Serielle Kommunikation
RS-232 + UART
ARM Cortex M4 / TM4C1294

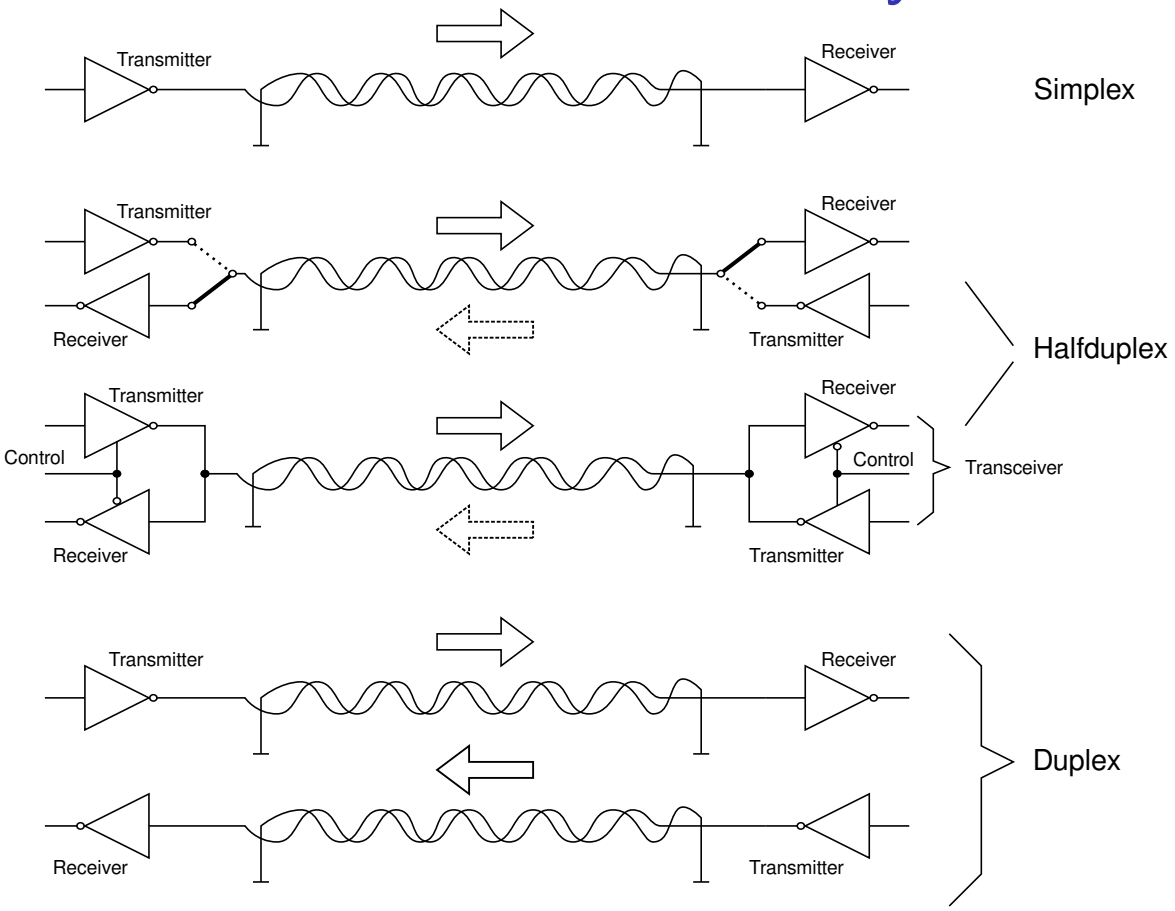
Vorlesung Mikroprozessortechnik

HAW Hamburg

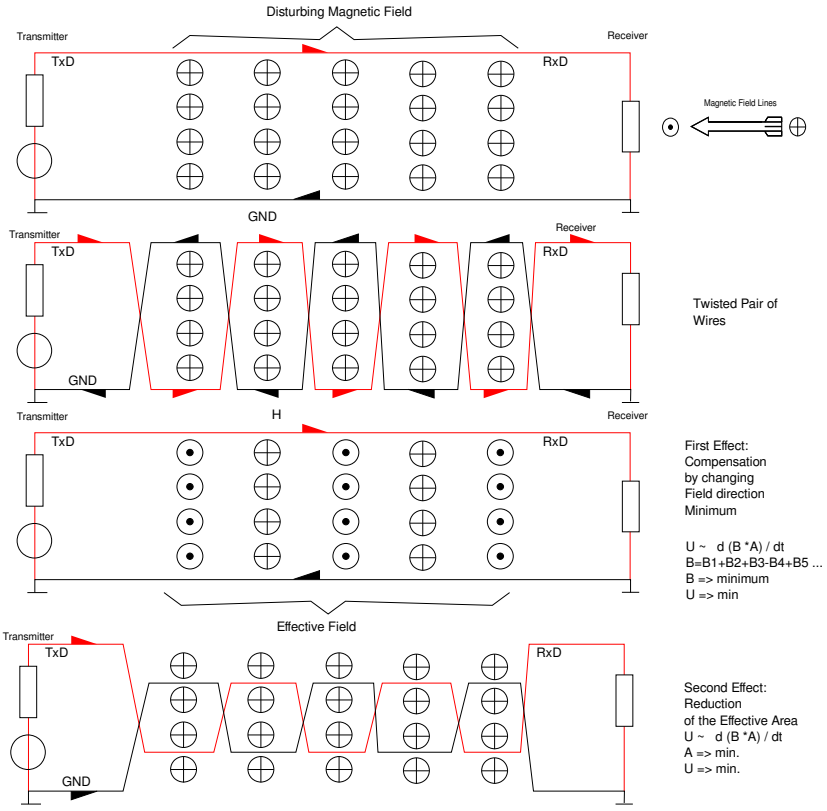
4. Januar 2018

Einige Merkmale der Computer-Kommunikation

Kommunikation zweier Prozessor-Systeme



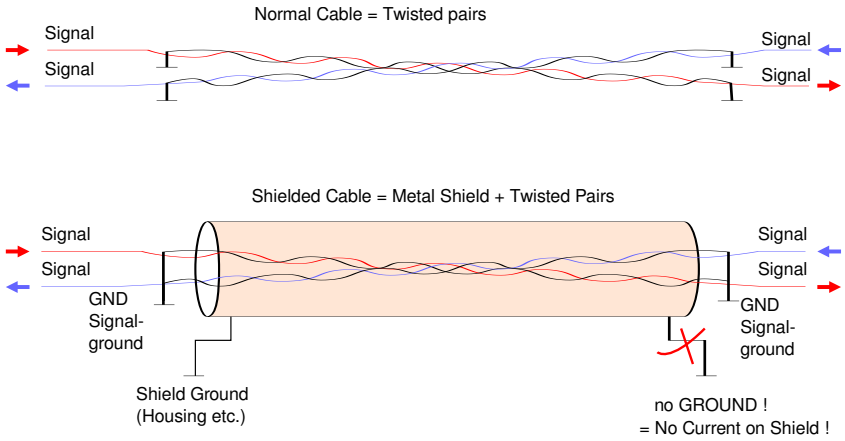
Twisted Pair: Verdrillung der Leitungsadern von Signal und Signalmasse



Paarweise passende Adern eng verdrillen
(typ. 5-15fach pro Meter)
Verminderte magnetische Störwirkung:
- Reduktion der effektiven Fläche zwischen Hin- und Rückleitung
- Aufheben der magnetischen Induktion durch Richtungswechsel

Leitungskapazität steigt = Tiefpasswirkung (Vorteil bei niedrigen Bitraten, Nachteil bei hohen Bitraten)

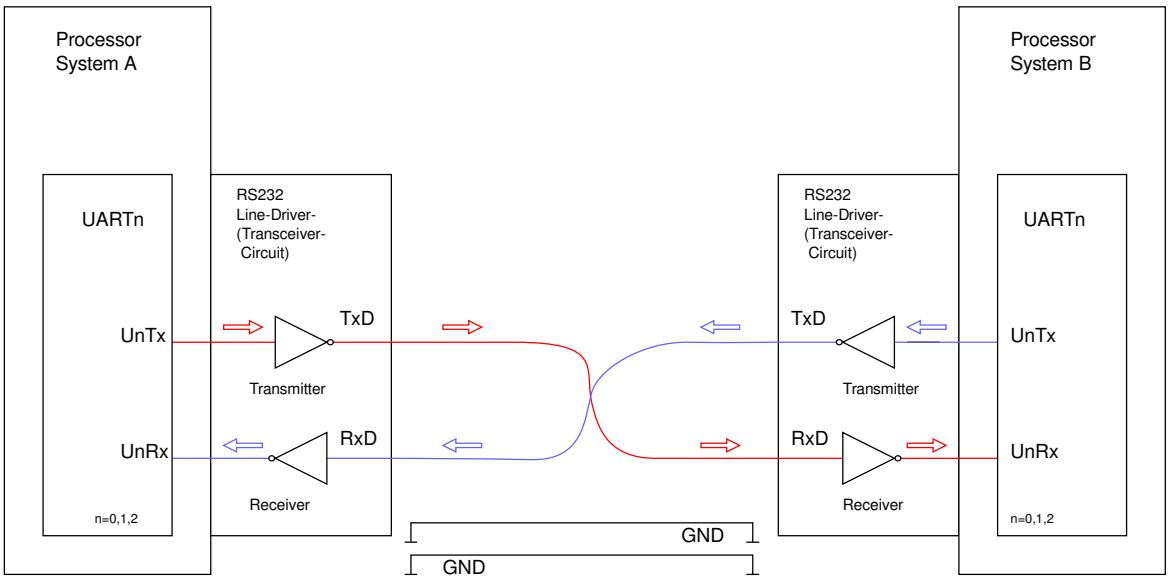
Schirmung des Kabels



Goldene Regeln:

