



Dokumentation Schwabenplan Protokoll Version 4.0

1. Einleitung

Das Schwabenplan MC-Protokoll wurde entwickelt, um einfache Steuerungsbefehle von einer Software oder einen Controller zu einem Busteilnehmer zu übermitteln. Das Protokoll basiert auf einem String und ist über ein Terminalprogramm einfach zu lesen. Die Länge des Protokolls ist fest vorgegeben.

Die Programmiersprache für die Anwendung muss nicht zwingend beachtet werden. Wir verwenden üblicherweise Visual Basic .Net und bieten das Protokoll auch in unserer eigenen DLL-Library an. Somit können Sie eine beliebige Einwicklungsplattform für Ihre Anwendung einsetzen.

Unsere Hauptplatinen können über eine RS-232, RS-485 oder eine CAN-Schnittstelle angebunden werden. Wir favorisieren die RS-485 Schnittstelle, da in diesem Bus mehrere Teilnehmer angebunden werden können. Die Konfiguration Schnittstelle ist frei definierbar. Unsere Programmbeispiele basieren auf folgenden Einstellungen:

Baud-Rate: 9600 / 19200 / 57600 / 115200 Baud

Datenbits: 8

Stopbits: 1

Parität: Keine

Intervallzeit: 80-150ms

Schnittstellen: RS-232 und RS-485 (RS-485 wird von unserer Seite empfohlen)

Das Schwabenplan Protokoll kann für eine einzelne Datenübertragung und im Polling-Modus betrieben werden.

Wichtiger Hinweis:

Die Intervallzeit für die permanente Datenabfrage hängt von verschiedenen Faktoren ab.

Diese sind:

- a) Der Taktgeschwindigkeit des Mikrocontrollers. -> Programmdurchlaufzeit**
- b) Der angeschlossenen Hardware z.B. mit oder ohne LC-Display. -> Programmdurchlaufzeit**
- c) Den Parameter der verwendeten Schnittstellenbausteine (vom Sender und Empfänger).**
- d) Der Verkabelung der Hardware selbst Länge, Abschlusswiderstände, Qualität der Kabel usw.**



1.1 Integration

Für die Integration dieses Übertragungsprotokolls stellen wir Ihnen sehr gerne unser Demo-Projekt „MC Agent“ zur Verfügung. Der MC Agent wurde mit Visual Basic .Net programmiert. Für den Microcontroller haben wir Beispiele in der Programmiersprache C für den PIC18F46K80 sowie den PIC18F97J60. Sprechen Sie uns im Bedarfsfall einfach an. Es müssen nicht alle Befehle in Ihre Anwendung integriert werden. Wir wollen Ihnen Anhand dieser umfangreichen Protokollbeschreibung die Möglichkeiten für Ihre tägliche Arbeit aufzeigen. Weiterentwicklungen wird es auch in Zukunft geben. Daher behalten wir uns Änderungen vor.

In unseren Projekten haben wir dieses Protokoll als Betriebssystem mit zwei Interrupts integriert. Diese Interrupts werden über die interne Prioritätsverwaltung gesteuert. Der seriellen Schnittstelle Nr. 1 ist die High-Priorität zugeordnet. So wird jede Abfrage oder Steuerungsbefehl verarbeitet. Dem Timer Nr. 0 ist die Low-Priorität zugeordnet, und erzeugt jede Sekunde ein Taktsignal. Dieses Taktsignal wird für die Systemuhr und den Temperatursensor verwendet. So können die Uhrzeit und das Datum unabhängig vom restlichen Programmablauf angezeigt oder geschrieben werden. Tritt ein zeitgleiches Ereignis ein, so werden zuerst die Daten über die serielle Schnittstelle verarbeitet. Im Nachgang werden die Daten für die Echtzeituhr und die Temperaturwerte auf der Platine aktualisiert. Die nachfolgende Tabelle zeigt die einzelnen Einstellungen der Interrupt-Steuerung auf.

Interrupt Bezeichnung	Interrupt Priorität	Interrupt Bit	Interrupt Flag	Parameter	Info
Allgemein	Prioritätsverwaltung	RCONbits.IPEN	-	1	-
Global Interrupt High (Hauptschalter High-Priorität)	INTCONbits.GIEH	INTCONbits.GIEH	-	0 oder 1	-
Global Interrupt Low (Hauptschalter Low-Priorität)	INTCONbits.GIEL	INTCONbits.GIEL	-	0 oder 1	-
USART Nr. 1 Datenempfang	High-Priorität	PIE1bits.RC1IE	PIR1bits.RC1IF	0 oder 1	-
Timer Nr. 0	Low-Priorität	INTCONbits.TMR0IE	INTCONbits.TMR0IF	0 oder 1	Takt 1 Hz.

- a) Eine CRC-Prüfung gibt es in diesem Protokoll nicht.
- b) Eine korrekt empfangene Nachricht wird unmittelbar mit einer Antwort (Quittierung) bestätigt.
- c) Erhält der Sender keine Quittierung so hat der Empfänger die Nachricht fehlerhaft oder nicht erhalten.
- d) In jedem Fall sind entsprechende Timeout-Zeiten integrierbar.
- e) Jede Nachricht in diesem Protokoll ist immer gleich groß.
- f) Eine dynamische Anpassung der zu reservierenden Datenfelder ist nicht notwendig.



2. Allgemeines

Das Protokoll ist grundsätzlich in drei Kategorien aufgeteilt. Es gibt:

✓ Status-Befehle

Über die Status-Befehle kann die Peripherie der gesamten Hard- und Firmware abgefragt werden.

✓ PORT-Befehle

Mit diesen Befehlen kann jeder beliebige PORT-Zustand eingelesen oder gesteuert werden.

✓ I²C-Bus bzw. SPI-Bus Befehle

Über diese Befehle können eine Vielzahl von Bausteinen direkt abgefragt und gesteuert werden.

Wichtiger Hinweis:

Es ist zu empfehlen das die empfangenen Daten als globale Variablen zu deklarieren oder mit Pointen zu arbeiten.



