IP S7 LINK SDK
SIMATIC connected Development

IP S7 LINK SDK for .NET

Tested? You want it?

License Model Prices Quotation Order Now

Book - The whole Manual as eBook

Development Guides

Development Guide FAQs

Download

The IP S7 LINK .NET SDK comes with an **evaluation license which can be used unlimited for each application run for 30 minutes**. If this restriction limits your evaluation options, you can request **another evaluation license** from us **for free**.
Just ask our support (via support@traeger.de) or let us consult you directly and clarify open questions with our developers!

Update Information

- starting from v2.2 a new license key is required!
- starting from v2.2 some classes with the prefix “Siemens” were renamed to “Simatic”!

IP S7 LINK .NET SDK – Evaluation Package[1)]
Download ZIP Archive of IPS7LnkNet.Advanced (Version: 2.5.0.0 – 2024-01-23)
Download NuGet Package of IPS7LnkNet.Advanced (Version: 2.5.0.0 – 2024-01-23)

S7 Watch (Version: 2.5.0.0 – 2024-01-23)
Ein kostenloser und einfacher, aber professioneller S7 Daten Monitor für den Datenzugriff auf S7 Steuerungen.

Version History - The list of improvements in each version

Preview Download

There are currently no preview versions available. In case you're interested in some feature the SDK may not fulfill in the latest version: **Do not hesitate and just contact us via support@traeger.de!**

IP S7 LINK

Development Guide

Example Code: Monitoring the Operating Temperature

- C#
- VB

```
namespace App
{
    using System;
    using System.Threading;

    using IPS7Lnk.Advanced;

    public class Program
    {
        public static void Main()
        {
            var device = new SimaticDevice("192.168.0.80");

            using (var connection = device.CreateConnection()) {
                connection.Open();

                while (true) {
                    var temperature = connection.ReadDouble("DB10.DBD 20");
                    Console.WriteLine($"Current Temperature is {0} °C", temperature);

                    Thread.Sleep(1000);
                }
            }
        }
    }
}
```

```
Imports System
Imports System.Threading

Imports IPS7Lnk.Advanced

Namespace App
    Public Class Program
        Public Shared Sub Main()
            Dim device = New SimaticDevice("192.168.0.80")

            Using connection = device.CreateConnection()
                connection.Open()

                While True
                    Dim temperature = connection.ReadDouble("DB10.DBD 20")
                    Console.WriteLine("Current Temperature is {0} °C", temperature)

                    Thread.Sleep(1000)
                End While
            End Using
        End Sub
    End Class
End Namespace
```

[1] Your “License Code” turns the package into a productive full version.

[2] Not recommended for productive use.

IP S7 LINK SDK
SIMATIC connected Development

Development Introduction

Tested? You want it?

License Model Prices Quotation Order Now

Addressing

Operands

The framework supports addressing of Inputs, Periphery Input, Outputs, Periphery Output, Flags, Data Blocks, Counters and Timers. Which of the components just mentioned is to be accessed is described via the operand portion of the PLC data address. The following table describes the different abbreviations, which are referred to as operands:

Name	Term (Siemens, DE)	Term (IEC)
Input	E	I
Periphery Input	PE	PI
Output	A	Q
Periphery Output	PA	PQ
Flag	M	M
Data Block	DB	DB
Counter	Z	C
Timer	T	T

In the case of a data block, the operand is followed by the number of the data block. This is followed by a period and repeats the abbreviation for the operand for data blocks.

The backus-naur form of an operand is defined as follows:

- `<Data-Block-Number> ::= 0-65535`
- `<Siemens-Operand> ::= E | PE | A | PA | M | DB<Data-Block-Number>.DB | Z | T`
- `<IEC-Operand> ::= I | PI | Q | PQ | M | DB<Data-Block-Number>.DB | C | T`

Offsets

The actual addressing part of the PLC address comes after the operand and is divided into the data type identifier (in raw data format) and the offset for the byte to be addressed as well as the bit to be addressed if necessary (separated by a dot). The supported data type identifiers and their valid terms (for the raw data types) are described in the following table:

Datatype	Term	Bits	Value Range	Description	Array
BOOL	X	1	0 to 1	A single Bit to represent true (1) or false (0)	yes
BYTE	B	8	0 to 255	Unsigned 8-Bit Integer	yes
WORD	W	16	0 to 65.535	Unsigned 16-Bit Integer (Word)	yes
DWORD	D	32	0 to 2^{32} -1	Unsigned 32-Bit Integer (Double Word)	yes
CHAR	B	8	A+00 to A+ff	Unsigned 8-Bit Zeichen as ASCII-Code	yes
INT	W	16	-32.768 to 32.767	Signed 16-Bit Integer	yes
DINT	D	32	-2^{31} to 2^{31}-1	Signed 32-Bit Integer	yes
REAL	D	32	+-1.5e-45 to +-3.4e38	32-Bit Floating point number with single precision (according to IEEE754)	yes
S5TIME	W	16	00.00:00:00.100 to 00.02:46:30.000	Binary Coded Decimal-Number (BCD), representing a span of time	
TIME	D	32	00.00:00:00.000 to 24.20:31:23.647	Signed 16-Bit Integer, representing a span of time in milliseconds	