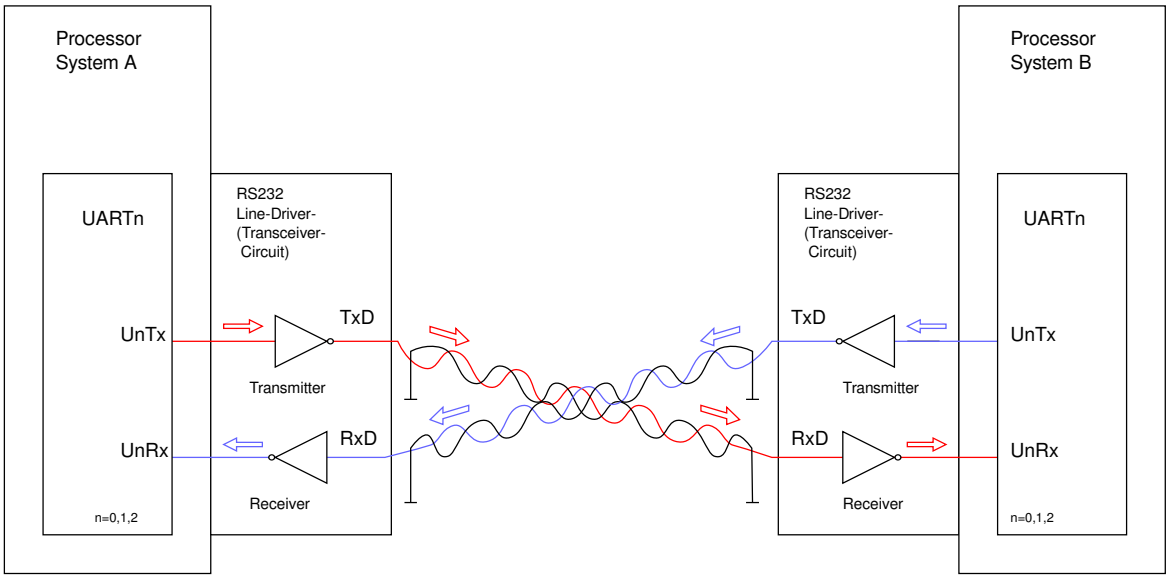
- Keine Ströme auf der Schirmung
- Schirmung nie als Signalmasseverbindung (GND) nutzen
- Oft getrennte Signal- und Geräte/Gehäusemassepotentiale vorliegend, diese nicht am Kabel verbinden
- Masseschleifen zwischen Geräten vermeiden, Schleifen durch mehrere Verbindungen möglich, inkl. Netz- u. Schutzleitern

Duplex-Kommunikation zwischen zwei Prozessor-Systemen: RS232-Standard



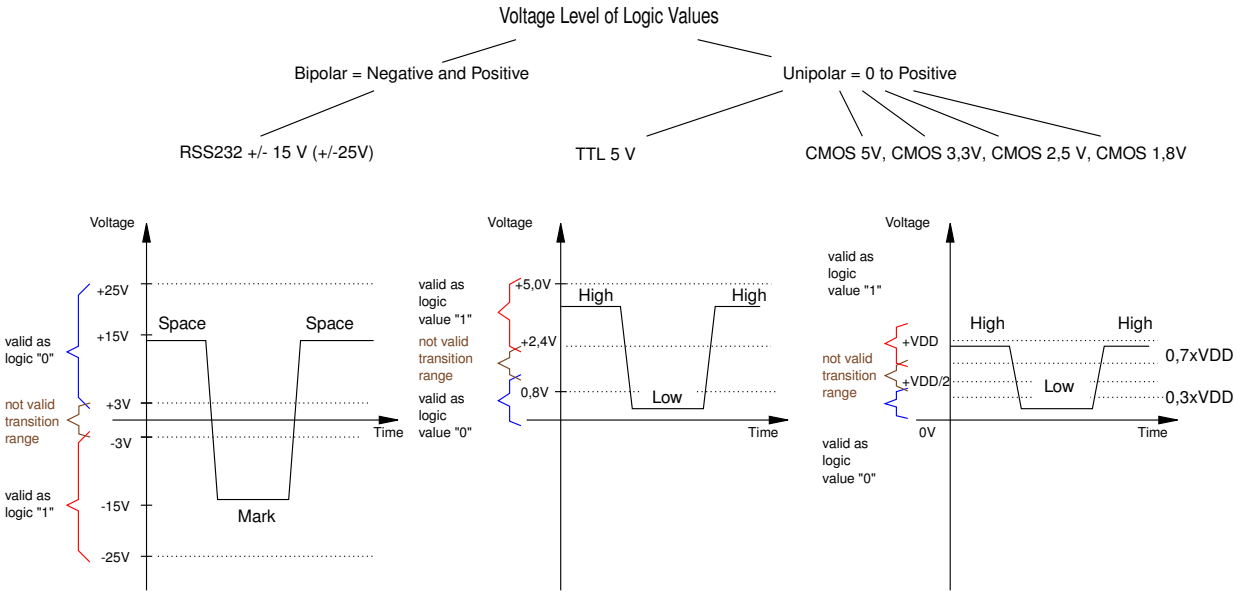
Die UART sind nicht direkt verbunden. Auf beiden Seiten sind RS232-Line-Driver-Chips zwischen Prozessor und Stecker geschaltet. Jedes Signal bekommt eine eigene Masseleitung GND.

RS232-Verbindung

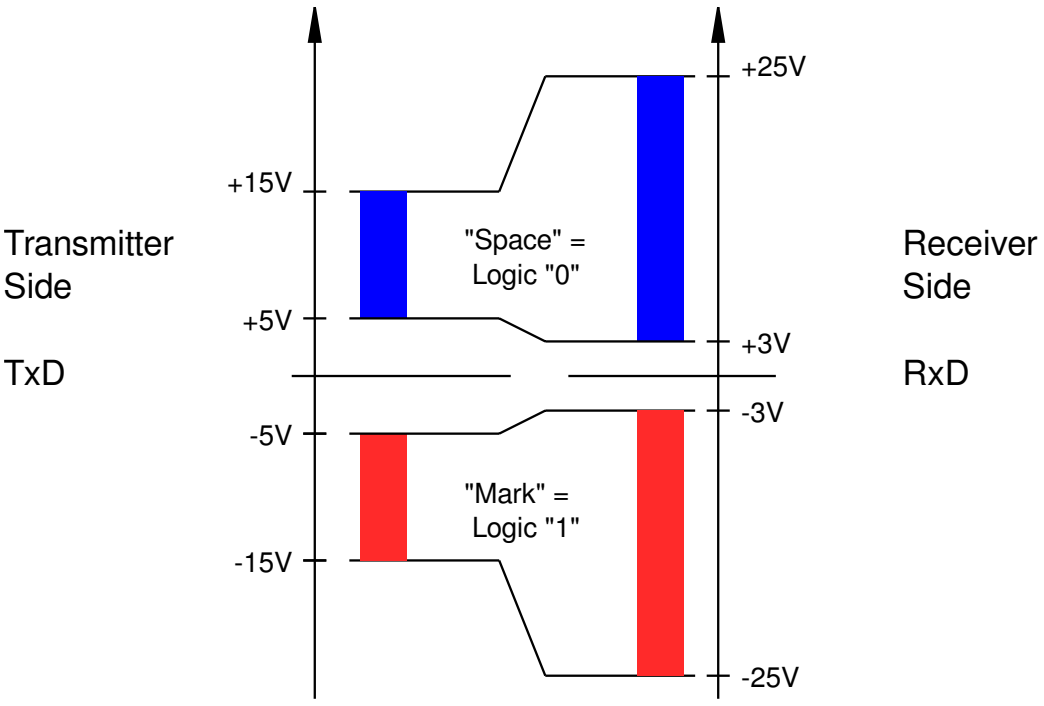


Die Namen TxD (Transmit Data) und RxD (Receive Data) bezeichnen Steckerpins, es sind keine Signal- oder Leitungsnamen !

