APN 048
JetSym-STX-Funktionen



Application Note

Jetter

Application Note: 048
Artikelnummer: 60879791
Version 1.00
August 2014 / Printed in Germany

Dieses Dokument hat die Jetter AG mit der gebotenen Sorgfalt und basierend auf dem ihr bekannten Stand der Technik erstellt.

Bei Änderungen, Weiterentwicklungen oder Erweiterungen bereits zur Verfügung gestellter Produkte wird ein überarbeitetes Dokument nur beigefügt, sofern dies gesetzlich vorgeschrieben oder von der Jetter AG für sinnvoll erachtet wird. Die Jetter AG übernimmt keine Haftung und Verantwortung für inhaltliche oder formale Fehler, fehlende Aktualisierungen sowie daraus eventuell entstehende Schäden oder Nachteile.

Die im Dokument aufgeführten Logos, Bezeichnungen und Produktnamen sind geschützte Marken der Jetter AG, der mit ihr verbundenen Unternehmen oder anderer Inhaber und dürfen nicht ohne Einwilligung des jeweiligen Inhabers verwendet werden.

Inhaltsverzeichnis

1 JetSym-STX-Funktionen 5

1.1 Formatkonvertierung ... 6
BCD nach binär ... 7
Binär nach BCD ... 8
1.2 Sortieren von Daten ... 9
Funktionsbeschreibung ... 10
Programmbeispiel ... 12
1.3 Modbus RTU CRC-Prüfsumme ... 14
Berechnung der CRC ... 15
Prüfung der CRC ... 17
1.4 Modbus/TCP-Client mit STX-Variablen ... 18
ModbusReadReg(): Azyklisch Register lesen ... 20
ModbusWriteReg(): Azyklisch Register schreiben ... 22
Programmbeispiel: Azyklische Kommunikation zum Modbus/TCP-Server ... 24
RemoteScanConfig(): RemoteScan konfigurieren ... 26
RemoteScanStart(): RemoteScan starten ... 30
RemoteScanStop(): RemoteScan stoppen ... 31
Programmbeispiel: Zyklische Kommunikation zum Modbus/TCP-Server ... 32
1.5 Datendateien ... 34
Datendatei schreiben ... 35
1.6 E-Mail versenden ... 37
E-Mail versenden ... 38
1.7 NetCopyList ... 40
Programmierung der Funktionalität NetCopyList ... 41
Eine Liste konfigurieren ... 43
Eine Liste senden ... 45
Eine Liste löschen ... 46
Eine Liste konfigurieren, senden und löschen ... 47
Programmbeispiel NetCopyList ... 49

1 JetSym-STX-Funktionen

Einleitung

Diese Application Note beschreibt spezielle STX-Funktionen, die in ihrer Funktionalität den Systemfunktionen entsprechen. Während `Systemfunction` nur mit Registern arbeiten, arbeiten die hier beschriebenen STX-Funktionen nur mit STX-Variablen, lokalisierte und nicht lokalisierte.

Inhalt

Thema	Seite
Formatkonvertierung	6
Sortieren von Daten	9
Modbus RTU CRC-Prüfsumme	14
Modbus/TCP-Client mit STX-Variablen	18
Datendateien	34
E-Mail versenden	37
NetCopyList	40

1.1 Formatkonvertierung

Einleitung

Dieses Kapitel beschreibt die STX-Funktionen für die Formatkonvertierung von BCD nach binär und umgekehrt.

Inhalt

Thema	Seite
BCD nach binär	7
Binär nach BCD	8

BCD nach binär

Einleitung

Diese Funktion können Sie z. B. benutzen, mehrstellige BCD-Schalter in die interne Zahlendarstellung (binär) umzuwandeln. Der BCD-Schalter ist dazu an digitalen Eingängen angeschlossen.

Funktionsdeklaration

```
Function Bcd2Hex(Bcd: Int): Int;
```

Parameter	Beschreibung
Bcd	Wert in BCD-Kodierung

Rückgabewert

Der Rückgabewert ist der Zahlenwert in interner Darstellung (binär).

JetSym-STX-Programm

- Am ersten JX3-DI16-Modul an einer JetControl 340 ist ein 4-stelliger BCD-Schalter angeschlossen.
- Die Einerstelle an Eingang 1 bis 4, die Zehnerstelle an Eingang 5 bis 8, usw.
- Der Zugriff auf den Schalter erfolgt über das den Eingängen überlagerte Register 100004062.
- Die Einstellung am Schalter ist "5432"

```
Var
    rInputs:    Int At %VL 100004062;
    nBCDValue: Int;
    nValue:     Int;
End_Var;

Task tMain Autorun
    ...

    nBCDValue := rInputs;
    nValue    := Bcd2Hex(nBCDValue);

    ...
End_Task;
```

Variable	Dezimalwert	Hexadezimalwert
nBCDValue	21554	0x5432
nValue	5432	0x1538

Binär nach BCD

Einleitung

Diese Funktion können Sie z. B. benutzen, einen Zahlenwert für die Ausgabe auf einer BCD-Anzeige umzuwandeln. Die BCD-Anzeige ist dazu an digitalen Ausgängen angeschlossen.

Funktionsdeklaration

```
Function Hex2Bcd(Hex: Int): Int;
```

Parameter	Beschreibung
Hex	Wert in interner Darstellung

Rückgabewert

Der Rückgabewert ist der Zahlenwert in BCD-Kodierung. Da eine Zahl vom Typ "Int" maximal acht BCD-Ziffern enthalten kann, sind bei einem Zahlenüberlauf die höherwertigen Halbbytes abgeschnitten.

JetSym-STX-Programm

- Am ersten JX3-DO16-Modul an einer JetControl 340 ist eine 4-stellige BCD-Anzeige angeschlossen.
- Die Einerstelle an Ausgang 1 bis 4, die Zehnerstelle an Ausgang 5 bis 8, usw.
- Der Zugriff auf den Schalter erfolgt über das den Ausgängen überlagerte Register 100004262.
- Der auszugebende Wert ist 5432.

```
Var
    rOutputs:    Int At %VL 100004262;
    nBCDValue:    Int;
    nValue:      Int;
End_Var;

Task tMain Autorun
    ...

    nBCDValue := Hex2Bcd(nValue);
    rOutputs  := nBCDValue;

    ...
End_Task;
```

Variable	Dezimalwert	Hexadezimalwert
nValue	5432	0x1538
nBCDValue	21554	0x5432

1.2 Sortieren von Daten

Einleitung

Dieses Kapitel beschreibt die STX-Funktion, um Daten in Variablen zu sortieren.

Inhalt

Thema	Seite
Funktionsbeschreibung	10
Programmbeispiel	12

Funktionsbeschreibung

Einleitung

Um für den Anwender eine größtmögliche Flexibilität zu eröffnen, sind Sortieralgorithmus und Daten voneinander getrennt. Der Sortieralgorithmus ist im Betriebssystem der Steuerung abgelegt. Der Zeiger auf die zu sortierenden Daten und die Sortierparameter werden beim Funktionsaufruf mitgegeben.

