Systembus

Anwender-Register

Benutzer-Information

ArtikelNr.: -
Auflage 1.03

August 2004 / Printed in Germany

Dieses Dokument besitzt nur in Verbindung mit dem zugehörigen Dokument zu den Sicherheitshinweisen volle Gültigkeit.

Die Firma JETTER AG behält sich das Recht vor, Änderungen an ihren Produkten vorzunehmen, die der technischen Weiterentwicklung dienen. Diese Änderungen werden nicht notwendigerweise in jedem Einzelfall dokumentiert.

Dieses Handbuch und die darin enthaltenen Informationen wurden mit der gebotenen Sorgfalt zusammengestellt. Die Firma JETTER AG übernimmt jedoch keine Gewähr für Druckfehler oder andere daraus entstehende Schäden.

Inhaltsverzeichnis

1 Einleitung

Dieses Dokument beschreibt die Konfiguration und den Zugriff auf Systembus Anwender-Register. Diese Register bieten dem Anwender die Möglichkeit auf beliebige Objekte von CANopen Erweiterungsmodulen zuzugreifen.

Normalerweise bilden die Steuerungen der Jetter AG alle relevanten Objekte der CANopen Erweiterungsmodule in Registern ab. Speziell bei Anwendungen, in denen nicht abgebildete Objekte benötigt werden, kann auf diese mit den Systembus Anwender-Registern zugegriffen werden.

Hinweis

Bei der Konfiguration von Systembus Anwender-Registern ist mit großer Sorgfalt vorzugehen. Über diese Register ist ein Zugriff auf alle zur Kommunikation erforderlichen Objekten der CANopen Module möglich. Deshalb ist darauf zu achten, dass keine Objekte im Bereich zwischen 1000_{hex} und $1FFF_{hex}$ benutzt werden.

1.1 Systemvoraussetzungen

Die in diesem Dokument beschriebene Funktionalität der Systembus Anwender-Register werden von folgenden Steuerungen ab der angegebenen SW-Version unterstützt.

Software-Versionen der Steuerungen	
Steuerung	**ab SW-Version**
Nano-B	V 3.50
Nano-C	V 3.50
Nano-D	V 3.50
JetControl 24X	V 3.13
JX6-SB / JX6-SB-I	V 2.12

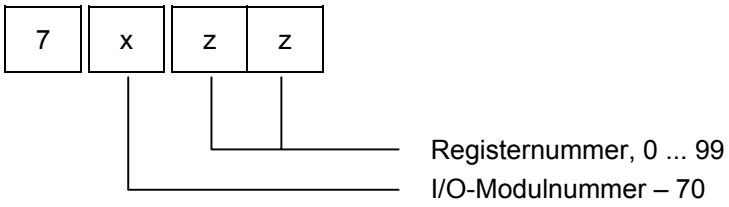
2 Software und Programmierung

2.1 Registernummern

2.1.1 Adressierung der Register

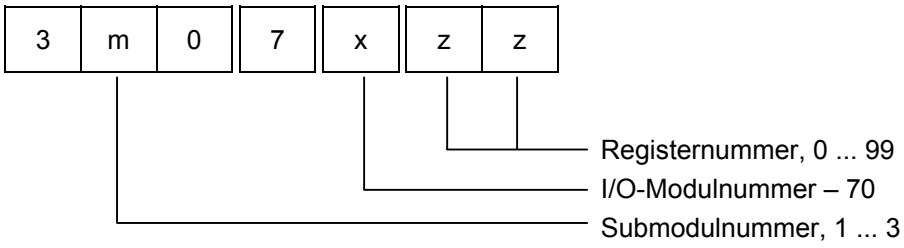
Codierung der Registernummern bei Nano und JC 24X

Die Registernummern bei Nano und JC 24X sind vierstellig. Sie ergeben sich aus der I/O-Modulnummer und der Nummer des jeweiligen Registers.