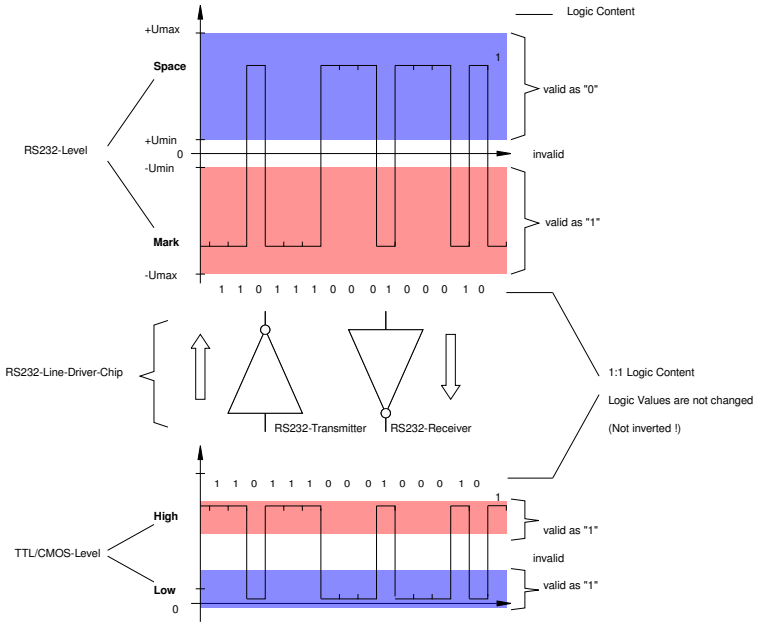
Repräsentation logischer Werte



RS232-Pegel



Umsetzung zwischen RS232 und TTL/CMOS-Pegeln



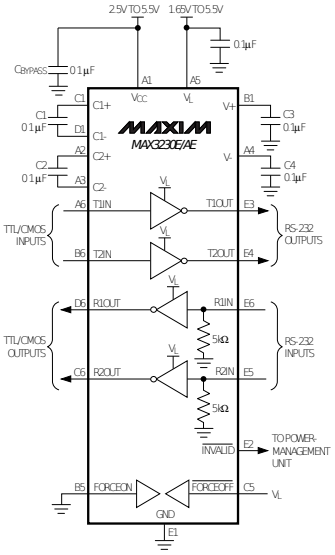
Funktion des Line-Driver-Chip:

CMOS/TTL-Pegel⇒RS232-Pegel durch den Transmitter-Teil

RS232-Pegel⇒CMOS/TTL-Pegel durch den Receiver-Teil

⇝ Es wird nur die physikalische Repräsentation der binären logischen Werte geändert, nicht der logische Wert selbst (keine Negation/Invertierung !).

Beispiel: Moderner RS232-Line-Driver-Chip I



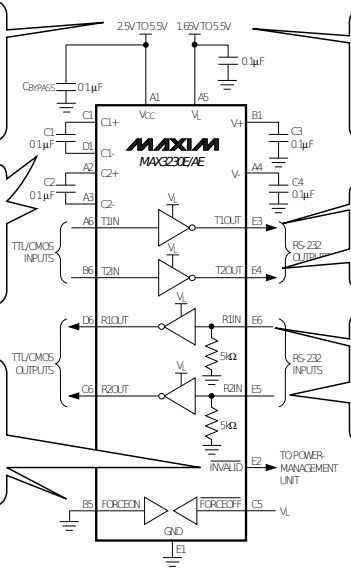
Source: MAXIM Datasheet MAX3230

Beispiel: Moderner RS232-Line-Driver-Chip II

The transceivers have a proprietary low-dropout transmitter output stage, delivering RS-232-compliant performance from a +3.1V to +5.5V supply, and RS-232-compatible performance with a supply voltage as low as +2.5V.

Dual charge pump requires only four small 0.1µF capacitors for operation from a +3.0V supply, see C1-C4.

The MAX3230E/AE and MAX3231E/AE achieve a 1µA supply current with Maxim's AutoShutdown™ feature. They save power without changing the existing BIOS or operating systems by entering low-power shutdown mode when the RS-232 cable is disconnected, or when the transmitters of the connected peripherals are off.



Configurable logic levels on the receiver outputs and transmitter inputs. Operating over a +1.65V to VCC range provides the MAX3230E/AE and MAX3231E/AE compatibility with multiple logic families.

The transmitters are level translators that convert CMOS logic levels to RS-232 levels. ...

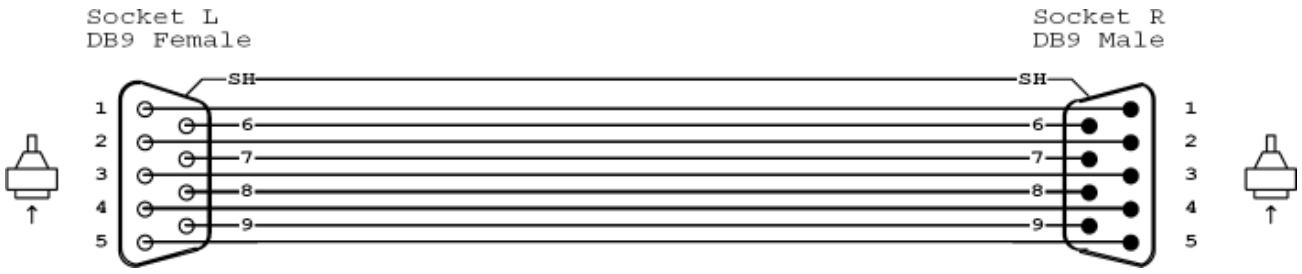
The MAX3230E/AE and MAX3231E/AE receivers convert RS-232 signals to logic-output levels. .....

Source: MAXIM Datasheet MAX3230

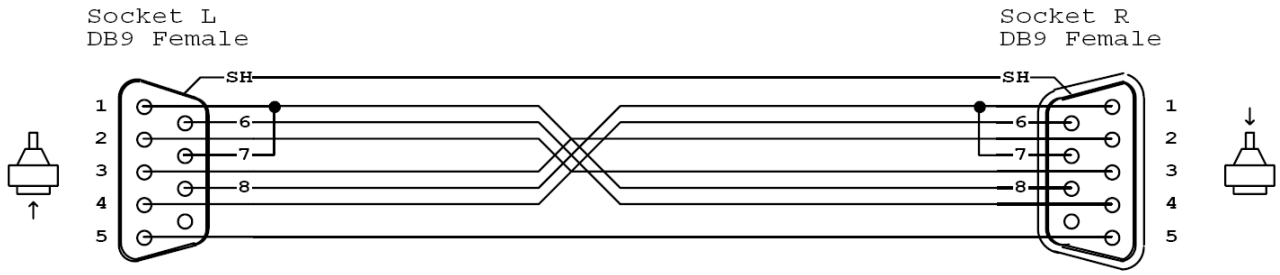
RS232 Anschluss am PC

N9-Poliger Stecker (male connector) am PC, Laptop oder Controllerboard

Serielle Kabel (Steckersystem DSUB 9-polig)