3. Befehlsaufbau

Die Befehle sind sehr einfach gehalten und bestehen immer aus 39 Zeichen. Es gibt in diesem Protokoll Lese- und Schreibbefehle. Diese Befehle sind immer gleich formatiert. Sie beginnen mit einem Startzeichen (Header „S“ oder „R“) und werden durch einen „:“ voneinander getrennt. Zum Abschluss von jedem Befehl folgen die Steuerzeichen „\r\n“, welche „Carrige Return“, 0x0D und dem Befehl „Newline“, 0x0A bedeutet.

Die Sende- und Empfangsbefehle werden zur vereinfachten Lesbarkeit mit einem „S“ für **Send** bzw. mit einem „R“ für **Receive** für Empfangen gegenzeichnet.

So können Sie direkt erkennen, ob es sich bei diesem Befehl um einen Sende- oder Rückgabebefehl handelt. Im Schwabenplan-Protokoll der Version 4.0 ist das Trennzeichen der unterschiedlichen Felder als „:“ ausgeführt. Alle zu übertragenden Zahlenwerte sind immer zweistelligen hexadezimalen Werten zu übertragen.

3.1 Protokoll Header

Bezeichnung	ASCII Wert	Dez. Wert	Hex. Wert
Senden	S	83	0x53
Empfangen	R	82	0x52
Trennzeichen	:	58	0x3A
Endzeichen Nr. 1	\r	13	0x0D
Endzeichen Nr. 2	\n	10	0x0A

3.2 Read / Write (R/W) Umschaltung

Im letzten Block des übertragenen Protokoll-Strings ist definiert, ob es sich um einen Lese- oder einen Schreibbefehl handelt.

Befehlsart	Hex. Wert
Schreiben	0x00
Lesen	0x01



Beispiele:

Sendebefehl

S : Adresse des Empfängers: Adresse des Senders : Befehl : P0 : P1 : P2 : P3 : P4 : P5 : P6 : P7 : RW \r \n

Antwort des Empfängers

R : Adresse des Empfängers: Adresse des Senders : Befehl : P0 : P1 : P2 : P3 : P4 : P5 : P6 : P7 : RW \r \n

Hier ein exemplarisches Beispiel mit dem Befehl 0x20 zum Lesen:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
S	:	0	1	:	0	0	:	2	C	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	\r	\n
53	3A	30	31	3A	30	30	3A	32	43	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A
083	058	048	049	058	048	048	058	050	067	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	049	013	010
R	:	0	0	:	0	1	:	2	C	:	0	1	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	\r	\n			
52	3A	30	30	3A	30	31	3A	32	43	3A	30	31	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A
082	058	048	048	058	048	049	058	050	067	058	048	049	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	048	058	048	049	013	010

S:01:00:20:P0:P1:P2:P3:P4:P5:P6:P7:01:\r\n

// Der Server mit der Adresse 0 sendet einen Status-Befehl zum Lesen an den Teilnehmer mit der Adresse 1

R:00:01:20:P0:P1:P2:P3:P4:P5:P6:P7:01:\r\n

//Die Adresse 1 antwortet dem Server mit einem führenden „R:“ und die entsprechende Adresse 0

Wichtiger Hinweis:

Bitte achten Sie darauf, dass alle hexadezimalen Werte zweistellige Längen angegeben werden müssen. Sollten Sie über unsere DLL-Datei arbeiten, wird dies automatisch angepasst.

Die Bearbeitungszeit für einen Befehl ist von verschiedenen Faktoren wie Microcontroller, Taktgeschwindigkeit, Speicher oder Baudrate abhängig. Diese Intervallzeit kann mit Hilfe von unserem Software-Tool beliebig eingestellt werden.



4. Allgemeine Befehlsübersicht

4.1 Status Befehle

Befehl	Dez	Beschreibung
0x20	32	MC-Typ
0x21	33	Hardware Version
0x22	34	Taktfrequenz
0x23	35	Firmware Version
0x24	36	Protokoll Version
0x25	37	RS-485 Adresse
0x26	38	Bootloader Modus
0x27	39	Debug Modus
0x28	40	Interface on Board
0x29	41	Baudrate
0x2A	42	I ² C-Geschwindigkeit
0x2B	43	RTC-Baustein
0x2C	44	Temperatur-Baustein
0x2D	45	Board-Edition

4.2 Netzwerk-Befehle

Befehl	Dez	Beschreibung
0x30	48	IP-Adresse
0x31	49	Subnet Maske
0x32	50	Router
0x33	51	Gateway
0x34	52	DNS-Server
0x35	53	Server IP
0x36	54	Kommunikations-Port
0x37	55	MAC-Adresse



4.3 I²C-Bus-Befehle

Befehl	Dez	Beschreibung
0x40	64	LM75 Temperatur Lesen
0x41	65	DS1621 Temperatur Lesen
0x43	66	MCP23008
0x44	67	MCP23017
0x45	68	PCF8574
0x46	69	PCF8591
0x50	80	Uhrzeit
0x51	81	Datum und Wochentag
0x52	82	Stunden
0x53	83	Minuten
0x54	84	Sekunden
0x55	85	Tag
0x56	86	Monat
0x57	87	Jahr
0x58	88	Wochentag
0xAD	173	A/D-Wandler
0xDD	221	Digitalbefehl
0xFF	255	Reset



5. Status-Befehle

Inzwischen haben die meisten Microcontroller einen großen Speicher. Daher werden in Ihnen zum Teil einige Zusatzinformationen wie Microcontroller-Typ, Hardware-Version usw. hinterlegt.

Die „Status-Befehle“ von dem Schwabenplan Protokoll umfassen Befehle, die Ihnen Auskünfte zu dem Microcontroller und der jeweiligen Hardware und Peripherie beinhalten.

Hierzu werden die Befehlsparameter P0 und P1 genutzt. Für die Abfrage können die Parameterfelder P0 bis P7 mit einem beliebigen Wert befüllt werden. Wir verwenden hier im Normalfall immer 0x00.

Befehl	Beschreibung	P0	P1
0x20	MC-Typ	Integer	
0x21	Hardware-Version	Integer	Integer (0x2D)
0x22	Taktfrequenz	Integer	
0x23	Firmware Version	Integer	
0x24	Protokoll Version	Integer	
0x25	RS-485 Adresse	Integer	
0x26	Bootloader Modus	Integer	
0x27	Debug Modus	Integer	
0x28	Interface on Board	Integer	Integer (0x2A)
0x29	Baudrate	Integer	
0x2A	I ² C-Geschwindigkeit	Integer	
0x2B	RTC-Baustein	Integer	
0x2C	Temperatur-Baustein	Integer	
0x2D	Board-Edition	Integer	

Wichtiger Hinweis:

Einige diese Paramater sollten lediglich als „ein reiner Lesebefehl“ in Ihr Projekt integriert werden.



