

SIEMENS

Ingenuity for life

Industry Online Support

Home

MQTT-Publisher (unverschlüsselt) für die S7-1500/ S7-1200 und S7-300

Bausteine für S7-1500/ S7-1200, S7-300, Version 1.1

<https://support.industry.siemens.com/cs/ww/de/view/109748872>

Siemens
Industry
Online
Support



Rechtliche Hinweise

Nutzung der Anwendungsbeispiele

In den Anwendungsbeispielen wird die Lösung von Automatisierungsaufgaben im Zusammenspiel mehrerer Komponenten in Form von Text, Grafiken und/oder Software-Bausteinen beispielhaft dargestellt. Die Anwendungsbeispiele sind ein kostenloser Service der Siemens AG und/oder einer Tochtergesellschaft der Siemens AG („Siemens“). Sie sind unverbindlich und erheben keinen Anspruch auf Vollständigkeit und Funktionsfähigkeit hinsichtlich Konfiguration und Ausstattung. Die Anwendungsbeispiele stellen keine kundenspezifischen Lösungen dar, sondern bieten lediglich Hilfestellung bei typischen Aufgabenstellungen. Sie sind selbst für den sachgemäßen und sicheren Betrieb der Produkte innerhalb der geltenden Vorschriften verantwortlich und müssen dazu die Funktion des jeweiligen Anwendungsbeispiels überprüfen und auf Ihre Anlage individuell anpassen.

Sie erhalten von Siemens das nicht ausschließliche, nicht unterlizenzierbare und nicht übertragbare Recht, die Anwendungsbeispiele durch fachlich geschultes Personal zu nutzen. Jede Änderung an den Anwendungsbeispielen erfolgt auf Ihre Verantwortung. Die Weitergabe an Dritte oder Vervielfältigung der Anwendungsbeispiele oder von Auszügen daraus ist nur in Kombination mit Ihren eigenen Produkten gestattet. Die Anwendungsbeispiele unterliegen nicht zwingend den üblichen Tests und Qualitätsprüfungen eines kostenpflichtigen Produkts, können Funktions- und Leistungsmängel enthalten und mit Fehlern behaftet sein. Sie sind verpflichtet, die Nutzung so zu gestalten, dass eventuelle Fehlfunktionen nicht zu Sachschäden oder der Verletzung von Personen führen.

Haftungsausschluss

Siemens schließt seine Haftung, gleich aus welchem Rechtsgrund, insbesondere für die Verwendbarkeit, Verfügbarkeit, Vollständigkeit und Mangelfreiheit der Anwendungsbeispiele, sowie dazugehöriger Hinweise, Projektierungs- und Leistungsdaten und dadurch verursachte Schäden aus. Dies gilt nicht, soweit Siemens zwingend haftet, z.B. nach dem Produkthaftungsgesetz, in Fällen des Vorsatzes, der groben Fahrlässigkeit, wegen der schuldhaften Verletzung des Lebens, des Körpers oder der Gesundheit, bei Nichteinhaltung einer übernommenen Garantie, wegen des arglistigen Verschweigens eines Mangels oder wegen der schuldhaften Verletzung wesentlicher Vertragspflichten. Der Schadensersatzanspruch für die Verletzung wesentlicher Vertragspflichten ist jedoch auf den vertragstypischen, vorhersehbaren Schaden begrenzt, soweit nicht Vorsatz oder grobe Fahrlässigkeit vorliegen oder wegen der Verletzung des Lebens, des Körpers oder der Gesundheit gehaftet wird. Eine Änderung der Beweislast zu Ihrem Nachteil ist mit den vorstehenden Regelungen nicht verbunden. Von in diesem Zusammenhang bestehenden oder entstehenden Ansprüchen Dritter stellen Sie Siemens frei, soweit Siemens nicht gesetzlich zwingend haftet.

Durch Nutzung der Anwendungsbeispiele erkennen Sie an, dass Siemens über die beschriebene Haftungsregelung hinaus nicht für etwaige Schäden haftbar gemacht werden kann.

Weitere Hinweise

Siemens behält sich das Recht vor, Änderungen an den Anwendungsbeispielen jederzeit ohne Ankündigung durchzuführen. Bei Abweichungen zwischen den Vorschlägen in den Anwendungsbeispielen und anderen Siemens Publikationen, wie z. B. Katalogen, hat der Inhalt der anderen Dokumentation Vorrang.

Ergänzend gelten die Siemens Nutzungsbedingungen (<https://support.industry.siemens.com>).

Securityhinweise

Siemens bietet Produkte und Lösungen mit Industrial Security-Funktionen an, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen.

Um Anlagen, Systeme, Maschinen und Netzwerke gegen Cyber-Bedrohungen zu sichern, ist es erforderlich, ein ganzheitliches Industrial Security-Konzept zu implementieren (und kontinuierlich aufrechtzuerhalten), das dem aktuellen Stand der Technik entspricht. Die Produkte und Lösungen von Siemens formen nur einen Bestandteil eines solchen Konzepts.

Der Kunde ist dafür verantwortlich, unbefugten Zugriff auf seine Anlagen, Systeme, Maschinen und Netzwerke zu verhindern. Systeme, Maschinen und Komponenten sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn und soweit dies notwendig ist und entsprechende Schutzmaßnahmen (z.B. Nutzung von Firewalls und Netzwerksegmentierung) ergriffen wurden.

Zusätzlich sollten die Empfehlungen von Siemens zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Industrial Security finden Sie unter: <https://www.siemens.com/industrialsecurity>.

Die Produkte und Lösungen von Siemens werden ständig weiterentwickelt, um sie noch sicherer zu machen. Siemens empfiehlt ausdrücklich, Aktualisierungen durchzuführen, sobald die entsprechenden Updates zur Verfügung stehen und immer nur die aktuellen Produktversionen zu verwenden. Die Verwendung veralteter oder nicht mehr unterstützter Versionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Produkt-Updates informiert zu sein, abonnieren Sie den Siemens Industrial Security RSS Feed unter: <http://www.siemens.com/industrialsecurity>.

Inhaltsverzeichnis

Rechtliche Hinweise	2
1 Einführung.....	4
1.1 Überblick.....	4
1.2 Funktionsweise.....	5
1.3 Verwendete Komponenten	6
2 Engineering	7
2.1 Bausteinbeschreibung	7
2.1.1 Schnittstellenbeschreibung "LMqtt_Publisher"	7
2.1.2 Datenbaustein "LMqtt_Data"	8
2.2 Integration ins Anwenderprojekt.....	13
2.3 Parametrierung und Bedienung	14
2.4 Fehlerhandling.....	16
3 Wissenswertes	18
3.1 Grundlagen zu MQTT.....	18
3.1.1 Terminologie.....	18
3.1.2 Standard und Architektur.....	19
3.1.3 Features	20
3.1.4 MQTT-Kontrollpakete	22
3.2 Details zur Funktionsweise des FB "LMqtt_Publisher"	24
3.2.1 Voraussetzungen und Umsetzung	24
3.2.2 Zustandsautomat "TCP"	24
3.2.3 Zustandsautomat "MQTT"	26
3.2.4 Zustandsautomat "PUSH"	28
3.2.5 Funktionsdiagramm	30
4 Anhang.....	31
4.1 Service und Support.....	31
4.2 Links und Literatur	32
4.3 Änderungsdokumentation	32

1 Einführung

1.1 Überblick

Motivation

Die Digitalisierung hat einen großen Einfluss auf die Wirtschaft und die Gesellschaft und schreitet unaufhaltsam voran. Das "Internet of Things" (Internet der Dinge, Kurzform: IoT) ist einer der Haupttreiber der Digitalisierung. Der Begriff "Internet of Things" steht synonym für eine der größten aktuellen Veränderungsdynamiken: die zunehmende Vernetzung und Automatisierung von Geräten, Maschinen und Produkten.

Das Protokoll "Message Queue Telemetry Transport" (Kurzform: MQTT) wird im "Internet of Things" als Kommunikationsprotokoll eingesetzt. Durch seinen leichtgewichtigen Ansatz eröffnet es ganz neue Möglichkeiten in der Automatisierung.

Schlank und Schnell: MQTT

Das MQTT ist ein einfach aufgebautes binäres Publish- und Subscribe-Protokoll auf TCP/IP-Ebene. Es eignet sich für den Nachrichtenaustausch zwischen Geräten mit geringer Funktionalität und für die Übertragung über unzuverlässige Netze mit geringer Bandbreite und hoher Latenz. Mit diesen Charakteristiken spielt MQTT eine wichtige Rolle für das IoT und in der M2M-Kommunikation.