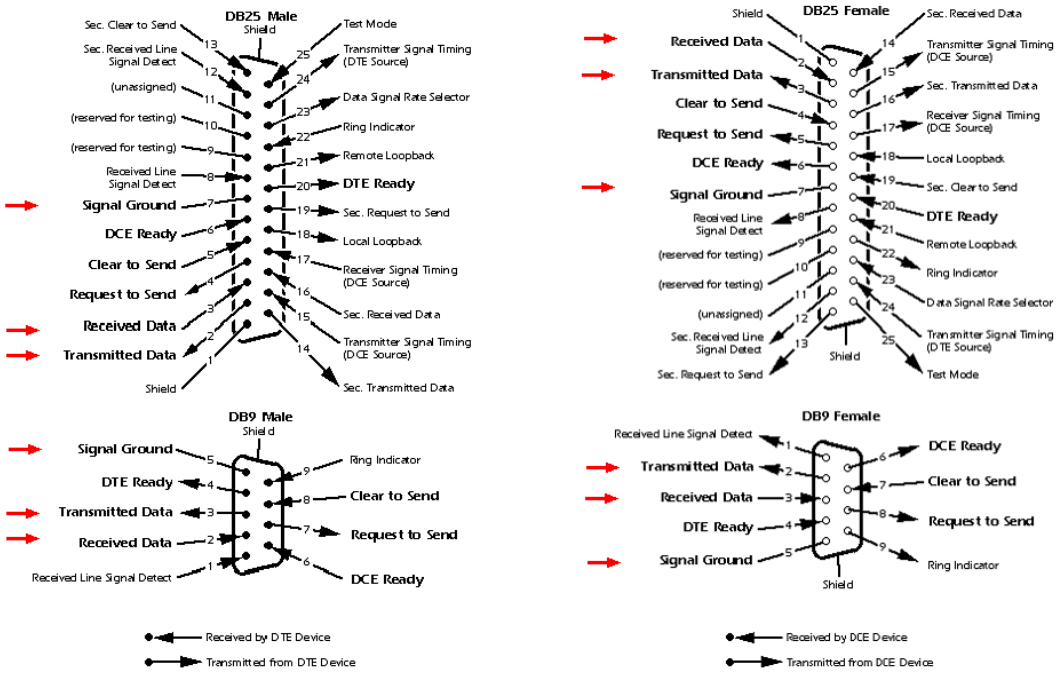


Standard Kabel, 1:1 verbunden, Stecker und Buchse



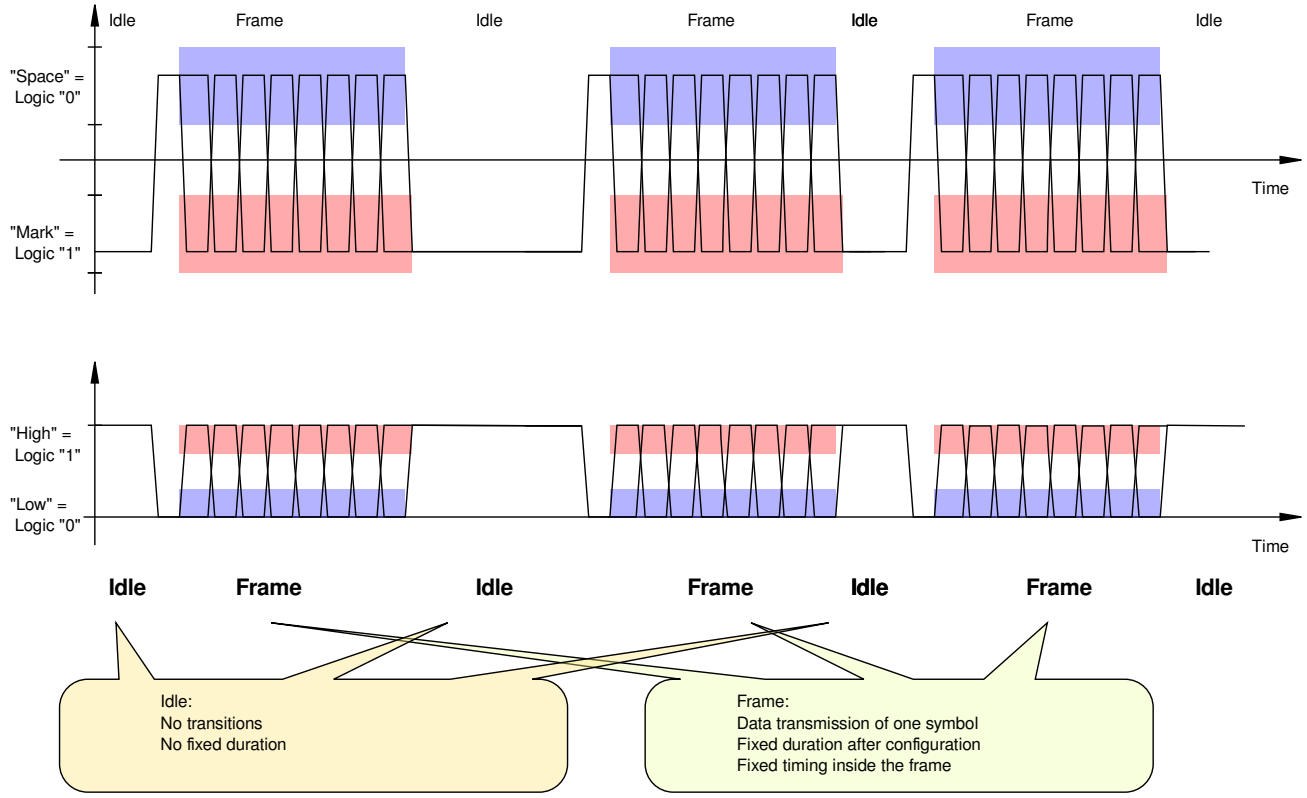
Null-Modem-Kabel, gekreuzt, Buchse und Buchse

Signale am Steckersystem DSUB 25- u. 9-polig



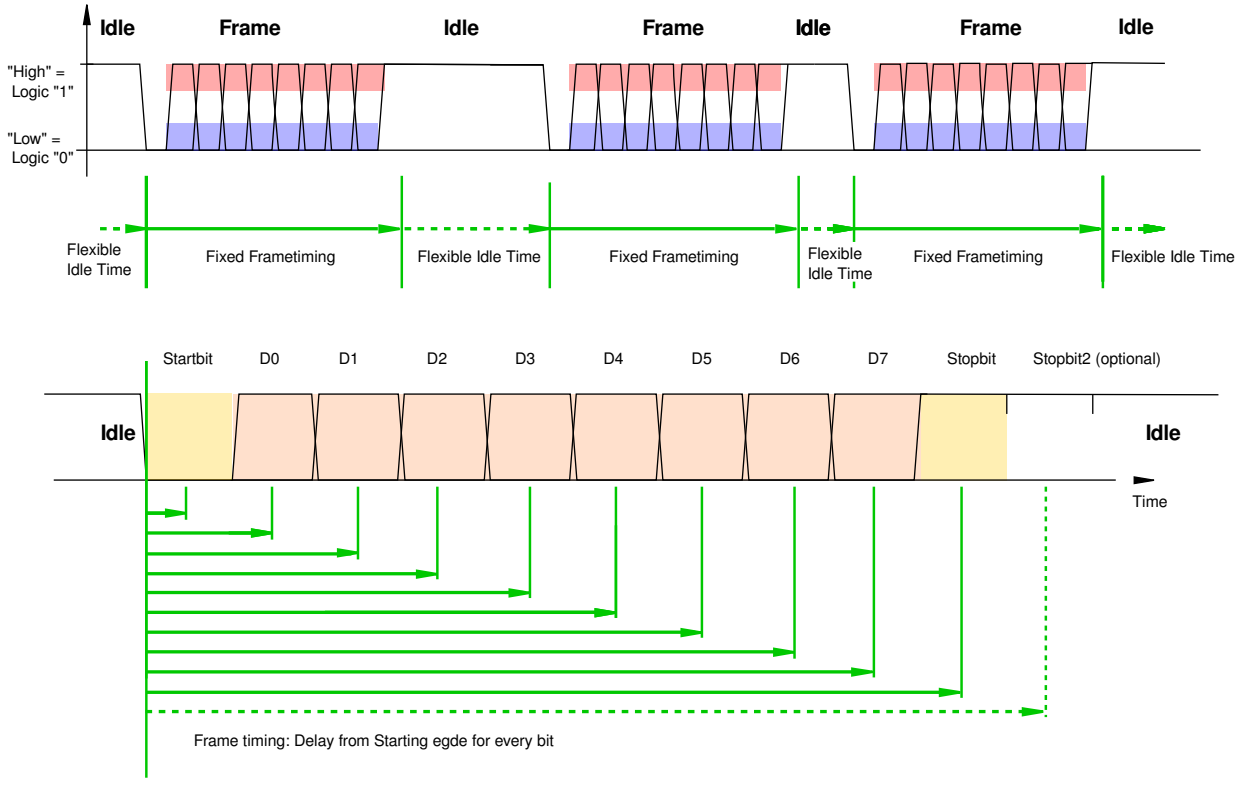
Wichtigste Signale: →Transmitted Data = TxD, → Received Data = RxD, → Signal Ground = GND, Rest wird heute oft nicht mehr genutzt

Frames und Idle States



Frame = Daten-Rahmen für ein Symbol; Dauer: Festgelegt durch konfigurierte Bitdauer u. Protokolloptionen Idle State = Ruhezustand zwischen den Frames; Dauer: 0 bis ∞,

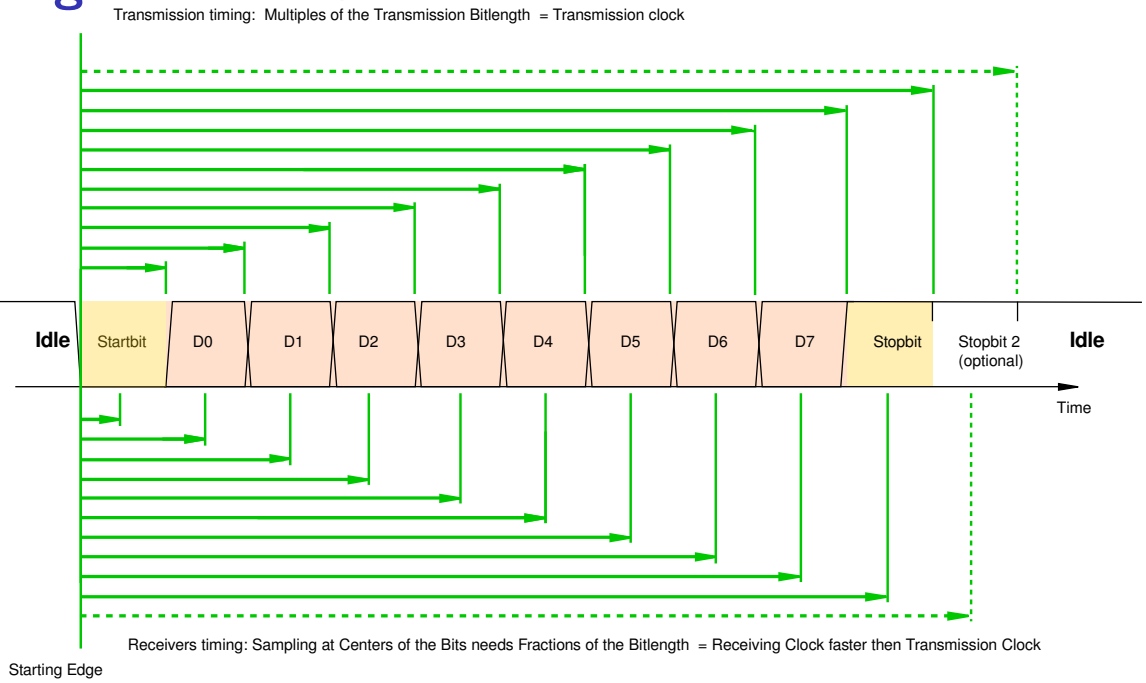
Zeitverhalten Idle States und Frames



Starting Edge = Falling Edge of the Startbit

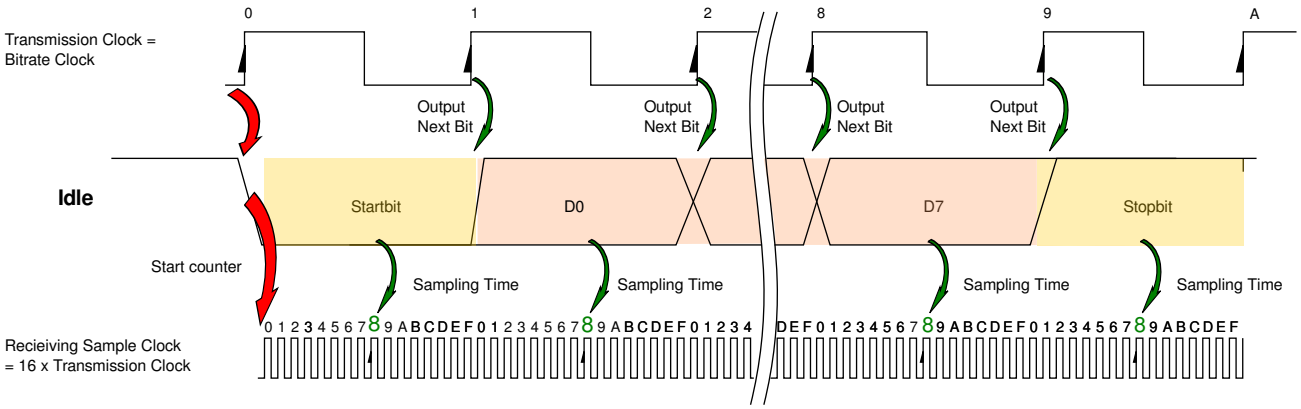
Das Timing des Frames bezieht sich auf die Startflanke des Startbits

