T_DIAG: Checking the connection

Description

You use the "T_DIAG" instruction to check the status of a connection and read further information on the local end point of this connection.

- The connection is referenced by the ID parameter. You can read both connection end points configured in the connection editor and programmed connection end points (e.g. with the "TCON" instruction).

  Temporary connection end points (for example end points created when you connect to an engineering station) cannot be diagnosed, as no connection ID is generated in this process.
- The connection information read is stored in a structure referenced by the RESULT parameter.
- The output parameter STATUS indicates whether it was possible to read the connection information. The connection information in the structure at the RESULT parameter is only valid if the "T_DIAG" instruction has been completed with STATUS = W#16#0000 and ERROR = FALSE.

  Connection information cannot be evaluated if an error occurs.
- As of firmware version V2.9, the S7-1500 CPU contains the instruction "TCONSettings". If you call "T_DIAG" with a connection ID provided by "TCONSettings", the structure referenced by the RESULT parameter contains the following values:

Name	Value
InterfaceID	64 (default value)
ID	Identical to Parameter ID
ConnectionType	0
ActiveEstablished	FALSE
State	1 (disconnected)
Child	3 (programmed connection)
SentBytes	0
ReceivedBytes	0
ConnTrials	0
ConnTrialsSuccess	0
LastConnErrReason	0
LastConnErrTimeStamp	LDT#1970-01-01-00:00:00
LastDisconnReason	0
LastDisconnTimeStamp	LDT#1970-01-01-00:00:00

- As of firmware version V4.5, the S7-1200 CPU contains the instruction "TCONSettings". If you call "T_DIAG" with a connection ID provided by "TCONSettings", the structure referenced by the RESULT parameter contains the following values:

Name	Value
InterfaceID	64 (default value)

ID	Identical to Parameter ID
ConnectionType	0
ActiveEstablished	FALSE
State	1 (disconnected)
Child	3 (programmed connection)
SentBytes	0
ReceivedBytes	0

Possible connection information

Two different structures can be used to read the connection information at the RESULT parameter:

- The "TDiag_Status" structure only contains the most important information about a connection end point, for example the protocol used, the connection status and the number of data bytes sent or received.
- The structure "TDiag_StatusExt" (with S7-1500 CPUs only) supplies not just the most important information, but also the number of connection attempts, the reason for a connection abort, etc.

The structure and parameters of the two structures are described below (see the "TDIAG_Status and TDIAG_StatusExt structures" table).

Parameters

The following table shows the parameters of the instruction "T_DIAG":

Parameter	Declaration	Data type	Memory area	Description
REQ	Input	BOOL	I, Q, M, D, L or constant	Starts the instruction to check the connection specified in the ID parameter when there is a positive edge.
ID	Input	CONN_OUC (WORD)	I, Q, M, D, L or constant	Reference to the assigned connection. Value range: W#16#0001 to W#16#0FFF
RESULT	InOut	VARIANT	D	Pointer to the structure in which the connection information is stored. The structures TDiag_Status or TDiag_Status-Ext (with S7-1500 CPUs only) can be used at the RESULT parameter (for a description, see the "TDIAG_Status and TDIAG_StatusExt structures" table).
DONE	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or still in progress. • 1: Instruction executed without errors.

BUSY	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: Instruction not yet started or already completed. • 1: Instruction not yet completed. A new job cannot be started.
ERROR	Output	BOOL	I, Q, M, D, L	Status parameter: • 0: No error. • 1: Error occurred.
STATUS	Output	WORD	I, Q, M, D, L	Status of the instruction

You can find additional information on valid data types under "**Overview of the valid data types**".

BUSY, DONE and ERROR parameters

You can check the status of "T_DIAG" instruction execution with the BUSY, DONE, ERROR and STATUS parameters. The BUSY parameter indicates the processing status. You use the DONE parameter to check whether or not an instruction has been executed successfully. The ERROR parameter is set if errors occur during execution of "T_DIAG".

The following table shows the relationship between the BUSY, DONE and ERROR parameters:

BUSY	DONE	ERROR	Description
1	-	-	The instruction is being processed.
0	1	0	The instruction has been executed successfully. The data in the structure referenced by RESULT are only valid if this is the case.
0	0	1	Instruction completed with an error. The cause of the error is output at the STATUS parameter.
0	0	0	No new instruction has been assigned.

Parameter STATUS

The following table shows the meaning of the values at the STATUS parameter:

STATUS* (W#16#...)	Explanation
0000	The instruction "T_DIAG" has been executed successfully. The data in the structure referenced at the RESULT parameter can be evaluated.
7000	No instruction processing active.
7001	Instruction processing launched.
7002	Connection information is being read (REQ parameter irrelevant).
8086	The value at the ID parameter is outside the valid range.
8089	The RESULT parameter points to an invalid data type (structures TDIAG_Status and TDIAG_StatusExt only).

80A3	The ID parameter references a connection end point which does not exist. With programmed connections, this error can also occur after the "TDISCON" instruction is called.
80B1	The RESULT parameter was changed before the execution of T_DIAG was completed. A change of RESULT is not permitted during execution of T_DIAG.
80C4	Internal error. Access to the connection end point is temporarily unavailable.
* The error codes can be displayed as integer or hexadecimal values in the program editor. For information on switching the display formats, refer to "See also".	

Structures TDIAG_Status and TDIAG_StatusExt

Note

TDIAG_StatusExt structure

The TDIAG_StatusExt structure only exists with S7-1500 CPUs.

