**BECKHOFF** New Automation Technology

Manual | EN

# TS6341

## TwinCAT 2 PLC Serial Communication 3964R/RK512

Supplement | Communication

2022-10-17 | Version: 1.1

# Table of contents

# 1 Foreword

## 1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

### Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

### Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

### Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

### Copyright

## 1.2 Safety instructions

### Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

### Exclusion of liability

All the components are supplied in particular hardware and software configurations appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

### Personnel qualification

This description is only intended for trained specialists in control, automation and drive engineering who are familiar with the applicable national standards.

### Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

**Serious risk of injury!**

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

**Risk of injury!**

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

**Personal injuries!**

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

*NOTE*

**Damage to the environment or devices**

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

ℹ **Tip or pointer**

This symbol indicates information that contributes to better understanding.

## 1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

# 2 Overview

The TwinCAT PLC library COMlib supplies function blocks and data structures for serial data communication. COMlib supports the serial Beckhoff KL6xxx bus terminal and the standard PC COMx interfaces.

The COMlib library is described in the document “COMlib PLC Library – Serial Data Communication”. The present document is to be seen as a supplement to that.

### 3964R communication

In addition to the basic functions of sending and receiving transparent data, COMlib3964R supports the 3964R protocol. The COMlib and COMlib3964R libraries thus replace the previous P3964lib library. Both libraries are included in the 3964R package.

The COMlib3964R communication library extends COMlib to implement an error-protected protocol for the transmission of any type of data. A checksum in combination with repetition of faulty telegrams provide the error protection.

### RK512 communication

The RK512 protocol is used to transmit data blocks such as are used in Siemens controllers. The COMlibRK512 communication library uses the 3964R protocol for this purpose and handles the telegram traffic that it requires. The user only must parameterize the RK512 function block.

The "Serial Communication RK512" communication package contains all the necessary libraries. It supports the transmission of data blocks with a length of up to 128 data words.

# 3 Installation

### Installlling COMlib3964R.LIB

To perform installation of t the COMlib3964R.LIB the following libraries are copied into the TwinCAT directory TwinCAT\PLC\LIB.

- COMlib.LIB
- ChrAsc.LIB
- ChrAsc.OBJ
- COMlib3964R.LIB

The associated test program (COMlib3964Test.pro, COMlib3964Test.wsm) should be copied to any project directory of your choice, e.g. to TwinCAT\PLC.

- https://infosys.beckhoff.com/content/1033/tcplclib3964r/Resources/12566747531/.zip

### Installing COMlibRK512.LIB

Installation of COMlibRK512.LIB is performed by copying the following library files into the TwinCAT directory TwinCAT\PLC\LIB.

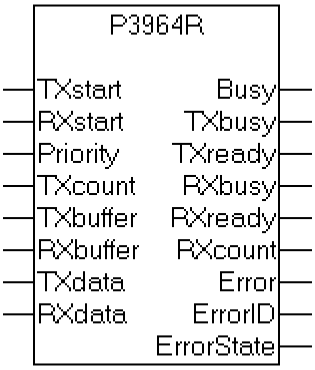
- COMlib.LIB
- ChrAsc.LIB
- ChrAsc.OBJ
- COMlib3964R.LIB
- COMlibRK512.LIB

The associated test program (COMlibRK512Test.pro, COMlibRK512Test.wsm) should be copied to any project directory of your choice, e.g. to TwinCAT\PLC.

- https://infosys.beckhoff.com/content/1033/tcplclib3964r/Resources/12566750219/.zip

# 4 COMlib3964R.LIB

## 4.1 P3964R



```
VAR_INPUT
    TXstart  : BOOL; (* Start signal, edge triggered *)
    RXstart  : BOOL; (* Start signal, edge triggered *)
    Priority : BOOL; (* 3964 priority *)
    TXcount  : INT; (* Number of data bytes in TxData *)
END_VAR
VAR_OUTPUT
    Busy        : BOOL;  (* Block active *)
    TXbusy      : BOOL; (* Sending now *)
    TXready     : BOOL; (* Sending finished *)
    RXbusy      : BOOL; (* Receiving now *)
    RXready     : BOOL; (* Receiving finished *)
    RXcount     : INT; (* Number of data bytes in TxData *)
    Error       : BOOL; (* Error *)
    ErrorID     : INT; (* Error number *)
    ErrorState : STRING(80); (* Internal error state *)
END_VAR
VAR_IN_OUT
    TXbuffer : COMbuffer; (* Intermediate buffer for hardware *)
    RXbuffer : COMbuffer; (* Intermediate buffer for hardware *)
    TXdata   : P3964buffer; (* Application data (to be sent) *)
    RXdata   : P3964buffer; (*Application data (to be received) *)
END_VAR
```

### Connection to Hardware

The P3964R function block handles the 3964R protocol. The background communication, which is handled by a separate block (PcComControl, KL6Control and KL6Control5B) decides on the hardware interface used. The associated data buffers [▶ 11], Txbuffer and Rxbuffer, are passed to the P3964R block.

### Sending

Send data are entered into the Txdata send data buffer by the PLC. The number of data bytes entered is passed on in Txcount, and then Txstart is set to TRUE. Txstart is edge-triggered, so that before a new attempt to send is made, the block must be called with Txstart=FALSE. While sending, Txbusy=TRUE. Once the data has been successfully transferred, Txready changes to TRUE.

### Receiving

RxStart is set to TRUE to receive data. After a complete data set has been received, the Rxready output goes TRUE and Rxcount data bytes are ready in the receive data buffer Rxdata. Rxstart is also edge-triggered. While receiving, Rxbusy=TRUE.

### Interrupt reception

The P3964R block can be used in either send or receive mode. It is worthwhile switching the block to receive when transmission is not taking place. As long as the block is in receive mode, but no start character has been received from the other end, a rising edge at Txstart can interrupt the reception and initiate the sending of data. After the transfer the block goes into its initial state (Busy=FALSE), and is only switched over to receive by a new edge at Rxstart.

### Priority

In a 3964R data connection, either end may send at any time. If both ends attempt to send at the same time, there is a collision. In that case, the end with the lower priority goes into receive mode, while the end with the higher priority sends again. For this reason, when priority is low the RxReady output can be TRUE, even though data are to be sent. The data in RxData may then require evaluation. The setting of the priority at the two ends is to be agreed.

### Error

In the event of an error the block goes into its initial state (Busy=FALSE), and the Error output goes TRUE. The ErrorId returns the error code. ErrorState provides information about the state in which the error occurred when the internal protocol structure is known.

### Also see about this

- P3964R error codes [▸ 11]

## 4.2 P3694buffer

Data buffers of type P3964buffer are used in association with the P3964R function block. These buffers are written and read by the user.

```
TYPE P3964buffer
  STRUCT
    D : ARRAY [0..16#0FFF] OF BYTE;
  END_STRUCT
END_TYPE
```

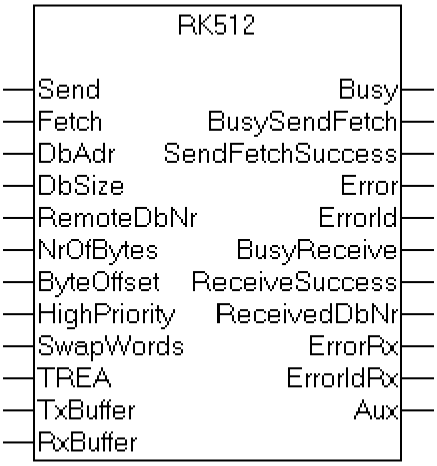
## 4.3 P3964R error codes

| No | Error constant | Description |
|---|---|---|
| 4 | P3964ERR_ZVZ | The character delay time was exceeded. CDT = 220 ms. The character delay time is the maximum time interval between the transmission of two succeeding characters. |
| 5 | P3964ERR_QVZ | The acknowledgement delay time was exceeded. The other end sent an acknowledgement character (DLE = 10H) twice as the telegram was being handled. ADT = 2 s.The first acknowledgement was expected at the start of the transmission. The output *ErrorState* when an error initially occurs is "Wait_DLE_TXstart". The cause might lie with a faulty physical connection or an incorrect interface parameter.The second acknowledgement is expected after the user data has been transmitted. If an error occurs, the *ErrorState* is "Wait_DLE_TXend". In this case the cause might, for instance, be data loss, data corruption or data bytes of a data word that have become swapped (see the *SwapWords* input). |
| 6 | P3964ERR_WVZ | The repeat delay time was exceeded. RDT = 4 s. A telegram is repeated if an error occurs. If the telegram is not repeated by the other device, the 3964R function block reports this error. |