Timing von Idle States und Frames



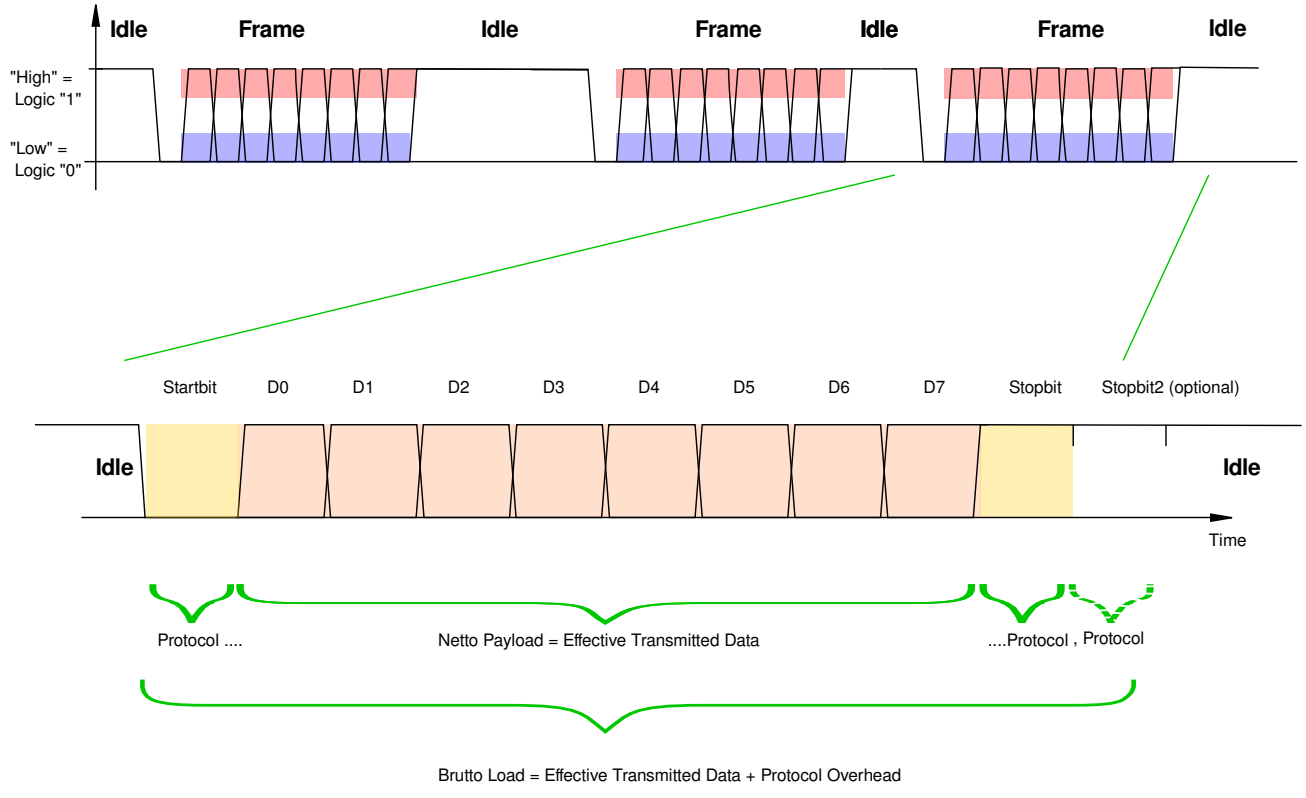
Die serielle Ausgabe jedes Bits erfolgt an den (steigenden) Flanken des Transmission Clocks. Das serielle Einlesen jedes Bits erfolgt in der 'Bitmitte'. Dort ist der günstigste Zeitpunkt: (max. Abstand zu Flanken, erlaubte Toleranz der Taktfrequenzen zwischen Sender und Empfänger wird maximal ausgenutzt).

Transmission and Receiver Sampling Clock



Zwei Takte in der UART: (1) Die serielle Ausgabe jedes Bits erfolgt an den (steigenden) Flanken des Transmission Clocks. (2) Der Receiver Sampling Clock ist 16fach schneller und startet mit der Anfangsflanke des Startbits einen umlaufenden Zähler. Vor dem Zählerstand 8 wird jedes Bit abgetastet.

Frame Struktur



Hinweis: Das Least Significant Bit LSB (niedrigstwertiges Datenbit D0) wird zuerst nach dem Startbit übertragen.

Anzahl der Datenbits

Die Anzahl der Daten Bits im Frame kann betragen:

- 5 Data Bits für den Baudot Code (Fernschreiber, ältere Funkprotokolle oder CNC-Steuerungen)
- 6 Data Bits, keine Anwendung bekannt
- 7 Data Bits, für Standard ASCII-Zeichen (127 Zeichen true ASCII)
- 8 Data Bits für Zeichen- (Extend ASCII) und Binärdaten-Übertragung (in Byte)
- 9 Data Bits - selten genutzt, mit adressierenden oder Kommando-Betriebsarten

→ Die Option mit 8 Data Bits wird universell in den heutigen Anwendungen benutzt.

→ Die Optionen mit 5 und 7 Data Bits sind verbreitet in alten/kompatiblen Systemen

→ Das Least Significant Bit der Daten (LSB = D0) wird zuerst übertragen.

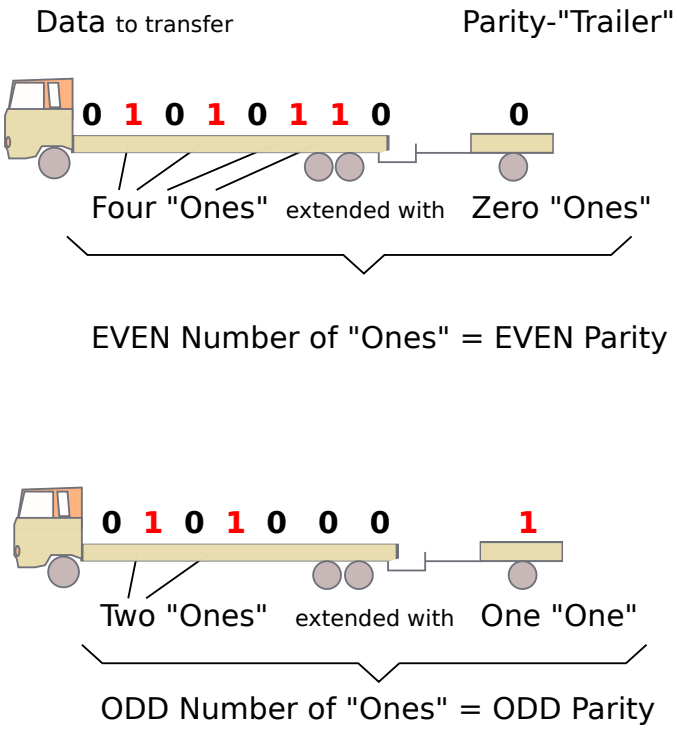
LSB Data Bit first !

Parity Bits

- Durch ein Paritätsbit können einzelne Bitfehler der Übertragung der Daten eines Frames erkannt werden [1].
- Ein Parity Bit ist ein optionaler Teil des Frame-Protokolls.
- Die Sende-Seite hängt das Parity Bit an die Datenbits vor dem (ersten) Stopbit an (Trailer-Bit).
- Der logische Wert des Paritätbits ergänzt die Gesamtanzahl der Einsen in den Daten und dem Trailerbit:
  - zur geraden Gesamtanzahl der Einsen, wenn die gerade Parität (even Parity) konfiguiert wurde
  - zur ungeraden Gesamtanzahl der Einsen, wenn die ungerade Parität (odd Parity) konfiguiert wurde
- Die Parität kann auf der Empfangsseite geprüft werden, dort muss gleiche (gerade/ungerade) Parität konfiguiert sein.

[1] Parity Bits werden auch bei Speichern, für arithmetische Schaltungen oder im Schaltungs-Selbsttest verwendet.