Funktionsdeklaration

```
Function QSort(DataPtr: Int, ElementCnt: Int, ElementSize: Int,
SortOffset: Int, SortType: STXBASETYPE, SortMode: QSORTMODE): Int;
```

Parameter	Beschreibung
DataPtr	Startadresse des zu sortierenden Variablenbereichs
ElementCnt	Anzahl der zu sortierenden Variablen
ElementSize	Größe einer der zu sortierenden Variablen
SortOffset	Adressoffset innerhalb einer Variablen, die den Sortierwert enthält
SortType	Typ der Variablen an "SortOffset"
SortMode	Sortiermodus

Typdeklaration

```
Type
    STXBASETYPE: Enum (
        TYPECHAR    = 0,
        TYPEBYTE    = 1,
        TYPEINT8    = 2,
        TYPESMALL   = 3,
        TYPEWORD    = 4,
        TYPEUSINT   = 5,
        TYPEINT16   = 6,
        TYPESINT    = 7,
        TYPESHORT   = 8,
        TYPEINT32   = 9,
        TYPELONG    = 10,
        TYPEINT     = 11,
        TYPEFLOAT   = 12,
        TYPEREAL    = 13,
        TYPEDOUBLE  = 14,
        TYPELREAL   = 15
    );

    QSORTMODE: Enum (
        ASCENDING   = 0,
        DESCENDING  = 1
    );
End_Type;
```

Funktionsparameter

Parameter	Wert	Bemerkung
DataPtr	Gültige Variablenadresse	Direkt angeben
ElementCnt	2 ... 1000	Anzahl der zu sortierenden Variablen
ElementSize	1 ... 1000	Größe einer der zu sortierenden Variablen in Byte
SortOffset	0 ... ("ElementSize" - 1)	Adressoffset innerhalb einer Variablen, die den Sortierwert enthält
SortType	0 ... 15	Typ der Variablen an "SortOffset"
SortMode	0 ... 1	Sortiermodus "aufsteigend" oder "absteigend"

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
> 0	Ausführungszeit [µs]
-3	"ElementCnt" kleiner 2 oder größer 1.000
-4	"ElementSize" kleiner 1 oder größer 1.000
-5	"SortMode" ungültig
-6	"SortOffset" größer "ElementSize" - 1
-7	"SortType" größer 15
-8	"DataPtr" ungültig

Programmbeispiel

Aufgabe

Die in einem Array abgelegten Datensätze sollen anhand des zweiten Datensatzelements aufsteigend sortiert werden.

Lösung

Das Sortieren mit der STX-Funktion QSort() durchführen.

JetSym-STX-Programm

```
#define SORTSIZE    4

Type
    SORTTYPE:    Struct
                    wData1:    Word;
                    nData1:    Int;
                    rData1:    Real;
                End_Struct;
End_Type;

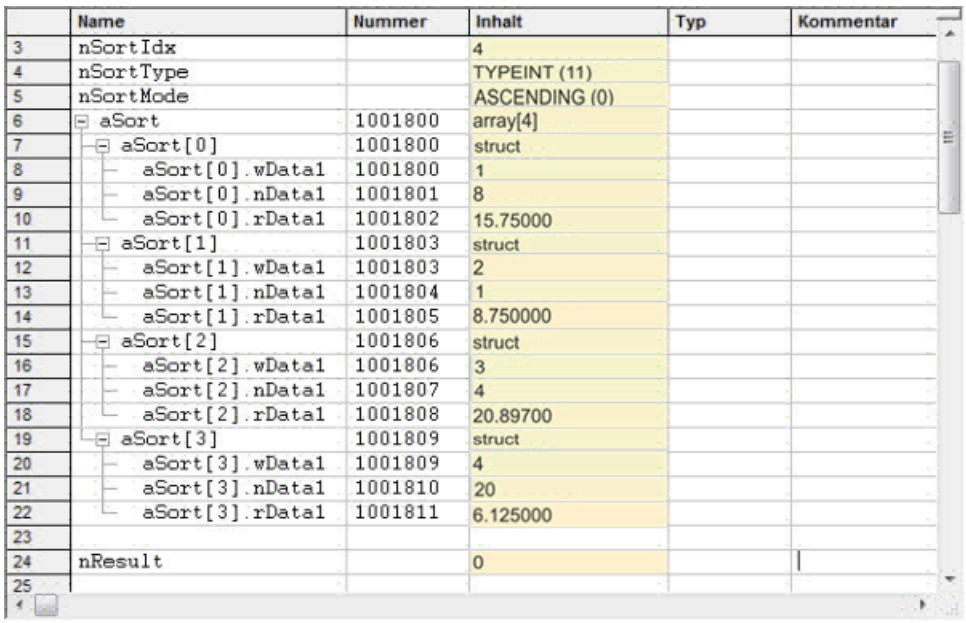
Var
    aSort:          Array[SORTSIZE] Of SORTTYPE;
    nSortIdx:      Int;
    nSortType:     STXBASETYPE;
    nSortMode:     QSORTMODE;
    nResult:        Int;
End_Var;

Task tMain Autorun
    ...
    // Offset des 2. Datensatzelements ermitteln
    nSortIdx  := &aSort[0].nData1 - &aSort;
    // Variablentyp festlegen
    nSortType := STXBASETYPE.TYPEINT;
    // Sortiermodus festlegen
    nSortMode := QSORTMODE.ASCENDING;
    // Datensätze sortieren
    nResult := QSort(&aSort, SORTSIZE, sizeof(SORTTYPE),
                     nSortIdx, nSortType, nSortMode);

    ...
End_Task;
```

Datensätze vor der Sortierung

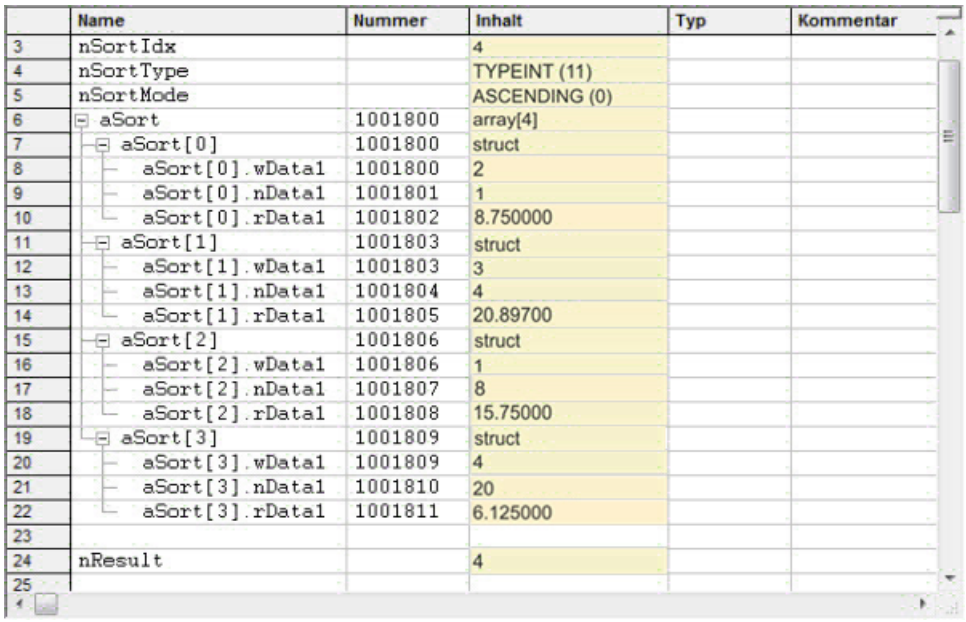
In diesem Setup-Fenster von JetSym sehen Sie die Datensätze vor der Sortierung:



	Name	Nummer	Inhalt	Typ	Kommentar
3	nSortIdx		4		
4	nSortType		TYPEINT (11)		
5	nSortMode		ASCENDING (0)		
6	aSort	1001800	array[4]		
7	aSort[0]	1001800	struct		
8	aSort[0].wData1	1001800	1		
9	aSort[0].nData1	1001801	8		
10	aSort[0].rData1	1001802	15.75000		
11	aSort[1]	1001803	struct		
12	aSort[1].wData1	1001803	2		
13	aSort[1].nData1	1001804	1		
14	aSort[1].rData1	1001805	8.750000		
15	aSort[2]	1001806	struct		
16	aSort[2].wData1	1001806	3		
17	aSort[2].nData1	1001807	4		
18	aSort[2].rData1	1001808	20.89700		
19	aSort[3]	1001809	struct		
20	aSort[3].wData1	1001809	4		
21	aSort[3].nData1	1001810	20		
22	aSort[3].rData1	1001811	6.125000		
23					
24	nResult		0		
25					

Datensätze nach der Sortierung

In diesem Setup-Fenster von JetSym sehen Sie die Datensätze nach der Sortierung:



	Name	Nummer	Inhalt	Typ	Kommentar
3	nSortIdx		4		
4	nSortType		TYPEINT (11)		
5	nSortMode		ASCENDING (0)		
6	aSort	1001800	array[4]		
7	aSort[0]	1001800	struct		
8	aSort[0].wData1	1001800	2		
9	aSort[0].nData1	1001801	1		
10	aSort[0].rData1	1001802	8.750000		
11	aSort[1]	1001803	struct		
12	aSort[1].wData1	1001803	3		
13	aSort[1].nData1	1001804	4		
14	aSort[1].rData1	1001805	20.89700		
15	aSort[2]	1001806	struct		
16	aSort[2].wData1	1001806	1		
17	aSort[2].nData1	1001807	8		
18	aSort[2].rData1	1001808	15.75000		
19	aSort[3]	1001809	struct		
20	aSort[3].wData1	1001809	4		
21	aSort[3].nData1	1001810	20		
22	aSort[3].rData1	1001811	6.125000		
23					
24	nResult		4		
25					

1.3 Modbus RTU CRC-Prüfsumme

Einleitung

Dieses Kapitel beschreibt die STX-Funktionen zur Berechnung und Prüfung der Modbus RTU CRC-Prüfsumme.