5.2.1 Befehl 0x20 -> Microcontroller-Typen

Mit diesem Befehl wird der Microcontroller-Typ der auf der Leiterplatte abgefragt. Dies ist ein Befehl, der nur als ein reiner Lesebefehl in Ihr Projekt integriert werden sollte.

Dieser Befehl liefert nur an dem Parameter Nr. 0 einen Rückgabewert. Diesen finden Sie in der nachfolgenden Tabelle.

Wert Hex.	Microcontroller-Typ	Bezeichnung
0x00	unbekannt	
0x01	PIC18F97J60	aktuelle Generation
0x02	PIC18F46K80	aktuelle Generation
0x03	PIC18F26K80	aktuelle Generation
0x04	PIC18F66K80	aktuelle Generation
0x05	PIC18F67K40	aktuelle Generation
0x0A	PIC18F25K50	USB-Typen
0x0B	PIC18F45K50	USB-Typen
0x0C	PIC18F2550	USB-Typen
0x0D	PIC18F4550	USB-Typen
0x0E	PIC18F67J50	USB-Typen
0x0F	PIC18F87J50	USB-Typen
0x14	PIC18F06Q41	neue Generation
0x15	PIC18F16FQ41	neue Generation
0x16	PIC18F27Q43	neue Generation
0x17	PIC18F47Q43	neue Generation
0x18	PIC18F57Q43	neue Generation
0x19	PIC18F27Q84	neue Generation
0x1A	PIC18F47Q84	neue Generation
0x1B	PIC18F57Q84	neue Generation
0x1E	PIC18F67K22	RTCC-Typen
0x1F	PIC18F87K22	RTCC-Typen
0x20	PIC18F67J94	RTCC-Typen
0x21	PIC18F87J94	RTCC-Typen
0x22	PIC18F97J94	RTCC-Typen
0x5A	PIC16F877A	alte Generation
0x5B	PIC18F452	alte Generation
0x5C	AT89C51CC03	alte Generation
0x5D	PIC16F18313	neue Generation
0x5E	PIC32MX795F512L	neue Generation



5.2.2 Befehl 0x21 und 0x2D -> Hardware Version

Mit diesem Befehl wird der Microcontroller-Typ der auf der Leiterplatte abgefragt. Dies ist ein Befehl, der nur als ein reiner Lesebefehl in Ihr Projekt integriert werden sollte. Änderungen ergeben sich in aller Regel nicht.

Dieser Befehl liefert zwei Rückgabewerte, als Parameter Nr. 0 und Nr. 1.

Die Versionsnummer 1.0 wird als dezimal 10 oder 0x0A in hexadezimal dargestellt. Mit der Erweiterung des Schwabenplan Protokolls auf die Version 4.0 wurde der Befehl [0x2D] in den Befehl [0x21] mit integriert. Der Befehl [0x2D] existiert als eigenständiger Befehl weiterhin, und antwortet dann als Parameter Nr. 0. In diesem Datenbyte ist die Hardwareausführung beschrieben. Unsere Hauptplatinen sind in verschiedenen Ausbaustufen erhältlich. Daher ist diese zusätzliche Unterscheidung möglich.

Diesen finden Sie in den nachfolgenden Tabellen. Hier ein Beispiel einer Hauptplatine der Version 2.0 in der Enterprise Ausführung.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
S	:	0	1	:	0	0	:	2	1	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v ₁	v ₁
53	3A	30	31	3A	30	30	3A	32	31	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A	v ₁		
R	:	0	0	:	0	1	:	2	1	:	1	4	:	4	5	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v ₁	v ₁
52	3A	30	30	3A	30	31	3A	32	31	3A	31	34	3A	34	35	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A	v ₁		

Befehl	P0 in Hex.	P1 in Hex.
0x21	0x14	0x45

ASCII-Wert	Dez. Wert	Hex. Wert	Bedeutung
B	66	0x42	Basic-Ausführung
P	80	0x50	Professional Ausführung
E	69	0x45	Enterprise Ausführung



5.2.3 Befehle 0x22 bis 0x25

Diese Befehle werden mit dem Parameter Nr. 0 in einer Ganzzahl (Integer) im hexadezimalen Zahlenformat beantwortet.

Befehl	Beschreibung	P0 in Dez.	Po in Hex.
0x22	Taktfrequenz	25	0x19
0x23	Firmware Version	10	0x0A
0x24	Protokoll Version	40	0x28
0x25	RS-485 Adresse	1	0x01

Für eine Taktfrequenz, Befehl [0x22] von 25 MHz wird der hexadezimale Wert 0x19 gesetzt. Die Taktfrequenz wird in aller Regel nicht vom Anwender verändert. Technisch gesehen, aber möglich. Daher sollten Sie entscheiden, wie Sie diesen Befehl in Ihre Applikation implementieren.

Für den Befehl [0x23] Firmware Version und dem Befehl [0x24] Protokoll Version wird der zu speichernde Wert mit einer durch 10 teilbaren Zahl befüllt. Für die Firmware Version 1.0 gilt hier der der dezimale Wert 10 bzw. hexadezimale Wert 0x0A. Diese Werte sollten Sie lesend und schreibend in Ihre Anwendung implementieren, damit die immer über den aktuellen Stand informiert werden können.

Hier ein Beispiel für den Befehle Firmware Version [0x23]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
S	:	0	1	:	0	0	:	2	3	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v	v
53	3A	30	31	3A	30	30	3A	32	33	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A
R	:	0	0	:	0	1	:	2	3	:	0	A	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v	v
52	3A	30	30	3A	30	31	3A	32	33	3A	30	41	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A

Für die Version 4.0 gilt in dezimal die Zahl 40 oder bzw. in Hex 0x28.