| No | Error constant | Description |
|---|---|---|
| 7 | P3964ERR_WRONGBCC | Checksum error during data reception. Each telegram is provided with a checksum. When the data is received, the calculated checksum is compared with the checksum that has been received. |
| 9 | P3964ERR_COMERRTX | Interface error when transmitting |
| 10 | P3964ERR_COMERRRX | Interface error when receiving |
| 11 | P3964ERR_NOTXDATA | Parameterisation error. The number of bytes that are to be sent, *TxCount*, is zero. |
| 20 | P3964ERR_NAK | A telegram was refused by the other device with a negative acknowledgement (NAK). This error can occur in the following transmission states (*ErrorState*), amongst others:*Wait_DLE_TXstart*: The local PLC begins a telegram with a start character. The other device refuses the telegram with NAK.*Wait_DLE_TXend*: The other device refuses the telegram with NAK after the user data has been transmitted. In this case the cause can, for example, be a checksum error at the other device. |

# 5 ComLibRK512.LIB

## 5.1 RK512



```
VAR_INPUT
    Send         : BOOL;     (* SEND command *)
    Fetch        : BOOL;     (* FETCH command *)
    DbAdr        : POINTER TO BYTE;   (* ADR(DB), SEND=source, FETCH=target *)
    DbSize       : UINT;     (* SIZEOF(DB), size in byte for Send or Fetch *)
    RemoteDbNr   :BYTE;     (* SEND=target, FETCH=source *)
    NrOfBytes    : UINT;     (* number of bytes to send or fetch *)
    ByteOffset   : UINT;     (* byte offset in remote DB *)
    HighPriority : BOOL;   (* 3964R priority *)
    SwapWords    : BOOL;     (* swaps every two bytes in the data stream before/
after transmission *)
    TREA         : TIME;     (* Timeout for answer *)
END_VAR
VAR_OUTPUT
    Busy             : BOOL;     (* FB is busy *)
(* send or fetch (active) mode *)
    BusySendFetch    : BOOL;   (* FB is with SendFetchCommand busy *)
    SendFetchSuccess : BOOL;   (* Send or Fetch successfully finished *)
    Error          : BOOL;     (* RK512 error *)
    ErrorId        : INT;     (* RK512 error id *)
(* receive (passive) mode only *)
    BusyReceive    : BOOL;    (* FB is busy with remote request *)
    ReceiveSuccess : BOOL;   (* DB has been received from the remote PLC *)
    ReceivedDbNr   : BYTE;   (* Number of the DB that has been received *)
    ErrorRx        : BOOL;     (* RK512 error. For receive mode only *)
    ErrorIdRx      : INT;     (* RK512 error id. For receive mode only *)
    Aux            : RK512_Auxiliary_t;   (* Additional errors for debugging *)
END_VAR
VAR_IN_OUT
    TxBuffer : ComBuffer;
    RxBuffer : ComBuffer;
END_VAR
```

### Operating Modes

The RK512 function block distinguishes two quite different operating modes, passive and active operation.

In the passive mode the block waits for and answers data telegrams from the communication partner. The communication partner can transmit the data blocks to the local controller, or can fetch data from it. In order for the RK512 function block to be able to accept data blocks or to return data that has been requested, it must previously be configured for passive operation by DBconfig [▶ 16].

In active mode, the RK512 function block either sends data blocks to the communication partner (SEND) or fetches them from it (FETCH).

As long as neither of the function block's *Send* or *Fetch* inputs switch it into active operation it waits for telegrams from communication partners, changing when appropriate into passive mode. The two operating modes can be mixed. In that case the RK512 function block attempts to synchronise the telegram traffic. Since mixed mode operation can introduce delays into the flow of data, it should be avoided if possible.

One instance of the RK512 function block serves just one serial interface. It is not possible for multiple instances to handle data traffic over the same serial interface at the same time.

## Input and Output Parameters

### Send

At a rising edge at the *Send* input data from the variable whose location is given by the *DbAdr* input is sent to the communication partner where it is saved in the data block whose number is given at *RemoteDbNr*. Send and fetch cannot be executed at the same time.

### Fetch

With a rising edge at the *Fetch* input, data from the data block specified by *RemoteDbNr* is requested from the communication partner and saved in the variable specified by the *DbAdr* input. Send and fetch cannot be executed at the same time.

### DbAdr

*DbAdr* is the memory address of a PLC variable determined by the ADR function.

e.g.: DbAdr := ADR( PLCvar );

The PLC variable can be of any type. An ARRAY OF WORD, for example, or a STRUCT data structure would be suitable.

Data from this variable is transmitted to the communication partner with a send command, and is fetched and stored in this variable with a fetch command.

### DbSize

DbSize is the size of the PLC variable at *DbAdr* in bytes, determined by the SIZEOF function.

e.g.: DbSize := SIZEOF( PLCvar );

### RemoteDbNr

*RemoteDbNr* contains the number of the data block at the communication partner to which the data is to be sent or from which it is to be fetched.

### NrOfBytes

*NrOfBytes* contains the number of data bytes to be transferred. The number can be less than or equal to the variable size *DbSize*.

### ByteOffset

The *ByteOffset* indicated the data byte within the communication partner's data block from where the data is to be fetched or where it is to be stored.

### HighPriority

The priority is relevant to the 3964R protocol. The two communication partners should have different priorities. If it happens that the two partners transmit at the same time, there is a collision. This collision is resolved by the partner with the lower priority switching to receive mode, allowing the higher priority partner to transmit again.

### SwapWords

Data blocks are usually organized by words. Depending on how the data words are stored in the memory, it may be necessary to swap data bytes within the transmitted words. If *SwapWords* is TRUE, the data bytes of each word in all the data telegrams are swapped.

### TREA

Every telegram from a communication partner is acknowledged by an answer telegram. The time required for this acknowledgement is monitored with respect to the *TREA* timeout period. *TREA* might be about 15 seconds.

### Busy

*Busy* becomes TRUE as soon as the block switches into either active or passive mode, i.e. as soon as it leaves the idle state. It cannot accept a new command as long as *Busy* remains TRUE.

### BusySendFetch

*BusySendFetch* becomes TRUE as soon as a rising edge at the *Send* or *Fetch* input switches the block into active mode. Once *BusySendFetch* has become FALSE the transmission has been completed, and either the *SendFetchSuccess* or the *Error* output is set.

### SendFetchSuccess

*SendFetchSuccess* indicates that a data transfer triggered by Send or Fetch has been successfully completed.

### Error

*Error* becomes TRUE if an error occurs during a data transfer triggered by *Send* or *Fetch*.

### ErrorId

*ErrorId* provides an error number [▸ 17] if an error occurs.

### BusyReceive

*BusyReceive* indicates that the RK512 function block is in receive mode, i.e. in the passive operating mode. The block changes automatically into passive mode from the idle state (*Busy* is FALSE) as soon as a telegram is received from the communication partner. As *BusyReceive* falls, either *ReceiveSuccess* or *ErrorRx* is set.

### ReceiveSuccess

*ReceiveSuccess* is set to TRUE after a data block has been successfully received from the communication partner. This signal is only relevant to passive operation. In other words, *ReceiveSuccess* is not set to TRUE if a data block has been actively fetched from the communication partner with *Fetch*.

### ReceivedDbNr

As soon as ReceiveSuccess becomes TRUE, *ReceivedDbNr* indicates the number of the data block that has been received.

### ErrorRx

The *ErrorRx* signal indicates that an error has occurred during data reception in passive mode.

### ErrorIdRx

If an error occurs in passive mode, ErrorIdRx indicates the error number [▸ 17].