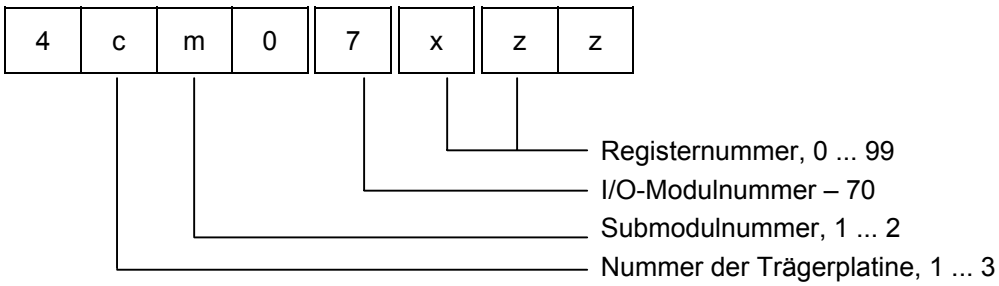
Codierung der Registernummern bei JX6-SB(-I) und JC64X

Die Registernummern beim JX6-SB Modul, eingebaut in einen JC64X, sind siebenstellig. Sie ergeben sich aus der Submodulnummer, der I/O-Modulnummer und der Nummer des jeweiligen Registers.



Codierung der Registernummern bei JX6-SB(-I) und JC800

Die Registernummern beim JX6-SB Modul, eingebaut in einen JC800, sind achtstellig. Sie ergeben sich aus der Nummer der Trägerplatine, der Submodulnummer auf der Trägerplatine und der Nummer des jeweiligen Registers.



In diesem Dokument werden generell die Registernummern bei Nano bzw. JetControl 24X verwendet. Für Anwendungen mit einem JX6-SB / JX6-SB-I Modul müssen die Ziffern „3m0“ bzw. „4cm0“ vorangestellt werden.

2.2 Systembus Anwender-Register

Die Systembus Anwender-Register bieten die Möglichkeit nahezu jedes beliebige Objekt von CANopen Modulen zu lesen oder zu schreiben. Eine Ausnahme bilden nur die Objekte mit den Datentypen „visible string“ und „float“, auf die nicht zugegriffen werden kann.

Mit den Standard-Registern und den Systembus Anwender-Registern hat der Anwender die größtmögliche Flexibilität um auf die Funktionen eines Gerätes zuzugreifen. Für jedes CANopen Modul, die das Geräteprofil DSP 401 für „Generic IO“ unterstützen, reserviert die Steuerung fünf frei definierbare Systembus Anwender-Register. Die Objekte werden den Registern 7x65 bis 7x69 zugeordnet.
Zusätzlich stehen bei CANopen Modulen, die das Geräteprofil DSP 402 für „Drives and Motion Control“ unterstützen, die Register 7x70 bis 7x85 als Systembus Anwender-Register zur Verfügung.

2.2.1 Funktionsweise

Bevor auf ein Objekt eines CANopen Moduls zugegriffen werden kann muss dieses zuerst einem der Register 7x65 bis 7x84 zugeordnet werden. Die Zuordnung ist sofort wirksam und kann beliebig oft geändert werden.