Merkmale von MQTT

Das MQTT-Protokoll hebt sich durch folgende Merkmale hervor:

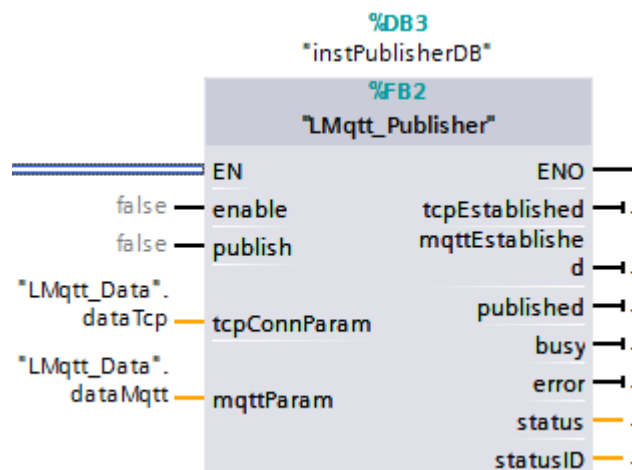
- Leichtgewichtiges Protokoll mit geringem Transport-Overhead
- Minimaler Bedarf an Netzwerk-Bandbreite durch Push-Mechanismus
- Funktion zum Wiederverbinden nach Abbruch der Verbindung
- Erneutes Ausliefern von Nachrichten nach Verbindungsabbruch
- Mechanismus zur Benachrichtigung von Interessenten nach einem unvorhergesehen Verbindungsabbruch eines Clients
- Einfache Nutzung und Implementierung durch einen geringen Satz an Befehl-Kommandos
- Qualitätssicherung (QoS-Level) mit verschiedenen Zuverlässigkeitsstufen für die Nachrichten-Zustellung
- Optionale Verschlüsselung der Nachrichten mit SSL/TLS
- Authentifizierung der Publisher und der Subscriber mit Benutzername und Passwort

Applikative Umsetzung

Um das MQTT-Protokoll in eine SIMATIC S7-Steuerung zu implementieren, wird Ihnen mit diesem Anwendungsbeispiel eine adäquate Lösung angeboten.

Das Anwendungsbeispiel stellt Ihnen je einen Funktionsbaustein für die SIMATIC S7-1500/ SIMATIC S7-1200 und für die SIMATIC S7-300 zur Verfügung. Der Funktionsbaustein "LMqtt_Publisher" integriert die MQTT-Client-Funktion und ermöglicht es Ihnen, MQTT-Nachrichten an einen Broker zu übermitteln (Publisher-Rolle).

Abbildung 1-1



Hinweis Der MQTT-Client unterstützt die MQTT-Protokollversion 3.1.

1.2 Funktionsweise

Schematische Darstellung

Die folgende Abbildung zeigt die wichtigsten Zusammenhänge zwischen den beteiligten Komponenten und Schritten, die für eine MQTT-Kommunikation notwendig sind.

Abbildung 1-2

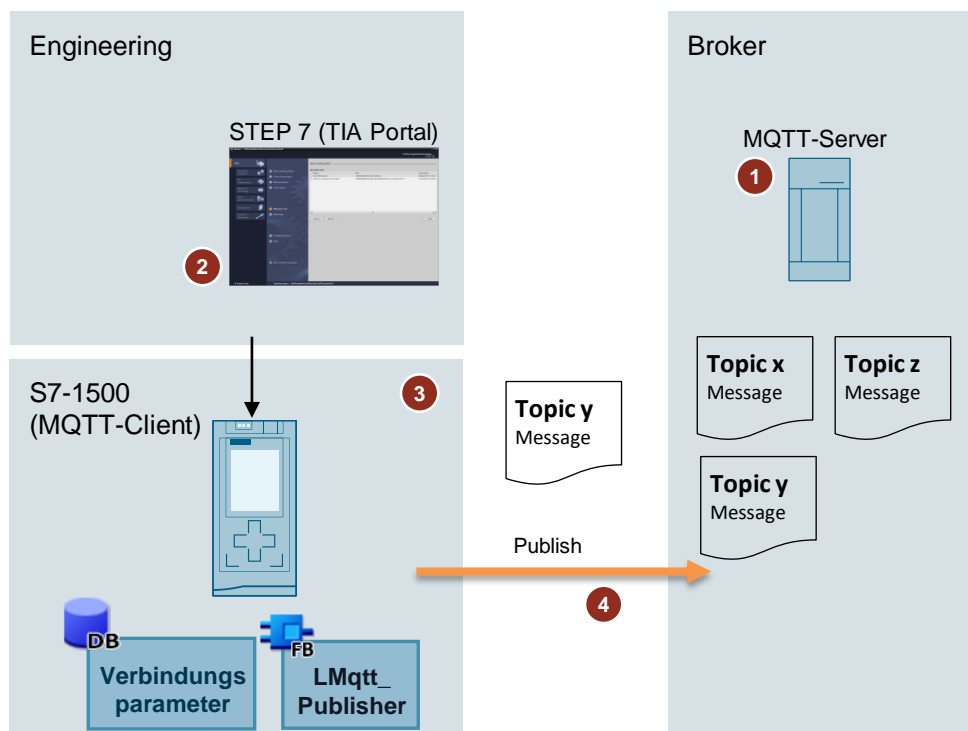


Tabelle 1-1

Schritt	Beschreibung
1	Installieren und konfigurieren Sie den MQTT-Broker.
2	Erstellen Sie ein Projekt in STEP 7 (TIA Portal) mit Ihrer CPU.
3	Der Funktionsbaustein „LMqtt_Publisher“ übernimmt die Rolle des Publisher und sendet MQTT-Nachrichten an den Broker.
4	Die MQTT-Nachricht wird im Broker gespeichert und an die Subscriber verteilt.

Hinweis

Eine detailliertere Funktionsbeschreibung des Funktionsbausteins "LMqtt_Publisher" und Informationen zum MQTT-Protokoll finden Sie im [Kapitel 3](#).

1.3 Verwendete Komponenten

Dieses Anwendungsbeispiel wurde mit diesen Hard- und Softwarekomponenten erstellt:

Tabelle 1-2

Komponente	Anzahl	Artikelnummer	Hinweis
CPU 1513-1 PN	1	6ES7513-1AL01-0AB0	Sie können auch eine andere CPU verwenden.
CPU 317-2 PN/DP	1	6ES7317-2EK14-0AB0	Sie können auch eine andere CPU verwenden.
TIA Portal V15	-	-	
MQTT-Broker	-	-	

Dieses Anwendungsbeispiel besteht aus folgenden Komponenten:

Tabelle 1-3

Komponente	Dateiname
Bibliothek "LMqtt" und "LMqttQdn" für SIMATIC S7-1500.	109748872_MqttClient_Publish_Unsecure_S71500_LIB_V1_1.zip
Bibliothek "LMqtt" für SIMATIC S7-300.	109748872_MqttClient_Publish_Unsecure_S7300_LIB_V1_1.zip
Dieses Dokument	109748872_MqttClient_Publish_Unsecure_DOKU_V1_1_de.pdf

Hinweis

Um den Broker über eine statische IP-Adresse zu erreichen, verwenden Sie die Bibliothek "LMqtt".

Um den Broker über einen Domännennamen ("Qualified Domain Name", kurz: QDN) zu erreichen, verwenden Sie die Bibliothek "LMqttQdn".

2 Engineering

Hinweis Das Engineering in diesem Kapitel fokussiert die MQTT-Client-Funktion, die dieses Anwendungsbeispiel realisiert.
Es wird vorausgesetzt, dass Sie den MQTT-Broker bereits installiert und konfiguriert haben.

2.1 Bausteinbeschreibung

2.1.1 Schnittstellenbeschreibung "LMqtt_Publisher"

Hinweis Der Funktionsbaustein "LMqtt_Publisher" ist in allen Bibliotheken vorhanden und identisch.
Bei der S7-1500 ist der Funktionsbaustein für einen "Optimierten Bausteinzugriff" konzipiert.

Im Folgenden werden die Ein- und Ausgangsparameter des Funktionsbausteins "LMqtt_Publisher" erläutert.

Eingangsparameter

Tabelle 2-1

Parameter	Datentyp	Funktion
enable	BOOL	Mit einer positiven Flanke wird der Funktionsbaustein aktiviert. Der Funktionsbaustein ist aktiv, solange "enable" den Zustand "true" hat. Über eine negative Flanke wird der Funktionsbaustein beendet und die TCP- und MQTT-Verbindung abgebaut.
publish	BOOL	Mit einer positiven Flanke wird eine Nachricht an den Broker geschickt
tcpConnParam	"typeTcpConnParam"	Datenbereich der TCP-Verbindungsinformationen
mqttParam	"typeMqttParam"	Datenbereich der MQTT-Verbindungs- und Nachrichteninformationen

Ausgangsparameter

Tabelle 2-2

Parameter	Datentyp	Funktion
tcpConnected	BOOL	True, wenn die TCP-Verbindung aufgebaut wurde
mqttConnected	BOOL	True, wenn die MQTT-Verbindung aufgebaut wurde
published	BOOL	True, wenn die Nachricht erfolgreich am Broker angekommen ist. Er ist nur ein Zyklus auf "true".
busy	BOOL	True, während eine Nachricht oder ein PING an den Broker geschickt wird
error	BOOL	True, wenn ein Fehler anliegt
statusID	INT	Zustand, der den Fehler ausgelöst hat
status	DWORD	Fehlermeldung

2.1.2 Datenbaustein "LMqtt_Data"

In der folgenden Abbildung sehen Sie die Deklaration des Datenbausteins für die SIMATIC S7-1500/ SIMATIC S7-1200:

Hinweis Der Datenbaustein ist für einen "Optimierten Bausteinzugriff" konzipiert.