#### Aux

*Aux* is a data structure containing additional error messages for diagnostic purposes.

#### TxBuffer

The transmit data buffer *TxBuffer* is of type COMbuffer. This buffer is used by the RK512 function block, and is not changed by the user.

#### RxBuffer

The receive data buffer *RxBuffer* is of type COMbuffer. This buffer is used by the RK512 function block, and is not changed by the user.

## 5.1.1 DBconfig

The RK512 function block has two operating modes [▶ 13]. In passive mode it receives data and request telegrams from a communication partner who is addressing data blocks by means of its data block number. Numbered data blocks are not known to IEC-1131, and therefore initially also not to TwinCAT. In TwinCAT, data blocks are variables of various types such as arrays or data structures (STRUCT).

In order to define a numbered data block, the method DBconfig belonging to the RK512 function block as called with all the necessary parameters during the initialisation phase. The method is called once for each data block that is to be addressed by communication partners. This configuration is not needed for the active operating mode (send and fetch).

In the active operating mode variable contents can be sent to or fetched from communication partners, independently of this data block configuration. In active mode the data block number refers to the partner device, and only needs to be known there.

#### Parameters

#### DbAdr

*DbAdr* is the memory address of a PLC variable that is to be defined as a data block. The address is determined using the ADR function.

e.g.: DbAdr := ADR( PLCvar );

The PLC variable can be of any type. An ARRAY OF WORD, for example, or a STRUCT data structure would be suitable.

#### DbSize

DbSize is the size of the PLC variable at *DbAdr* in bytes, determined by the SIZEOF function.

e.g.: DbSize := SIZEOF( PLCvar );

#### RemoteDbNr

*RemoteDbNr* contains the number of the data block at the communication partner.

#### TxBuffer

The transmit data buffer *TxBuffer* is of type COMbuffer. This parameter is not needed by the configuration, but must however be passed as an IN_OUT parameter.

#### RxBuffer

The receive data buffer *RxBuffer* is of type COMbuffer. This parameter is not needed by the configuration but must however be passed as an IN_OUT parameter.

### Example

```
VAR
(* declare some DB
(* the type of data doen't matter but the
(* size shouldn't be larger than 128 bytes
(**)
    DB1 : ARRAY[1..64] OF WORD; (* exemplary type of db *)
    DB5 : ARRAY[1..64] OF WORD; (* exemplary type of db *)
    DB10 : ARRAY[1..64] OF WORD; (* exemplary type of db *)

(* input and ouput data for the RK512 function block *)
RK512com : RK512;

    initialized : BOOL;
END_VAR

IF NOT initialized THEN
    RK512com.DBconfig( RemoteDbNr:=5, DbAdr:=ADR(DB5), DbSize:=SIZEOF(DB5), TxBuffer:=TxBuffer, RxBu
ffer:=RxBuffer );
    RK512com.DBconfig( RemoteDbNr:=10, DbAdr:=ADR(DB10), DbSize:=SIZEOF(DB10), TxBuffer:=TxBuffer, R
xBuffer:=RxBuffer );
    initialized := TRUE;
END_IF
```

The initialization in this example has the effect that the communication partner may read or write data blocks 5 and 10. Any attempt to access another data block is considered as an error and is refused.

## 5.2 RK512 error codes

| No | No. (hex) | Error constant | Description |
|---|---|---|---|
| 4 | 16#04 | RK512ERR_P3964ERR_ZVZ | The character delay time was exceeded. ZVZ = 220 ms. The character delay time is the maximum time interval between the transmission of two succeeding characters. |
| 5 | 16#05 | RK512ERR_P3964ERR_QVZ | The acknowledgement delay time was exceeded. The other end sent an acknowledgement character (DLE = 10H) twice as the telegram was being handled. QVZ = 2 s.The first acknowledgement was expected at the start of the transmission. The output *ErrorState* when an error initially occurs is "Wait_DLE_TXstart". The cause might lie with a faulty physical connection or an incorrect interface parameter.The second acknowledgement is expected after the user data has been transmitted. If an error occurs, the *ErrorState* is "Wait_DLE_TXend". In this case the cause might, for instance, be data loss, data corruption or data bytes of a data word that have become swapped (see the *SwapWords* input). |
| 6 | 16#06 | RK512ERR_P3964ERR_WVZ | The repeat delay time was exceeded. WVZ = 4 s. A telegram is repeated if an error occurs. If the telegram is not repeated by the other device, the 3964R function block reports this error. |

| No | No. (hex) | Error constant | Description |
|---|---|---|---|
| 7 | 16#07 | RK512ERR_P3964ERR_WRONGBCC | Checksum error during data reception. Each telegram is provided with a checksum. When the data is received, the calculated checksum is compared with the checksum that has been received. |
| 9 | 16#09 | RK512ERR_P3964ERR_COMERRTX | Interface error when transmitting. |
| 10 | 16#0A | RK512ERR_P3964ERR_COMERRRX | Interface error when receiving. |
| 11 | 16#0B | RK512ERR_P3964ERR_NOTXDATA | Parameterisation error. The number of bytes that are to be sent, *TxCount*, is zero. |
| 120 | 16#78 | RK512ERR_P3964ERR_NAK | A telegram was refused by the other device with a negative acknowledgement (NAK). This error can occur in the following transmission states (*ErrorState*), amongst others:*Wait_DLE_TXstart*: The local PLC begins a telegram with a start character. The other device refuses the telegram with NAK.*Wait_DLE_TXend*: The other device refuses the telegram with NAK after the user data has been transmitted. In this case the cause can, for example, be a checksum error at the other device. |
| 12 | 16#0C | RK512ERR_INVALIDDATATYPE | Invalid data type A telegram has been received whose data type identifier is not supported. Only data blocks with the identifier 'D' can be handled. |
| 16 | 16#10 | RK512ERR_ERRORPROTOCOLHEADER | Faulty protocol header The header of a data telegram is not in accordance with the RK512 specification. |
| 20 | 16#14 | RK512ERR_DBNOTAVAILIBLE | Data block not available A data block that is not available has been transmitted or requested. Data blocks that are accessed by the partner device must first be registered for this purpose with the DBconfig method. |
| 22 | 16#16 | RK512ERR_INVALIDCOMMAND | Invalid command A telegram with an invalid command identifier (SEND / FETCH) has been received. |
| 52 | 16#34 | RK512ERR_INVALIDSIZE | Invalid length quoted Either the *DbSize* or the *NrOfBytes* parameter is invalid, or a data telegram that is too long has been received. |
| 53 | 16#35 | RK512ERR_INVALIDDBADR | Invalid data block address The data block address *DbAdr* is invalid. |
| 54 | 16#36 | RK512ERR_SYNCERROR | Synchronisation error A synchronisation error can occur if both communication partners start to transmit at the same time. |
| 257 | 16#101 | RK512ERR_TIMEOUT | Timeout on the RK512 telegram level An expected reaction telegram has not been received within the waiting time *TREA*. |

| No | No. (hex) | Error constant | Description |
|---|---|---|---|
| 258 | 16#102 | RK512ERR_ERRORREMOTEREATEL | Error in the reaction telegram An error was reported in the communication partner's reaction telegram. The error number can be read from the additional error information in *AUX*. |
| 259 | 16#103 | RK512ERR_INVALIDSIZEREMOTEREATEL | The communication partner's reaction telegram has an incorrect length. |
| 260 | 16#104 | RK512ERR_TIMEOUTREAPEATSENDFETCH | Timeout when transmitting In spite of a number of repetitions, a send or fetch telegram could not be transmitted. |
| 261 | 16#105 | RK512ERR_3964ERROR | Error on the 3964R protocol level Predictable errors on this level are not reported as a general RK512ERR_3964ERROR error, but with a detailed error number. |
| 262 | 16#106 | RK512ERR_3964NOTBUSYNOTREADY | The transmission of a telegram has been halted without having been successfully completed. |