Angehängte Paritätsbits ERGÄNZEN die Datenbits zur geraden oder ungeraden Parität



Paritätsbits Regeln

Konfiguration des Senders und des Empfängers vor der Übertragung:		
	Even (gerade) Parity wurde eingestellt	Odd (ungerade) Parity wurde eingestellt

Während Übertragung für jeden Frame:		
aktuelle Anzahl der Einsen in den Datenbits	Wert des Parity Bits	Wert des Parity Bits
gerade	**0**	**1**
ungerade	**1**	**0**

Paritätsbits Beispiele

7 Bit Data	8 Bit: 7 Bit Data + 1 Trailer Bit	
	even (gerade)	odd (ungerade)
0000 000	0000 000 0	0000 000 1
1010 001	1010 001 1	1010 001 0
1101 001	1101 001 0	1101 001 1
1111 111	1111 111 1	1111 111 0

vspace*4mm

8 Bit Data	9 Bit: 8 Bit Data + 1 Trailer Bit	
	even (gerade)	odd (ungerade)
0000 0000	0000 0000 0	0000 0000 1
1010 0010	1010 0010 1	1010 0010 0
1101 0010	1101 0010 0	1101 0010 1
1111 1111	1111 1111 0	1111 1111 1

Parity Bit Konfiguration im RS232-Standard

- Im RS232-Frame-Protokoll kann das Parity Bit konfiguriert werden:
  - None Parity Bit (N): Es wird kein Paritätsbit angehängt.
  - Odd Parity Bit (O): Es wird ein Bit angehängt, so dass die Anzahl der Einsen in den Datenbits inkl. des Trailer-Bits ungerade ist.
  - Even Parity Bit (E): Es wird ein Bit angehängt, so dass die Anzahl der Einsen in den Datenbits inkl. des Trailer-Bits gerade ist.
- Ungebräuchlich sind Trailer Bits mit festen Werten, anstelle der Parity Bits.
  - Mark Trailer Bit (M):Festwert '1' anhängen
  - Space Trailer Bit (S): Festwert '0' anhängen
- Am häufigsten wird heute die Option 'none Parity Bit (N)' benutzt. Stattdessen werden Fehler in den höheren Schichten des Übertragungsverfahrens abgefangen.

Parity Generator & Parity Checker

- Das Paritätsbit wird auf der Seite des Transmitters erzeugt und an die Daten angehängt (Trailer Bit).
- Die Erzeugung erfolgt durch einen Parity-Generator. (z.B. XOR bzw. XNOR aller Bits oder serieller Einsenzähler)
- Das Paritätsbit wird auf der Seite des Receivers abgelöst und entsprechend der konfigurierten Parität (odd/even) auf Richtigkeit geprüft.
- Die Prüfung erfolgt durch einen Parity-Checker in drei Schritten:

- Ein Parity-Generator im Parity-Checker erzeugt eine Prüfparität der Daten.
- Ein Vergleicher vergleicht die Prüfparität und die übertragene Parität (Äquivalenz).
- Stimmt die Prüfparität und die übertragene Parität geschieht nichts, anderenfalls wird ein Parity-Error-Flag gesetzt.

- Hinweis: Ein falsches Paritätsbit kann einen Übertragungsfehler anzeigen, aber nicht 'reparieren'. $\rightarrow$ Mehrbitfehler, Effizienz

Stopbits

- Stopbits stehen am Ende des Frames und haben immer den logischen Wert '1' (d.h. Mark für RS232-Level, High für CMOS/TTL-Level) .
- Die Anzahl der Stopbits beträgt:

  1 Stopbit: '1' gehalten für mindestens 100% einer Bitdauer

  $1\frac{1}{2}$ Stop bits: '1' gehalten für mindestens 150% einer Bitdauer

  2 Stop bits: '1' gehalten für mindestens 200% einer Bitdauer
- Stopbits erlauben die Resynchronisation des Receivers. Das Stopbit hat den gleichen Wert '1' wie der Idle-State und geht lückenlos in diesen über.
- Ein Startbit kann unmittelbar auf das/die Stopbit/s folgen (Idle-Dauer = 0).
- Heutige Geräte und Controller nutzen i.allg. 1 Stopbit.

Kurznamen der Protokolloptionen

- Konvention für die Protokollbezeichnung D/P/S = Data/Parity/Stop
  - D = Anzahl der Datenbits im Frame: 5,6,7,8,9
  - P = Paritäts-Konfiguration: N=none, O=odd, E=even (selten M=Mark, S=Space)
  - S = Anzahl der Stopbits: 1 oder 1,5 oder 2
  - Die Trennstriche '/' können entfallen.
- Die Einstellungen am Empfänger (RxD) und Sender (TxD) müssen exakt gleich sein.
  - Die Einstellungen für beide Richtungen des Duplexbetriebs sind i.R. gleich (oft nicht anders möglich).
  - korrektes Beispiel: 7/E/1 bzw. 7E1 bedeutet 7 Data Bits, Even Parity, 1 Stop bit
  - fehlerhaftes Beispiel: 7E1 am Transmitter gesendet, 8N1 am Receiver erwartet
- Die UART oder andere serielle Module (SCI, SIO) implementieren nur eine Teilmenge des Standards.

Beispiele für die Protokolloptionen

Zusätzliche Daten Fluss Steuerung (Flow Control) in Hardware

- Zusätzliche Signale für Unterbrechung / Start bzw. Weiterlaufen der Sendung
- "Handshake Lines"(Anforderungs- und Quittungsbetrieb), z.B.:
  - Request to Send (RTS), Transmitter sendet 0 und kündigt dem Receiver den Datenempfang an
  - Clear to Send (CTS), Receiver sendet 0 und erlaubt die Sendung
  - Data Terminal Ready (DTR), Transmitter sendet 0 um die Verbindungsbereitschaft zu signalisieren
  - Data Set Ready (DSR), Transmitter sendet 0 um eine aktive Verbindung zu signalisieren, DSR kann man oft in Hardware festsetzen.

Früher wurde die Datenfluss Steuerung in Hardware mit gesonderten Leitungen häufiger genutzt, heute weitgehend entfallen.

Zusätzliche Daten Fluss Steuerung (Flow Control) in Software

- Zeichen mit Sonderfunktion, wie z.B.:
  - XON (transmit on) vom Receiver zum Transmitter: 'Bin bereit für den Empfang von (weiteren) Daten'
  - XOFF (transmit off) vom Receiver zum Transmitter: 'Sender soll warten bis der Empfänger fertig ist'
- Ist langsamer als Hardware Flow-Control, benutzt den Rückkanal.
- Beansprucht reservierte Zeichen im Rückkanal, daher Problem bei binären Daten.
- Geeignet für Geräte mit weitgehend 'freiem' Rückkanal (Drucker, Ausgabegeräte)
- Kabel kann auf 2x Signalleitungen und 2x GND reduziert werden.

Bitraten und max. Kabellängen

Bit/s	Bit time	max. Kabellänge*)
300	3,3 *ms*	>1km
1.200	833 μs	>1km
2.400	417 μs	900m
4.800	208 μs	300m
9.600	104 μs	152m
19.200	52μs	20-40m
38.400	26 μs	10-20m
57.600	17 μs	5-10m
115.200	8,68 μs	2-5m
230.400	4,34 μs	1-2m
460.800	2,17 μs	<1m

*) Es gibt unterschiedliche Angaben: Sie sind abhängig von Kabelqualität (Widerstand und Kapazität der Leitungen), Umfang der Störeinflusse, Anzahl der parallel verlegten Leitungsadern und der Einbausituationen. Es wurden typische Werte angeben, Quelle hier: Texas Instruments, www.ti.com

Baud vs. Bit/s

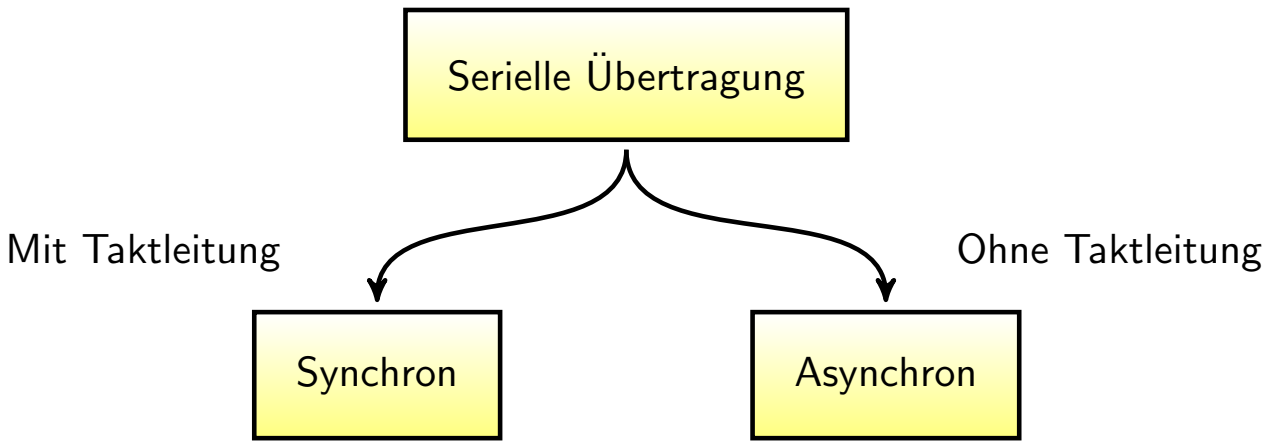
- Einheit Baud (auch kurz Bd) ist benannt nach J.M.E. Baudot: 1874 Baudot-Code Fernschreiber).
- Die Einheit Baud wird für Symbolrate (Symbole/s) benutzt.
- Ein Symbol entspricht je nach Codierung unterschiedlich vielen Bits eines Datenstromes.
- Als Symbol wird bei der seriellen Übertragung verstanden:

  a) Die gesamten Daten im Frame (z.B. ein ASCII-Symbol/ein Byte + Protokoll) oder

  b) Ein beliebiges Bit der Übertragung
- Die Einheit Bit/s betrachtet alle übertragenen Bit (Daten und Protokoll).
- Bei der Bennennung der seriellen Bitrate findet man in der Literatur sowohl die Baud als auch Bit/s. Meist (aber nicht immer) wird das gleichbedeutend benutzt. Die Einheit Baud also im Sinne von b).
- Eindeutig ist die Bezeichnung Bit/s (manchmal Bit per sec, Bps oder bps).