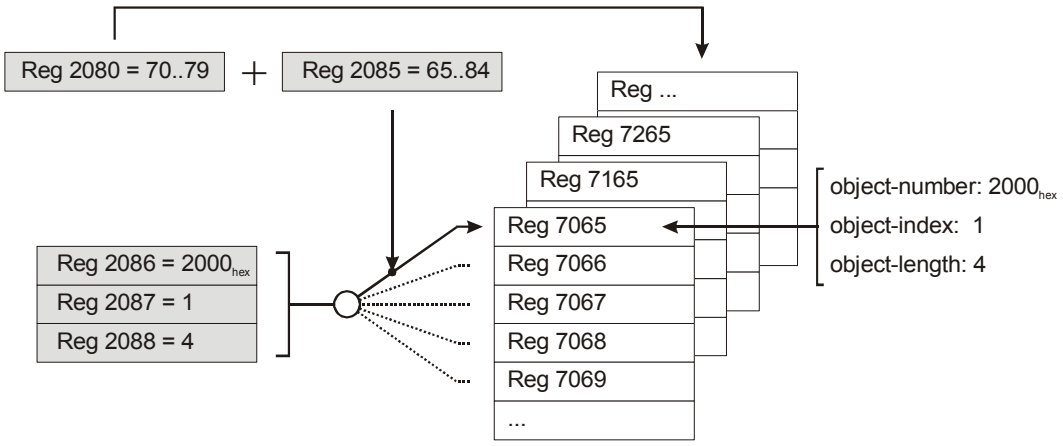


Abb. 1: Konfiguration der Systembus Anwender-Register

Die Registernummer 7x65 bis 7x84 ergibt sich aus dem Register 2080 und dem Register 2085. Über die Register 2086 bis 2088 wird Objektnummer, Länge und Sub-Index zugewiesen.
Nach der Konfiguration lässt sich das Objekt über die Register 7x65 bis 7x84 lesen und schreiben.

2.2.2 Konfiguration

Register 2080:	Systembus Modulnummer
Funktion	**Beschreibung**
Lesen	aktuelle Modulnummer
Schreiben	neue Modulnummer
Wertebereich	70 – 79
Wert nach Reset	70

Register 2085:	Index für Systembus Anwender-Register
Funktion	**Beschreibung**
Lesen	aktueller Index
Schreiben	neuer Index
Wertebereich	65 – 84
Wert nach Reset	65

Register 2086:	Objekt-Nummer für Systembus Anwender-Register
Funktion	**Beschreibung**
Lesen	aktuelle Objekt-Nummer
Schreiben	neue Objekt-Nummer
Wertebereich	sinnvoll: 2000_{hex} – $6FFF_{hex}$
Wert nach Reset	0

Register 2087:	Objekt-Index für Systembus Anwender-Register
Funktion	**Beschreibung**
Lesen	aktueller Objekt-Index
Schreiben	neuer Objekt-Index
Wertebereich	0 – 255
Wert nach Reset	0

Register 2088:	Objekt-Länge für Systembus Anwender-Register
Funktion	**Beschreibung**
Lesen	aktuelle Objekt-Länge
Schreiben	neue Objekt-Länge Bit 7 = 0, nicht vorzeichenbehafteter Wert Bit 7 = 1, vorzeichenbehafteter Wert
Wertebereich	0, 1 – 4, 129 - 132 1 = 8-Bit Wert unsigned 0 ... + 255 2 = 16-Bit Wert unsigned 0 ... + 65535 3 = 24-Bit Wert unsigned 0 ... + 16777215 4 = 32-Bit Wert - 2147483648 ... + 2147483647 129 = 8-Bit Wert signed -128 ... + 127 130 = 16-Bit Wert signed - 32768 ... 32767 131 = 24-Bit Wert signed - 8388608 ... + 8388607 132 = 32-Bit Wert - 2147483648 ... + 2147483647
Wert nach Reset	0

Beispiel 1: Konfiguration der Systembus Anwender-Register

Das Objekt 2201_{hex} mit dem Sub-Index 2 soll über das Register 7x67 direkt erreichbar sein. Das Objekt hat eine Länge von zwei Bytes. Das CANopen Modul hat die Modulnummer 71. Die Deklartion des STRUCTs „SYS_BUS“ befindet sich im Anhang.