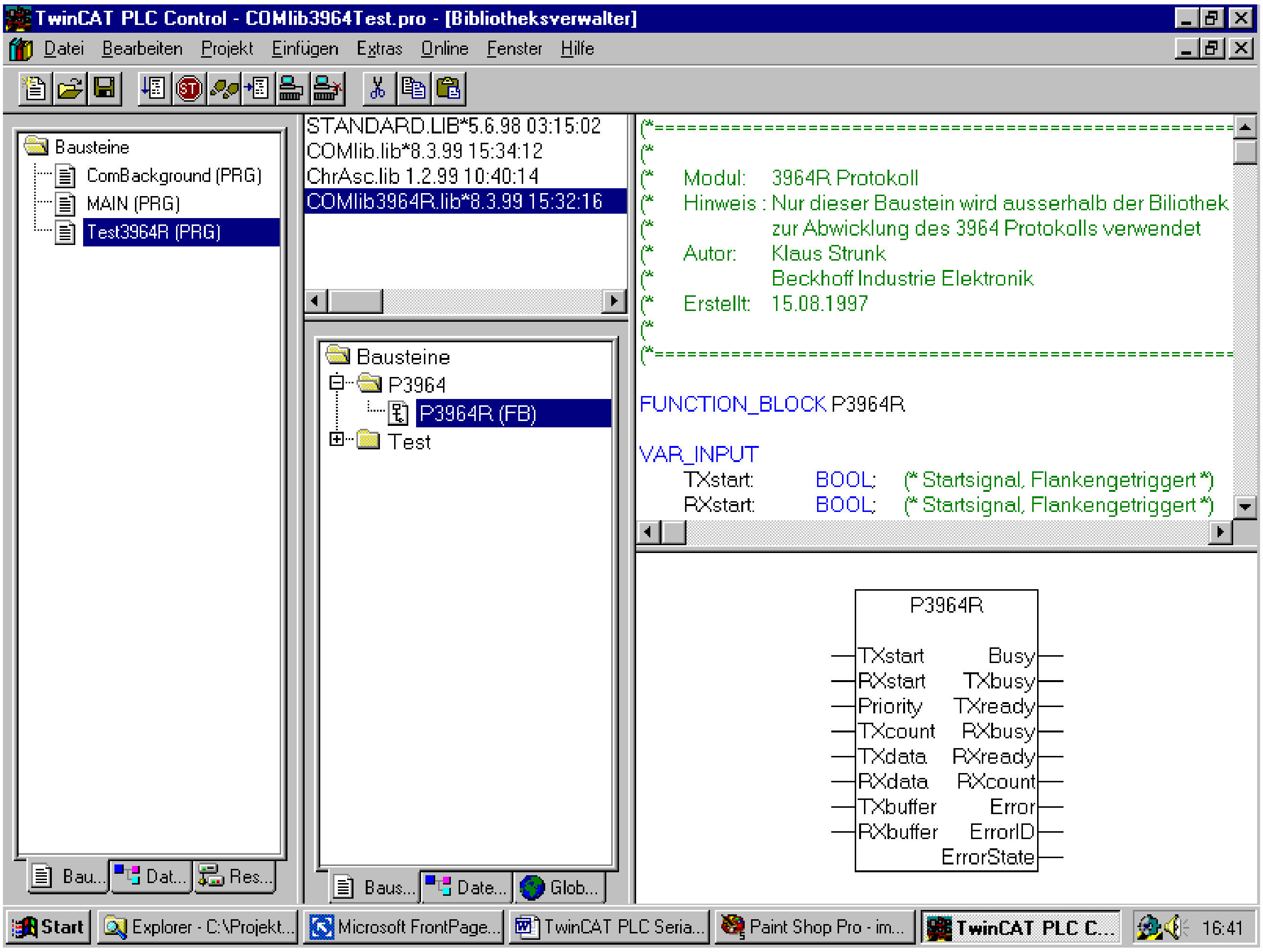
# 6 Examples

## 6.1 Link Libraries

### Linking ComLib3964R.LIB

Create a new PLC project with TwinCAT PLC Control to perform the library linking.

Go to library management and add the libraries ChrAsc.LIB, ComLib.LIB and ComLib3964R.LIB.



TwinCAT PLC Control - COMlib3964Test.pro - [Bibliotheksverwalter]

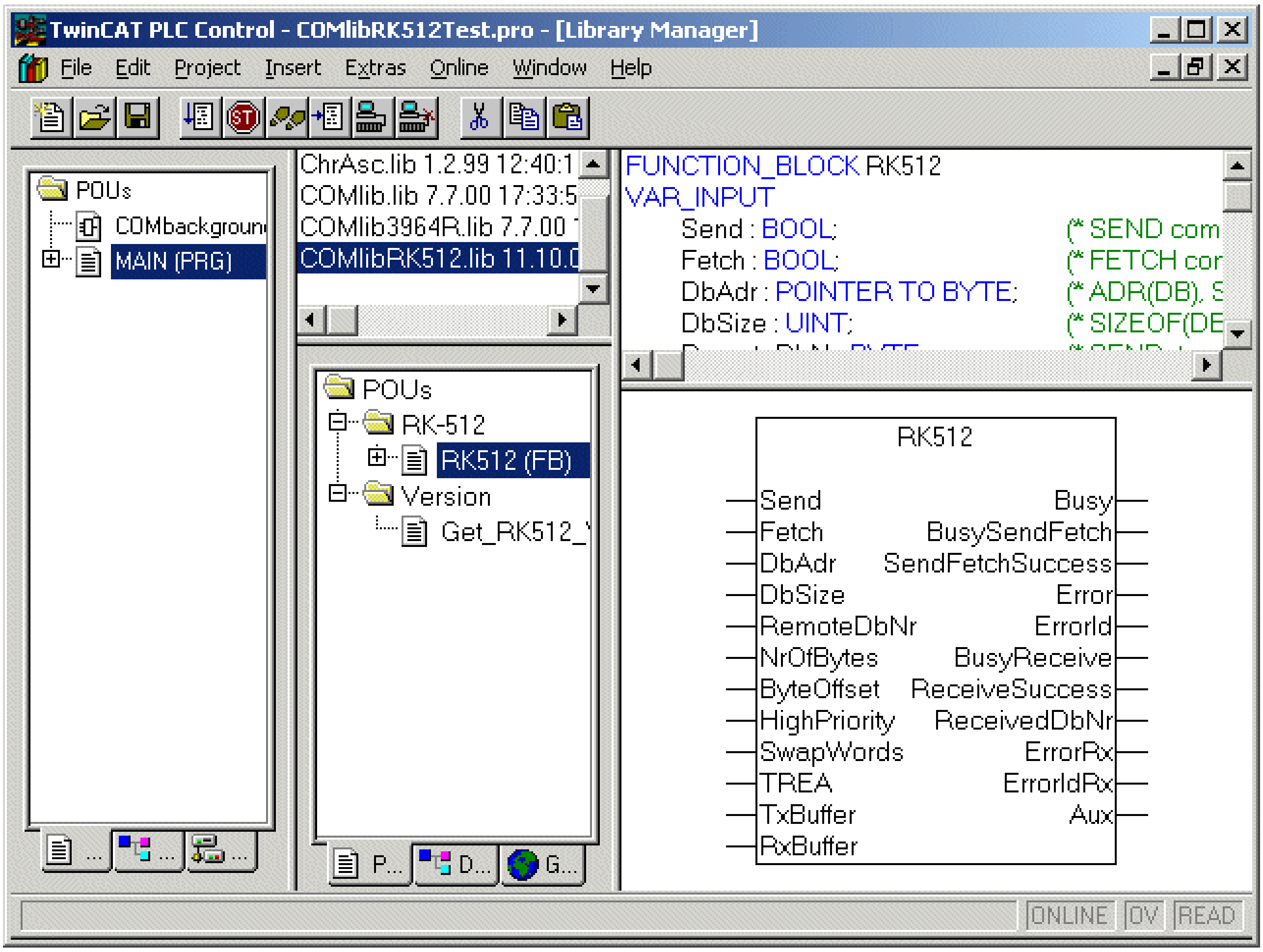
```
STANDARD.LIB*5.6.98 03:15:02
COMlib.lib*8.3.99 15:34:12
ChrAsc.lib 1.2.99 10:40:14
COMlib3964R.lib*8.3.99 15:32:16
```

```
(*=================================================
(*
(*    Modul:    3964R Protokoll
(*    Hinweis : Nur dieser Baustein wird ausserhalb der Biliothek
(*              zur Abwicklung des 3964 Protokolls verwendet
(*    Autor:    Klaus Strunk
(*              Beckhoff Industrie Elektronik
(*    Erstellt: 15.08.1997
(*
(*=================================================

FUNCTION_BLOCK P3964R

VAR_INPUT
    TXstart:     BOOL;    (* Startsignal, Flankengetriggert *)
    RXstart:     BOOL;    (* Startsignal, Flankengetriggert *)
```

### Linking ComLibRK512.LIB

Create a new PLC project with TwinCAT PLC Control to perform the library linking.