Inhalt

Thema	Seite
Berechnung der CRC	15
Prüfung der CRC	17

Berechnung der CRC

Einleitung

Bei der Modbus-RTU-Kommunikation muss die Datenübertragung durch eine 2-Byte-CRC-Prüfsumme gesichert werden. Diese Funktion berechnet die CRC und fügt sie hinten an den zu sendenden Frame an.

Funktionsdeklaration

```
Function ModbusCRCgen(FramePtr: Int, Length: Int): Int;
```

Parameter	Beschreibung
FramePtr	Adresse des zu sendenden Frames
Length	Anzahl Zeichen im Frame

Wichtiger Hinweis

Die Variable, die den Frame enthält, muss um mindestens 2 Bytes größer sein als der Wert in Parameter "Length". Die Funktion fügt die beiden Bytes für die CRC hinten an den Frame an.

Rückgabewert

Der Rückgabewert ist die CRC-Prüfsumme, die auch an den Frame angefügt wird.

JetSym-STX-Programm

Der Frame-Puffer enthält die Daten des Modbus-RTU-Telegramms mit einer Länge von 6 Bytes. Die Funktion fügt die beiden CRC-Bytes an, so dass der gesamte Frame mit einer Länge von 8 Bytes anschließend z. B. über die serielle Schnittstelle versandt werden kann.

```
Var
    nResult:     Int;
    abyFrame:    Array[20] Of Byte;
    nLen:         Int;
End_Var;

Task tMain Autorun
    ...

    nLen          := 6;
    abyFrame[0] := 0x02;
    abyFrame[1] := 0x03;
    abyFrame[2] := 0x00;
    abyFrame[3] := 0x20;
    abyFrame[4] := 0x00;
    abyFrame[5] := 0x04;
    nResult := ModbusCRCgen(&abyFrame, nLen);

   ...
End_Task;
```

	Name	Nummer	Inhalt	Typ	Kommentar
12	nResult		0x000045f0		
13	nLen		6		
14	abyFrame		array[20]		
15	abyFrame[0]		0x02		
16	abyFrame[1]		0x03		
17	abyFrame[2]		0x00		
18	abyFrame[3]		0x20		
19	abyFrame[4]		0x00		
20	abyFrame[5]		0x04		
21	abyFrame[6]		0x45		CRC high byte
22	abyFrame[7]		0xf0		CRC low byte

Prüfung der CRC

Einleitung

Bei der Modbus-RTU-Kommunikation muss die Datenübertragung durch eine 2-Byte-CRC-Prüfsumme gesichert werden. Diese Funktion prüft die CRC in den letzten beiden Bytes des Frames.

Funktionsdeklaration

```
Function ModbusCRCcheck(FramePtr: Int, Length: Int): Int;
```

Parameter	Beschreibung
`FramePtr`	Adresse des zu sendenden Frames
`Length`	Anzahl Zeichen im Frame einschließlich CRC

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	CRC nicht korrekt
1	CRC korrekt

1.4 Modbus/TCP-Client mit STX-Variablen

Einleitung

Der Modbus/TCP-Client in der Steuerung unterstützt nur Class 0 Conformance.

Bei dieser Klasse werden die Befehle zum Lesen und Schreiben von mehreren Registern genutzt. Ein Telegramm überträgt bis zu 125 Register mit 16-Bit-Breite.

Als Protocol-ID wird eine "0" verwendet. Die Zuordnung der gesendeten und empfangenen Telegramme geschieht über die Transaction-ID.

Dieses Kapitel beschreibt die Durchführung einer azyklischen oder zyklischen Übertragung zu einem Modbus/TCP-Server mit Hilfe von STX-Funktionen.

Anzahl möglicher Verbindungen

Gleichzeitig können Verbindungen zu elf unterschiedlichen Modbus/TCP-Servern geöffnet sein.

Azyklische Datenübertragung

Zur Herstellung eines azyklischen Übertragungskanals zu einem Modbus/TCP-Server werden die Funktionen `ModbusReadReg()` und `ModbusWriteReg()` genutzt.

Die Funktionen kopieren Daten zwischen Registern eines Modbus/TCP-Servers und STX-Variablen. Sie öffnen eine Verbindung zu dem angegeben Modbus/TCP-Server, übertragen die gewünschten Daten und schließen die Verbindung wieder.

Wenn eine durch RemoteScan aufgebaute Verbindung, zyklische Datenübertragung, besteht, dann wird die Verbindung benutzt und der Verbindungsaufbau und -abbau entfällt.

Zyklische Datenübertragung

Die zyklische Datenübertragung geschieht über die konfigurierbare Funktion `RemoteScanConfig()`. Zyklisch werden die Daten von STX-Variablen von und zu den Modbus/TCP-Servern übertragen.

Zu jedem Modbus/TCP-Server (IP-Adresse und Port) wird eine Verbindung aufgebaut. Der Aufbau geschieht unabhängig davon, wie viele Kommunikationseinheiten auf diesem Server konfiguriert sind.

Wenn mehrere Kommunikationseinheiten auf einem Modbus/TCP-Server konfiguriert sind, werden die Zugriffe serialisiert. Das ist so, weil die Server oftmals kein "command pipeling" unterstützen. Wenn mehrere Server konfiguriert sind, wird parallel mit ihnen kommuniziert.

Unit-ID

Bei Umsetzern von Modbus/TCP auf Modbus RTU wird die *Unit-ID* dazu verwendet, die Modbus-RTU-Server zu adressieren. Deshalb ist die Unit-ID einstellbar.

Inhalt

Dieses Kapitel enthält die folgenden Themen:

Thema	Seite
ModbusReadReg(): Azyklisch Register lesen	20
ModbusWriteReg(): Azyklisch Register schreiben	22
Programmbeispiel: Azyklische Kommunikation zum Modbus/TCP-Server	24
RemoteScanConfig(): RemoteScan konfigurieren	26
RemoteScanStart(): RemoteScan starten	30
RemoteScanStop(): RemoteScan stoppen	31
Programmbeispiel: Zyklische Kommunikation zum Modbus/TCP-Server	32

ModbusReadReg(): Azyklisch Register lesen

Einleitung

Die Funktion ModbusReadReg() kopiert einen Registerblock von einem Modbus/TCP-Server in die Variablen des lokalen Speichers.

Wichtige Hinweise

- Während diese STX-Funktion ausgeführt wird, werden parallele Aufrufe dieser Funktion in anderen Tasks blockiert.
- Während diese STX-Funktion ausgeführt wird, sollte kein TaskBreak oder TaskRestart auf dieser Task oder ein Programmneustart mit JetSym durchgeführt werden. Sonst bleibt die Verbindung in diesem Fall geöffnet und es finden unter Umständen keine weiteren Übertragungen statt.
- Die IP-Adresse ist immer direkt anzugeben. Die Angabe über Namen wird nicht unterstützt.