Hier ein Beispiel für den Befehle Protokoll Version [0x24]:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
S	:	0	1	:	0	0	:	2	4	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v	v
53	3A	30	31	3A	30	30	3A	32	34	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A
R	:	0	0	:	0	1	:	2	4	:	2	8	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	v	v
52	3A	30	30	3A	30	31	3A	32	34	3A	32	38	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A



Der Befehl [0x25] zum Setzen oder Lesen einer Bus Adresse z.B. für einen RS-485 Bus kann z.B. über DIP-Schalter per Hardware aber auch per Software gesetzt werden. Dafür dient dieser Befehl. So könne Sie Pins an einem Microcontroller einsparen und diese Funktion per Software realisieren.

Hier ein Beispiel für den Befehle [0x25] RS-485 Adresse Lesen und Schreiben:

Lesen:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
S	:	0	1	:	0	0	:	2	5	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	vr	vi
53	3A	30	31	3A	30	30	3A	32	35	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A
R	:	0	0	:	0	1	:	2	5	:	0	1	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	1	vr	vi
52	3A	30	30	3A	30	31	3A	32	35	3A	30	31	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	31	0D	0A

Schreiben Adresse 1 bleibt Adresse 1:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39			
S	:	0	1	:	0	0	:	2	5	:	0	1	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	vr	vi
53	3A	30	31	3A	30	30	3A	32	35	3A	30	31	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	0D	0A
R	:	0	0	:	0	1	:	2	5	:	0	1	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	:	0	0	vr	vi
52	3A	30	30	3A	30	31	3A	32	35	3A	30	31	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	3A	30	30	0D	0A

Das Schreiben einer neuen RS-485 Adresse bringt eine Besonderheit mit sich. Formell gesehen beantwortet der Microcontroller seinem Bus-Master zuerst den Befehl auf der bisherigen Teilnehmer-Adresse. Danach speichert der Controller die neue Adresse ab und liest diese ein. Je nachdem wie Schnell Ihr Controller, der Speicher, die Schnittstelle usw. ist, kann es sein, dass Ihr Controller eine zweite Bestätigung mit der neuen Teilnehmeradresse an den Bus-Master sendet.

5.2.4 Befehle 0x26 und 0x27

Diese Befehle dienen zur Steuerung eines Bootloaders oder einem zusätzlichen Debug Modus ohne Programmiergerät. Diese können im Bedarfsfall über Ihre Software direkt gesteuert werden. Der Wert 0x00 steht für Off und der Wert 0x01 steht für On.

Befehl	Beschreibung	P0 in Hex.
0x26	Bootloader Modus	0x00 oder 0x01
0x27	Debug Modus	0x00 oder 0x01



5.2.5 Befehle 0x28, 0x27 und 0x02A

Diese Befehle dienen zur Steuerung eines zusätzlichen Bussystems z.B. I²C- oder SPI-Bus. Der Befehl [0x2A] wurde im Zuge der Erweiterung im Befehl [0x28] integriert. Der Befehl [0x2A] steht als Einzelabfrage aber weiterhin zur Verfügung. Die Inhalte werden als Ganzzahl im zweistelligen Hex-Format übertragen.

Befehl	Beschreibung	P0 in Hex.	P1 in Hex.
0x28	Interface on Board	Integer	Integer (0x2A)
0x29	Baudrate	Integer	
0x2A	I ² C-Geschwindigkeit	Integer	

Dateninhalte Befehl 0x28 Interface on Board:

Wert	Interface-Typ
0x00	Off
0x01	I ² C Bus
0x02	SPI Bus

Dateninhalte Befehl 0x29 verfügbare Baudraten:

Eine RS-232 bzw. RS-485 Schnittstelle ist immer auf unseren Hauptplatinen verbaut.

Wert	Baudrate
0x00	9600
0x01	19200
0x02	57600
0x03	115200

Dieser Befehl hat ebenfalls eine Besonderheit. Wird die Baudrate im laufenden Betrieb geändert, sendet die Hauptplatine zuerst auf der Baudrate eine Bestätigung. Danach wird der neue Wert abgespeichert, und die Schnittstelle neu gestartet. Je nachdem wie schnell Ihr Controller ist, sendet dieser dann nochmals eine Bestätigung mit der neuen Baudrate,



Dateninhalte Befehl 0x29 verfügbare Baudraten:

Die Geschwindigkeit von dem I²C-Bus auf unseren Hauptplatinen ist steuerbar. Diese kann zwischen 100 und 400 KHz gesteuert werden. Der Inhalt von dieser Speicherstelle ist 0x01 für 100 KHz (Normal Speed) und 0x04 für 400 KHz (Fast Speed).

Wert Dez.	Wert Hex.	Interface-Typ
0	0x00	Off
1	0x01	100 KHz
4	0x04	400 KHz
5	0x05	500 KHz
101	0x65	1 MHz
102	0x66	2 MHz
103	0x67	4 MHz
104	0x68	8 MHz
105	0x69	10 MHz



5.2.6 Befehle 0x2B, 0x2C und 0x2D

Diese Befehle dienen zur Abfrage und Steuerung eines externen Bus-Teilnehmers wie einer Echtzeituhr (RTC), einem Temperaturbausteins oder einem Speicherbausteins. Die Inhalte werden als Ganzzahl im zweistelligen Hex-Format übertragen. Diese Befehle sind im Wesentlichen nur zum Lesen gedacht.

Befehl	Beschreibung
0x2B	RTC-Baustein
0x2C	Temperatur-Baustein
0x2D	Board-Edition

Die Inhalte bzw. Rückgabewerte für die unterstützten RTC-Typen (Befehl [0x2B]) sind wie folgt:

Hex. Wert	RTC-Typ
0x00	nicht vorhanden
0x01	DS1337
0x02	PCF8583
0x03	MCP794XX
0x04	DS3231

Die Inhalte bzw. Rückgabewerte für die unterstützten Temperatur-Sensoren (Befehl [0x2C]) sind wie folgt:

Hex. Wert	Temperatur-Baustein
0x00	nicht vorhanden
0x01	LM75
0x02	DS1621
0x03	MCP9808
0x04	TC74
0x05	LM35
0x06	DS18B20
0x07	TQS3 / TQS4
0x08	AM2315

Die Inhalte bzw. Rückgabewerte für die unterstützten Board-Editionen (Befehl [0x2D]) sind wie folgt:

ASCII-Wert	Dez. Wert	Hex. Wert	Bedeutung
B	66	0x42	Basic-Ausführung
P	80	0x50	Professional Ausführung
E	69	0x45	Enterprise Ausführung

Dieser Befehl ist im Zuge der Weiterentwicklung auf die Version 4.0 im Befehl [0x21] für die Hardware-Version mit integriert. Steht aber weiterhin als Einzelbefehl zur Verfügung.