Abbildung 2-1

▼ Static	
■ ▼ dataTcp	*typeTcpConnParam*
■ hwIdentifier	HW_ANY
■ connectionID	CONN_OUC
■ ▼ ipAdressBroker	Array[0..3] of Byte
■ ipAdressBroker[0]	Byte
■ ipAdressBroker[1]	Byte
■ ipAdressBroker[2]	Byte
■ ipAdressBroker[3]	Byte
■ localPort	UInt
■ mqttPort	UInt
■ ▼ dataMqtt	*typeMqttParam*
■ ▼ connectFlag	*typeMqttConnectFlags*
■ cleanSession	Bool
■ will	Bool
■ willQoS_1	Bool
■ willQoS_2	Bool
■ willRetain	Bool
■ password	Bool
■ userName	Bool
■ ▼ publishFlag	*typeMqttPublishFlags*
■ qualityOfService	Int
■ retain	Bool
■ keepAlive	Word
■ packetIdentifier	Word
■ clientIdentifier	String[23]
■ willTopic	String[100]
■ willMessage	String[100]
■ userName	String[20]
■ password	String[20]
■ topic	String[100]
■ message	String

In der folgenden Abbildung sehen Sie die Deklaration des Datenbausteins für die SIMATIC S7-300:

Abbildung 2-2

▼ Static	
■ ▼ dataTcp	"typeTcpConnParam"
■ localDeviceId	Byte
■ connectionID	Word
■ ▶ ipAddressBroker	Array[0..3] of Int
■ localPort	Word
■ mqttPort	Word
■ ▼ dataMqtt	"typeMqttParam"
■ ▼ connectFlag	"typeMqttConnectFlags"
■ cleanSession	Bool
■ will	Bool
■ willQoS_1	Bool
■ willQoS_2	Bool
■ willRetain	Bool
■ password	Bool
■ userName	Bool
■ ▼ publishFlag	"typeMqttPublishFlags"
■ qualityOfService	Int
■ retain	Bool
■ keepAlive	Word
■ packetIdentifier	Word
■ clientIdentifier	String[23]
■ willTopic	String[100]
■ willMessage	String[100]
■ userName	String[20]
■ password	String[20]
■ topic	String[100]
■ message	String

Übersicht der Datentypen

Um die Datenmenge übersichtlich zu strukturieren, wurden mehrere Datentypen erstellt. Welche Datentypen im Programm verwendet werden, sehen Sie in der folgenden Auflistung:

- "typeTcpConnParam"
- "typeMqttParam"; untergliedert in
 - "typeMqttConnectFlags"
 - "typeMqttPublishFlags"

Datentyp "typeTcpConnParam"

In diesem Datentyp werden alle Informationen abgelegt, die zum Aufbau der TCP-Verbindung notwendig sind. Sie können diese Parameter nach Ihren Vorgaben einstellen.

Die folgende Tabelle zeigt die Parameter der SIMATIC S7-1500/ SIMATIC S7-1200:

Tabelle 2-3

Parameter	Datentyp	Bedeutung
hwIdentifizier	HW_ANY	HW-ID der PROFINET-Schnittstelle der CPU
connectionID	CONN_OUC	ID der TCP-Verbindung
ipAdressBroker	Array[0..3] of BYTE	IP-Adresse des Brokers, z. B. für die Adresse 192.168.0.10 ipAdressBroker[0] gleich "192" ipAdressBroker[1] gleich "168" ipAdressBroker[2] gleich "0" ipAdressBroker[3] gleich "10"
localPort	UINT	Lokale Portnummer in der CPU
mqttPort	UINT	Remote Port am MQTT-Broker

Hinweis

Wenn Sie die Bausteine aus der Bibliothek "LMqttQdn" verwenden, dann ersetzt der Parameter "qdnAddressBroker" den Parameter "ipAdressBroker".

Die folgende Tabelle zeigt die Parameter der SIMATIC S7-300:

Tabelle 2-4

Parameter	Datentyp	Bedeutung
localDeviceID	Byte	Steckplatzbezeichnung der PROFINET-Schnittstelle der CPU (siehe Kapitel 4.2)
connectionID	CONN_OUC	ID der TCP-Verbindung
ipAdressBroker	Array[0..3] of INT	IP-Adresse des Brokers, z. B. für die Adresse 192.168.0.10 ipAdressBroker[0] gleich "192" ipAdressBroker[1] gleich "168" ipAdressBroker[2] gleich "0" ipAdressBroker[3] gleich "10"
localPort	UINT	Lokale Portnummer in der CPU
mqttPort	UINT	Remote Port am MQTT-Broker

Datentyp "typeMqttParam"

Dieser Datentyp beinhaltet alle Informationen zu MQTT. Welche Informationen Sie hier ablegen können, sehen Sie in der folgenden Auflistung:

- Flags für den Verbindungsaufbau
- Flags für den Versand der Nachrichten
- Anmeldeinformationen am Broker
- Topic
- Nachrichtentext

Um die große Menge der Parameter strukturierter darzustellen, wurden für die Flags eigene Datentypen angelegt.

Mit dem Datentyp "typeMqttConnectFlags" können Sie die Flags für den Aufbau der Verbindung zum MQTT-Broker bestimmen.

Tabelle 2-5

Parameter	Datentyp	Bedeutung
cleanSession	BOOL	True, wenn alle Daten aus einer vorherigen Session gelöscht werden sollen.
will	BOOL	Aktiviert die "Last Will and Testament"-Funktion.
willQoS_1	BOOL	True, wenn das QoS für den letzten Willen die Stufe 1 hat.
willQoS_2	BOOL	True, wenn das QoS für den letzten Willen die Stufe 2 hat.
willRetain	BOOL	True, wenn der letzte Wille gespeichert werden soll, sobald er verschickt wurde.
password	BOOL	True, wenn der MQTT-Broker eine Anmeldung (Name und Passwort) des Client erfordert.
username	BOOL	True, wenn der MQTT-Broker eine Anmeldung (Name und Passwort) des Client erfordert.

Mit dem Datentyp "typeMqttPublishFlags" können Sie die Flags für die MQTT-Nachricht bestimmen.

Tabelle 2-6

Parameter	Datentyp	Bedeutung
qualityOfService	INT	Definiert den QoS-Level für die MQTT-Nachricht. Mögliche Werte sind: <ul style="list-style-type: none"> • "0" für QoS-Stufe 0 • "1" für QoS-Stufe 1 • "2" für QoS-Stufe 2
retain	BOOL	True, wenn die Nachricht am Broker gespeichert werden soll.

In der folgenden Tabelle sehen Sie die weiteren Parameter des Datentyps "typeMqttParam", die Sie für MQTT bestimmen können.

Tabelle 2-7

Parameter	Datentyp	Bedeutung
keepAlive	WORD	Zeitintervall der KeepAlive-Funktion in Sekunden. Die Zeit wird in hexadezimaler Schreibweise angegeben. Ein keepAlive mit Wert "0" deaktiviert die KeepAlive-Funktion. Die maximal zulässige Zeit ist 18h 12min 15 s.
packetIdentifier	WORD	Startwert für die Paketnummern. Die Nummer wird automatisch im Programm inkrementiert.
clientIdIdentifier	String [23]	Eindeutiger Name des Clients. Mit diesem Namen identifiziert sich der Client am Broker beim Verbindungsaufbau. Erlaubt sind: <ul style="list-style-type: none"> • Zahlen • Klein- und Großbuchstaben
willTopic	String [100]	Wenn das will-Flag gesetzt ist, muss an dieser Stelle das Topic für den letzten Willen definiert werden.
willMessage	String [100]	Wenn das will-Flag gesetzt ist, muss an dieser Stelle die Nachricht für den letzten Willen definiert werden.
userName	String [20]	Wenn das Username-Flag gesetzt ist, muss hier der Benutzername für das Login am Broker definiert werden.
password	String [20]	Wenn das Passwort-Flag gesetzt ist, muss hier das Passwort für das Login am Broker definiert werden.
topic	String [100]	Name für das Topic
message	String	Nachrichtentext

Hinweis

Beachten Sie folgende Regelungen:

1. Wenn Sie das "will"-Flag auf "true" setzen, müssen Sie bei den Variablen "willMessage" und "willTopic" eine Zeichenkette hinterlegen.
2. Wenn Sie das "will"-Flag auf "false" setzen, müssen Sie die folgenden Flags ebenfalls auf "false" setzen:
 - "willQoS_1"
 - "willQoS_2"
 - "willRetain"
3. Wenn Sie die Flags "username" und "password" auf "true" setzen, müssen Sie bei den Variablen "userName" und "password" eine Zeichenkette mit den Logindaten hinterlegen. Diese Logindaten müssen mit den Logindaten übereinstimmen, die Sie am MQTT-Broker hinterlegt haben.

2.2 Integration ins Anwenderprojekt

TIA Portal-Projekt erstellen

Erstellen Sie ein TIA Portal-Projekt mit der CPU, die Sie für das Anwendungsbeispiel verwenden möchten. Parametrieren Sie die Ethernet-Schnittstelle der CPU mit einer IP-Adresse, die im gleichen Subnetz wie der MQTT-Broker liegt.