Funktionsdeklaration

```
Function ModbusReadReg(Const Ref MbParam: MODBUS_PARAM): Int;
```

Parameter	Beschreibung
MbParam	Struktur vom Typ MODBUS_PARAM

Typdeklaration

```
Type
    MODBUS_TYPES: Enum (
                    MBTYPE_WORD          = 0,
                    MBTYPE_LONGLITTLE    = 1,
                    MBTYPE_LONGBIG       = 2,
                    MBTYPE_FLOATLITTLE   = 3,
                    MBTYPE_FLOATBIG          = 4
                  );
    MODBUS_PARAM: Struct
                    IpAddr:              Int;
                    IpPort:              Int;
                    UnitID:              Int;
                    RemoteReg:           Int;
                    RemoteRegCnt:        Int;
                    LocalVar:            Int;
                    LocalType:           MODBUS_TYPES;
                    Timeout:             Int;
                    Res:                 Int;
                  End_Struct;
End_Type;
```

Funktionsparameter

Parameter	Wert	Bemerkung
IpAddr	IP-Adresse des Modbus/TCP-Servers	Direkt angeben
IpPort	502	
UnitID	0 ... 255	Unit-ID
RemoteReg	Startregisternummer	Registernummer Quelle
RemoteRegCnt	1 ... 125	Anzahl Register
LocalVar	Lokale Speicheradresse	Adresse Ziel
LocalType	MODBUS_TYPES	Typ der lokalen Variablen
Timeout	In ms	Timeout
Res	0	Interne Verwendung

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-1 oder -2	Fehler beim Verbindungsaufbau
-4	Fehler bei Datenübertragung
-5	Fehlermeldung vom Modbus/TCP-Server
-8	Zeitüberschreitung
-10	Modbus/TCP ist nicht lizenziert

ModbusWriteReg(): Azyklisch Register schreiben

Einleitung

Die Funktion ModbusWriteReg() kopiert Variablenwerte des lokalen Speichers in einen Registerblock von einem Modbus/TCP-Server.

Wichtige Hinweise

- Während diese STX-Funktion ausgeführt wird, werden parallele Aufrufe dieser Funktion in anderen Tasks blockiert.
- Während diese STX-Funktion ausgeführt wird, sollte kein TaskBreak oder TaskRestart auf dieser Task oder ein Programmneustart mit JetSym durchgeführt werden. Sonst bleibt die Verbindung in diesem Fall geöffnet und es finden unter Umständen keine weiteren Übertragungen statt.
- Die IP-Adresse ist immer direkt anzugeben. Die Angabe über Namen wird nicht unterstützt.

Funktionsdeklaration

```
Function ModbusWriteReg(Const Ref MbParam: MODBUS_PARAM): Int;
```

Parameter	Beschreibung
MbParam	Struktur vom Typ MODBUS_PARAM

Typdeklaration

```
Type
    MODBUS_TYPES: Enum (
                    MBTYPE_WORD          = 0,
                    MBTYPE_LONGLITTLE    = 1,
                    MBTYPE_LONGBIG       = 2,
                    MBTYPE_FLOATLITTLE   = 3,
                    MBTYPE_FLOATBIG      = 4
                  );
    MODBUS_PARAM: Struct
                    IpAddr:              Int;
                    IpPort:              Int;
                    UnitID:              Int;
                    RemoteReg:           Int;
                    RemoteRegCnt:        Int;
                    LocalVar:            Int;
                    LocalType:           MODBUS_TYPES;
                    Timeout:             Int;
                    Res:                 Int;
                  End_Struct;
End_Type;
```

Funktionsparameter

Parameter	Wert	Bemerkung
IpAddr	IP-Adresse des Modbus/TCP-Servers	Direkt angeben
IpPort	502	
UnitID	0 ... 255	Unit-ID
RemoteReg	Startregisternummer	Registernummer Ziel
RemoteRegCnt	1 ... 125	Anzahl Register
LocalVar	Lokale Speicheradresse	Adresse Quelle
LocalType	MODBUS_TYPES	Typ der lokalen Variablen
Timeout	In ms	Timeout
Res	0	Interne Verwendung

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-1 oder -2	Fehler beim Verbindungsaufbau
-4	Fehler bei Datenübertragung
-5	Fehlermeldung vom Modbus/TCP-Server
-8	Zeitüberschreitung
-10	Modbus/TCP ist nicht lizenziert

Programmbeispiel: Azyklische Kommunikation zum Modbus/TCP-Server

Aufgabe

Auf externe Anforderung soll der Wert eines einzelnen Registers an den Kommunikationspartner verschickt werden.

Lösung

Der Wert eines einzelnen Registers wird mit Hilfe der STX-Funktionen `ModbusWriteReg()` und `ModbusReadReg()` azyklisch an den Kommunikationspartner verschickt.

Das Setzen des Merkers "bWrite" löst einmalig das Schreiben eines Registerblocks an den Modbus/TCP-Server aus.

Das Setzen des Merkers "bRead" löst einmalig das Lesen eines Registerblocks, den der Modbus/TCP-Server gesendet hat, aus.

JetSym-STX-Programm

```
Var
    bWrite, bRead         : Bool;
    MyData2Send           : Array[0..4] Of Word;
    MyData2Receive        : Array[0..4] Of Word;
    MyResult              : Int;

    MyModbusTCPWrite      : MODBUS_PARAM;
    MyModbusTCPRead       : MODBUS_PARAM;
End_Var;

Task tModBusTCPazyklisch Autorun

Loop
If bwrite Then;
    bWrite := False;
    // Parameter belegen
    // IP-Adresse des Servers
    MyModbusTCPWrite.IpAddr       := IP#192.168.10.17;
    // Portnummer (502)
    MyModbusTCPWrite.IpPort       := 502;
    // UnitID
    MyModbusTCPWrite.UnitID       := 0x01;
    // Startregisternummer im Server
    MyModbusTCPWrite.RemoteReg    := 5;
    // Anzahl 16-Bit-Register
    MyModbusTCPWrite.RemoteRegCnt := 5;
    // Zeiger auf lokalen Speicher
    MyModbusTCPWrite.LocalVar     := &MyData2Send;
    MyModbusTCPWrite.LocalType    := MODBUS_TYPES.MBTYPE_WORD;
    // Timeout in ms
    MyModbusTCPWrite.Timeout      := 200;
    // Reserve
    // MyModbusTCPWrite.Res       := 0;

    MyResult := ModbusWriteReg(MyModbusTCPWrite);
End_If;
```

```
If bRead Then;
    bRead := False;
    // Parameter belegen
    // IP-Adresse des Servers
    MyModbusTCPRead.IpAddr       := IP#192.168.10.17;
    // Portnummer (502)
    MyModbusTCPRead.IpPort       := 502;
    // UnitID
    MyModbusTCPRead.UnitID       := 0x01;
    // Startregisternummer im Server
    MyModbusTCPRead.RemoteReg    := 5;
    // Anzahl 16-Bit-Register
    MyModbusTCPRead.RemoteRegCnt := 5;
    // Zeiger auf lokalen Speicher
    MyModbusTCPRead.LocalVar     := &MyData2Receive;
    MyModbusTCPRead.LocalType    := MODBUS_TYPES.MBTYPE_WORD;
    // Timeout in ms
    MyModbusTCPRead.Timeout      := 200;
    // Reserve
    // MyModbusTCPRead.Res       := 0;

    MyResult := ModbusReadReg(MyModbusTCPRead);
End_If;
End_Loop;

End_Task;
```

Verwandte Themen

- **STX-Funktion ModbusWriteReg(): Azyklisch Register schreiben** (siehe Seite 22)
- **STX-Funktion ModbusReadReg(): Azyklisch Register lesen** (siehe Seite 20)

RemoteScanConfig(): RemoteScan konfigurieren

Einleitung

Die Funktion RemoteScanConfig() konfiguriert den RemoteScan. Die Kommunikation (zyklische Datenübertragung) beginnt aber noch nicht.

Die Funktion RemoteScanStart() startet und die Funktion RemoteScanStop() stoppt die Kommunikation.

Wichtige Hinweise

- Die IP-Adresse ist immer direkt anzugeben. Die Angabe über Namen wird nicht unterstützt. Die IP-Adresse ist jeweils der erste Eintrag einer Kommunikationseinheit.
- Bei Modulen ohne Eingangs- oder Ausgangsregister, ist die entsprechende Anzahl auf 0 zu setzen.
- Wenn sowohl Eingänge als auch Ausgänge konfiguriert sind, werden zunächst die Ausgänge gesendet und danach die Eingänge gelesen.