5.2.7 I²C-Bus, SPI-Bus, One-Wire

Die Steuerung einer Echtzeituhr (RTC), eines Temperatur-Sensors oder einer digitalen oder analogen Erweiterung sind inzwischen nichts außergewöhnliches mehr. Im Schwabenplan Protokoll Version 4.0 können Sie eine Vielzahl von Bauteilen anbinden und steuern. Hierfür bietet dieses Protokoll eine breite Auswahl an Möglichkeiten. Diese können auch erweitert werden.

Die Inhalte werden als Ganzzahl im zweistelligen Hex-Format übertragen. Diese Befehle sind im Wesentlichen zum Lesen und Schreiben gedacht. Bitte beachten Sie hierfür das zusätzliche Datenfeld R/W. Aufgrund der vielen verschiedenen Sensoren, die verfügbar sind, beschränken sich derzeit die Temperaturmessungen auf positive Werte, die als Ganzzahl ausgegeben werden.

Befehl	Bezeichnung	P0	P1	P2	P3
0x40	LM75 Temperatur Lesen	Adresse			
0x41	DS1621 Temperatur Lesen	Adresse			
0x43	MCP23008	Adresse			Port-Wert
0x44	MCP23017	Adresse		Port Nr. 1	Port Nr. 2
0x45	PCF8574	Adresse			Port-Wert
0x46	PCF8591	Adresse	Analog-Port Nr.		
0x50	Uhrzeit	Stunden	Minuten	Sekunden	
0x51	Datum und Wochentag	Tag	Monat	Jahr	Wochentag
0x52	Stunden	Stunden			
0x53	Minuten	Minuten			
0x54	Sekunden	Sekunden			
0x55	Tag	Tag			
0x56	Monat	Monat			
0x57	Jahr	Jahr			
0x58	Wochentag	Wochentag			



5.3 Digitale und Analoge PORT-BEFEHLE

Die Ansteuerung bzw. Abfrage von einzelnen Pins und Ports eines Microcontrollers sind wohl die häufigste Anwendung in der Steuerungstechnik. So auch hier. Unabhängig von Prozessor, Programmiersprache oder Entwicklungsumgebung können Sie mit nur einem Befehl den Port eines Microcontrollers einlesen oder ansteuern.

Befehl	Beschreibung	P0	P1	P2	P3
0xAD	A/D-Wandler	Adresse		ADW_High	ADW_Low
0xDD	Digitalbefehl	Port			Wert

Für die Steuerung eines Digital-Ports stehen folgende Ports zur Auswahl:

Befehl	PORT-Nummer	PORT-Name	Bemerkung
0x00	0	PORTA	
0x01	1	PORTB	
0x02	2	PORTC	
0x03	3	PORTD	
0x04	4	PORTE	
0x05	5	PORTF	
0x06	6	PORTG	
0x07	7	PORTH	
0x08	8	PORTI	nicht implementiert
0x09	9	PORTJ	
0x0A	10	PORTK	über PORT-Expander möglich
0x0B	11	PORTL	über PORT-Expander möglich
0x0C	12	PORTM	über PORT-Expander möglich
0x0D	13	PORTN	über PORT-Expander möglich
0x0E	14	PORTO	über PORT-Expander möglich
0x0F	15	PORTP	über PORT-Expander möglich
0x10	16	PORTQ	über PORT-Expander möglich
0x11	17	PORTR	über PORT-Expander möglich
0x12	18	PORTS	über PORT-Expander möglich
0x13	19	PORTT	über PORT-Expander möglich
0x14	20	PORTU	über PORT-Expander möglich
0x15	21	PORTV	über PORT-Expander möglich



5.4 Sammelruf (Broadcast Message)

Über ein RS-485 Schnittstelle oder CAN-Busschnittstelle können mehrere Teilnehmer miteinander verbunden und gesteuert werden. Hierzu wird die Teilnehmeradresse 0xFF bzw. 255 in Dezimal genutzt.

Ein Sammelruf (Broadcast-Message) wird von keinem Busteilnehmer bestätigt. In der Praxis wird dieser Befehl zum Aktualisieren der Uhrzeit, des Datum, einer Statusmeldung für alle Teilnehmer oder zur Durchführung eines Resets verwendet.

Wichtiger Hinweis:

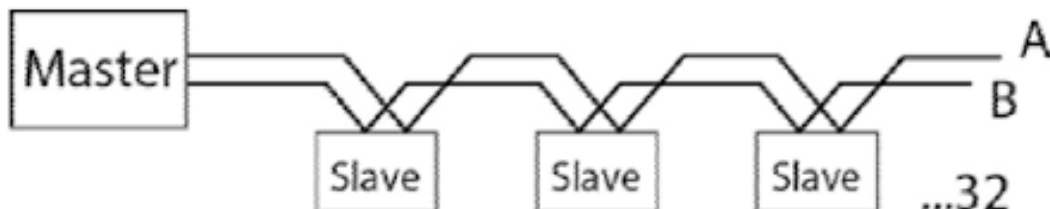
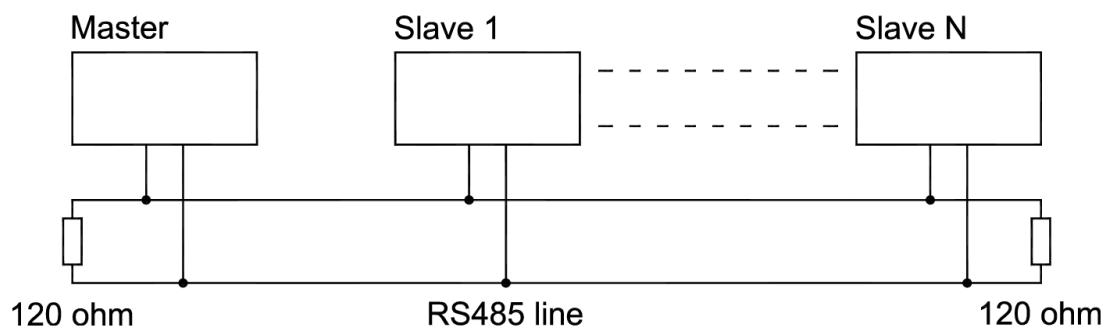
- a) Die Adresse eines Busteilnehmers kann mit dem Befehl 0x25 verwendet werden.**
- b) Achten Sie bitte darauf, dass Sie diesen Befehl nicht als Sammelruf absenden.**
- c) In diesem Fall hätten alle am Bus angeschlossenen Teilnehmer die gleiche Adresse.**



