Jetter
automation

Getting Started

Erstinbetriebnahme eines JXM-IO-EX30 in C
mit einer HY-TTC580-Steuerung von TTControl

60886706_00

We automate your success.

Dieses Dokument wurde von der Jetter AG mit der gebotenen Sorgfalt und basierend auf dem ihr bekannten Stand der Technik erstellt. Änderungen und technische Weiterentwicklungen an unseren Produkten werden nicht automatisch in einem überarbeiteten Dokument zur Verfügung gestellt. Die Jetter AG übernimmt keine Haftung und Verantwortung für inhaltliche oder formale Fehler, fehlende Aktualisierungen sowie daraus eventuell entstehende Schäden oder Nachteile.

Jetter AG
Gräterstraße 2
71642 Ludwigsburg
Germany

Telefon:

Zentrale	+49 7141 2550-0
Vertrieb	+49 7141 2550-531
Technische Hotline	+49 7141 2550-444

E-Mail:

Technische Hotline	hotline@jetter.de
Vertrieb	sales@jetter.de

Originaldokument

Dokumentenversion	1.01
Ausgabedatum	29.10.2021

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Erstinbetriebnahme eines JXM-IO-EX30 von Jetter an einer Steuerung HY-TTC 580 von TTControl via CANopen.

Die Programmierung des CANopen-Protokolls erfolgt über die Entwicklungsumgebung Code Composer Studio mit der Programmiersprache C.

1.1 Informationen zum Dokument

Dieses Dokument ist Teil des Produkts und muss vor dem Einsatz des Geräts gelesen und verstanden werden. Es enthält wichtige und sicherheitsrelevante Informationen, um das Produkt sachgerecht und bestimmungsgemäß zu betreiben.

Zielgruppen

Dieses Dokument richtet sich an Fachpersonal.
Das Gerät darf nur durch fachkundiges und ausgebildetes Personal in Betrieb genommen werden.
Der sichere Umgang mit dem Gerät muss in jeder Produktlebensphase gewährleistet sein. Fehlende oder unzureichende Fach- und Dokumentenkenntnisse führen zum Verlust jeglicher Haftungsansprüche.

Verfügbarkeit von Informationen

Stellen Sie die Verfügbarkeit dieser Informationen in Produktnähe während der gesamten Einsatzdauer sicher.
Informieren Sie sich im Downloadbereich unserer Homepage über Änderungen und Aktualität des Dokuments. Das Dokument unterliegt keinem automatischen Änderungsdienst.

Start | Jetter - We automate your success.

1.2 Darstellungskonventionen

Unterschiedliche Formatierungen erleichtern es, Informationen zu finden und einzuordnen. Im Folgenden das Beispiel einer Schritt-für-Schritt-Anweisung:

✓ Dieses Zeichen weist auf eine Voraussetzung hin, die vor dem Ausführen der nachfolgenden Handlung erfüllt sein muss.

► Dieses Zeichen oder eine Nummerierung zu Beginn eines Absatzes markiert eine Handlungsanweisung, die vom Benutzer ausgeführt werden muss. Arbeiten Sie Handlungsanweisungen der Reihe nach ab.

⇨ Der Pfeil nach Handlungsanweisungen zeigt Reaktionen oder Ergebnisse dieser Handlungen auf.

ⓘ INFO	**Weiterführende Informationen und praktische Tipps**
	In der Info-Box finden Sie weiterführende Informationen und praktische Tipps zu Ihrem Produkt.

2 Systemvoraussetzungen

Folgende Hardware- und Softwarevoraussetzungen müssen zur Nutzung des JXM-IO-EX30 erfüllt sein:

JXM-IO-EX30

Min. HW-Revision	1.02
Min. OS-Version	2.23.0.00

HY-TTC 580

Min. HW-Revision	1.03-C
Min. OS-Version	2.10.1
Bemerkung	Hersteller TTControl

Code Composer Studio

Min. Version	6.2

3 Compiler herunterladen

Um das Steuerungsprogramm korrekt builden zu können, benötigen Sie einen funktionsfähigen Compiler.