Datatype	Term	Bits	Value Range	Description	Array
TIME_OF_DAY	D	32	00.00:00:00.000 to 00.23:59:59.999	Unsigned 16-Bit Integer, representing a span of time in milliesconds	
DATE	W	16	01.01.1990 to 31.12.2168	Unsigned 16-Bit Integer, representing a date in days	
DATE_AND_TIME	D	64	00:00:00.000 01.01.1990 to 23:59:59.999 31.12.2089	Binary Coded Decimal-Number (BCD), representing a date and time	
S7STRING	B	-	A+00 to A+ff	A ASCII-Code encoded String of 254 Bytes maximum	

Raw Data Types

The term stands for:

- X = Bit, to address a single bit *(when addressing a bit, the X is optional)*
- B = Byte, to address 8 Bits
- W = Word, to address 16 Bits
- D = Double Word (short DWord), to address 32 Bits

If the operand part and the data type identifiers just mentioned are put together, the following Backus-Naur form results:

```
<Operand with data type identifier> ::= (<Siemens-Operand> | <IEC-Operand>) [ X ] | B | W | D
```

Examples

If you add the offset information to the PLC address, the address is complete and can look like this:

Example	Datatype	PLC Address (Siemens, DE)	PLC Address (IEC)
First Bit in first Byte in Input	BOOL	E 1.0	I 1.0
Seventh Bit in first Byte in Output	BOOL	A 1.7	Q 1.7
First Bit in tenth Byte in Flag	BOOL	M 10.1	M 10.1
Zero Bit in first Byte in Data Block with number 1	BOOL	DB1.DBX 1.0	DB1.DBX 1.0
First byte in Input	BYTE	EB 1	IB 1
Thenth Byte in Output	BYTE	AB 10	QB 10
Hundres of Byte in Flag	BYTE	MB 100	MB 100
Zero Byte in Periphery Input	BYTE	PEB 0	PIB 0
First Byte in Periphery Output	BYTE	PAB 1	PQB 1
First Byte in Data Block with number 2	BYTE	DB2.DBB 1	DB2.DBB 1
Zero Double Word in Periphery Input	DWORD	PED 0	PID 0

With the Backus-Naur form for the byte and bit offset, the complete PLC address is as follows:

- `<Byte Offset> ::= 0-65535`
- `<Bit Offset> ::= 0-7`
- `<PLC Address> ::= <Operand with data type identifier> <Byte Offset> [. <Bit Offset> ]`

The connection to the PLC

Open

This happens when you call 'Open':

1. It is checked whether an **end point has been set** (EndPoint property).
2. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Opening***.
3. The connection **checks its configuration** for validity and coherence.
4. The connection then prepares for the first communication with the PLC.
5. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Opened***.

Connect

This happens when calling 'Connect':

1. It is checked whether the connection has already been opened (state property).
2. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Connecting***.
3. The connection establishes a physical connection for the PLC and now occupies one of the maximum possible connections to the PLC.
4. Then, precautions are taken for **monitoring the connection**:
    1. “KeepAlive-Tracking” for the detection of dropped connections
    2. ~~“Notification-Tracking” to receive notifications~~
5. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Connected***.

Close

This happens when you call 'Close':

1. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Closing***.
2. The connection **releases** all **acquired resources**
3. **Completes** the monitoring of the physical connection
4. Any physical **connection to the PLC that has been established is terminated** and is thus available to other participants.
5. The socket created during connect is **closed and discarded**.
6. The connection changes its status (**PlcDeviceConnection.State** property) to the value ***Closed***.

BreakDetection

The “BreakDetection” designates the mechanism that is responsible for the detection of connection aborts. The KeepAlive process is used to **recognize** a timeout of the connection to the PLC. If the timeout occurs, the framework **automatically tries to re-establish a connection to the PLC**. While KeepAlive messages are sent to the PLC at regular intervals in order to “test” and “maintain” the connection, if the response times are too long (= timeout reached?) to receive a KeepAlive message it is assumed that the connection is interrupted. If this is the case, another KeepAlive message is sent at ever greater intervals. If these remain unanswered, a broken connection is assumed and the previously described mechanism for re-establishing the connection comes into effect. The termination detection, which is not activated by default, is activated via the **PlcDeviceConnection.UseBreakDetection** property.

Connection Parameters

The correct parameters must be set so that a connection to the PLC can be established. **Generally** the **IP address of the PLC** (**IPDeviceEndPoint.Adress** property) **is required**, under which it can be reached. The endpoint instance expected from the **SiemensDevice** setups the connection with all the information that is primarily necessary to connect to the PLC. Instead of the static IP address, the DNS name of the PLC can also be used, so “plc_man_drill_001” could be used instead of “192.168.0.80”. In addition to the IP address of the PLC, the end point requires the **number of the rack** (the position of the mounting rail) and the **number of the slot** of the CPU. If no rack or slot number is configured at the end point, the framework **automatically tries to set the appropriate numbers**. Which number has to be entered manually in the individual case always depends on the setup of the system / control cabinet. The rack number (starting at 0) defines the position of the mounting rail on which the PLC was attached in the body. The slot number (starting at 1) defines the position of the CPU in the setup of the modules of the PLC. An S7-300 mounted on the first top hat rail therefore has rack number 0.

The arrangement of the modules (on the mounting rail) corresponds to the diagram:

1. “Power Supply (SP)” Modul
2. “CPU” Modul
3. “Interface” Modul (IM)
4. “Signal” Modul 1 (SM)
5. “Signal” Modul 2 (SM)
6. “Signal” Modul 3 (SM)
7. ...

Then slot number 2 applies to this PLC because the CPU was positioned at this position.

Table of Contents