6. Verkabelung

Bitte achten Sie bei einem RS-485 bzw. einen CAN-Bus Vernetzung auf eine korrekte Verkabelung. Die maximale Länge der Stichleitungen in so einem Bussystem sowie die korrekte Terminierung des Bussystems sind wichtig. Ohne die Abschlusswiderstände am Anfang und am Ende der Busverkabelung funktioniert der Bus nicht stabil.

Auf unseren Platinen können Sie diese über DIP-Schalter vornehmen. Die nachfolgenden Skizzen sollen Ihnen bei der Projektierung helfen.





7. Umwandlung von HEX zu Integer (Char -> Integer)

Die Daten zwischen den einzelnen Teilnehmern werden als String versendet. Für einen Computer können Sie mit Hilfe von unserer Schwabensplan DLL Datei die Daten entsprechend empfangen und umwandeln. In einem Microcontroller empfiehlt es sich die Daten als Ganzzahlen (Integer Werten) verarbeiten. Das nachfolgende Beispiel soll Ihnen die Arbeit während der Integration erleichtern.

```

//*****
// Hex to Integer
// Programmierer:   Ingo Schick, Fa. Schwabensplan
// Datum:          19.12.2020
//*****

//*****
// Include-Dateien
//*****
#include <xc.h>
#include "Hex2Int.h"
//*****

//*****
unsigned short int Hex2Int(char* hex)
{
    unsigned short int val = 0;

    while (*hex) {
        // get current character then increment
        uint8_t byte = *hex++;
        // transform hex character to the 4bit equivalent number, using the ascii table
indexes
        if (byte >= '0' && byte <= '9') byte = byte - '0';
        else if (byte >= 'a' && byte <= 'f') byte = byte - 'a' + 10;
        else if (byte >= 'A' && byte <= 'F') byte = byte - 'A' + 10;
        // shift 4 to make space for new digit, and add the 4 bits of the new digit
        val = (val << 4) | (byte & 0xF);
    }
    return val;
}
//*****

```



8. Erweiterung auf das Ethernet Protokoll

Das Schwabenplan Protokoll 4.0 wurde im Zuge der Weiterentwicklung für den Microcontroller PIC18F97J60 auf Ethernet Übertragung abgeändert. Die Treiber in der Software für den PC und der Firmware des Microcontroller selbst erfordern keine Kennzeichnung mehr das Senden, Empfangen, Empfänger, Absender oder Steuerzeichen. Diese werden automatisch über die Layer (Befehlsstruktur) des TCP-Protokolls hinzugefügt bzw. entfernt. Daher wurden diese Felder für die Ethernet Datenübertragung entfernt, und die einzelnen Befehle vereinfacht.

Mit dem PIC18F97J60 können die Daten auch per Ethernet über das TCP-Protokoll und einen vordefinierten Port zwischen Server und Client übertragen werden. Der Controller antwortet nur an einen vordefinierten Server im Netzwerk.

In den Clients (PIC18F97J60) werden eine manuelle IP-Adresse, die Subnetzmaske, Gateway usw. manuell hinterlegt. Der PIC18F97J60 verfügt über ein internes 10Mbit Ethernet Interface, welches Halbduplex arbeitet. Die MAC-Adresse bezieht der PIC18F97J60 aus einem externen Speicherbaustein, welche von Microchip einprogrammiert worden ist. Diese kann auch nicht verändert oder kopiert werden. Somit ist sichergestellt, dass es diese MAC-Adresse nur einmal weltweit gibt. Die Hardware arbeitet an jedem 10/100/1000 Mbit Ethernet-Netzwerk. IM PIC18F97J60 ist eine Autonegotiation (Auto-Sensing) implementiert. Ein Router, Switch usw. erkennen das den Controller automatisch.



Auf der Hauptplatine befindet sich zusätzlich eine Echtzeituhr (RTC) und ein Temperatur-Baustein und eine RS-232 oder RS-485 Schnittstelle.

Die Parameter für die seriellen Schnittstellen und den I²C- bzw. den SPI-Bus sind im Quellcode und über die externen Schnittstellen frei konfigurierbar.