Go to Library Management and add the libraries ChrAsc.LIB, ComLib.LIB, ComLib3964R.LIB and ComLibRK512.LIB.



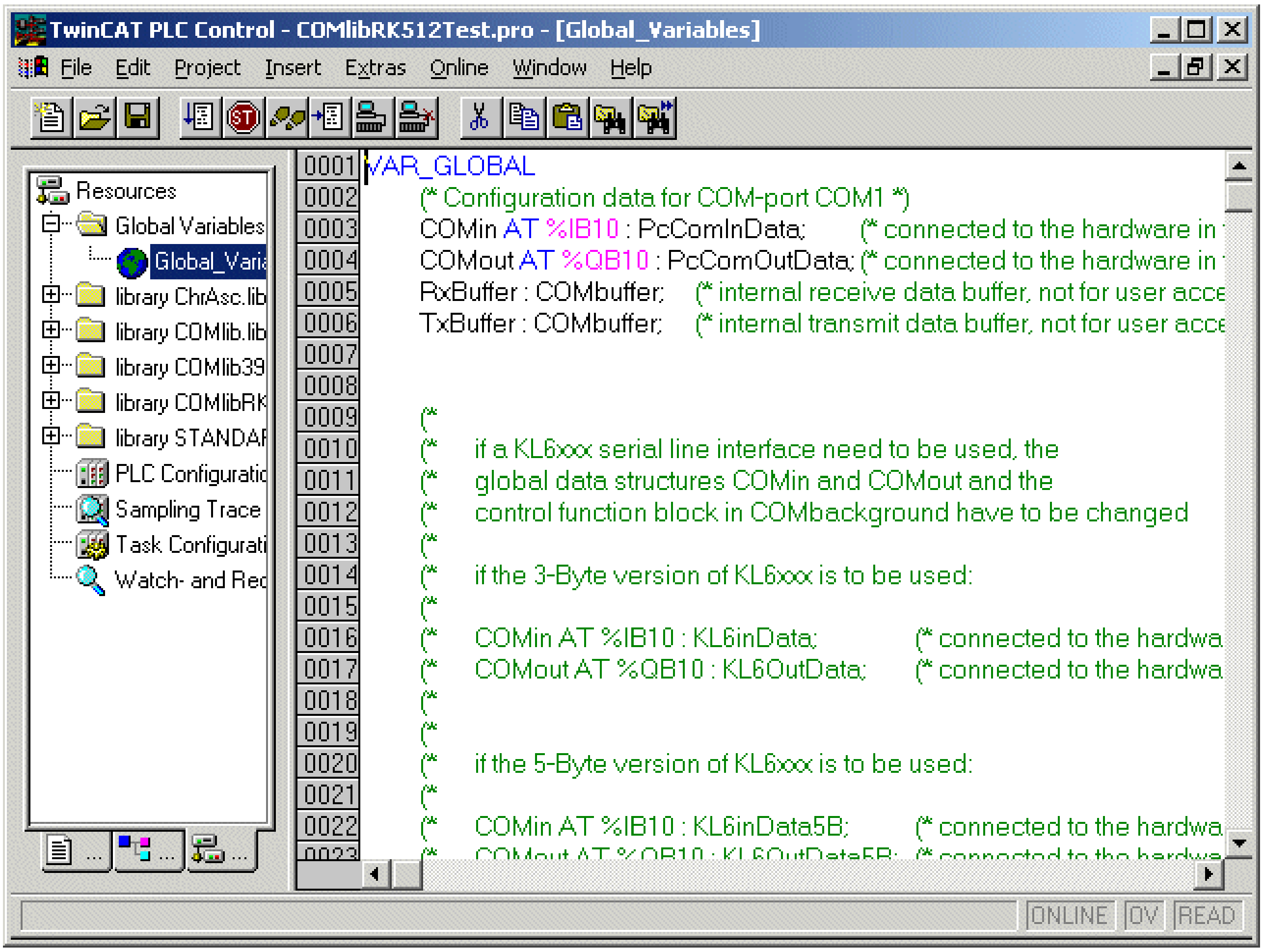
TwinCAT PLC Control - COMlibRK512Test.pro - [Library Manager]
File Edit Project Insert Extras Online Window Help
POUs
COMbackgroun
MAIN (PRG)
ChrAsc.lib 1.2.99 12:40:1
COMlib.lib 7.7.00 17:33:5
COMlib3964R.lib 7.7.00
COMlibRK512.lib 11.10.0
POUs
RK-512
RK512 (FB)
Version
Get_RK512_
P...
D...
G...
ONLINE
OV
READ

```
FUNCTION_BLOCK RK512
VAR_INPUT
    Send : BOOL;                    (* SEND com
    Fetch : BOOL;                   (* FETCH cor
    DbAdr : POINTER TO BYTE;        (* ADR(DB), S
    DbSize : UINT;                  (* SIZEOF(DE
```

## 6.2 Global Variables

Four global data structures are needed to access a serial interface. Two provide the connection to the hardware in the send and receive directions. Two data buffers are also necessary for intermediate storage.