The table below details the form of the TDIAG_Status and TDIAG_StatusExt structures:

- From the InterfaceID to the ReceivedBytes parameter, the "TDIAG_StatusExt" and "TDIAG_Status" structures are identical.
- The structure TDIAG_StatusExt also includes the parameters ConnTrials to LastDisconnTimeStamp.

Note

Usage of the structures TDIAG_StatusExt and TDIAG_StatusExt_V2

It is strongly recommended to use TDIAG_StatusExt_V2 as a structure for the detailed information in new projects.

If you are using TDIAG_StatusExt, accessing the "LastConnErrTimeStamp" structure element will cause your CPU to enter the "Defective" operating state.

The value of each element is only valid if the instruction has been executed without errors. If an error occurs, the content of the parameters will not change.

Name	Data type	Description
The following parameters are in both the TDIAG_Status and the TDIAG_StatusExt structure:		
InterfaceID	HW_ANY	Interface ID (LADDR) of the CPU or the CM/CP.
ID	CONN_OUC	ID of the connection diagnosed. Following a successful call, the value of this element is identical to the parameter ID of the "T_DIAG" instruction.
ConnectionType	BYTE	Protocol type used for connection: • 0x01: Not used. • ... • 0x0B: TCP protocol (IP_v4) • 0x0C: ISO-on-TCP protocol (RFC1006) • 0x0D: TCP protocol (DNS) • 0x0E: Dial-in protocol • 0x0F: WDC protocol

		• 0x10: SMTP protocol • 0x11: TCP protocol • 0x12: TCP and ISO-on-TCP protocol (RFC1006) • 0x13: UDP protocol • 0x14: Reserved • 0x15: PROFIBUS bus access protocol (FDL) • 0x16: ISO 8073 transport protocol (ISO native) • ... • 0x20: SMTP or SMTPS protocol - based on IPv4 • 0x21 SMTP or SMTPS protocol - based on IPv6 • 0x22: SMTP or SMTPS protocol based on FQDN (**F**ully **Q**ualified **D**omain **N**ame) • ... • 0x70: S7 connection • Other: Reserved
ActiveEstablished	BOOL	• FALSE: Locally, the passive connection end point • TRUE: Locally, the active connection end point
State	BYTE	Current status of the connection end point • 0x00: Not used. • 0x01: Connection terminated. Temporary status, for example, after the "T_RESET" instruction is called. The system then automatically attempts to reestablish the connection. • 0x02: The active connection end point is attempting to establish a connection to the remote communication partner. • 0x03: The passive connection end point is waiting for establishment of the connection to the remote communication partner. • 0x04: Connection established. • 0x05: The connection is being terminated. This may be because the "T_RESET" or "T_DISCON" instruction has been called. Other possible reasons are protocol errors and line breaks. • 0x06..0xFF: Not used.
Kind	BYTE	Mode of the connection end point: • 0x00: Not used. • 0x01: Configured, static connection which has been configured and loaded to the CPU. • 0x02: Configured, dynamic connection which has been configured and loaded to the CPU (not currently supported). • 0x03: Programmed connection generated in the user program with the instruction "TCON". A call of the instruction "TDISCON" or a transition to CPU STOP status has destroyed the connection end point. • 0x04: Temporary, dynamic connection established by the engineering station (ES) or operator station (OS) (this connection currently not be diagnosed due to the missing ID). • 0x05..0xFF: Not used.

SentBytes	UDINT	Number of data bytes sent.
Received-Bytes	UDINT	Number of data bytes received.
The following parameters only occur in the TDiag_StatusExt structure:		
ConnTrials	UDINT	Number of connection attempts. Once a connection has been successfully established, ConnTrials has the value 0. If this element is not equal to 0, this indicates that there are connection problems. Note: With a passive connection end point, this value is never greater than 1.
ConnTrials-Success	UDINT	Number of successful connection attempts. This element is never reset during the life cycle of a connection end point, and returns to 0 after reaching 0xFFFF FFFF. Note: This parameter is 1 if there has never been a problem in this connection.
LastConnErr-Reason	UDINT	Error ID output during the last connection attempt with errors (the error messages are identical to those at the LastDisconn-Reason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment). • 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastConnErr-TimeStamp	LDT	Time of the last connection attempt with errors.
LastDisconn-Reason	UDINT	Error ID which led to the last connection termination (the error messages are identical to those at the LastConnErrReason parameter): • 0x4F01: Remote connection end point cannot be reached (this error usually occurs during connection establishment).

		• 0x4F02: Connection terminated locally. • 0x4F03: Connection terminated by remote communication partner. • 0x4F04: Connection terminated by a protocol error. • 0x4F05: Connection terminated by a network problem detected locally. • 0x4F06: Connection terminated by a network problem detected remotely. • 0x4F07: Connected terminated due to timeout in protocol. • 0x4F08: Incorrect parameter assignment: Connection to be established to local partner's own address. • 0x4F09: Connection temporarily reset by a call of the instruction "T_RESET". • 0x4F0A: Insufficient connection resources available (quantity exceeded) • 0x4F0B: Internal error: Incorrect addressing parameters • 0x4F0C: Internal CPU communication error • 0x4F0D: Internal AS communication error between CPU and CM/CP • 0x4F0E: The local TCP/UDP port (or RFC1006-T selector) specified is already in use.
LastDisconnTimeStamp	LDT	Time of the last connection termination.

Example

You can find the example here: **Program example for T_DIAG**.

You can find additional information and the program code for the example here: **Sample Library for Instructions**.

See also

Difference between synchronous and asynchronous instructions (S7-1200, S7-1500)