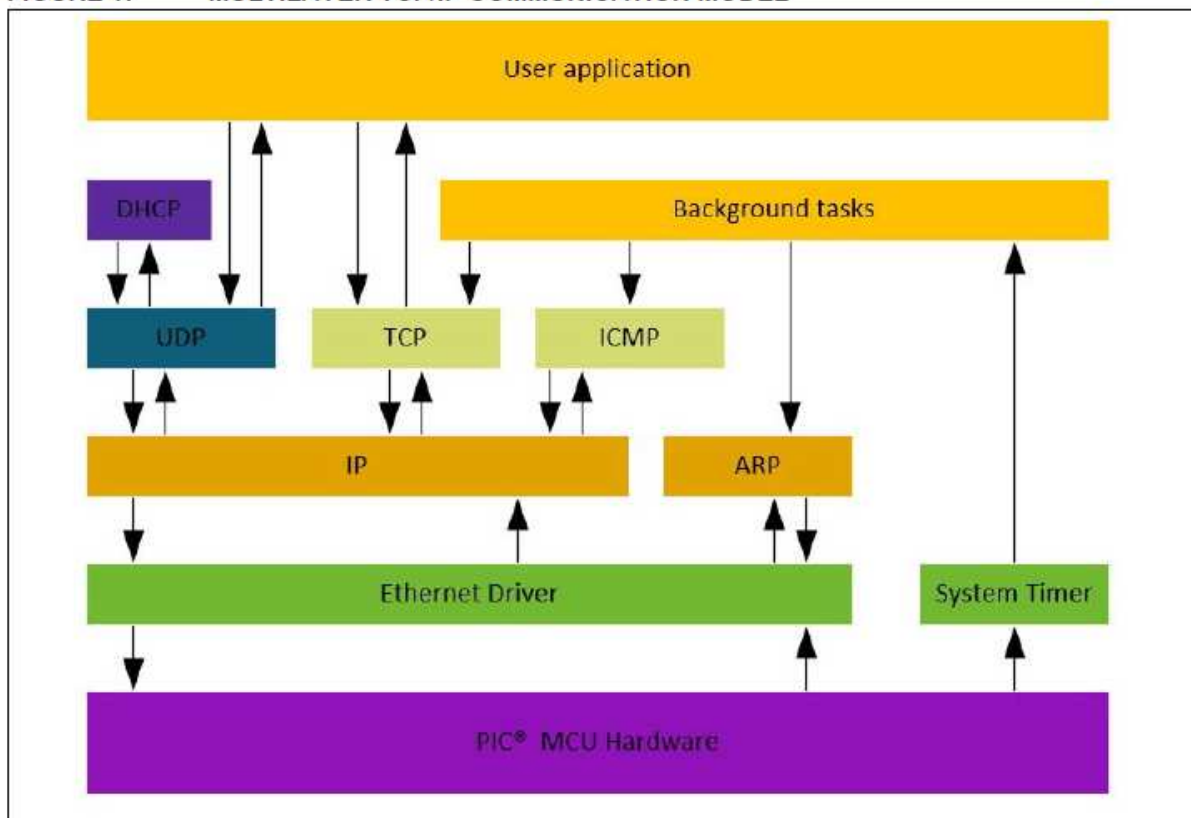
Die Firma Microchip hat hierfür drei MAC-Adressbereiche für sich reserviert. Diese sind:

- 00:04:A3:
- 54:10:EC:
- 80:1F:12:

 PIC18F97J60-1.fritz.box	192.168.100.20	Microchip Technology Inc.	80:1F:12:D4:0E:5A
 PIC18F97J60-2.fritz.box	192.168.100.21	Microchip Technology Inc.	80:1F:12:D4:0C:8A
 PIC18F97J60-3.fritz.box	192.168.100.25	Microchip Technology Inc.	54:10:EC:E4:98:60
 PIC18F97J60-4.fritz.box	192.168.100.26	Microchip Technology Inc.	54:10:EC:E4:9A:01
 PIC18F97J60-5.fritz.box	192.168.100.27	Microchip Technology Inc.	54:10:EC:E4:99:9E
 PIC18F97J60-6.fritz.box	192.168.100.28	Microchip Technology Inc.	54:10:EC:E4:9A:7F
 PIC18F97J60-7.fritz.box	192.168.100.29	Microchip Technology Inc.	54:10:EC:E4:DB:40
 PIC18F97J60-8.fritz.box	192.168.100.30	Microchip Technology Inc.	54:10:EC:E4:9A:DD

Der TCP/IP-Stack stammt von Microchip selbst ([AN1921](#)), und wurde auf die Hardwareversion 1.2 und 2.0 angepasst. Die Kommunikation der einzelnen Netzwerkteilnehmer erfolgt über IPv4. Sämtliche Zusatzplatinen können eingesetzt werden.

FIGURE 1: MULTILAYER TCP/IP COMMUNICATION MODEL





Befehlsaufbau

Alle Befehle werden im hexadezimalen Format als String übertragen. Der Doppelpunkt dient als Trennzeichen. Der letzte Parameter teilt dem Controller mit, ob es sich um ein Lese- oder Schreibbefehl handelt. Andere Steuerzeichen werden nicht benötigt. Dies wird von den Netzwerkteilnehmern selbst erledigt.

CMD : P0 : P1 : P2 : P3 : P4 : P5 : P6 : P7 : R/W

P0 bis P7 -> Parameter für die Übergabe bzw. Rückgabe

R/W -> Parameter für das Lesen oder Schreiben eines Befehles

Befehlsart	Hex. Wert
Schreiben	0x00
Lesen	0x01

Beispiel Nr. 1: 20:00:00:00:00:00:00:00:01

Beispiel Nr. 2: DD:08:00:00:FF:00:00:00:00

Das Protokoll basiert auf dem Schwabenplan Protokoll 4.0, und kann mit sehr wenig Aufwand in eine eigene Anwendung implementiert werden. Der Header für die serielle Schnittstelle kann für Ethernet in entfallen. Die Aushandlung der Teilnehmeradresse/n erfolgt direkt in den Ethernet Treibern der jeweiligen Teilnehmer im Netzwerk.

Eine einfache und unkomplizierte Hausautomatisierung ist über Ethernet in beliebiger Größe möglich. Eine einfache VB.net Applikation inkl. Einer eigenen DLL-Datei mit vielen Funktionen steht ebenfalls zur Verfügung.



Sämtliche Busaktivitäten können über eine optional zuschaltbare Protokoll-Datei mitgeschrieben werden.

```
14.03.2024 20:58:35 Protokoll-Datei ON
14.03.2024 20:58:35 DEBUG-Mode ON
14.03.2024 20:58:35 Broadcast-Message ON
14.03.2024 20:58:35 Der Server 192.168.100.101 ist gestartet.
14.03.2024 20:58:35 Der Client 192.168.100.26 ist verbunden.
14.03.2024 20:58:35 Der Client 192.168.100.30 ist verbunden.
14.03.2024 20:58:35 Der Client 192.168.100.21 ist verbunden.
14.03.2024 20:58:35 Der Client 192.168.100.20 ist verbunden.
14.03.2024 20:58:35 Der Client 192.168.100.29 ist verbunden.
14.03.2024 20:58:36 Der Client 192.168.100.27 ist verbunden.
14.03.2024 20:58:36 Der Client 192.168.100.28 ist verbunden.
14.03.2024 20:58:36 Der Client 192.168.100.25 ist verbunden.
```

Der Datenverkehr kann ebenfalls mit dem Tool Wireshark analysiert werden.