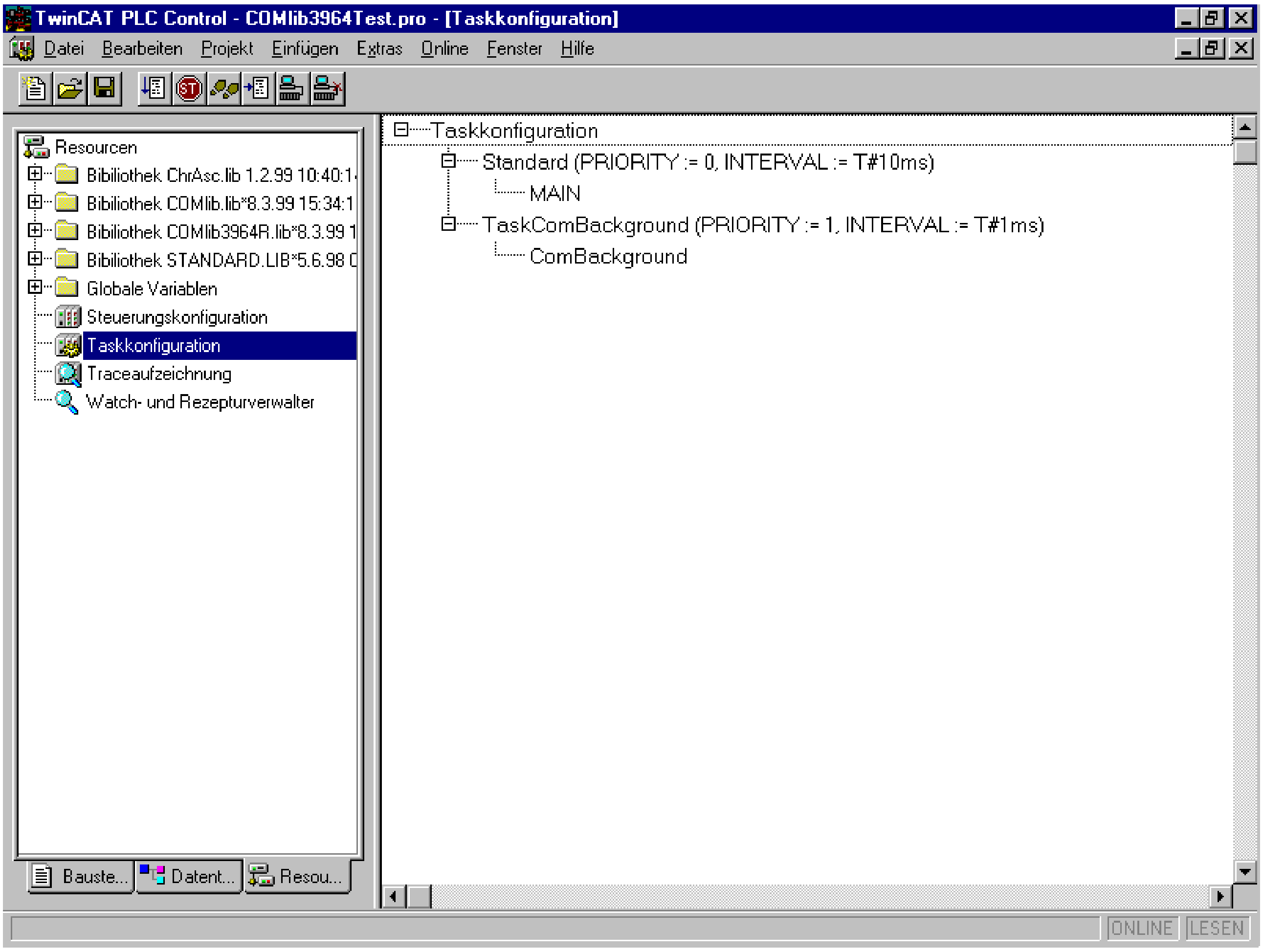
The data structures of type PcComInData or PcComOutData are linked with the hardware in the TwinCAT System Manager. Please read the corresponding section in the COMlib documentation in this regard.



```
VAR_GLOBAL
        (* Configuration data for COM-port COM1 *)
        COMin AT %IB10 : PcComInData;       (* connected to the hardware in
        COMout AT %QB10 : PcComOutData; (* connected to the hardware in
        RxBuffer : COMbuffer;    (* internal receive data buffer, not for user acce
        TxBuffer : COMbuffer;    (* internal transmit data buffer, not for user acce


        (*
        (*      if a KL6xxx serial line interface need to be used, the
        (*      global data structures COMin and COMout and the
        (*      control function block in COMbackground have to be changed
        (*
        (*      if the 3-Byte version of KL6xxx is to be used:
        (*
        (*      COMin AT %IB10 : KL6inData;            (* connected to the hardwa
        (*      COMout AT %QB10 : KL6OutData;          (* connected to the hardwa
        (*
        (*
        (*      if the 5-Byte version of KL6xxx is to be used:
        (*
        (*      COMin AT %IB10 : KL6inData5B;          (* connected to the hardwa
        (*      COMout AT %QB10 : KL6OutData5B;        (* connected to the hardwa
```

## 6.3 Task Configuration

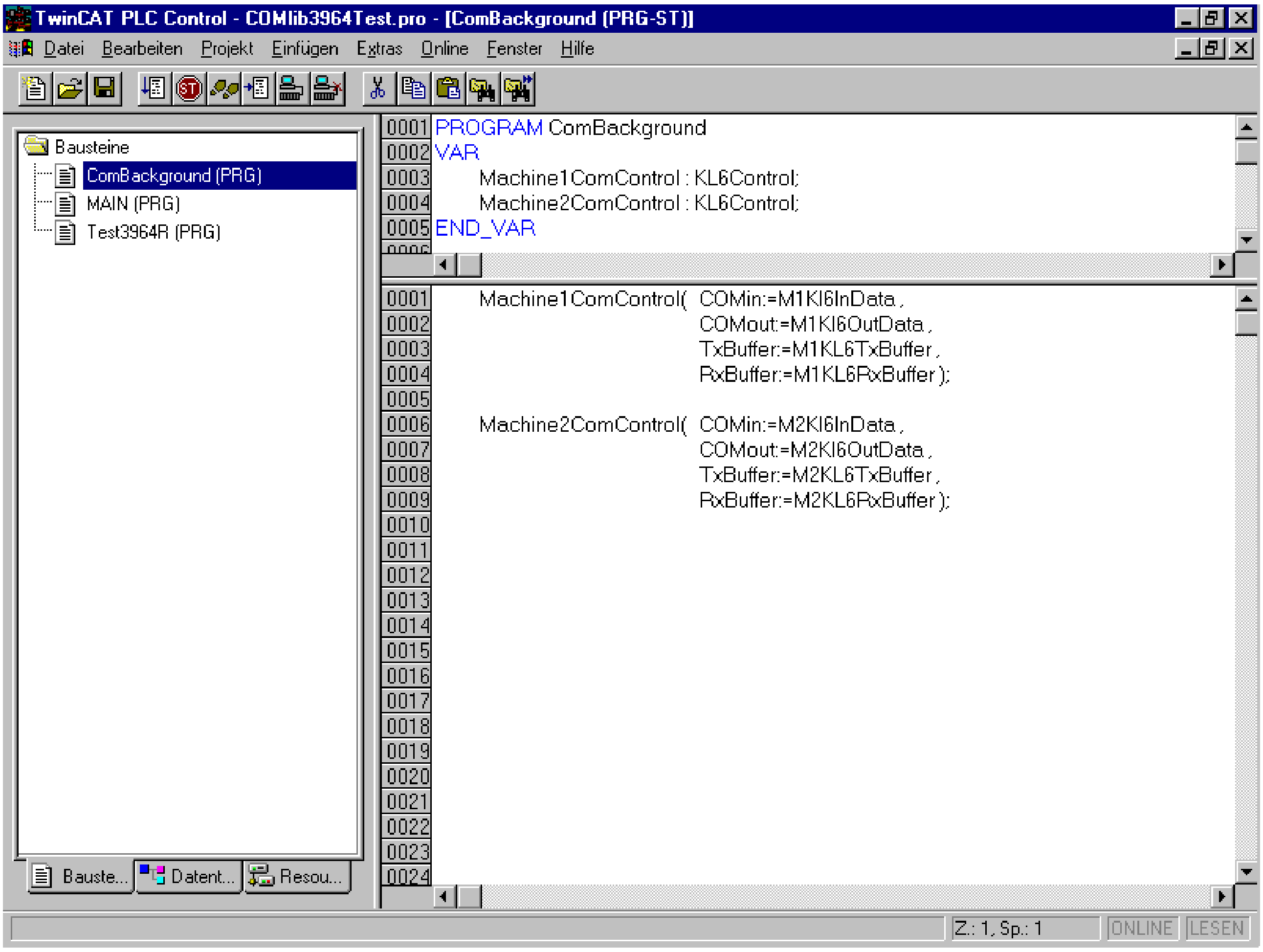
The speed of the serial interfaces must be considered in the task configuration. For example, in order that, with 9600 bps at the serial bus terminal, the data can all actually be processed at this speed, the associated communication block must be active at least once per millisecond. The task that operates the block must have a correspondingly fast setting. The simplest case is when the entire PLC program runs in this fast task. If the task is set slower, then as long as the interface operates with a hardware handshake, the communication will function at reduced speed. Without handshake, data to be received can be lost.


TwinCAT PLC Control - COMlib3964Test.pro - [Taskkonfiguration]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Resourcen
Bibiliothek ChrAsc.lib 1.2.99 10:40:1
Bibiliothek COMlib.lib*8.3.99 15:34:1
Bibiliothek COMlib3964R.lib*8.3.99 1
Bibiliothek STANDARD.LIB*5.6.98 C
Globale Variablen
Steuerungskonfiguration
Taskkonfiguration
Traceaufzeichnung
Watch- und Rezepturverwalter
Bauste...
Datent...
Resou...
ONLINE
LESEN

Taskkonfiguration
Standard (PRIORITY := 0, INTERVAL := T#10ms)
MAIN
TaskComBackground (PRIORITY := 1, INTERVAL := T#1ms)
ComBackground

## 6.4 Background Communication

Communication between the serial hardware and the data buffer, whose type is ComBuffer, is handled in a separate fast task.