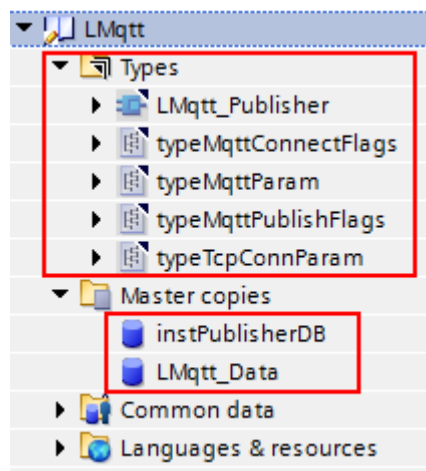
Verbinden Sie die SIMATIC-Steuerung und den MQTT-Broker über Ethernet.

Die Bausteine kopieren

Die Bausteine "LMqtt_Publish" und "LMqtt_Data" sowie die benötigten Datentypen stehen Ihnen in der Bibliothek "LMqtt" zur Verfügung.

Um die Bausteine in Ihr TIA-Projekt zu kopieren, folgen Sie dieser Anleitung:

1. Entpacken Sie die ZIP-Datei aus dem Downloadbereich dieses Anwendungsbeispiels (siehe \1\ im [Kapitel 4.2](#)) in ein lokales Verzeichnis auf Ihrem PC.
2. Öffnen Sie im TIA Portal die Bibliothekansicht. Klicken Sie in der Funktionsleiste der Palette "Globale Bibliotheken" ("Global library") auf das Symbol "Globale Bibliothek öffnen" ("Open global library"). Der Dialog "Globale Bibliothek öffnen" ("Open global library") wird geöffnet.
3. Navigieren Sie zu Ihrem Verzeichnis und wählen Sie die globale Bibliothek "LMqtt". Klicken Sie auf "Öffnen" ("Open").
4. Kopieren Sie den Inhalt aus "Typen" ("Types") und aus „Kopiervorlagen“ („Master copies“) in Ihr Projekt:

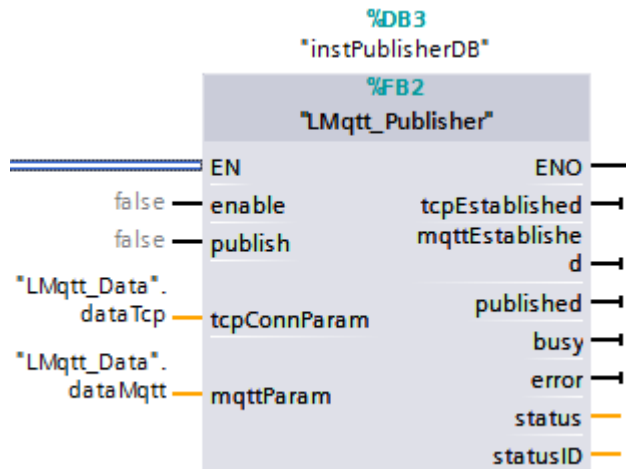


Den Funktionsbaustein aufrufen und verschalten

Wenn Sie die Bausteine in Ihr Projekt integriert haben, müssen Sie den Funktionsbaustein noch in Ihrem Programm aufrufen und verschalten.

1. Rufen Sie den Funktionsbaustein "LMqtt_Publisher" auf z. B. im OB 1 und weisen Sie ihm einen Instanzdatenbaustein zu.
2. Verschalten Sie die Ein- bzw. Ausgangsvariablen nach Belieben. Einzig die Verschaltung der Ein- und Ausgangsvariablen ist Ihnen vorgegeben:

- Ein- und Ausgangsvariable "tcpConnParam" mit "LMqtt_Data".dataTCP
- Ein- und Ausgangsvariable "mqttParam" mit "LMqtt_Data".dataMqtt



2.3 Parametrierung und Bedienung

Einstellen der Parameter

Bevor Sie das Anwendungsbeispiel testen können, müssen Sie die Parameter für die TCP-Verbindung und für MQTT nach Ihren Vorgaben einstellen.

Alle Parameter, die Sie selbst definieren können, befinden sich im Datenbaustein "LMqtt-Data". Stellen Sie die Parameter in der Spalte "Startwert" ("Start value") ein.

Sie müssen vor allem bei folgenden Parametern Ihren eigenen Wert eintragen:

- IPv4-Adresse bzw. der Domainname des MQTT-Brokers. Der Domainname muss mit einem "." abgeschlossen werden.
- remote Port, auf dem der MQTT-Broker die Nachrichten empfängt
- sämtliche MQTT-Parameter, z. B.
 - Flags für den Verbindungsaufbau
 - Flags für den Versand der Nachrichten
 - Anmeldeinformationen am Broker
 - Topic
 - Nachrichtentext

Laden Sie anschließend das Projekt in Ihre CPU.

Hinweis

Wenn Sie die Bibliothek "LMqttQdn" verwenden, dann müssen Sie in der CPU einen DNS-Server konfigurieren.

Anwendungsbeispiel bedienen

Wenn Sie alle Parameter eingestellt haben, können Sie das Anwendungsbeispiel testen.

Bevor Sie das Anwendungsbeispiel testen, überprüfen Sie folgende Punkte:

1. das Projekt ist in die CPU geladen.
2. die CPU und der MQTT-Broker sind miteinander über Ethernet verbunden und erreichbar.
3. der MQTT-Broker ist ordnungsgemäß konfiguriert und gestartet.
4. das Logging am MQTT-Broker ist bei Bedarf gestartet, um die Anmeldung des MQTT-Clients und den Publish-Mechanismus mitzuversorgen.

Wenn die genannten Punkte erfüllt sind, können Sie die MQTT-Kommunikation zwischen CPU und MQTT-Broker anstoßen. Setzen Sie die Variable "enable" am Funktionsbaustein "LMqtt_Publisher" auf das Signal „1“.

Im positiven Fall werden die internen Zustandsautomaten durchlaufen und stellen eine TCP- bzw. MQTT-Verbindung zum MQTT-Broker her. Die Ausgangsvariablen "tcpConnected" und "mqttConnected" werden gesetzt und signalisieren eine bestehende TCP- bzw. MQTT-Verbindung.

Nun können Sie eine MQTT-Nachricht senden. Triggern Sie dafür die Eingangsvariable "publish".

Wenn die Verbindung zum MQTT-Broker nicht aufgebaut wird, überprüfen Sie die Ausgangsvariable "status" und "statusID", um den Fehler zu diagnostizieren.

Welche Bedeutung die Werte der beiden Variablen haben, finden Sie im [Kapitel 2.4](#).

2.4 Fehlerhandling

Wenn ein Fehler im Programm auftritt, wird der aktuelle Zustand der Zustandsautomaten und die Fehlerursache in den Ausgangsparametern "statusID" und "status" geschrieben.

"statusID"

Am Ausgang "statusID" wird die Nummer des Zustand ausgegeben, in dem der Fehler aufgetreten ist. Die Zustände sind durchnummeriert und haben folgende Bedeutung.

Tabelle 2-8

Wert	Beschreibung
-12	MQTT_ERROR
-11	MQTT_DISCONNECTED
-2	TCP_ERROR
-1	TCP_DISCONNECT
0	IDLE
1	TCP_PARAM
2	TCP_CONNECTING
3	TCP_CONNECTED
10	MQTT_CONNECT_FLAG_CHECK
11	MQTT_CONNECT
12	MQTT_CONNACK
13	MQTT_PUBLISH
14	MQTT_PUBACK
15	MQTT_DISCONNECT
16	MQTT_CONNECTED
17	MQTT_PING
18	MQTT_PINGRESP
19	MQTT_PUBREL
20	MQTT_PUBCOMP
20	TIME_MONITORING

"status"

Der Ausgangsparameter "status" zeigt den Fehlercode an:

Tabelle 2-9

statusID	status	Beschreibung	Abhilfe
-1	Statusmeldung vom Baustein „TDISCON“	Siehe Handbuch	-
2	Statusmeldung vom Baustein „TCON“	Siehe Handbuch	Erreichbarkeit des Brokers prüfen. IP-Adresse, Port, Firewall
3	Statusmeldung vom Baustein „TRCV“	Siehe Handbuch	Netzwerkverbindung prüfen
10	W#16#80F0	Fehler beim „Will“-Flag	Flags im Datentyp „typeMqttConnectFlags“ prüfen;
	W#16#80F1	Fehler beim „WillQoS“-Flag	
	W#16#80F3	Fehler beim „KeepAlive“-Flag	KeepAlive muss größer als 2 Sekunden sein.
11	Statusmeldung vom Baustein „TSEND“	Siehe Handbuch	-
12	1	Der Broker akzeptiert nicht das MQTT-Protokoll Level	Zugangsdaten im Datentyp „typeMqttParam“ prüfen
	2	ClientIdentifier wird nicht akzeptiert	
	3	MQTT Service nicht vorhanden	
	4	Daten im Username/Passwort sind inkorrekt	
	5	Client ist nicht berechtigt	
13	Statusmeldung vom Baustein „TSEND“	Siehe Handbuch	-
14,19,20	W#16#80F2	Falschen PacketIdentifier empfangen	-
20	Statusmeldung vom Baustein „TCON“ oder „TRCV“	Zeitüberschreitung	Verbindungsparameter prüfen

3 Wissenswertes

3.1 Grundlagen zu MQTT

Hinweis Eine genaue Beschreibung zu MQTT finden Sie in der MQTT-Spezifikationsbeschreibung (siehe \3\ im [Kapitel 4.2](#)).