ST Programmfile

```
#include "sysbus.stp"

var
    stSysBus        : SYS_BUS   AT  %vl 2000;
    nJXSIO_UserReg  : INT       AT  %vl 7165;
end_var

TASK 0
    // …
    stSysBus.nsModulIdx         := 71;      // Modulnummer
    stSysBus.nsUserRegIdx       := 65;      // Systembus Anwender-Reg.
    stSysBus.nsUserRegObj       := 0x2201;  // Objekt-Nummer
    stSysBus.nsUserRegSubIdx    := 2;       // Objekt-Index
    stSysBus.nsUserRegLen       := 2;       // Objekt-Länge, unsigned

    nJXSIO_UserReg  := 500;         // Wert 500 in Objekt 2201 schreiben
    // …
END_TASK;
```

Programmfile

```
REGISTER_LOAD (2080, 71)        // Modulnummer
REGISTER_LOAD (2085, 67)        // Systembus Anwender-Register
REGISTER_LOAD (2086, 0x2201)    // Objekt-Nummer
REGISTER_LOAD (2087, 2)         // Objekt-Index
REGISTER_LOAD (2088, 2)         // Objekt-Länge, unsigned
// neuen Wert in Objekt 2201 schreiben
REGISTER_LOAD (7167, 500)       // Wert schreiben
```

Beispiel 2: Konfiguration eines 16-Bit Analogwertes

Von einem JX-SIO soll der erste Analogeingang-Kanal einer Klemme über ein Systembus Anwender-Register gelesen werden. An der Analogklemme werden Spannungen zwischen -10 V und +10 V mit einer Auflösung von 16-Bit eingelesen. Der im Systembus Anwender-Register dargestellte Wert ist somit 16-Bit groß und vorzeichenbehaftet.
Die Analogeingangswerte eines JX-SIO werden im Objekt 6401_{hex} abgebildet. Sie sind über die Sub-Indizes 1 und folgende lesbar.
Das Beispiel macht zwar wenig Sinn, da der Analogeingangswert der Klemme in Register über das Register 5060 ebenfalls lesbar ist. Andererseits muss in diesem Beispiel der Wert in den Registern 5060 und 7065 übereinstimmen.

ST Programmfile

```
#include "sysbus.stp"

var
    stSysBus        : SYS_BUS   AT  %vl 2000;
    nJXSIO_UserReg  : INT       AT  %vl 7065;
    nJXSIO_AnalogIn : INT       AT  %vl 5060;
    nAnalogValue    : INT       AT  %vl 100;
end_var

TASK 0
    // …
    stSysBus.nsModulIdx         := 70;      // Modulnummer
    stSysBus.nsUserRegIdx       := 65;      // Systembus Anwender-Reg.
    stSysBus.nsUserRegObj       := 0x6401;  // Objekt-Nummer
    stSysBus.nsUserRegSubIdx    := 1;       // Objekt-Index
    stSysBus.nsUserRegLen       := 82;      // Objekt-Länge, signed

    nAnalogValue := nJXSIO_UserReg;     // Analogwert lesen
    // nun ist nAnalogValue = nJXSIO_AnalogIn
    // …
END_TASK;
```

Programmfile

```
    REGISTER_LOAD (2080, 70)        // Modulnummer
    REGISTER_LOAD (2085, 65)        // Systembus Anwender-Register
    REGISTER_LOAD (2086, 0x6401)    // Objekt-Nummer
    REGISTER_LOAD (2087, 1)         // Objekt-Index
    REGISTER_LOAD (2088, 0x82)      // Objekt-Länge, signed
    // Analogwert in Register 100 ablegen
    REGISTER_LOAD (100, @7065)
    // Wert muss mit Register 5060 übereinstimmen
```

2.3 Modul-Info Schnittstelle

Über die Modulinfo-Schnittstelle lassen sich einzelne Prozessdatenkanäle aktivieren und deaktivieren, die Timeout-Überwachung deaktivieren und die angeschlossenen CANopen Module in den Konfigurationsmodus bzw. in den Betriebsmodus schalten.