No.	Time	Source	Destination	Protocol	Length	Info
83	14.307143	192.168.100.101	192.168.100.21	TCP	87	60 → 1028 [PSH, ACK] Seq=1 Ack=1 Win=65361 Len=33
84	14.310061	192.168.100.21	192.168.100.101	TCP	60	1028 → 60 [ACK] Seq=1 Ack=34 Win=57 Len=0
85	14.324983	192.168.100.21	192.168.100.101	TCP	85	[TCP ZeroWindow] 1028 → 60 [PSH, ACK] Seq=1 Ack=34 Win=0 Len=31
86	14.376391	192.168.100.101	192.168.100.21	TCP	54	60 → 1028 [ACK] Seq=34 Ack=32 Win=65330 Len=0

Frame 83: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface \Device\NPF_{0B306608-799F-4FFC-8A68-E416E0A6C53C}, id 0		0000	80	1f	12	d4	0c	8a	94	de	80	0f	0f	f5	08	00	45	00E
Ethernet II, Src: GigaByteTech_0f:0f:f5 (94:de:80:0f:f5), Dst: MicrochipTec_d4:0c:8a (80:1f:12:d4:0c:8a)		0010	00	49	33	e1	40	00	40	06	00	00	c0	a8	64	65	c0	a8	:I3@:de-
Internet Protocol Version 4, Src: 192.168.100.101, Dst: 192.168.100.21		0020	64	15	00	3c	04	04	09	90	35	ef	00	00	00	23	50	18	d-<-<-< 5....#P-
Transmission Control Protocol, Src Port: 60, Dst Port: 1028, Seq: 1, Ack: 1, Len: 33		0030	ff	51	4a	07	00	00	32	30	3a	30	30	3a	30	30	3a	30	Q1...20 :00:00:0
Data (33 bytes)		0040	30	3a	30	30	3a	30	30	3a	30	30	3a	30	30	3a	30	30	0:00:00: 00:00:00
		0050	3a	30	31	0d	0a	0d	0a										:01....

Vorteile von TCP

Das Transmission Control Protocol (TCP) ist das Protokoll für maximale Zuverlässigkeit und Qualität. Es ist vielleicht nicht das schnellste, aber es erledigt seine Arbeit ordentlich. Hier finden Sie ein paar typische Beispiele:

- ✓ Es baut eine Verbindung zwischen Sender und Empfänger auf und hält sie aufrecht.
- ✓ Es arbeitet unabhängig vom Betriebssystem.
- ✓ Es unterstützt viele Routing-Protokolle.
- ✓ Es prüft auf Fehler und garantiert, dass die Daten unverändert am Ziel ankommen.
- ✓ Es bestätigt den Eingang der Daten nach der Zustellung bzw. versucht, sie erneut zu übertragen.
- ✓ Es ist in der Lage, Daten in einer bestimmten Reihenfolge zu senden.
- ✓ Es optimiert die Geschwindigkeit der Datenübertragung in Abhängigkeit vom Empfänger.
- ✓ Trotz seiner geringeren Geschwindigkeit ist TCP das einzige Protokoll, das verlorene Datenpakete erneut übertragen kann. Wenn Zuverlässigkeit entscheidend ist, ist TCP die beste Option.



9. Ergänzungen ASCII-Zeichensatz

Zahlen

ASCII-Zeichen	Wert Hex.	Wert Dez.
0	0x30	48
1	0x31	49
2	0x32	50
3	0x33	51
4	0x34	52
5	0x35	53
6	0x36	54
7	0x37	55
8	0x38	56
9	0x39	57

Beliebte Trenn- und Endzeichen

ASCII-Zeichen	Wert Hex.	Wert Dez.
:	0x3A	58
;	0x3B	59



Großbuchstaben

ASCII-Zeichen	Wert Hex.	Wert Dez.
A	0x41	65
B	0x42	66
C	0x43	67
D	0x44	68
E	0x45	69
F	0x46	70
G	0x47	71
H	0x48	72
I	0x49	73
J	0x4A	74
K	0x4B	75
L	0x4C	76
M	0x4D	77
N	0x4E	78
O	0x4F	79
P	0x50	80
Q	0x51	81
R	0x52	82
S	0x53	83
T	0x54	84
U	0x55	85
V	0x56	86
W	0x57	87
X	0x58	88
Y	0x59	89
Z	0x5A	90



Kleinbuchstaben

ASCII-Zeichen	Wert Hex.	Wert Dez.
a	0x61	97
b	0x62	98
c	0x63	99
d	0x64	100
e	0x65	101
f	0x66	102
g	0x67	103
h	0x68	104
i	0x69	105
j	0x6A	106
k	0x6B	107
l	0x6C	108
m	0x6D	109
n	0x6E	110
o	0x6F	111
p	0x70	112
q	0x71	113
r	0x72	114
s	0x73	115
t	0x74	116
u	0x75	117
v	0x76	118
w	0x77	119
x	0x78	120
y	0x79	121
z	0x7A	122



Steuerzeichen

ASCII-Zeichen	Wert Hex.	Wert Dez.	Bedeutung	Kurzzeichen
NUL	0x0	0	(Null)	
SOH	0x1	1	Start of Header	
STX	0x2	2	Start of Text	
ETX	0x3	3	End of Text	
EOT	0x4	4	End of Transmit	
ENQ	0x5	5	Enquiry	
ACK	0x6	6	Acknowledge	
BEL	0x7	7	Bell	
BS	0x8	8	Backspace	
HT	0x9	9	Horizontal Tab	
LF	0xA	10	Line Feed	\n
VT	0xB	11	Vertical Tab	
FF	0xC	12	Form Feed	
CR	0xD	13	Carriage Return	\r
SO	0xE	14	Shift Out	
SI	0xF	15	Shift In	
DLE	0x10	16	Data Line Escape	
DC1	0x11	17	Device Control 1	
DC2	0x12	18	Device Control 2	
DC3	0x13	19	Device Control 3	
DC4	0x14	20	Device Control 4	
NAK	0x15	21	Negative Acknowledge	
SYN	0x16	22	Synchronous Idle	
ETB	0x17	23	End of Transmit Block	
CAN	0x18	24	Cancel	
EM	0x19	25	End of Medium	
SUB	0x1A	26	Substitute	
ESC	0x1B	27	Escape	
FS	0x1C	28	File Separator	
GS	0x1D	29	Group Separator	
RS	0x1E	30	Record Separator	
US	0x1F	31	Unit Separator	
SP	0x20	32	Space	