3.1.1 Terminologie

Im Folgenden werden die wichtigsten Begriffe bei dem Telemetrie-Protokoll MQTT erläutert.

MQTT-Nachricht

Eine Nachricht bei MQTT besteht aus mehreren Teilen:

- Einem definierten Betreff ("Topic")
- Ein zugewiesenes Merkmal zur Qualitätssicherung ("Quality of Service")
- Dem Nachrichtentext

MQTT-Client

Ein MQTT-Client ist ein Programm oder ein Gerät, das MQTT nutzt. Ein Client baut immer aktiv die Verbindung zum Broker auf. Ein Client kann folgende Funktionen ausführen:

- Nachrichten mit einem definierten Betreff ("Topic") an den Broker senden, an die anderen Clients interessiert sein könnten (Publish-Mechanismus)
- Nachrichten beim Broker abonnieren, die einem bestimmten Betreff ("Topic") folgen (Subscriber-Mechanismus)
- Sich selbst von abonnierten Nachrichten abmelden
- Die Verbindung zum Broker trennen

Hinweis Der Funktionsbaustein "LMqtt_Publisher" in diesem Anwendungsbeispiel unterstützt folgende Funktionen:

- Publish-Mechanismus
- Abmeldung am Broker.

MQTT-Broker

Ein MQTT-Broker ist die zentrale Komponente von MQTT und kann ein Programm oder ein Gerät sein. Der Broker agiert als Vermittler zwischen dem sendenden MQTT-Client und dem abonnierenden MQTT-Client. Der MQTT-Broker verwaltet die Topics inklusive der darin enthaltenen Nachrichten und regelt den Zugriff auf die Topics. Der Broker hat folgende Funktionen:

- Netzwerkverbindungen von den Clients akzeptieren
- Nachrichten eines MQTT-Clients entgegennehmen
- Abonnement-Anfragen von MQTT-Clients bearbeiten
- Nachrichten an die MQTT-Clients weiterleiten, die mit Ihrem Abonnement übereinstimmen

Hinweis

Der MQTT-Broker ist nicht Teil dieses Anwendungsbeispiels und wird als gegeben vorausgesetzt.

Topics

MQTT-Nachrichten sind in Topics organisiert. Ein Topic "beschreibt" ein Themengebiet. Die Topics können von den MQTT-Clients abonniert werden (Subscriber-Mechanismus). Dem Absender einer Nachricht (Publisher-Mechanismus) obliegt die Aufgabe, Inhalt und Topic beim Versand der Nachricht festzulegen. Der Broker kümmert sich dann darum, dass die Subscriber die Nachrichten von den abonnierten Topics bekommen. Die Topics folgen einem definierten Schema. Sie ähneln einem Verzeichnispfad und bilden eine Hierarchie ab.

3.1.2 Standard und Architektur**ISO Standard**

MQTT definiert einen OASIS- bzw. ISO-Standard (ISO/IEC PRF 20922).

Je nach verwendeten Sicherheitsprotokollen läuft MQTT auf unterschiedliche Zugriffs-Ports. Angebotene Ports sind:

- 1883: MQTT, unverschlüsselt
- 8883: MQTT, verschlüsselt
- 8884: MQTT, verschlüsselt, Client Zertifikat notwendig
- 8080: MQTT über WebSockets, unverschlüsselt
- 8081: MQTT über WebSockets, verschlüsselt

Architektur

MQTT ist ein Publish- und Subscribe-Protokoll. Dieser Mechanismus entkoppelt einen Client, der Nachrichten sendet (Publisher) von einen oder mehreren Clients, welche die Nachrichten empfangen (Subscriber). Das bedeutet auch, dass die "Publisher" nichts von der Existenz der "Subscriber" wissen (und umgekehrt). Es gibt eine dritte Komponente in der MQTT-Architektur, der MQTT-Broker. Der MQTT-Broker befindet sich zwischen "Publisher" und "Subscriber". Der MQTT-Broker sorgt für die Kommunikationssteuerung.

3.1.3 Features

MQTT bietet durchaus nützliche Features an.

Quality of Service

Für die Qualitätssicherung bei der Nachrichtenübermittlung sieht die MQTT-Spezifikation drei Levels vor:

- QoS "0": Bei der niedrigsten Stufe 0 handelt es sich um ein "fire'n'forget"-Verfahren. Es gibt also keine Garantie, dass die Nachricht überhaupt ankommt.
- QoS "1": Bei dem QoS-Level 1 ist sichergestellt, dass die Nachricht mindestens einmal in der Topic-Queue landet. Der Broker quittiert den Erhalt der Nachricht.
- QoS "2": In der höchsten Stufe 2 garantiert der Broker durch mehrfachen Handshake mit dem Client, dass die Nachricht genau einmal abgelegt wird.

Last will

MQTT unterstützt die "Last Will and Testament"-Funktion. Diese Funktion wird verwendet, um anderen Clients zu unterrichten, wenn die Verbindung zu einem Client unvorhergesehen getrennt wurde.

Jeder Client kann während des Verbindungsaufbaus zum Broker seinen letzten Willen spezifizieren und dem Broker mitteilen. Dieser letzte Wille ist wie eine normale MQTT-Nachricht aufgebaut, inklusive Topic, QoS und Payload. Der Broker speichert den letzten Willen. Sobald der Broker merkt, dass die Verbindung mit dem betreffenden Client unvorhergesehen abgebrochen wurde, schickt der Broker an alle Abonnenten, die sich für das Topic registriert haben, den letzten Willen als MQTT-Nachricht heraus. Auf diese Weise erfahren auch die Abonnenten, dass der Client getrennt wurde.

KeepAlive

MQTT unterstützt die KeepAlive-Funktion. Die KeepAlive-Funktion gewährleistet, dass die Verbindung noch offen ist sowie Client und Broker miteinander verbunden sind.

Für das KeepAlive definieren die Clients ein Zeitintervall und teilen es dem Broker während ihres Verbindungsaufbaus mit. Dieses Intervall ist die größtmögliche, geduldete Zeitperiode, in welcher der Client und der Broker ohne Kontakt verharren dürfen. Wird die Zeit überschritten, muss der Broker die Verbindung trennen.

Das bedeutet, solange der Client regelmäßig Nachrichten innerhalb des KeepAlive-Intervalls an den Broker schickt, muss der Client keine besondere Aktion ausführen, um die Verbindung aufrecht zu erhalten. Wenn der Client aber keine Nachrichten innerhalb des KeepAlive-Intervalls sendet, muss er vor Ablauf der Frist ein Ping-Paket an den Broker absetzen. Mit diesem PING signalisiert der Client dem Broker, dass er weiterhin verfügbar ist.

Wenn eine Nachricht oder ein Ping-Paket an den Broker geschickt wurde, beginnt die Zeitmessung für das KeepAlive-Intervall von vorne.

Hinweis

- Der Client bestimmt das KeepAlive-Intervall. So kann er das Intervall seiner Umgebung anpassen, z. B. wegen einer langsamen Bandbreite.
- Der größtmögliche Wert für das KeepAlive-Intervall ist 18 h 12 m 15 s
- Wenn der Client das KeepAlive-Intervall auf den Wert "0" setzt, wird der KeepAlive-Mechanismus deaktiviert.

Message Persistence

Wird die Verbindung zu einem Client unterbrochen, so kann der Broker neue Nachrichten für diesen Client für eine spätere Zustellung zwischenspeichern.

Retained Messages

Abonniert ein MQTT-Client zum ersten Mal ein Topic, bekommt er normalerweise erst dann eine Nachricht, wenn ein anderer MQTT-Client das nächste Mal eine Nachricht mit dem abonnierten Topic sendet. Mit "Retained Messages" bekommt der Abonnent den letzten Wert, der vor seiner Abonnementsanfrage an das Topic gesendet wurde, sofort zugestellt.

3.1.4 MQTT-Kontrollpakete

Die meisten MQTT-Kontrollpakete arbeiten nach dem Handshake-Verfahren. Der MQTT-Client ist immer das aktive Element und setzt einen Auftrag an den Broker ab. Der Broker bestätigt je nach Auftrag die Anfrage.

Die Struktur eines MQTT-Kontrollpaketes ist fest vorgegeben. Die folgende Grafik zeigt die Struktur:

Abbildung 3-1

<p>Fixed header Pflicht für alle Kontrollpakete</p>
<p>Variable header Pflicht für einige Kontrollpakete</p>
<p>Payload Pflicht für alle Kontrollpakete</p>

Der "Fixed Header" besteht immer aus folgenden Elementen:

- Eine Kennungsnummer für den MQTT-Kontrollpakettyp
- Ein Bereich für mögliche Flags; falls keine Flags für das Kontrollpaket vorgesehen sind, werden die Bits als "reserved" gekennzeichnet
- Die Anzahl der folgenden Bytes nach dem "Fixed Header"

Der "Variable Header" ist nur bei einigen Kontrollpaketen erforderlich. Der Inhalt des "Variable Header" ist abhängig vom Kontrollpakettyp.