Einschränkung

Sie können nur eine beschränkte Anzahl an Kommunikationseinheiten bei RemoteScan anlegen:

Parameter	Beschreibung
Anzahl Kommunikationsein-heiten	1 ... 10

Funktionsdeklaration

```
Function RemoteScanConfig(Protocol: RSCAN_PROTOCOL, Elements: Int,
Const Ref Configuration: RSCAN_DSCR): Int;
```

Parameter	Beschreibung
`Protocol`	Protokollkennung (Modbus/TCP = 5)
`Elements`	Anzahl der Kommunikationseinheiten in der Struktur, die durch den dritten Parameter referenziert wird (1 ... 10)
`Configuration`	Datenstruktur zur Konfiguration

Typdeklaration

```
Type
    RSCAN_STATBIT:  Enum (
                        RUNNING         = 0,
                        Error           = 1
                    );

    RSCAN_PROTOCOL: Enum (
                        MODBUSTCP       = 5
                    );
```

```
RSCAN_ERRTYPE:  Enum (
                    MBTIMEOUT       = -8,
                    MBTRANSACTION   = -7,
                    MBRECEIVE       = -6,
                    MBEXCEPTION     = -5,
                    MBREADINPUTS    = -4,
                    MBWRITEOUTPUTS  = -3,
                    MBCONNECT       = -2,
                    MBNETDRIVER     = -1,
                    NOERROR         = 0,
                    TIMEOUT         = 101,
                    LOCALACCESS     = 102,
                    COMMUNICATION1  = 103,
                    COMMUNICATION2  = 104
                );

RSCAN_STATUS:   Struct
                    Status:         RSCAN_STATBIT;
                    ErrType:        RSCAN_ERRTYPE;
                    ErrCnt:         Int;
                End_Struct;

RSCAN_DSCR:     Struct
                    IpAddr:         Int;
                    IpPort:         Int;
                    UnitID:         Int;
                    Update:         Int;
                    OutRegCnt:      Int;
                    OutRemoteReg:   Int;
                    OutLocalVar:    Int;
                    OutLocalType:   Int;
                    InRegCnt:       Int;
                    InRemoteReg:    Int;
                    InLocalVar:     Int;
                    InLocalType:    Int;
                    Timeout:        Int;
                    Status:         Pointer To RSCAN_STATUS;
                    Res:            Int;
                End_Struct;
End_Type;
```

Aufbau der Struktur zur Konfiguration

Der dritte Parameter der Funktion RemoteScanConfig() ist die Adresse der Beschreibungsstruktur von der ersten Kommunikationseinheit. Die Beschreibungsstrukturen der weiteren Kommunikationseinheiten (bis zur Anzahl im zweiten Parameter) müssen im Speicher nahtlos hintereinander abgelegt sein. Das Anlegen eines entsprechenden Arrays vom Typ RCAN_DSCR bietet sich an.

Parameter	Wert	Bemerkung
Kommunikationseinheit 1		
IpAddr	IP-Adresse des Modbus/TCP-Servers	Direkt angeben
IpPort	502	Modbus/TCP: 502
UnitID	0 ... 255	
Update	10 ... 65.535 ms	
OutRegCnt	0 ... 125	Anzahl Ausgangsregister
OutRemoteReg	0 ... 65.535	Erste Ausgangsregister-nummer
OutLocalVar	Lokale Adresse	Erste Ausgangsvariable
OutLocalType	MODBUS_TYPES	Typ der lokalen Variablen
InRegCnt	0 ... 125	Anzahl Eingangsregister
InRemoteReg	0 ... 65.535	Erste Eingangsregister-nummer
InLocalVar	Lokale Adresse	Erste Eingangsvariable
InLocalType	MODBUS_TYPES	Typ der lokalen Variablen
Timeout	In ms	
Status	Lokale Adresse	Status bei laufender Kom-munikation
Res	0	Interne Verwendung
Kommunikationseinheit 2		
IpAddr	IP-Adresse des Modbus/TCP-Servers	Direkt angeben
...	...	...

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
> 0	Anzahl der initialisierten Kommunikationseinheiten
-1	Nicht unterstütztes Protokoll
-2	Eingestellte Anzahl Kommunikationseinheiten > 10
-3	Ungültige lokale Adresse
-4	Remote-Registerwert ist größer als 65.535.
-5	RemoteScan läuft bereits
-10	Modbus/TCP ist nicht verfügbar (R 202930 Bit 5 = 0)

Aufbau Statusregisterblock RSCAN_STATUS

Bei laufendem RemoteScan werden im Statusregisterblock die Fehlermeldungen dieser Kommunikationseinheit abgelegt.

Parameter	Wert	Bemerkung
Status (bitkodiert)	Bit 0 = 1	RemoteScan läuft; das Bit wird nach jedem Updatezyklus gesetzt
	Bit 1 = 1	Fehler aufgetreten; das Bit wird nach jedem Fehler gesetzt
Error		Die Kennung des letzten aufgetretenen Fehlers ist immer lesbar
	0	Kein Fehler
	< 0	Applikationsspezifischer Fehler
		Bei Modbus/TCP:
	-1	Fehler im Netzwerktreiber
	-2	Fehler in der Verbindungsverwaltung
	-3	Fehler beim Senden der Ausgangsregisterwerte
	-4	Fehler beim Lesen der Eingangsregisterwerte
	-5	Exception-Rückmeldung
	-6	Fehler beim Empfang der Rückmeldung
	-7	Falsche Transaction ID
	-8	Timeout
	101	Timeout
	102	Fehler beim Lesen/Schreiben der lokalen Variablen
	103/104	Fehler in der unterlagerten Kommunikationsschicht
ErrCnt		Fehleranzahl; wird nach jedem Fehler inkrementiert

RemoteScanStart(): RemoteScan starten

Einleitung

Die Funktion RemoteScanStart() startet den RemoteScan.

Zuvor wird der RemoteScan mit der Funktion RemoteScanConfig() konfiguriert.

Funktionsdeklaration

```
Function RemoteScanStart(Protocol: Int): Int;
```

Parameter	Beschreibung
`Protocol`	Protokollkennung RSCAN_PROTOCOL (Modbus/TCP = 5)

Rückgabewert

Die Funktion RemoteScanStart() liefert stets den Rückgabewert 0 zurück.

RemoteScanStop(): RemoteScan stoppen

Einleitung

Die Funktion RemoteScanStop() stoppt einen laufenden RemoteScan und schließt alle geöffneten Kommunikationsverbindungen.

Funktionsdeklaration

```
Function RemoteScanStop(Protocol: Int): Int;
```

Parameter	Beschreibung
`Protocol`	Protokollkennung RSCAN_PROTOCOL (Modbus/TCP = 5)

Rückgabewert

Die Funktion RemoteScanStop() liefert stets den Rückgabewert 0 zurück.

Programmbeispiel: Zyklische Kommunikation zum Modbus/TCP-Server

Aufgabe

Zwischen Modbus/TCP-Client und Modbus/TCP-Server sollen zyklisch I/O-Daten ausgetauscht werden.

Lösung

Für die zyklische Datenübertragung wird die Funktion RemoteScan eingesetzt. Zunächst wird die Funktion RemoteScan durch Aufruf der Funktion `RemoteScanConfig()` konfiguriert. Das geschieht, indem die benötigten Strukturen mit den Konfigurationsdaten belegt werden. Die Anfangsadresse dieser Strukturen wird dann beim Aufruf der Funktion `RemoteScanConfig()` mitgegeben. Bei erfolgreicher Initialisierung wird über die Funktion `RemoteScanStart()` die Funktion RemoteScan gestartet und mit der zyklischen Kommunikation begonnen. Bei Bedarf wird die zyklische Kommunikation durch Aufruf der Funktion `RemoteScanStop()` gestoppt.