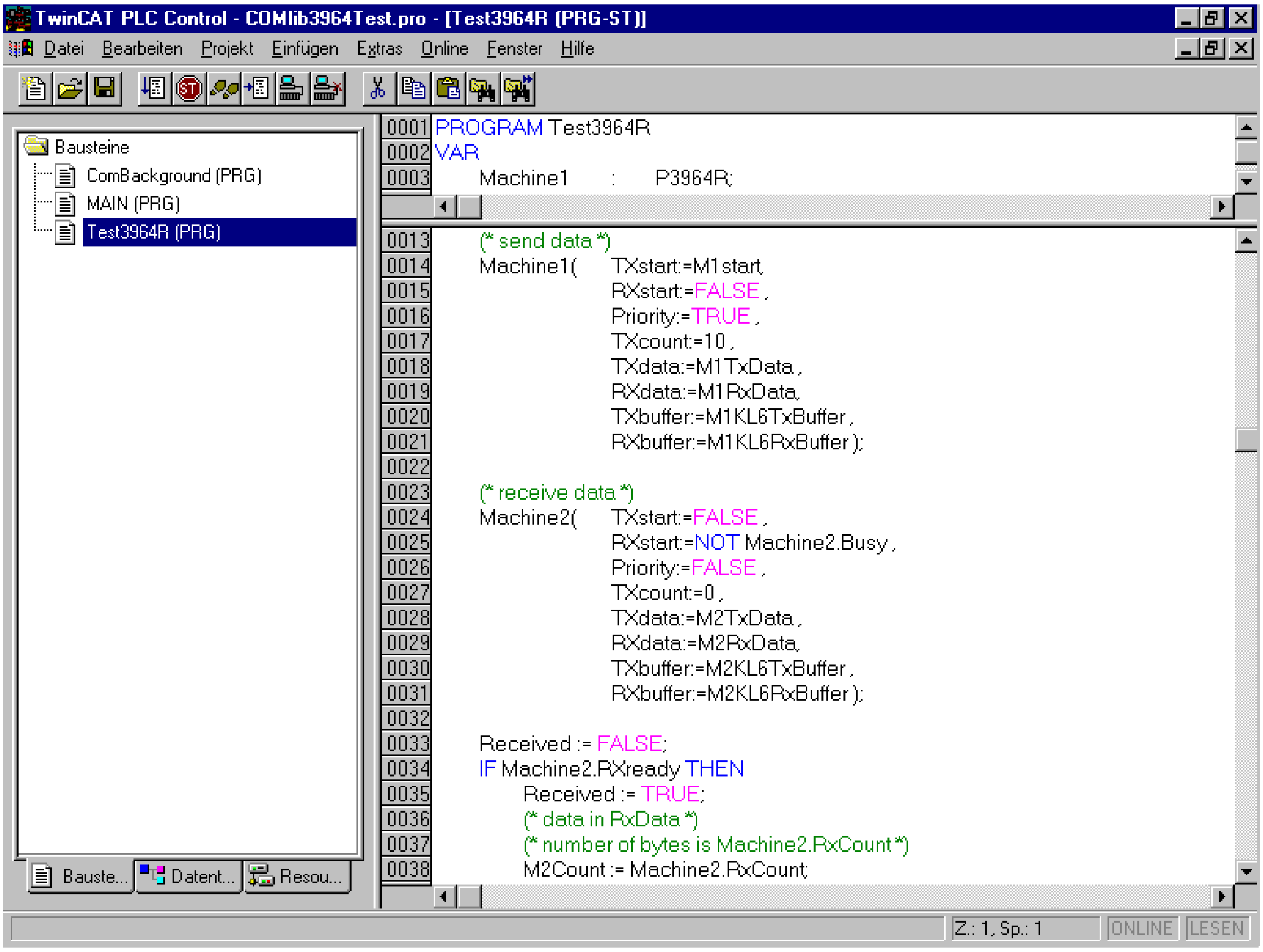


```
PROGRAM ComBackground
VAR
	Machine1ComControl : KL6Control;
	Machine2ComControl : KL6Control;
END_VAR
```

```
	Machine1ComControl(	COMin:=M1Kl6InData ,
				COMout:=M1Kl6OutData ,
				TxBuffer:=M1KL6TxBuffer ,
				RxBuffer:=M1KL6RxBuffer );

	Machine2ComControl(	COMin:=M2Kl6InData ,
				COMout:=M2Kl6OutData ,
				TxBuffer:=M2KL6TxBuffer ,
				RxBuffer:=M2KL6RxBuffer );
```

## 6.5 3964R Communication

The example program sends, for the purposes of demonstration, data to machine 1 every 10 seconds, and receives data constantly from machine 2