Der Payload ist bei den meisten Kontrollpaketen Pflicht. Auch hier ist der Inhalt vom Kontrollpakettyp abhängig. Für jedem Kontrollpakettyp gibt es klare Regelungen, mit was und in welcher Reihenfolge der Payload befüllt werden kann.

Hinweis Eine genaue Beschreibung zu den MQTT-Kontrollpaketen finden Sie in der MQTT-Spezifikationsbeschreibung (siehe \3\ im [Kapitel 4.2](#)).

Die MQTT-Kontrollpakete aus diesem Anwendungsbeispiel werden im Folgenden kurz erläutert.

MQTT-Verbindung

Eine MQTT-Verbindung wird immer zwischen einem Client und dem Broker hergestellt. Eine direkte Client-Client-Verbindung ist nicht möglich.

Die Verbindung wird von einem Client initiiert, sobald der Client ein "CONNECT"-Paket an den Broker schickt. Im positiven Fall antwortet der Broker mit einem "CONNACK"-Paket und einem Statuscode.

In folgenden Fällen schließt der Broker sofort die Verbindung:

- Wenn das "CONNECT"-Paket schadhaft ist
- Wenn der Aufbau des "CONNECT"-Pakets nicht der Spezifikation entspricht
- Wenn der Verbindungsaufbau zu lange dauert

Ein "CONNECT"-Paket enthält im "Variable Header" einen Bereich für Flags. Das "CONNECT"-Flag-Byte beinhaltet eine Reihe an Parametern, die das Verhalten der MQTT-Verbindung spezifizieren. Zudem zeigt das "CONNECT"-Flag-Byte auch, welche optionalen Felder im "Payload" vorhanden sind oder nicht.

Im "Payload" sind folgende Felder zwingend notwendig:

- Die "ClientID" dient zur Identifikation des Clients am Broker
- Mit der "CleanSession" kann die Verbindungsart geregelt werden
- Mit der KeepAlive-Zeit wird das Zeitintervall bestimmt, in der sich der Client verpflichtend beim Broker melden muss. Das kann entweder durch das Senden einer Nachricht oder einem PING-Kommando erfolgen. Meldet sich der Client nicht in dem Zeitintervall, baut der Broker die Verbindung zum Client ab.

Optionale Felder sind z. B. Username, Passwort und Informationen über den letzten Willen ("Last Will").

MQTT-Push-Mechanismus

Sobald sich ein MQTT-Client mit dem Broker verbunden hat, kann er Nachrichten an den Broker schicken. Dazu nutzt der Client das "PUBLISH"-Paket. Da die Nachrichten bei MQTT topic-basiert gefiltert und verwaltet werden, muss jede MQTT-Nachricht ein Topic beinhalten. Das Topic ist Teil des "Variable Header". Der eigentliche Nachrichtentext ist im "Payload" untergebracht.

Abhängig von der Einstellung der Qualitätssicherung ("QoS") endet an dieser Stelle der Push-Mechanismus oder es werden weitere Kontrollpakete ausgetauscht:

Bei QoS gleich "0" endet an dieser Stelle der Sendeauftrag.

Bei QoS gleich "1" quittiert der Broker das "PUBLISH"-Paket mit einem "PUBACK".

Bei QoS gleich "2" quittiert der Broker das "PUBLISH"-Paket mit einem "PUBREC". Daraufhin erfolgt ein weiterer Handshake zwischen Client und Broker. Der Client beantwortet das "PUBREC" mit einem "PUBREL"-Paket. Der Broker komplettiert das zweifache Handshake mit einem "PUBCOM"-Paket.

Hinweis Nähere Informationen zu der Qualitätssicherung QoS finden Sie in [Kapitel 3.1.3](#).

MQTT-Ping-Mechanismus

Wenn die KeepAlive-Funktion aktiv ist (das KeepAlive-Intervall ist größer als "0"), muss der Client innerhalb des KeepAlive-Intervalls wenigstens eine Nachricht an den Broker schicken. Ist das nicht der Fall, muss der Broker die Verbindung zum Client beenden. Um solch einen Zwangsabbruch zu verhindern, muss der Client vor Ablauf der KeepAlive-Zeit ein Ping-Request an den Broker absetzen. Dafür dient das Kontrollpaket "PINGREQ". Der Broker antwortet mit einem "PINGRESP"-Paket und signalisiert dem Client damit seine Verfügbarkeit.

Hinweis Dieses Anwendungsbeispiel setzt eine aktive KeepAlive-Funktion voraus. Das KeepAlive-Intervall muss größer als zwei Sekunden eingestellt sein.

MQTT-Verbindungsabbau

Ein Client kann die Verbindung zu einem Broker schließen, indem er ein "DISCONNECT"-Paket an den Broker schickt. Der Broker löscht daraufhin alle "Last Will and Testament"-Informationen. Da der Client aktiv und aus freien Willen die Verbindung hat, verschickt der Broker nicht seinen letzten Willen an die registrierten Abonnenten.

3.2 Details zur Funktionsweise des FB "LMqtt_Publisher"

3.2.1 Voraussetzungen und Umsetzung

Für eine Kommunikationsbeziehung zwischen einem MQTT-Client und einem MQTT-Broker müssen folgende Voraussetzungen erfüllt sein:

1. Es wurde erfolgreich eine TCP-Verbindung zum MQTT-Broker aufgebaut (Status: "TCP_CONNECTED").
2. Der Funktionsbaustein "LMqtt_Publisher" hat sich über die bestehende TCP-Verbindung als MQTT-Client am Broker angemeldet und sich mit diesem verbunden (Status: "MQTT_CONNECTED").
3. Der Trigger zum Senden der Nachricht oder zum Erhalt der MQTT-Verbindung ("KeepAlive") ist aktiv. Abhängig von der gewünschten Qualitätssicherung wird die Nachricht an den Broker über die bestehende MQTT-Verbindung gesendet.

Hinweis

Ein MQTT-Verbindungsaufbau ist nur möglich, wenn die TCP-Verbindung zum Broker erfolgreich aufgebaut ist und anschließend gehalten wird.

Eine MQTT-Nachricht oder ein KeepAlive kann nur gesendet werden, wenn eine TCP- und MQTT-Verbindung zum Broker besteht.

Übersicht

Um die genannten Voraussetzungen zu erfüllen, wurden im Programm mehrere Zustandsautomaten realisiert:

- Zustandsautomat "TCP": Verwaltung der TCP-Verbindung
- Zustandsautomat "MQTT": Verwaltung der MQTT-Verbindung
- Zustandsautomat "PUSH": Abwicklung des Sendevorgangs

3.2.2 Zustandsautomat "TCP"

Der Zustandsautomat "TCP" wird gestartet, wenn am Eingangsparameter "enable" eine positive Flanke erkannt wurde. Dieser Zustandsautomat hat folgende Funktionen:

- Er regelt den Aufbau der TCP-Verbindung
- Er überwacht die bestehende TCP-Verbindung auf Verbindungsfehler, z. B. Kabelbruch
- Wenn ein Fehler aufgetreten ist oder am Eingangsparameter "enable" keine positive Flanke erkannt wurde, setzt er alle statischen Variablen und die anderen Zustandsautomaten in einen definierten Zustand.

Der Zustandsautomat "TCP" beinhaltet folgende Zustände:

- IDLE
- TCP_PARAM
- TCP_CONNECTING
- TCP_CONNECTED
- TCP_DISCONNECT
- TCP_ERROR

Die Bedeutung der Zustände listet die folgende Tabelle

Tabelle 3-1

Zustand	Beschreibung
IDLE	Im Ruhezustand "IDLE" werden alle Parameter zurückgesetzt. Der im Zustandsautomat wartet solange in diesem Zustand, bis er eine positive Flanke am Eingangsparameter "enable" erkennt. Sobald eine positive Flanke am Eingang anliegt, wird der Zustandsautomat in den Zustand "TCP_PARAM" versetzt.
TCP_PARAM	In diesem Zustand werden alle Verbindungsparameter eingelesen. Der Funktionsbaustein wechselt ohne Weiterschaltbedingung in den Zustand "TCP_CONNECTING".
TCP_CONNECTING	In diesem Zustand wird die TCP-Verbindung zum MQTT-Broker aufgebaut. Wenn die Verbindung mit "TCON" erfolgreich aufgebaut ist, wechselt der FB in den Zustand "TCP_CONNECTED" und die Ausgangsvariable "tcpConnected" wird gesetzt. Die TCP-Verbindung bleibt solange bestehen, bis sie mit "TDISCON" abgebaut wird. Wenn beim Verbindungsaufbau ein Fehler auftritt, wechselt der Zustandsautomat in den Zustand "TCP_ERROR".
TCP_CONNECTED	In diesem Zustand verwahrt der Funktionsbaustein solange bis folgende Ereignisse eintreten: <ul style="list-style-type: none"> Der Baustein "TRCV" erkennt ein Verbindungsabbruch, z. B. durch Ziehen des Netzkabels, und meldet einen Fehler. Der Eingangsparameter "enable" wird zurückgesetzt und stößt damit den Verbindungsabbau an. Wenn der "TRCV"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat in den Zustand "TCP_ERROR". Der Zustand "TCP_CONNECTED" ist Voraussetzung für die Abarbeitung des Zustandsautomaten "MQTT".
TCP_DISCONNECT	In diesem Zustand wird die TCP-Verbindung abgebaut. Wenn der "TDISCON"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat in den Zustand "TCP_ERROR".
TCP_ERROR	Wenn ein Fehler im Zustandsautomaten "TCP" eintritt, ist der Zustand "TCP_ERROR" die zentrale Anlaufstelle. Hier werden die erforderlichen Parameter (statische Variablen und Ausgangsvariablen) gesetzt bzw. zurückgesetzt und die MQTT-Verbindung abgebrochen. Zudem werden folgende Aktionen ausgeführt: <ul style="list-style-type: none"> Die Fehlermeldung des beteiligten T-Bausteins wird am Ausgang "status" übergeben. Am Ausgang "statusID" wird die Nummer des Zustands ausgegeben, in dem der Fehler aufgetreten ist Der Zustandsautomat kehrt zurück in den "IDLE"-Zustand. Wenn bereits eine TCP-Verbindung besteht, wird diese vorher abgebaut. Die Ausgangsvariable "tcpConnected" wird zurückgesetzt. Der Zustandsautomat "MQTT" wird in den Zustand "MQTT_DISCONNECTED" versetzt.

Hinweis

Der Funktionsbaustein "LMqtt_Publisher" ist im Fehlerfall nicht „selbstheilend“. Das bedeutet, dass der Funktionsbaustein zurück in den Zustand "IDLE" verfällt und solange dort verweilt, bis eine erneute positive Flanke am Eingangsparameter "enable" erkannt wurde.

3.2.3 Zustandsautomat "MQTT"

Der Zustandsautomat "MQTT" wird automatisch gestartet, wenn der Zustandsautomat "TCP" den Zustand "TCP_CONNECTED" erreicht. Dieser Zustandsautomat hat folgende Funktionen:

- Er regelt das Handshake-Verfahren zum Aufbau der MQTT-Verbindung
- Er sorgt für den Verbindungsabbau
- Er verwaltet den internen Zustandsautomaten "PUSH", um Nachrichten zu senden
- Er kümmert sich, dass vor Ablauf des KeepAlive-Intervalls ein PING-Paket verschickt wird.

Der Zustandsautomat "MQTT" beinhaltet folgende Zustände

- MQTT_DISCONNECTED
- MQTT_CONNECT_FLAG_CHECK
- MQTT_CONNECT
- MQTT_CONNACK
- MQTT_CONNECTED
- MQTT_DISCONNECT
- MQTT_ERROR

Die Bedeutung der Zustände listet die folgende Tabelle:

Tabelle 3-2

Zustand	Beschreibung
MQTT_DISCONNECTED	Solange keine TCP-Verbindung besteht, ist der Zustand immer "MQTT_DISCONNECTED". Erst wenn eine TCP-Verbindung aufgebaut ist, wird automatisch die Weberschaltbedingung auf den Zustand "MQTT_CONNECT_FLAG_CHECK" aktiviert.
MQTT_CONNECT_FLAG_CHECK	In diesem Zustand werden die Flags und Parameter für den MQTT-Verbindungsaufbau eingelesen und validiert. Wenn es bei der Überprüfung zu Unstimmigkeiten kommt, wechselt der Zustandsautomat in den Zustand "MQTT_ERROR" und es wird eine entsprechende Fehlermeldung am Ausgangsparameter "status" ausgegeben. Im fehlerfreien Zustand wechselt der Zustandsautomat ohne Weberschaltbedingung in den Zustand "MQTT_CONNECT".
MQTT_CONNECT	In diesem Zustand wird die MQTT-Verbindung zum MQTT-Broker aufgebaut. Dafür wird ein "CONNECT"-Paket mit den eingelesenen Parametern zusammengebaut und an den Broker mit dem "TSEND"-Baustein geschickt. Wenn beim Senden des "CONNECT"-Pakets ein Fehler auftritt, wechselt der Zustandsautomat in den Zustand "MQTT_ERROR". Wenn das "CONNECT"-Paket erfolgreich versendet wurde, wechselt der Zustandsautomat in den Zustand "MQTT_CONNACK".
MQTT_CONNACK	In diesem Zustand verwahrt der Zustandsautomat solange, bis der Baustein "TRCV" eine Nachricht empfängt. Es wird überprüft, ob es sich um ein "CONNACK"-Paket handelt. Wenn der Broker den Verbindungswunsch mit "CONNACK" bestätigt hat, wechselt der Zustandsautomat in den Zustand "MQTT_CONNECTED" und die Ausgangsvariable "mqttConnected" wird gesetzt. Das KeepAlive-Intervall wird bei Bedarf gestartet.

Zustand	Beschreibung
	Wenn der "TRCV"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat in den Zustand "MQTT_ERROR".
MQTT_CONNECTED	<p>In diesem Zustand verwahrt der Funktionsbaustein solange, bis die MQTT-Verbindung oder TCP-Verbindung abgebaut wird. In dem Zustand "MQTT_CONNECTED" werden folgende Punkte zyklisch geprüft:</p> <ul style="list-style-type: none"> • Liegt ein Sendeanstoß für eine MQTT-Nachricht vor? • Endet bald das KeepAlive-Intervall und ein PING-Befehl muss an den Broker geschickt werden? <p>Je nach Ausgang der Prüfung wird der interne Zustandsautomat "PUSH" in den entsprechenden Zustand versetzt, um die gewünschte Routine auszuführen.</p>
MQTT_DISCONNECT	<p>Wenn der Eingangsparameter "enable" zurückgesetzt wird, wird die MQTT-Verbindung abgebaut. Dafür wird ein "DISCONNECT"-Paket zusammengebaut und mit dem "TSEND"-Baustein an den Broker verschickt.</p> <p>Wenn beim Senden des "DISCONNECT"-Pakets ein Fehler auftritt, wechselt der Zustandsautomat in den Zustand "MQTT_ERROR".</p> <p>Wenn das "DISCONNECT"-Paket erfolgreich versendet wurde, wechselt der Zustandsautomat in den Zustand "MQTT_DISCONNECTED". Zeitgleich wird der Zustandsautomat "TCP" in den Zustand "TCP_DISCONNECT" versetzt. Damit wird auch die TCP-Verbindung beendet.</p>
MQTT_ERROR	<p>Wenn ein Fehler im Zustandsautomaten "MQTT" eintritt, ist der Zustand "MQTT_ERROR" die zentrale Anlaufstelle. Hier werden die erforderlichen Parameter (statische Variablen und Ausgangsvariablen) gesetzt bzw. zurückgesetzt. Zudem werden folgende Aktionen ausgeführt:</p> <ul style="list-style-type: none"> • Die Fehlermeldung des beteiligten MQTT-Befehls wird am Ausgang "status" übergeben. • Am Ausgang "statusID" wird die Nummer des Zustandsausgegeben, in dem der Fehler aufgetreten ist • Der Zustandsautomat kehrt zurück in den "MQTT_DISCONNECTED"-Zustand.

3.2.4 Zustandsautomat "PUSH"

Der Zustandsautomat "PUSH" wird nur durchlaufen, wenn sich der Zustandsautomat "MQTT" im Zustand "MQTT_CONNECTED" befindet. Denn hier wird entschieden, von welcher Stelle aus der Zustandsautomat "PUSH" gestartet wird. Wenn ein Sendeanstoß für eine MQTT-Nachricht vorliegt, dann wird die Senderoutine aktiv. Wenn die KeepAlive-Zeit in Kürze endet, wird die PING-Routine gestartet.