Register 2080: Modulnummer	
Funktion	**Beschreibung**
Lesen	aktuelle Modulnummer
Schreiben	neue Modulnummer
Wertebereich	70 - 79
Wert nach Reset	70

Register 2075: Index auf Info-Tabelle	
Funktion	**Beschreibung**
Lesen	aktueller Modul-Index
Schreiben	neuer Modulindex
Wertebereich	0 - 255
Wert nach Reset	0

Register 2076: Info-Tabelle	
Funktion	**Beschreibung**
Lesen	aktuelle Info Reg 2075 = 1 -> Reg 2076 = Status Reg 2075 = 2 -> Reg 2076 = Pollzeit PDO (tx) usw.
Schreiben	neue Info
Wertebereich	32 Bit
Wert nach Reset	in Abhängigkeit vom Info-Index

Auf die Info-Tabelle eines CANopen Modules wird über die Register 2080, 2075 und 2076 zugegriffen. In Register 2080 wird die Modulnummer des CANopen Modules eingetragen. In Register 2075 dann der Index auf die Info-Tabelle. Über Register 2076 lassen sich dann einzelne Einträge in der Info-Tabelle lesen und schreiben.

Übersicht der Info-Tabelle:

0 keine Funktion

1 Status des CANopen Modules

Bit 0 : 1 = CANopen Modul ist gültig

Bit 1 : 1 = Mindestabstand beim Lesen der PDO (tx) / Eingangsdaten aktiv

Bit 2 : 1 = Mindestabstand beim Schreiben der PDO (tx) / Ausgangsdaten aktiv

Bit 8 : 1 = Timeout-Überwachung aktiv

2 Mindestzeitabstand beim Lesen der PDO (tx) / Eingangsdaten in ms

3 Mindestzeitabstand beim Schreiben der PDO (rx) / Ausgangsdaten in ms

4 aktive Prozessdatenkanäle für PDO (tx) / Eingangsdaten

Bit 0 : 1 = PDO1 (tx) aktiv

Bit 1 : 1 = PDO2 (tx) aktiv

Bit 2 : 1 = PDO3 (tx) aktiv

Bit 3 : 1 = PDO4 (tx) aktiv

99 NMT-Message

1 = Wechsel in Betriebsbereit „operational“

128 = Wechsel in Konfigurationsmodus „pre-operational“

Beispiel 3: Deaktivierung der Prozessdatenkommunikation

Die Steuerung erkennt bei der Inbetriebnahme des Systembusses selbständig, welche Prozessdatenkanäle vom CANopen Modul unterstützt werden. Alle aktiven Prozessdatenkanäle für Eingänge werden dabei von der Steuerung zyklisch gelesen.
Das zyklische Lesen kann über die Modul-Info Schnittstelle deaktiviert werden. Dazu muss in den vierten Eintrag der Modul-Info Tabelle eine Null geschrieben werden.

ST Programmfile

```
#include "sysbus.stp"

var
    stSysBus        : SYS_BUS   AT  %vl 2000;
end_var

TASK 0
    // …
    stSysBus.nsModulIdx     := 70;      // Modulnummer
    stSysBus.nsInfoTabIdxx  := 4;       // Index für Modulinfo
    stSysBus. nsInfoTab     := 0;       // alle PDO deaktivieren
    // …
END_TASK;
```

A Header-Dateien für JetSym ST

A.1 Import der Header-Dateien in JetSym

1. Legen Sie eine neue JetSymST-Programm-Datei mit dem Namen „SysBus“ im Include-Ordner des Programms an.
2. Markieren Sie den Inhalt des Kapitels A.2 Systembus mit der Maus, und kopieren Sie Ihn in die Zwischenablage. Anschließend wechseln Sie zurück zu JetSym und fügen den Inhalt der Zwischenablage in das Fenster der JetSymST-Programm-Datei „SysBus.stp“ ein.
3. Fügen Sie die Zeile `#include „sysbus.stp“` in Ihr Programm ein. Nun können Sie den `STRUCT SYS_BUS` in Ihrem Anwenderprogramm benutzen.