Vorgehensweise

1. Laden Sie einen funktionsfähigen Compiler herunter (hier: ARM Compiler 5.1.6).
2. Passen Sie ggf. den Pfad zum Compiler in der Datei *settings.mk* an.

```
# path with C compiler
ifndef C_COMP_PATH
    C_COMP_PATH = C:\TI\ccsv6\tools\compiler\arm_5.1.6\bin
endif
```

Falls Sie einen anderen Compiler verwenden, dann passen Sie dies in der Datei *cgt_version_check.bat* an.

```
rem check against the following version
set CGT_VERSION=5.1.6
```

4 CAN-Schnittstelle initialisieren

Das folgende Code-Beispiel zeigt, wie Sie die CAN-Schnittstelle initialisieren.

Schritt 1 – Benötigte APIs von TTControl im Programm einbinden

```
/*****************************************************************
 * Includes
 *****************************************************************/
#include "APDB.h"
#include "IO_Driver.h"
#include "IO_CAN.h"
```

Schritt 2 – Handle und Variablen anlegen

```
/* CAN handles */
ubyte2 can_handle_writeEx30 = 0;
ubyte2 can_handle_r_50 = 0;

/* CAN error counters */
ubyte1 rx_error_counter = 0;
ubyte1 tx_error_counter = 0;

/* buffer for CAN frame*/
IO_CAN_DATA_FRAME can_frame_ex30 = { 0 };
```

Schritt 3 – CAN-Schnittstelle initialisieren

```
/* initialize CAN channel */
io_error = IO_CAN_Init(CAN_ECHO_CHANNEL, /* CAN channel */
                       IO_CAN_BIT_250_KB,       /* 250 kBit/s */
                       0,                       /* default */
                       0,                       /* default */
                       0,                       /* default */
                       0);                      /* default */

/* get CAN channel status */
io_error = IO_CAN_Status(CAN_ECHO_CHANNEL,      /* CAN channel */
                         &rx_error_counter,     /* receive error counter */
                         &tx_error_counter);    /* transmit error counter */

/* initialize CAN handle for writing */
io_error = IO_CAN_ConfigMsg(&can_handle_w,      /* handle variable */
                            CAN_ECHO_CHANNEL,   /* CAN channel */
                            IO_CAN_MSG_WRITE,   /* handle for writing */
                            IO_CAN_STD_FRAME,   /* standard ID */
                            0,                  /* not relevant for writing */
                            0);                 /* not relevant for writing */
```

```
/* initialize CAN handle for reading */
io_error = IO_CAN_ConfigMsg(&can_handle_r_50,   /* handle variable */
                            CAN_ECHO_CHANNEL,   /* CAN channel */
                            IO_CAN_MSG_READ,    /* handle for reading */
                            IO_CAN_STD_FRAME,   /* standard ID */
                            0x50,               /* listen for ID 0x50 */
                            0x07FFU);           /* acceptance mask */
```

Schritt 4 – Programmschleife programmieren

CAN-Nachrichten lesen:

```
/* read CAN message */
io_error = IO_CAN_ReadMsg(can_handle_r_70,  /* use handle for ID 0x70 */
                          &can_frame);      /* local CAN frame buffer */
```

CAN-Nachrichten schreiben:

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);   /* local CAN frame buffer */
```

5 JXM-IO-EX30 konfigurieren

Die Anschlüsse am JXM-IO-EX30 konfigurieren Sie per SDO.
Die entsprechenden Indexe, Subindexe und Werte entnehmen Sie der Betriebsanleitung des JXM-IO-EX30.

Schritt 1 – Daten für die gewünschte Konfiguration eintragen

```
/* DO konfigurieren */
can_frame_ex30.length = 8;
can_frame_ex30.id = 0x630;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x23; // Command-Sprecifier
can_frame_ex30.data[1] = 0x18; // Index Low
can_frame_ex30.data[2] = 0x21; // Index High
can_frame_ex30.data[3] = 0x01; // SubIndex
can_frame_ex30.data[4] = 0x07; // Wert
can_frame_ex30.data[5] = 0x00; // Wert
can_frame_ex30.data[6] = 0x00; // Wert
can_frame_ex30.data[7] = 0x00; // Wert
```

Schritt 2 – Konfiguration an den JXM-IO-EX30 senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);    /* local CAN frame buffer */
```