JetSym-STX-Programm

```
Var
    bStart, bStop           : Bool;
    MyData2Send             : Array[0..9] Of Word;
    MyData2Receive          : Array[0..9] Of Word;
    MyResult                : Int;
    MyStatus                : RSCAN_STATUS;

    MyModbusTCPConnection1  : RSCAN_DSCR;
End_Var;

Task tModBusTCPzyklisch Autorun

// Parameter belegen
// IP-Adresse des Servers
MyModbusTCPConnection1.IpAddr         := IP#192.168.10.17;
// Portnummer (502)
MyModbusTCPConnection1.IpPort         := 502;
// UnitID
MyModbusTCPConnection1.UnitID         := 0x01;
// Update-Rate in ms
MyModbusTCPConnection1.Update         := 1000;
// Anzahl Register
MyModbusTCPConnection1.OutRegCnt      := 5;
// Startregisternummer
MyModbusTCPConnection1.OutRemoteReg   := 0;
// Zeiger auf lokalen Speicher
MyModbusTCPConnection1.OutLocalVar    := &MyData2Send;
MyModbusTCPConnection1.OutLocalType   :=
MODBUS_TYPES.MBTYPE_WORD;
// Anzahl Register
MyModbusTCPConnection1.InRegCnt       := 5;
// Startregisternummer
MyModbusTCPConnection1.InRemoteReg    := 100;
// Zeiger auf lokalen Speicher
MyModbusTCPConnection1.InLocalVar     := &MyData2Receive;
```

```
MyModbusTCPConnection1.InLocalType     :=
MODBUS_TYPES.MBTYPE_WORD;
// Timeout in ms
MyModbusTCPConnection1.Timeout         := 20;
// Status
MyModbusTCPConnection1.Status          := &MyStatus;
// Reserve
// MyModbusTCPConnection1.Res          := 0;

Loop
When bStart Continue;
bstart := False;

// Konfigurieren
MyResult := RemoteScanConfig(RSCAN_PROTOCOL.MODBUSTCP, 1,
MyModbusTCPConnection1);

// Starten
If MyResult > 0 Then
    RemoteScanStart(RSCAN_PROTOCOL.MODBUSTCP);

    // Stoppen
    When bStop Continue;
    bStop := False;
    RemoteScanStop(RSCAN_PROTOCOL.MODBUSTCP);
End_If;
End_Loop;

End_Task;
```

Verwandte Themen

- **STX-Funktion RemoteScanConfig(): RemoteScan konfigurieren** (siehe Seite 26)
- **STX-Funktion RemoteScanStart(): RemoteScan starten** (siehe Seite 30)
- **STX-Funktion RemoteScanStop(): RemoteScan stoppen** (siehe Seite 31)

1.5 Datendateien

Einleitung

Mit Hilfe zweier Funktionen ist es möglich aktuelle Registerwerte und Merkerzustände, gesteuert durch das Anwendungsprogramm, in eine Datei zu schreiben oder Werte aus einer vorhandenen Datei zu lesen.

Verweise

- Das Dateiformat wird auch von JetSym und Nicht-STX-Steuerungen unterstützt.
- Die so erstellten Dateien können z. B. von JetSym oder mit der Funktion `FileDARead()` gelesen werden.
- Das Dateiformat ist in den Betriebsanleitungen der Steuerungen beschrieben.

Inhalt

Thema	Seite
Datendatei schreiben	35

Datendatei schreiben

Einleitung

Sie können diese Funktion verwenden, um eine Datendatei (Dateierweiterung .da, nicht .stxda) zu erstellen oder Werte an eine bestehende Datei anzuhängen. Dabei ermöglicht das, Register und Merker zu sichern.

Funktionsdeklaration

```
Function FileDAWrite(Const Ref FileName: String, Const Ref Mode:
String, VarType: DAWRITE_TYPE, First: Int, Last: Int): Int;
```

Parameter	Beschreibung
FileName	Name der Datendatei
Mode	Datei erstellen oder Werte anfügen
VarType	Register oder Merker
First	Nummer des ersten Registers/Merkers
Last	Nummer des letzten Registers/Merkers

Modus

Mit dem Modus wird angegeben, ob die Datei für einen Schreibzugriff oder einen Lesezugriff geöffnet wird.

- Der Modus "fWrite" öffnet eine bestehende Datei oder legt eine neu an. Der Inhalt einer bestehenden Datei wird dabei gelöscht.
- Soll an den Inhalt einer bestehenden Datei weiterer Inhalt angefügt werden, muss als Modus "fAppend" angegeben werden.
- Für einen Lesezugriff muss der Modus "fRead" angegeben werden.

fWrite, fAppend und fRead sind vordefinierte Makros, die als folgende Konstanten definiert sind:

```
Const
    fAppend = 'a';
    fWrite  = 'w';
    fRead   = 'r';
End_Const;
```

Typdeklaration

```
Type
    DAWRITE_TYPE: Enum (
        DAREGSITER    = 1,
        DAFLAG         = 3,
    );
End_Type;
```

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-1	Fehler bei der Öffnung/Anlegung der Datei
-2	Fehler beim Schreibvorgang der Datei
-6	Register-/Merkernummer außerhalb des gültigen Bereichs
-7/-13	Dateiname zu lang
-10	Datendateifunktion nicht verfügbar
-11/-12	Interner Betriebssystemfehler

JetSym-STX-Programm

Die Zustände der Merker 10 bis 25 und die Werte der Register 1000100 bis 1000199 sollen in einer Datendatei abgespeichert werden.

```
Var
    nResult:    Int;
End_Var;

Task tMain Autorun
    ...

    nResult := FileDAWrite('MyDataFile.da', fWrite,
                           DAWRITE_TYPE.DAREGISTER,
                            1000100, 1000199);
    nResult := FileDAWrite('MyDataFile.da', fAppend,
                           DAWRITE_TYPE.DAFLAG,
                           10, 25);

    ...
End_Task;
```

1.6 E-Mail versenden

Einleitung

Dieses Kapitel beschreibt die STX-Funktion, die es ermöglicht, zuvor erstellte E-Mails im Anwendungsprogramm zu versenden.

Beim Versenden aus dem Anwendungsprogramm stellt das Gerät aus einer E-Mail-Vorlagendatei eine E-Mail zusammen und fügt ggf. Variablenwerte ein.

Verarbeitung im Anwendungsprogramm

Der E-Mail-Versand kann relativ lange dauern. Deshalb werden, während diese Funktion ausgeführt wird, die anderen Tasks des Anwendungsprogramms bearbeitet. Nur ein E-Mail-Funktionsaufruf ist gleichzeitig möglich. Während ein E-Mail-Versand eines Tasks gerade läuft, werden daher die Tasks, die die E-Mail-Funktion aufrufen, solange blockiert, bis der E-Mail-Versand abgeschlossen ist.

Inhalt

Thema	Seite
E-Mail versenden	38

E-Mail versenden

Einleitung

Der Anwender erstellt Vorlagendateien für E-Mails, in die die Steuerung beim Versenden ggf. Variablenwerte einfügt. Die Steuerung sendet die E-Mails an einen E-Mail-Server, der die Weiterleitung übernimmt.

Voraussetzungen

Die E-Mail-Funktion in der Steuerung ist verfügbar und erfolgreich konfiguriert.

Wichtige Hinweise

- Während diese STX-Funktion ausgeführt wird, werden parallele Aufrufe dieser Funktion in anderen Tasks blockiert.
- Während diese STX-Funktion ausgeführt wird, darf kein TaskBreak oder TaskRestart auf diesen Task oder ein Programmneustart mit JetSym ausgeführt werden.

Verweis

Eine detaillierte Beschreibung der E-Mail Funktionalität finden Sie in den Betriebsanleitungen der jeweiligen Steuerung.

Funktionsdeklaration

```
Function EmailSend(Const Ref FileName: String): Int;
```

Parameter	Beschreibung
`FileName`	Dateiname der E-Mail-Vorlagendatei

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-1	Interner Betriebssystemfehler
-2	FROM nicht definiert
-3	TO nicht definiert
-4	Keine Verbindung zum E-Mail-Server
-10	E-Mail-Funktion nicht verfügbar
-11/-12	Interner Betriebssystemfehler

JetSym-STX-Programm

Wenn ein Fehler in der Anlage auftritt, soll die Steuerung eine E-Mail verschicken.

```
Var
    nResult:    Int;
    bSendError: Bool;
End_Var;

Task tMain Autorun
    ...

    If bSendError Then
        nResult := EmailSend('/ErrorMails/MainError.email');
        bSendError := FALSE;
    End_If;

    ...
End_Task;
```

1.7 NetCopyList

Einleitung

Die Funktionen NetCopyList können mehrere Schreib-/Leseanfragen von Registern auf einer anderen Steuerung in einem Ethernet-Telegramm zusammenfassen. Das führt in folgenden Fällen zu einem deutlichen Performance-Gewinn:

- Wenn mehrere Registerwerte übertragen werden sollen, die verteilt im Speicher liegen
- Wenn mit einem Zugriff Register gelesen und geschrieben werden sollen.