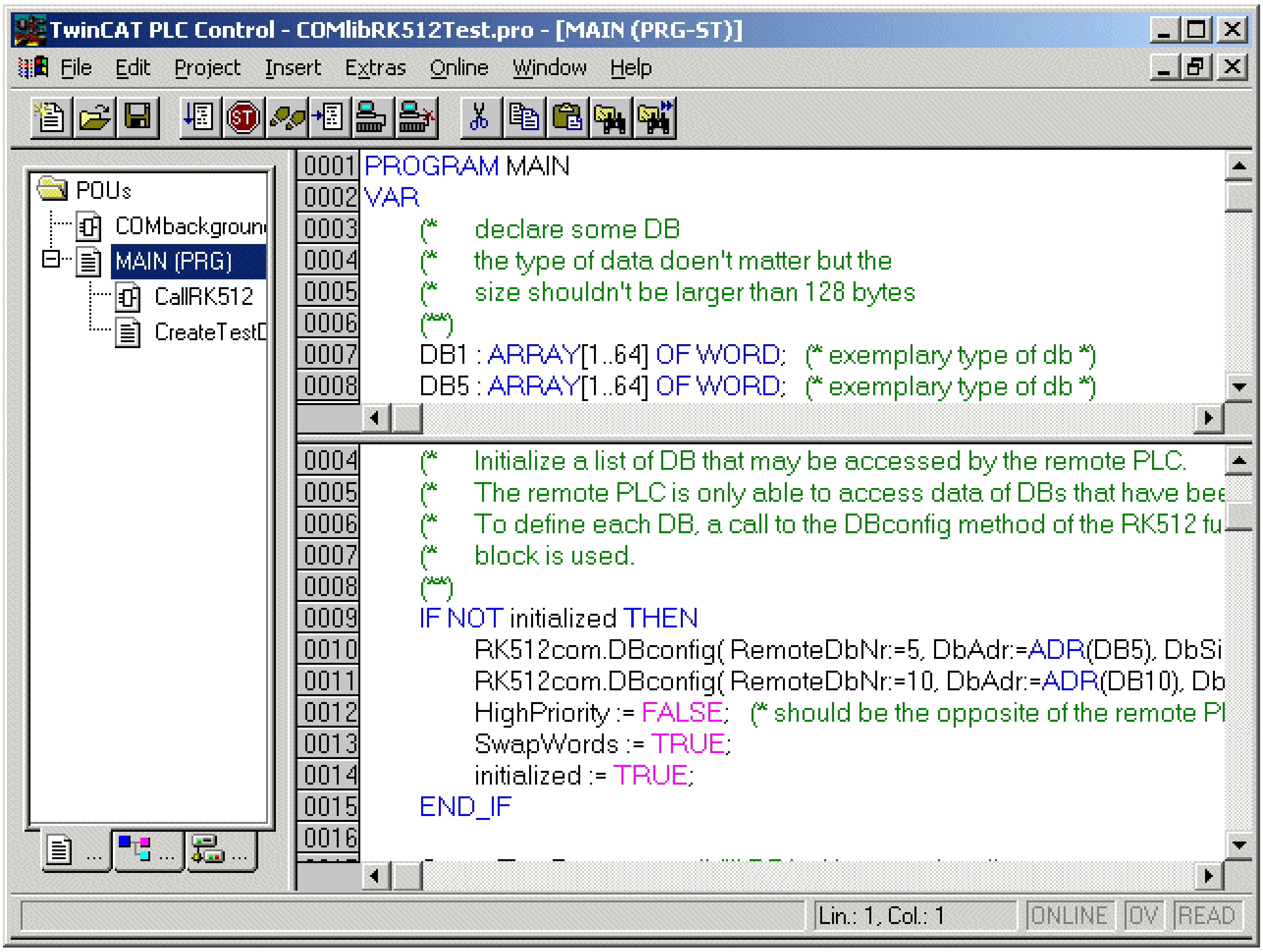

TwinCAT PLC Control - COMlib3964Test.pro - [Test3964R (PRG-ST)]
Datei Bearbeiten Projekt Einfügen Extras Online Fenster Hilfe
Bausteine
ComBackground (PRG)
MAIN (PRG)
Test3964R (PRG)
Bauste...
Datent...
Resou...
Z.: 1, Sp.: 1
ONLINE
LESEN

```
PROGRAM Test3964R
VAR
        Machine1        :       P3964R;
```

```
        (* send data *)
        Machine1(       TXstart:=M1start,
                        RXstart:=FALSE ,
                        Priority:=TRUE ,
                        TXcount:=10 ,
                        TXdata:=M1TxData ,
                        RXdata:=M1RxData,
                        TXbuffer:=M1KL6TxBuffer ,
                        RXbuffer:=M1KL6RxBuffer );

        (* receive data *)
        Machine2(       TXstart:=FALSE ,
                        RXstart:=NOT Machine2.Busy ,
                        Priority:=FALSE ,
                        TXcount:=0 ,
                        TXdata:=M2TxData ,
                        RXdata:=M2RxData,
                        TXbuffer:=M2KL6TxBuffer ,
                        RXbuffer:=M2KL6RxBuffer );

        Received := FALSE;
        IF Machine2.RXready THEN
            Received := TRUE;
            (* data in RxData *)
            (* number of bytes is Machine2.RxCount *)
            M2Count := Machine2.RxCount;
```

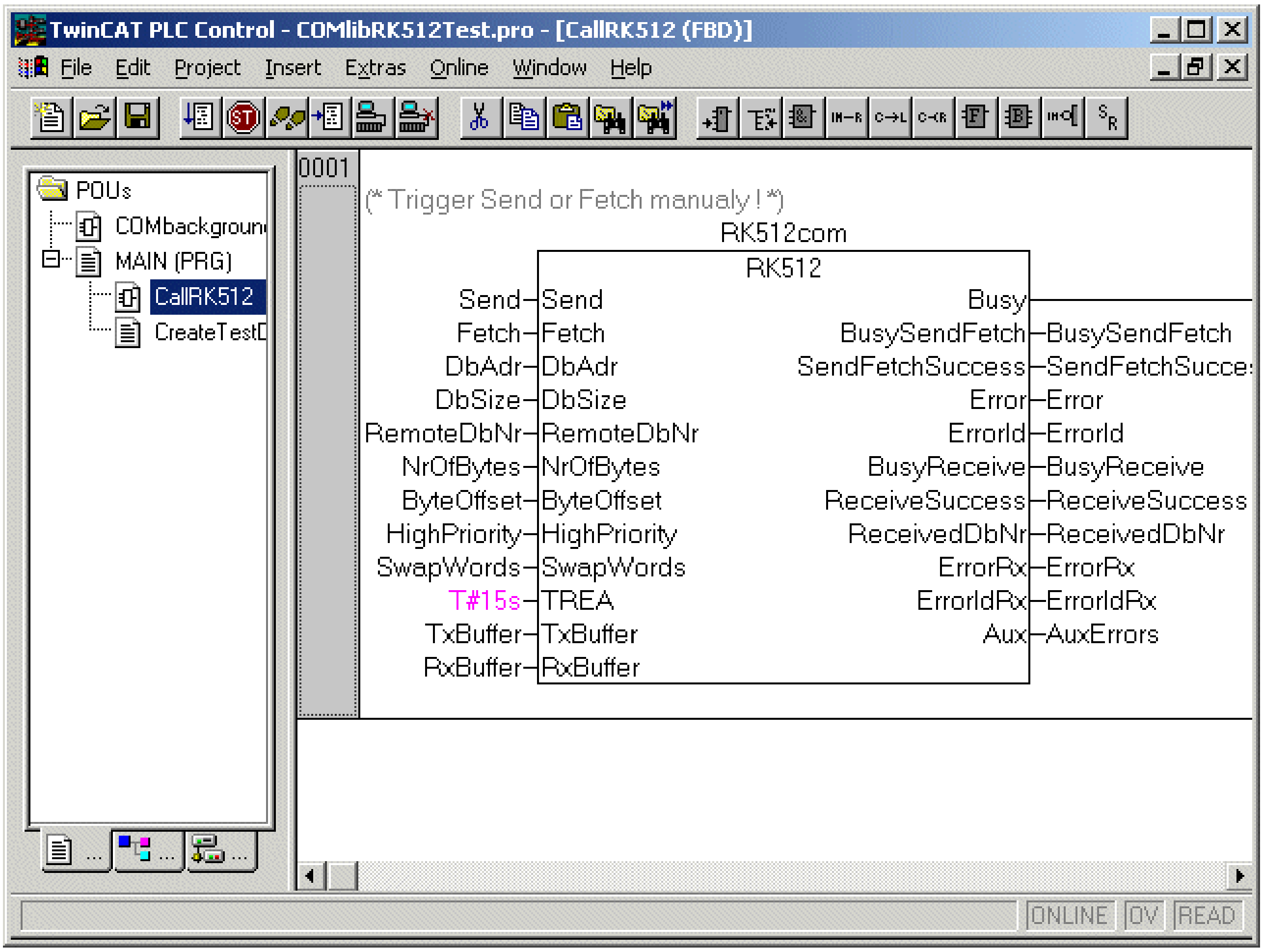
## 6.6 RK512 Communication

The example program defines three WORD arrays, in order to use them as data blocks. During initialisation the data blocks 5 and 10 are registered for access by the communication partners by calling the DBconfig method. The function block RK512 behaves passively, and is ready to accept and to answer data telegrams from the partner device. The test program allows active sending or fetching of data to be initiated by writing to the *Send* or *Fetch* variables.



```
PROGRAM MAIN
VAR
        (*      declare some DB
        (*      the type of data doen't matter but the
        (*      size shouldn't be larger than 128 bytes
        (**)
        DB1 : ARRAY[1..64] OF WORD;   (* exemplary type of db *)
        DB5 : ARRAY[1..64] OF WORD;   (* exemplary type of db *)
```

```
        (*      Initialize a list of DB that may be accessed by the remote PLC.
        (*      The remote PLC is only able to access data of DBs that have bee
        (*      To define each DB, a call to the DBconfig method of the RK512 fu
        (*      block is used.
        (**)
        IF NOT initialized THEN
                RK512com.DBconfig( RemoteDbNr:=5, DbAdr:=ADR(DB5), DbSi
                RK512com.DBconfig( RemoteDbNr:=10, DbAdr:=ADR(DB10), Db
                HighPriority := FALSE;   (* should be the opposite of the remote Pl
                SwapWords := TRUE;
                initialized := TRUE;
        END_IF
```



```
(* Trigger Send or Fetch manualy ! *)
                              RK512com
                               RK512
          Send─Send                           Busy─
         Fetch─Fetch                BusySendFetch─BusySendFetch
         DbAdr─DbAdr             SendFetchSuccess─SendFetchSucce
        DbSize─DbSize                        Error─Error
    RemoteDbNr─RemoteDbNr                  ErrorId─ErrorId
     NrOfBytes─NrOfBytes               BusyReceive─BusyReceive
    ByteOffset─ByteOffset           ReceiveSuccess─ReceiveSuccess
  HighPriority─HighPriority          ReceivedDbNr─ReceivedDbNr
     SwapWords─SwapWords                   ErrorRx─ErrorRx
        T#15s─TREA                       ErrorIdRx─ErrorIdRx
      TxBuffer─TxBuffer                        Aux─AuxErrors
      RxBuffer─RxBuffer
```

More Information:
**www.beckhoff.com/ts6341**

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com