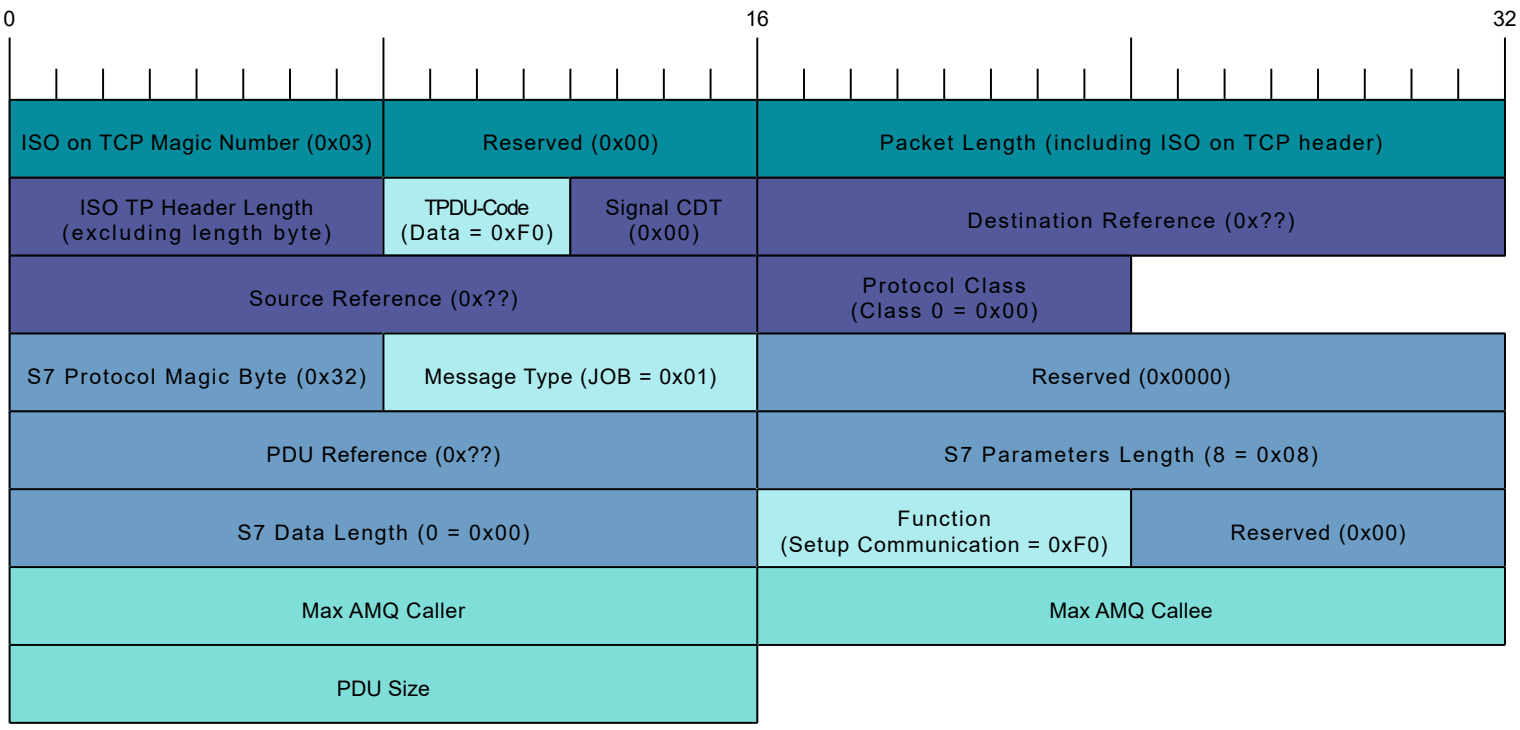
S7 Comm (0x32)

General

While a lot of information was available on the general structure of S7 communication, only little information was available on the constant values this protocol uses. If information was available, this was mostly provided with a GPL license and therefore was disqualified for being used in this project. The information on the S7 constants in this project were therefore generated by a little tool that generates "pcap" files **WireShark** can process. The tool then generated 256 versions of a given template with the only difference being the one byte having all possible values. Using the **tshark** commandline tool, the generated packets were decoded to an XML format. For each examined byte an XPath expression was created to detect valid values. As soon as a valid value was found the tool then output the detected constant value to the console.

The tool for generating this is located in the **plc4j/protocols/s7-utils** project.

Structure of a Setup Communication Request



Legend:

- ISO on TCP Packet Header
- ISO Transport Protocol Packet Header
- S7 Protocol
- Part of the packet that identifies the type of request
- Variable Parts of the ISO Transport Protocol Packet Header

Structure of a Setup Communication Response

The **Setup Communication Response** is identical to the **Setup Communication Request** with the only difference that the **Message Type** has an ACK_DATA code of **0x03**.