  $\Longrightarrow$ Bit/s ist zu bevorzugen

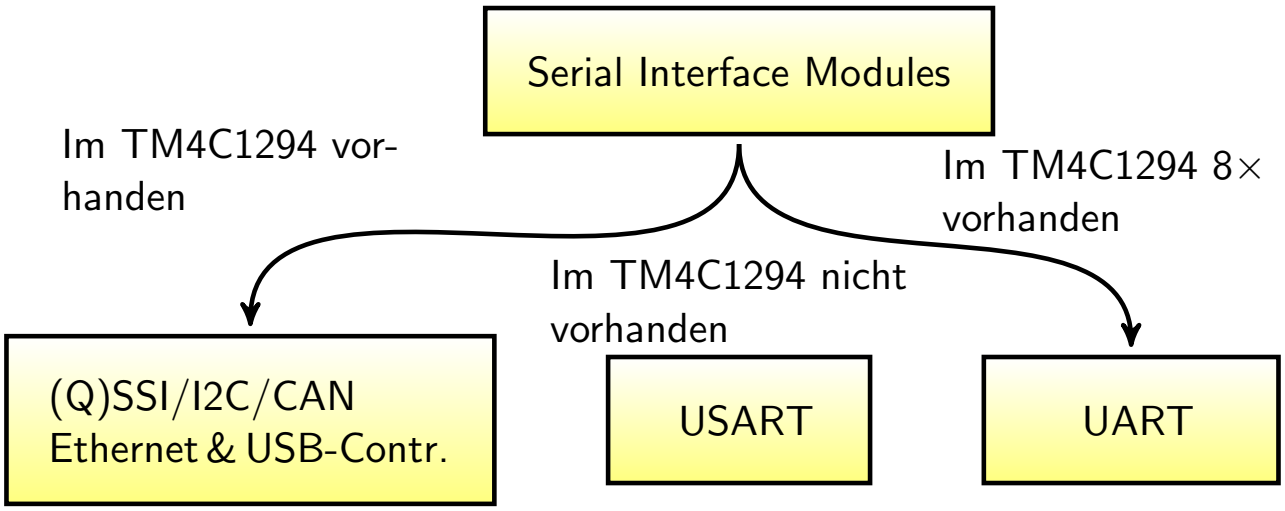
Synchron vs. Asynchron



gültiger Empfangszeitpunkt stabiler Bits mit Taktflanken auf zusätzlicher Taktleitung vom Sender vorgeben

gültiger Empfangszeitpunkt stabiler Bits durch Protokollbits auf der Sendeleitung vorgeben

Peripheral Units: UART, USART, SSI, I2C ...

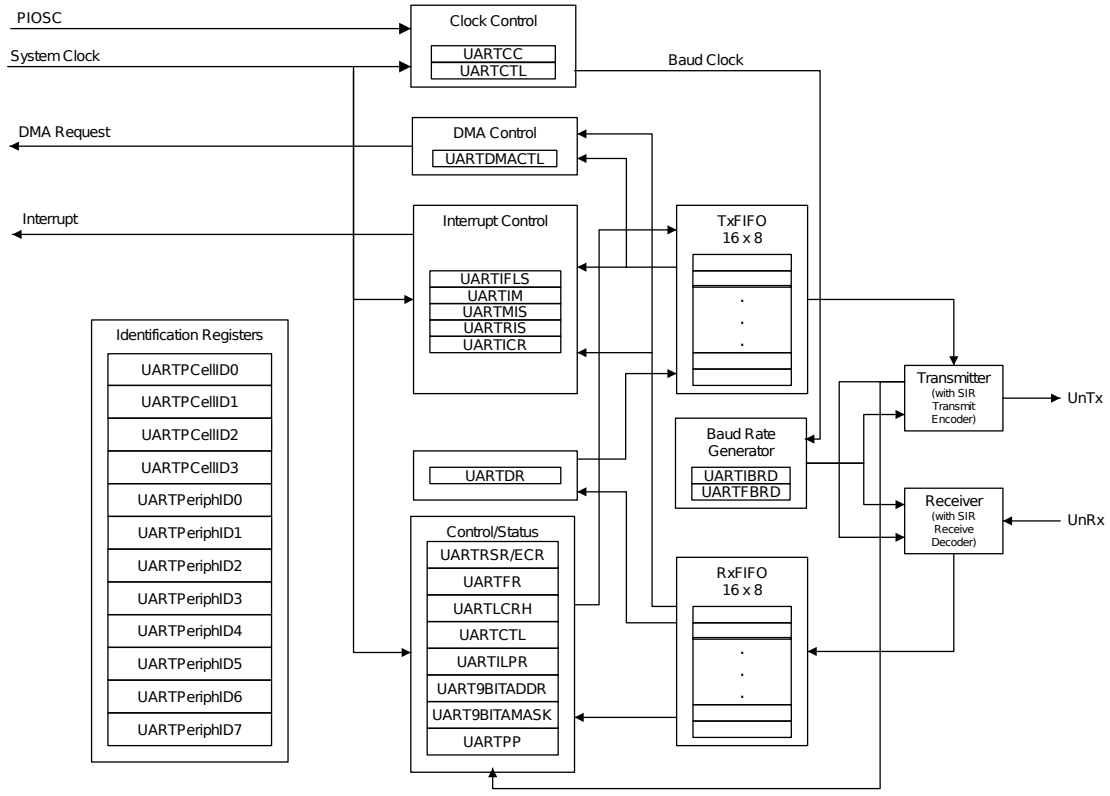


(Quad) Synchronous Serial Interface (SPI)/ Inter-Integrated Circuit (I2C) Interface u.a.

Universal Synchronous and Asynchronous Serial Receiver Transmitter

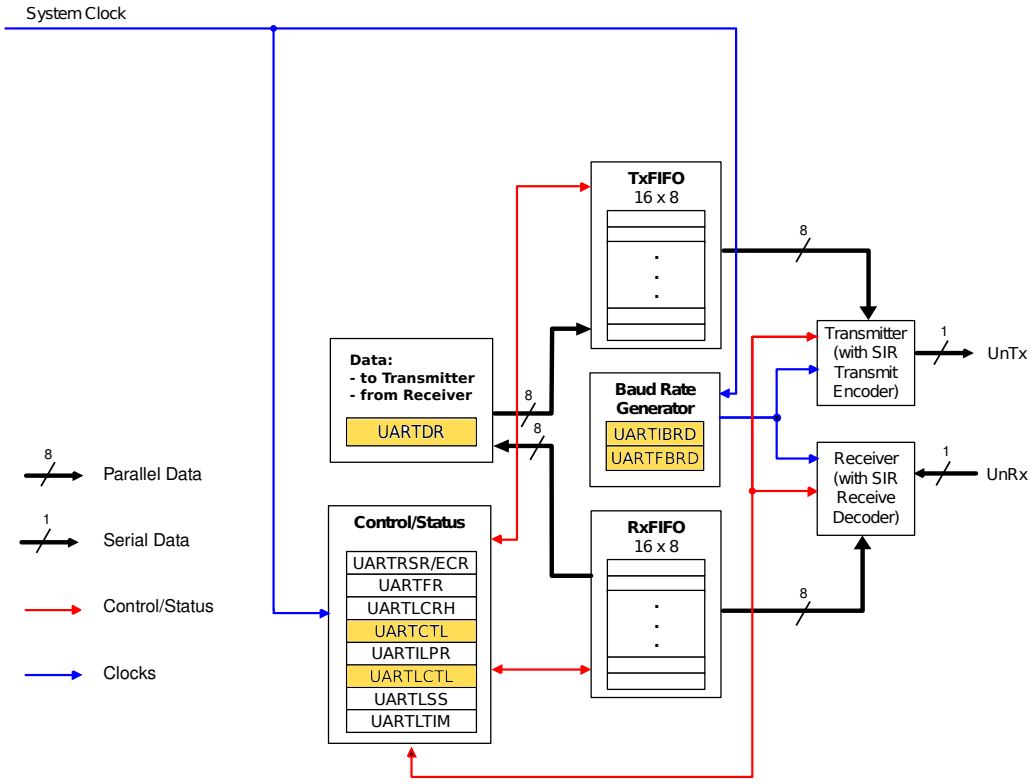
Universal Asynchronous Receiver Transmitter

Übersicht UART



Gesamtübersicht lt Datasheet

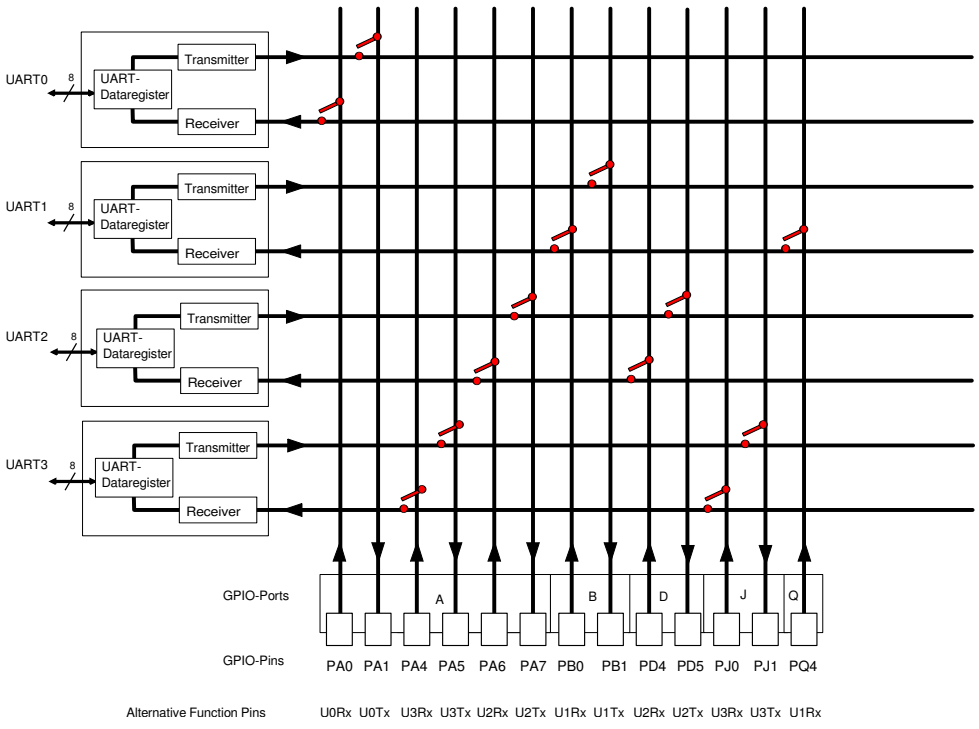
UART reduzierte Ansicht



modified from Texas Instruments TM4c1294 Datasheet

Reduzierte Darstellung: nur in Gelb markierte Register für Minimalfunktion notwendig.

