sich im Zustand 'Start-Datatransfer'
STATE_STOPDT	die TCS104-Verbindung befindet sich im Zustand 'Stop-Datatransfer'
STATE_TEST	die TCS104-Verbindung befindet sich im Zustand 'Verbindungstest'
STATE_STARTWAIT	nur Controlling station: die TCS104-Verbindung wartet auf die Bestätigung eines 'Start-Datatransfer'-Frames von der controlled station
STATE_STOPWAIT	nur Controlling station: die TCS104-Verbindung wartet auf die Bestätigung eines 'Stop-Datatransfer'-Frames von der controlled station

STATE\_BUSY

Antwort der Verbindungsschicht auf einen IECAPP\_OPEN-Request von der Anwendung, wenn die Verbindungsschicht bereits eine zugeordnete Anwendung hat

Die Zustandsübergänge sind in den folgenden Diagrammen dargestellt.



**Abbildung 1:** Zustandsdiagramm Verbindungsschicht Controlling Station (li) und Controlled Station (re)

## 9 Beispielprogramme

Das **bitcontrol® IEC 60870-5-104** Toolkit enthält Beispielprogramme für eine Controlled-Station (TCS104subst.c) und für eine Controlling-Station (test104s.c) im Unterverzeichnis 'samples'.