Schritt 3 – CAN-Bus auf OPERATIONAL setzen

```
/* NMT setzen Operational */
can_frame_ex30.length = 2;
can_frame_ex30.id = 0x0000;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x1;
can_frame_ex30.data[1] = 0x30;
```

Schritt 4 – Daten an den CAN-Bus senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30); /* local CAN frame buffer */
```

6 SDO-Nachricht – Beispiel

Das folgende Code-Beispiel zeigt, wie Sie die digitalen Ausgänge mit einer SDO-Nachricht an- oder ausschalten.

Schritt 1 – SDO-Nachricht zum Anschalten der digitalen Ausgänge vorbereiten

```
/* DO anschalten per SDO setzen */
can_frame_ex30.length = 8;
can_frame_ex30.id = 0x630;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x2F;
can_frame_ex30.data[1] = 0x00;
can_frame_ex30.data[2] = 0x62;
can_frame_ex30.data[3] = 0x02;
can_frame_ex30.data[4] = 0x3C;
can_frame_ex30.data[5] = 0x00;
can_frame_ex30.data[6] = 0x00;
can_frame_ex30.data[7] = 0x00;
```

Schritt 2 – SDO-Nachricht senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);    /* local CAN frame buffer */
```

Schritt 3 – SDO-Nachricht zum Ausschalten der digitalen Ausgänge vorbereiten

```
/* DO ausschalten per SDO setzen */
can_frame_ex30.length = 8;
can_frame_ex30.id = 0x630;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x2F;
can_frame_ex30.data[1] = 0x00;
can_frame_ex30.data[2] = 0x62;
can_frame_ex30.data[3] = 0x02;
can_frame_ex30.data[4] = 0x00;
can_frame_ex30.data[5] = 0x00;
can_frame_ex30.data[6] = 0x00;
can_frame_ex30.data[7] = 0x00;
```

Schritt 4 – SDO-Nachricht senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);    /* local CAN frame buffer */
```

7 PDO-Nachricht – Beispiel

Das folgende Code-Beispiel zeigt, wie Sie die digitalen Ausgänge mit einer PDO-Nachricht an- oder ausschalten.

Schritt 1 – PDO-Nachricht zum Anschalten der digitalen Ausgänge vorbereiten

```
/* DO anschalten per PDO setzen */
can_frame_ex30.length = 2;
can_frame_ex30.id = 0x230;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x00;
can_frame_ex30.data[1] = 0x3C;
```

Schritt 2 – PDO-Nachricht senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);    /* local CAN frame buffer */
```

Schritt 3 – PDO-Nachricht zum Ausschalten der digitalen Ausgänge vorbereiten

```
/* DO ausschalten per PDO setzen */
can_frame_ex30.length = 2;
can_frame_ex30.id = 0x230;
can_frame_ex30.id_format = IO_CAN_STD_FRAME;

can_frame_ex30.data[0] = 0x00;
can_frame_ex30.data[1] = 0x00;
```

Schritt 4 – PDO-Nachricht senden

```
io_error = IO_CAN_WriteMsg(can_handle_writeEx30, /* use handle for writing */
                           &can_frame_ex30);    /* local CAN frame buffer */
```

8 Service

8.1 Kundendienst

Bei Fragen, Anregungen oder Problemen stehen Ihnen die Experten unseres Kundendienstes zur Verfügung. Diese können Sie telefonisch über unsere Technische Hotline oder über unser Kontaktformular auf unserer Homepage erreichen:

Technische Hotline | Jetter - We automate your success.

Oder schreiben Sie eine E-Mail an die Technische Hotline:

hotline@jetter.de

Bei E-Mail- oder Telefonkontakt benötigt die Hotline folgende Informationen:

- Hardwareversion und Seriennummer
  Die Seriennummer und Hardwareversion Ihres Produkts entnehmen Sie dem Typenschild.

Jetter automation

Jetter AG
Gräterstraße 2
71642 Ludwigsburg
www.jetter.de

E-Mail info@jetter.de
Telefon +49 7141 2550-0

We automate your success.