Mindestvoraussetzungen

Für die Verwendung der Funktionen NetCopyList ist JetSym ab der Version 4.1.3 erforderlich.

Funktionsweise

- In Kommunikationseinheiten legen Sie fest:
  - Welche Registerbereiche werden übertragen?
  - Welche Register werden gelesen?
  - Welche Register werden geschrieben?
- Eine oder mehrere Kommunikationseinheiten werden in einer Liste zusammengefasst.
- Mehrere Listen unterschiedlicher Länge sind konfigurierbar.
- Beim Versand einer Liste werden alle Kommunikationseinheiten dieser Liste in einem Ethernet-Telegramm verschickt.

Technische Daten

Eigenschaft	Wert
Maximale Anzahl Listen	10
Gesamtzahl der Kommunikationseinheiten	500
Maximale Anzahl Register in einer Liste	64

Inhalt

Thema	Seite
Programmierung der Funktionalität NetCopyList	41
Eine Liste konfigurieren	43
Eine Liste senden	45
Eine Liste löschen	46
Eine Liste konfigurieren, senden und löschen	47
Programmbeispiel NetCopyList	49

Programmierung der Funktionalität NetCopyList

Einleitung

Bei der Programmierung der Funktionalität NetCopyList werden Funktionen verwendet, die im Sprachumfang der Steuerung enthalten sind. Führen Sie zur Programmierung folgende Schritte aus:

Schritt	Vorgehen
1	Funktionalität NetCopyList initialisieren (einmalig bei Programmstart)
2	Liste(n) konfigurieren
3	Daten übertragen durch Senden der Liste(n)
4	Eventuell Liste(n) löschen

Funktionsübersicht

Folgende Funktionen sind verfügbar:

Nummer	Funktion	Bedeutung
1	NetCopyListInit	Initialisierung, die mindestens einmal bei jedem Anwendungsprogrammstart durchgeführt werden muss
2	NetCopyListConfig	Konfigurieren einer Liste
3	NetCopyListSend	Senden einer Liste
4	NetCopyListDelete	Löschen einer Liste
5	NetCopyList	Bearbeitung der Funktionen 2 bis 4 in einem Aufruf

Vordefinierte Datentypen

Der folgende Datentyp für eine Kommunikationseinheit ist vordefiniert. Um die Parameterliste für die Funktion 2 und 5 anzulegen, kann dieser Datentyp im Anwendungsprogramm verwendet werden:

```
Type
    TNetCopyListElement : Struct
        nCommand     : Int;
        nMode        : Int;
        nCount       : Int;
        nLocalReg    : Int;
        nRemoteReg   : Int;
    End_Struct;

    TNetCopyListL    : Array Of TNetCopyListElement;
End_Type;
```

Kommunikationseinheit

In einer Kommunikationseinheit ist jeweils ein Netzwerkbefehl definiert. Mehrere Kommunikationseinheiten ergeben eine Liste von Netzwerkbefehlen. Die Kommunikationseinheiten einer Liste müssen hintereinander im Speicher abgelegt sein. Die Befehlsparameter haben folgenden Bedeutung:

TNetCopyListElement	
nCommand	
0 =	Ende der Liste
1 =	kopiere Werte aus fernen Registern in lokale Register
2 =	kopiere Werte aus lokalen Registern in ferne Register
3 =	kopiere Werte aus fernen Registern in lokale Variablen
4 =	kopiere Werte aus lokalen Variablen in ferne Register
nMode	
0 =	Zieladresse wird nicht verändert
1 =	Autoinkrement der Zieladresse
2 =	Autodekrement der Zieladresse
nCount	
1 ... 64	Anzahl zu kopierender Register
nLocalReg	
nCommand = 1 oder 2	Nummer des ersten lokalen Registers
nCommand = 3 oder 4	Adresse der ersten lokalen Variablen
nRemoteReg	
	Nummer des ersten entfernten Registers

Eine Liste konfigurieren

Einleitung

Bevor eine Liste verwendet werden kann, muss sie konfiguriert werden. Diese Funktion konfiguriert die Liste.

Funktionsdeklaration

```
Function NetCopyListConfig(IPAddr : Int,
                           IPPort : Int,
                  Const Ref List : TNetCopyListL) : Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
IPAddr	Gültige IP-Adresse	Adresse des Teilnehmers, an den die Liste geschickt wird
IPPort	Gültiger IP-Port	
List	Variablenadresse	Beginn der Liste mit den Kommunikationseinheiten

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
> 0	Handle zu dieser Liste; der Wert muss gespeichert werden, da er beim Senden und Löschen der Liste als Parameter benötigt wird
-1	Alle Listen sind benutzt; keine freie Liste mehr gefunden
-2	Alle Kommunikationseinheiten sind benutzt; keine freie Kommunikationseinheit mehr gefunden
-3	Leere Liste
-4	Liste ungültig
-5	Ungültige IP-Adresse
-6	Ungültiges Kommando
-8	Ungültiger Modus
-9	Anzahl Register zu groß
-10	Der angeforderte Sendepuffer ist zu groß
-11	Der angeforderte Empfangspuffer ist zu groß

Verwenden der Funktion

So können Sie die Funktion aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
hList := NetCopyListConfig(IP#192.168.25.123, 50000,
                           NetCopyListParam);
```

Funktionsweise

Die Steuerung arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung	
1	Die Steuerung legt in ihrem internen Speicher eine neue Liste an.	
2	Die Steuerung liest die Kommunikationseinheiten aus der übergebenen Variablen. Sie fügt die Befehle in die Liste ein, bis sie eine Kommunikationseinheit findet, bei der `nCommand = 0` oder ein Fehler aufgetreten ist.	
3	**Wenn ...**	**... dann ...**
	... kein Fehler aufgetreten ist,	... wird die Funktion beendet und ein positiver Wert als Handle zum weiteren Zugriff auf die Liste zurückgeliefert.
	... ein Fehler aufgetreten ist,	... wird die Funktion beendet und ein negativer Wert zurückgeliefert.

Eine Liste senden

Einleitung

Diese Funktion sendet eine zuvor konfigurierte Liste.

Wichtiger Hinweis

Während diese STX-Funktion ausgeführt wird, darf kein `TaskBreak` oder `TaskRestart` auf die Tasks, die die Listen senden, oder ein Programmneustart mit JetSym ausgeführt werden. Das kann zu Datenverlusten beim Empfang führen.

Funktionsdeklaration

```
Function NetCopyListSend(Handle : Int) : Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
Handle	Handle zur Liste	Rückgabewert der Funktion `NetCopyListConfig()`

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-7	Ungültiges Handle
-12	Fehler bei der Datenübertragung

Verwenden der Funktion

So können Sie die Funktion aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
nResult := NetCopyListSend(hList);
```

Funktionsweise

Die Steuerung arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung
1	Die Steuerung stellt ein Ethernet-Telegramm mit den zu schreibenden Werten zusammen und schickt es an den Kommunikationsteilnehmer.
2	Der Task wartet an dem Befehl, bis eine Antwort empfangen wurde und weist die empfangenen Werte den lokalen Variablen/Registern zu.
3	Die Funktion speichert das Ergebnis in der angegebenen Variablen.

Eine Liste löschen

Einleitung

Diese Funktion löscht eine zuvor konfigurierte Liste wieder.

Funktionsdeklaration

```
Function NetCopyListDelete(Handle : Int) : Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
Handle	Handle zur Liste	Rückgabewert der Funktion `NetCopyListConfig()`

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
0	Kein Fehler
-7	Ungültiges Handle

Verwenden der Funktion

So können Sie die Funktion aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
nResult := NetCopyListDelete(hList);
```

Funktionsweise

Die Steuerung arbeitet die Funktion in folgenden Stufe ab:

Stufe	Beschreibung
1	Die Steuerung löscht die Liste aus ihrem internen Speicher.

Eine Liste konfigurieren, senden und löschen

Einleitung

Die Funktion ruft nacheinander die zuvor beschriebenen Funktionen zum Konfigurieren, Senden und Löschen einer Liste auf.