Also does the response eventually provide different values for **Max AMQ Caller**, **Max AMQ Callee** and **PDU Size**.

The values might be lower than in the request, but never higher.

> **TIP** One thing about **Setup Communication Responses** which is kind of strange, is that usually S7 response messages have additional **error class** and **error code** fields, which this type of response doesn't seem to have.

Sizes of requests

During the connection to a S7 PLC the client and PLC agree on 3 important parameters:

- PDU Size
- Max AMQ Caller
- Max AMQ Callee

The PDU Size is the size of a data packet the PLC is willing to accept. Here note, that in reality there are two PDU sizes involved: On **ISO TP/COTP** protocol level (TPDU Size) and a second time on the **S7** protocol level (PDU Size). Most implementations treat them as somewhat equal, but this doesn't have to be the case. A PLC could accept a higher PDU size on **ISO TP** level than on **S7** level.

The **Max AMQ** parameters define how many unacknowledged requests a PLC (Callee) is able to accept from a client (Caller).

If the **ISO TP** TPDU size is bigger than the **S7** PDU size, then theoretically multiple **S7** PDUs can be contained in one **ISO TP** packet. But at max **Max AMQ** packets. Most drivers don't utilize this option however (We won't either).

When issuing a read request there are other things that have to be taken into account:

If on **ISO TP** level a max PDU size has been agreed on, the max PDU Size of the **S7** packet will be smaller. So if a TPDU size of 256 bytes has been agreed upon, then the ISO TP header takes 3 bytes and the header of the S7 packet takes 10 bytes. The header of a **Read Var** parameter takes 2 bytes. So if the TPDU size is 256 bytes, this leaves 256 - (3 + 10 + 2) bytes = 241 bytes

An **S7ANY** type variable specification takes 12 bytes, so the maximum number of memory areas that can be accessed in one S7 request packet is: 241 / 12 = 20 That's also the reason why most available drivers limit the number of addresses to 20 (Some go down to 18 because some devices seem to calculate the boundaries differently).

To make things even more complicated, we have to ensure the data retrieved by a read request fits into a PDU. So if a block of data requested in a read request would would exceed the agreed upon PDU size, the PLC will respond with an **Access Violation** and not return any data. So the size limit for reading data with a 256 byte PDU size would be 241 - 12 = 229 bytes.

Another example would be if you read different memory blocks, each smaller than the max PDU size, the read PDU could be quite small, but if these blocks are quite large, the read response could exceed the max PDU size. In this case the item responses that exceed the size will simply be responded with an **Access violation** by the PLC.

The typical **Max AMQ** of **8** further complicates things as we can't simply split up the one message into multiple ones and blindly fire them at the PLC. If the number of messages exceeds this number, we have to queue the excess PDUs and wait till the PLC confirmed some and can then continue sending further fragments.

Most commercial drivers reduce the stress by rearranging and changing the requests. If for example a request would read 8 consecutive bits, it would automatically change the 8 items into reading one byte. Also if the gap between addresses is smaller than the overhead for a new definition, it would extend the first to include the second memory area. Same should sometimes be possible in the other direction. So if we wanted to read 300 bytes with a 256 byte TPDU size, the driver would split this up into two smaller byte arrays and internally merge them. However when reading Real (Number) values there could be issues with this. However in this case the driver has to completely take care of this optimization.

Links

Providing some additional information without directly being used:

- High Level description: http://snap7.sourceforge.net/siemens_comm.html (http://snap7.sourceforge.net/siemens_comm.html)
- https://support.industry.siemens.com/cs/document/26483647/welche-eigenschaften-vorteile-und-besonderheiten-bietet-das-s7-protokoll-?dti=0&lc=de-WW (https://support.industry.siemens.com/cs/document/26483647/welche-eigenschaften-vorteile-und-besonderheiten-bietet-das-s7-protokoll-?dti=0&lc=de-WW)
- Interesting presentation mentioning a new protocol flavor 0x72 instead of the old 0x32: https://www.research.ibm.com/haifa/Workshops/security2014/present/Avishai_Wool_AccurateModelingoftheSiemensS7SCADAProtocol-v5.pdf (https://www.research.ibm.com/haifa/Workshops/security2014/present/Avishai_Wool_AccurateModelingoftheSiemensS7SCADAProtocol-v5.pdf)
- Open Source SCADA System: https://www.eclipse.org/eclipsescada/ (https://www.eclipse.org/eclipsescada/)