Der Zustandsautomat "PUSH" beinhaltet folgende Zustände:

- IDLE
- MQTT_PUBLISH
- MQTT_PUBACK
- MQTT_PUBREL
- MQTT_PUBCOMP
- MQTT_PING
- MQTT_PINGRESP

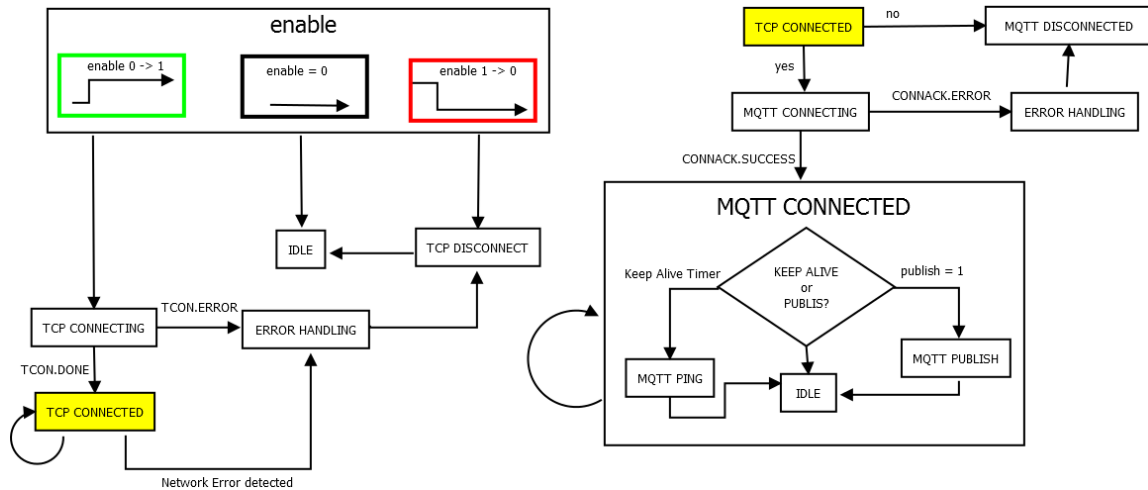
Zustand	Beschreibung
IDLE	Solange kein Sendeanstoß vorliegt oder das KeepAlive-Intervall nicht abläuft, ist der Zustand immer "IDLE".
MQTT_PUBLISH	<p>Wenn im Zustand "MQTT_CONNECTED" am Eingangsparameter "publish" eine positive Flanke erkannt wurde, wird der interne Zustandsautomat "PUSH" in den Zustand "MQTT_PUBLISH" versetzt. Hier startet die Senderoutine in Abhängigkeit von der Qualitätssicherung QoS.</p> <p>Zuerst wird ein "PUBLISH"-Paket mit den vorgegebenen Parametern, dem Topic und dem Nachrichtentext zusammengebaut und anschließend mit dem "TSEND"-Baustein an den Broker geschickt.</p> <p>Wenn beim Senden des "PUBLISH"-Pakets ein Fehler auftritt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und der Zustandsautomat geht zurück in "IDLE".</p> <p>Wenn das "PUBLISH"-Paket erfolgreich versendet wurde, ist der nächste Schritt von dem gewählten QoS abhängig:</p> <ul style="list-style-type: none"> • Bei QoS gleich "0" endet an dieser Stelle der Sendevorgang und dieser Zustandsautomat geht zurück in "IDLE". Das KeepAlive-Intervall wird bei Bedarf neu gestartet. • Bei QoS gleich "1" und QoS gleich "2" wechselt dieser Zustandsautomat in den Zustand "MQTT_PUBACK", um eine Quittierung vom Broker zu erhalten.
MQTT_PUBACK	<p>Wenn der QoS größer "0" ist, erwartet der Client eine Quittierung des Brokers auf das "PUBLISH"-Paket.</p> <p>In diesem Zustand verwahrt dieser Zustandsautomat solange, bis der Baustein "TRCV" eine Nachricht empfängt. Es wird überprüft, ob es sich um ein "PUBACK"-Paket handelt.</p> <p>Wenn der Broker den Erhalt der Nachricht bestätigt hat, ist der nächste Schritt von dem gewählten QoS abhängig:</p> <ul style="list-style-type: none"> • Bei QoS gleich "1" endet an dieser Stelle der Sendevorgang und dieser Zustandsautomat geht zurück in "IDLE". Das KeepAlive-Intervall wird bei Bedarf neu gestartet. • Bei QoS gleich "2" wechselt dieser Zustandsautomat in den Zustand "MQTT_PUBREL", um den Quittierungseingang zu bestätigen. <p>Wenn der "TRCV"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und</p>

Zustand	Beschreibung
	der Zustandsautomat geht zurück in "IDLE".
MQTT_PUBREL	<p>Bei QoS gleich "2" erfolgt ein zweifacher Handshake mit dem Broker.</p> <p>Nachdem der Client das "PUBACK" erhalten hat, wird es durch das "PUBREL"-Paket bestätigt. Dafür wird ein "PUBREL"-Paket zusammengebaut und anschließend mit dem "TSEND"-Baustein an den Broker geschickt. Wenn beim Senden des "PUBREL"-Pakets ein Fehler auftritt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und der Zustandsautomat geht zurück in "IDLE".</p> <p>Wenn das "PUBREL"-Paket erfolgreich versendet wurde, wechselt der Zustandsautomat in den Zustand "PUBCOMP".</p>
MQTT_PUBCOMP	<p>Dieser Zustand ist der letzte Teil des zweifachen Handshakes-Verfahren bei QoS gleich "2". Der Client erwartet eine Quittierung des Brokers auf das "PUBREL"-Paket.</p> <p>In diesem Zustand verwahrt dieser Zustandsautomat solange, bis der Baustein "TRCV" eine Nachricht empfängt. Es wird überprüft, ob es sich um ein "PUBCOMP"-Paket handelt.</p> <p>Wenn der Broker den Erhalt der Nachricht bestätigt hat, wechselt dieser Zustandsautomat zurück in "IDLE" und das KeepAlive-Intervall wird bei Bedarf neu gestartet. Das Handshake-Verfahren ist damit abgeschlossen.</p> <p>Wenn der "TRCV"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und dieser Zustandsautomat geht zurück in "IDLE".</p>
MQTT_PING	<p>Wenn im Zustand "MQTT_CONNECTED" festgestellt wird, dass das KeepAlive-Intervall abläuft, wird der interne Zustandsautomat in den Zustand "MQTT_PING" versetzt. Hier startet die Pingroutine.</p> <p>Zuerst wird ein "PING"-Paket zusammengebaut und anschließend mit dem "TSEND"-Baustein an den Broker geschickt.</p> <p>Wenn beim Senden des "PING"-Pakets ein Fehler auftritt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und dieser Zustandsautomat geht zurück in "IDLE".</p>
MQTT_PINGRESP	<p>Nach dem "PING"-Paket erwartet der Client eine Quittierung des Brokers.</p> <p>In diesem Zustand verwahrt der Zustandsautomat solange, bis der Baustein "TRCV" eine Nachricht empfängt. Es wird überprüft, ob es sich um ein "PINGRESP"-Paket handelt.</p> <p>Wenn der Broker den Erhalt der Nachricht bestätigt hat, geht der Zustandsautomat zurück in "IDLE". Das KeepAlive-Intervall wird neu gestartet.</p> <p>Wenn der "TRCV"-Baustein einen Fehler erkennt, wechselt der Zustandsautomat "MQTT" in den Zustand "MQTT_ERROR" und dieser Zustandsautomat geht zurück in "IDLE".</p>

3.2.5 Funktionsdiagramm

Folgende Abbildung zeigt das Diagramm der Funktionsweise mit den drei Zustandsautomaten:

Abbildung 3-2



4 Anhang

4.1 Service und Support

Industry Online Support

Sie haben Fragen oder brauchen Unterstützung?

Über den Industry Online Support greifen Sie rund um die Uhr auf das gesamte Service und Support Know-how sowie auf unsere Dienstleistungen zu.

Der Industry Online Support ist die zentrale Adresse für Informationen zu unseren Produkten, Lösungen und Services.

Produktinformationen, Handbücher, Downloads, FAQs und Anwendungsbeispiele – alle Informationen sind mit wenigen Mausklicks erreichbar:

<https://support.industry.siemens.com>

Technical Support

Der Technical Support von Siemens Industry unterstützt Sie schnell und kompetent bei allen technischen Anfragen mit einer Vielzahl maßgeschneiderter Angebote – von der Basisunterstützung bis hin zu individuellen Supportverträgen.

Anfragen an den Technical Support stellen Sie per Web-Formular:

www.siemens.de/industry/supportrequest

SITRAIN – Training for Industry

Mit unseren weltweit verfügbaren Trainings für unsere Produkte und Lösungen unterstützen wir Sie mit innovativen Lernmethoden.

Mehr zu den angebotenen Trainings und Kursen sowie deren Standorte und Termine erfahren Sie unter:

www.siemens.de/sitrain

Serviceangebot

Unser Serviceangebot umfasst folgendes:

- Plant Data Services
- Ersatzteilservices
- Reparaturservices
- Vor-Ort und Instandhaltungsservices
- Retrofit- und Modernisierungsservices
- Serviceprogramme und Verträge

Ausführliche Informationen zu unserem Serviceangebot finden Sie im Servicekatalog:

<https://support.industry.siemens.com/cs/sc>

Industry Online Support App

Mit der App "Siemens Industry Online Support" erhalten Sie auch unterwegs die optimale Unterstützung. Die App ist für Apple iOS, Android und Windows Phone verfügbar:

<https://support.industry.siemens.com/cs/ww/de/sc/2067>

4.2 Links und Literatur

Tabelle 4-1

Nr.	Thema
\1\	Siemens Industry Online Support https://support.industry.siemens.com
\2\	Link auf die Beitragsseite des Anwendungsbeispiels https://support.industry.siemens.com/cs/ww/de/view/109748872
\3\	MQTT Spezifikation http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
\4\	Informationen zur „local_device_id“ https://support.industry.siemens.com/cs/ww/de/view/51339682

4.3 Änderungsdokumentation

Tabelle 4-2

Version	Datum	Änderung
V1.0	03/2018	Erste Ausgabe
V1.1	08/2018	Bibliothek "LMqttQdn" für SIMATIC S7-1500