Funktionsdeklaration

```
Function NetCopyList(IPAddr : Int,
                     IPPort : Int,
              Const Ref List : TNetCopyListL) : Int;
```

Funktionsparameter

Beschreibung der Funktionsparameter:

Parameter	Wert	Bemerkung
IPAddr	Gültige IP-Adresse	Adresse des Teilnehmers, an den die Liste geschickt wird
IPPort	Gültiger IP-Port	
List	Variablenadresse	Beginn der Liste mit den Kommunikationseinheiten

Rückgabewert

Folgende Rückgabewerte sind möglich:

Rückgabewert	
> 0	Handle zu dieser Liste; der Wert muss gespeichert werden, da er beim Senden und Löschen der Liste als Parameter benötigt wird
-1	Alle Listen sind benutzt; keine freie Liste mehr gefunden
-2	Alle Kommunikationseinheiten sind benutzt; keine freie Kommunikationseinheit mehr gefunden
-3	Leere Liste
-4	Liste ungültig
-5	Ungültige IP-Adresse
-6	Ungültiges Kommando
-8	Ungültiger Modus
-9	Anzahl Register zu groß
-10	Der angeforderte Sendepuffer ist zu groß
-11	Der angeforderte Empfangspuffer ist zu groß
-12	Fehler bei der Datenübertragung

Verwenden der Funktion

So können Sie die Funktion aufrufen und den Rückgabewert einer Variablen zur weiteren Auswertung zuweisen:

```
nResult := NetCopyList(IP#192.168.25.123, 50000,
                       NetCopyListParam);
```

Funktionsweise

Die Steuerung arbeitet die Funktion in folgenden Stufen ab:

Stufe	Beschreibung
1	Die Steuerung ruft die Funktion `NetCopyListConfig()` auf.
2	Die Steuerung ruft die Funktion `NetCopyListSend()` auf.
3	Die Steuerung ruft die Funktion `NetCopyListDelete()` auf.

Verwandte Themen

- **Eine Liste konfigurieren** (siehe Seite 43)
- **Eine Liste senden** (siehe Seite 45)
- **Eine Liste löschen** (siehe Seite 46)

Programmbeispiel NetCopyList

Aufgabe

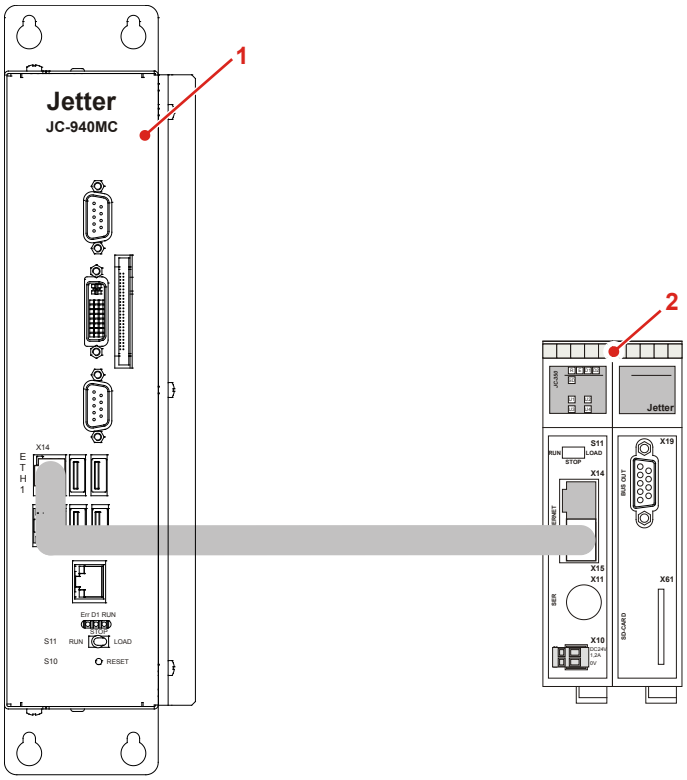
Von einem JetControl 940MC sollen Register- und Variableninhalte auf die Register eines JetControl 350 kopiert werden. Weiterhin sollen vom JC-940MC Registerinhalte des JC-350 gelesen und in lokale Register und Variablen kopiert werden. Das alles soll zyklisch erfolgen.

Lösung

Konfigurieren Sie im Anwendungsprogramm des JC-940MC eine Liste mit Hilfe der NetCopyList-Funktionen, die in einem Telegramm die Register des JC-350 liest und beschreibt.

Beispielkonfiguration

Dieses Beispiel basiert auf folgender Konfiguration:



Nummer	Teil	Beschreibung
1	JC-940MC	Steuerung
2	JC-350	Steuerung

Da die Funktionalität NetCopyList nicht plattformspezifisch implementiert ist, ist dieses Beispiel durch Änderung der Registernummern auch auf andere Konfigurationen übertragbar.

JetSym-STX-Programm

```
Const
    cNetCopyReadReg     = 1;
    cNetCopyWriteReg    = 2;
    cNetCopyReadVar     = 3;
    cNetCopyWriteVar    = 4;
End_Const;

Var
    nLocalVar          : Int;
    anLocalVar        : Array[5] Of Int;
    LocalReg           : Int At %VL 100;
    aLocalReg          : Array[3] Of Int At %VL 200;

    NetCopyListParam : Array[5] Of TNetCopyListElement;
    hList, nRes        : Int;
End_Var;

Task tNetCopyListText Autorun

    NetCopyListInit();

    // Eine Liste mit vier Elementen erstellen
    NetCopyListParam[0].nCommand   := cNetCopyReadReg;
    NetCopyListParam[0].nMode       := 1;
    NetCopyListParam[0].nCount      := 1;
    NetCopyListParam[0].nLocalReg  := &LocalReg;
    NetCopyListParam[0].nRemoteReg := 1000100;

    NetCopyListParam[1].nCommand   := cNetCopyWriteReg;
    NetCopyListParam[1].nMode       := 1;
    NetCopyListParam[1].nCount      := 3;
    NetCopyListParam[1].nLocalReg  := &aLocalReg;
    NetCopyListParam[1].nRemoteReg := 1000200;

    NetCopyListParam[2].nCommand   := cNetCopyReadVar;
    NetCopyListParam[2].nMode       := 1;
    NetCopyListParam[2].nCount      := 5;
    NetCopyListParam[2].nLocalReg  := &anLocalVar;
    NetCopyListParam[2].nRemoteReg := 1000300;

    NetCopyListParam[3].nCommand   := cNetCopyWriteVar;
    NetCopyListParam[3].nMode       := 1;
    NetCopyListParam[3].nCount      := 1;
    NetCopyListParam[3].nLocalReg  := &nLocalVar;
    NetCopyListParam[3].nRemoteReg := 1000400;

    // Ende der Parameterliste
    NetCopyListParam[4].nCommand   := 0;
```

```
    // Liste konfigurieren
    hList := NetCopyListConfig(IP#192.168.10.208,
                               50000,
                               NetCopyListParam);

    Loop
        If hList > 0 Then
            // Eine Liste senden
            nRes := NetCopyListSend(hList);
        End_If;
        Delay(T#1s);
    End_Loop;
End_Task;
```

Jetter AG

Gräterstraße 2

D-71642 Ludwigsburg

Deutschland

Telefon:	+49 7141 2550-0
Telefon Vertrieb:	+49 7141 2550-433
Fax Vertrieb:	+49 7141 2550-484
Hotline:	+49 7141 2550-444
Internet:	http://www.jetter.de
E-Mail:	sales@jetter.de

Tochtergesellschaften

Jetter (Schweiz) AG

Straussbergstrasse 11

CH-8570 Weinfelden

Schweiz

Telefon:	+41 71 91879-50
Fax:	+41 71 91879-59
E-Mail:	info@jetterag.ch
Internet:	http://www.jetterag.ch

Jetter Automation Technology (Shanghai) Co., Ltd.

NO.49, Lane 2927, Lai Yang Road,

Pudong New District

CN-200137 Shanghai

China

Telefon:	+86 215 869-1233
Fax:	-
E-Mail:	contact@jetterat.cn
Internet:	http://www.jetterat.cn

Jetter USA Inc.

13075 US Highway 19 North

Florida - 33764 Clearwater

U.S.A

Telefon:	+1 727 532-8510
Fax:	+1 727 532-8507
E-Mail:	bschulze@jetterus.com
Internet:	http://www.jetter.de