A.2 Systembus

```
TYPE
    SYS_BUS:
        STRUCT
            nDummy1                 : ARRAY[8] OF INT;
            nsError                 : INT;                  // Reg 2008
            nDummy2                 : ARRAY[2] OF INT;
            nsToutIOModule          : INT;                  // Reg 2011
            nsToutSlaveModule       : INT;                  // Reg 2012
            nsNumIOModule           : INT;                  // Reg 2013
            nsNumSlaveModule        : INT;                  // Reg 2014
            nsModuleArrayIdx        : INT;                  // Reg 2015
            nsModuleArray           : INT;                  // Reg 2016
            nDummy3                 : ARRAY[6] OF INT;
            nsDummyIOModule         : INT;                  // Reg 2023
            nsDummySlaveModule      : INT;                  // Reg 2024
            nDummy4                 : ARRAY[2] OF INT;
            nsErrOutputDriver       : INT;                  // Reg 2027
            nsMonitoringIntervalIO  : INT;                  // Reg 2028
            nsBaudrate              : INT;                  // Reg 2029
            nsDummyIOModuleEx       : INT;                  // Reg 2030
            nDummy5                 : INT;
            nsPwrUpDelay            : INT;                  // Reg 2032
            nDummy6                 : ARRAY[37] OF INT;
            nsNumJX_SIO             : INT;                  // Reg 2070
            nsNumIO                 : INT;                  // Reg 2071
            nsVersion               : INT;                  // Reg 2072
            nsJXSIOTout             : INT;                  // Reg 2073
            nsSYNC                  : INT;                  // Reg 2074
            nsInfoTabIdx            : INT;                  // Reg 2075
            nsInfoTab               : INT;                  // Reg 2076
            nDummy7                 : ARRAY[3] OF INT;
```

```
            nsModulIdx              : INT;                  // Reg 2080
            nDummy8                 : ARRAY[4] OF INT;
            nsUserRegIdx            : INT;                  // Reg 2085
            nsUserRegObj            : INT;                  // Reg 2086
            nsUserRegSubIdx         : INT;                  // Reg 2087
            nsUserRegLen            : INT;                  // Reg 2088
            nDummy9                 : ARRAY[671] OF INT;
            nsIOModuleToutCfg       : INT;                  // Reg 2760
            nsIOModuleToutArrayIdx  : INT;                  // Reg 2761
            nsIOModuleToutArray     : INT;                  // Reg 2762
            nsIOModuleTout          : INT;                  // Reg 2763
            nsIOModulRegTout        : INT;                  // Reg 2764
            nsSlaveModulRegTout     : INT;                  // Reg 2765
        END_STRUCT;
END_TYPE;
```

Jetter AG

Gräterstrasse 2
D-71642 Ludwigsburg

Deutschland

Telefon: +49 7141 2550-530
Telefon Vertrieb: +49 7141 2550-433
Fax: +49 7141 2550-484
Fax Vertrieb: +49 7141 2550-484
Hotline: +49 7141 2550-444
Internet: http://www.jetter.de
E-Mail: sales@jetter.de

Tochtergesellschaften

Jetter Asia Pte. Ltd.

32 Ang Mo Kio Industrial Park 2
#07-03 Sing Industrial Complex
Singapore 569510

Singapore

Telefon: +65 6483 8200
Fax: +65 6483 3881
E-Mail: sales@jetter.com.sg

Jetter AG Schweiz

Münchwilerstrasse 19
CH-9554 Tägerschen

Schweiz

Telefon: +41 719 1879-50
Fax: +41 719 1879-69
E-Mail: info@jetterag.ch

Jetter Automation Inc.

165 Ken Mar Industrial Parkway
Broadview Heights
OH 44147-2950

U.S.A

Telefon: +1 440 8380860
Fax: +1 440 8380861
E-Mail: bernd@jetterus.com