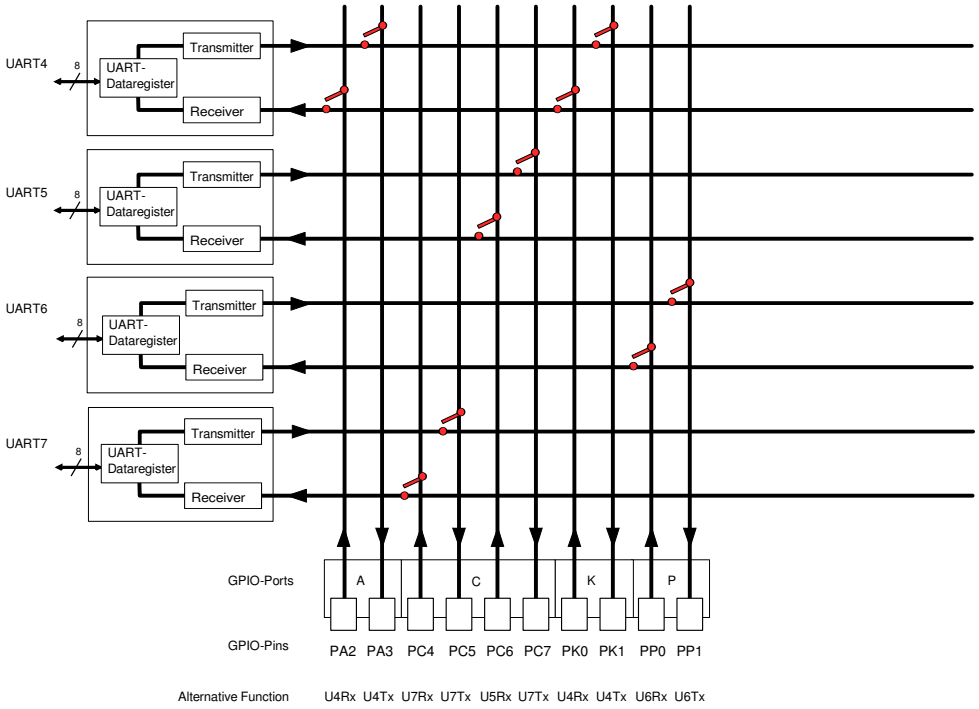
Crossbar zwischen GPIO-Port Pins und UART0-3



Die Schalter werden mit GPIO_PORT_AFSEL_R und GPIO_PORTx_PCTL_R gesetzt.

Source: Texas Instruments TM4C1294 Datasheet June 18, 2014, P. 1163

Crossbar zwischen GPIO-Port Pins und UART4-7



Die Schalter werden mit GPIO_PORT_AFSEL_R und GPIO_PORTx_PCTL_R gesetzt.

Source: Texas Instruments TM4C1294 Datasheet June 18, 2014, P. 1163

GPIO Pin Alternate Port Control GPIO_PORTx_PCTL_R

Table 10-2. GPIO Pins and Alternate Functions (128TQFP)

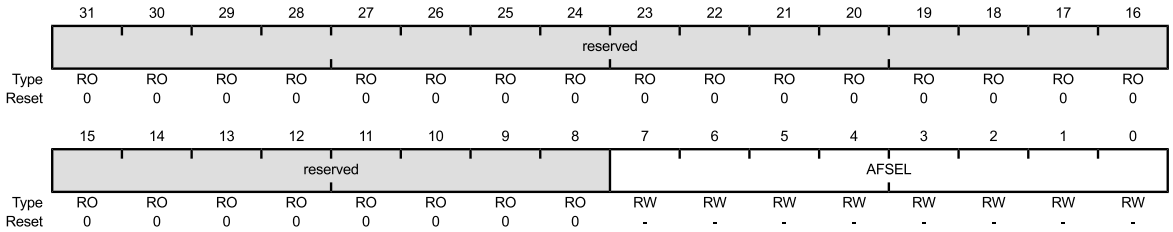
IO	Pin	Analog or Special Function[a]	Digital Function (GPIOPCTL PMCx Bit Field Encoding)[b]											
			1	2	3	4	5	6	7	8	11	13	14	15
PA0	33	-	U0Rx	I2C9SCL	T0CCP0	-	-	-	CAN0Rx	-	-	-	-	-
PA1	34	-	U0Tx	I2C9SDA	T0CCP1	-	-	-	CAN0Tx	-	-	-	-	-
PA2	35	-	U4Rx	I2C8SCL	T1CCP0	-	-	-	-	-	-	-	-	SSI0Clk
			U1RI	U3CTS	I2C2SCL	-	-							
PP0	118	C2+	U6Rx	-	-	-	-	-	-	-	-	-	-	SSI3XDAT2
PP1	119	C2-	U6Tx	-	-	-	-	-	-	-	-	-	-	SSI3XDAT3
PP2	103	-	U0DTR	-	-	-	-	-	-	-	-	-	USB0NXT	EPI0S29
PP3	104	-	U1CTS	U0DCD	-	-	-	-	RTCCLK	-	-	-	USB0DIR	EPI0S30
PQ4	102	-	U1Rx	-	-	-	-	-	DIVSCLK	-	-	-	-	-

Source: Modifed from Pages 743-746 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Die Schalter werden mit GPIO_PORT_AFSEL_R und GPIO_PORTx_PCTL_R gesetzt.

GPIO Alternate Port Function Select GPIO_PORTx_AFSEL_R

GPIO Alternate Function Select (GPIOAFSEL)



Bit/Field	Name	Type	Reset	Description
7:0	AFSEL	RW	-	GPIO Alternate Function Select

Value	Description
0	The associated pin functions as a GPIO and is controlled by the GPIO registers.
1	The associated pin functions as a peripheral signal and is controlled by the alternate hardware function.

The reset value for this register is 0x0000.0000 for GPIO ports that are not listed in Table 10-1 on page 743.

AFSEL: Ein Bit selektiert die alternative digitale Funktion

Source: Texas Instruments TM4C1294 Datasheet June 18, 2014, P.771

Spezifikation des On-Chip R/C Oszillators

Table 27-19. PIOSC Clock Characteristics

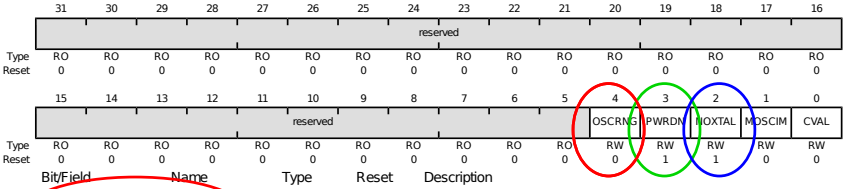
Parameter	Parameter Name	Min	Nom	Max	Unit
F_{PIOSC}	Factory calibration, 0 to +105°C: Internal 16-MHz precision oscillator frequency variance across voltage and temperature range when factory calibration is used	-	-	±4.5%	-
	Factory calibration, -40°C to <0°C	-	-	±10%	-
	Recalibration: Internal 16-MHz precision oscillator frequency variance when recalibration is used at a specific temperature	-	-	±1%	-
T_{START}	PIOSC startup time[a]	-	-	1	µs

a. PIOSC startup time is part of reset and is included in the internal reset timeout value (T_{IRTOUT}) given in Table 27-14 on page 1831. Note that the T_{START} value is based on simulation.

Source: Texas Instruments TM4C1294 Datasheet June 18, 2014, P.1837

Main Oscillator Control SYSCTL_MOSCCTL_R

Main Oscillator Control (MOSCCTL)



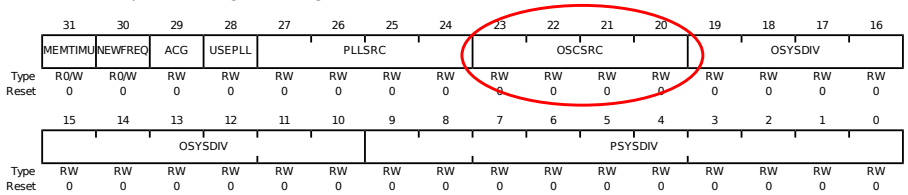
Bit/Field	Name	Type	Reset	Description
4	OSCRNG	RW	0	Specifies the frequency range of operation of the oscillator. Value Description 0 Low Frequency Range 1 High Frequency Range (equal to or greater than 10 MHz).
3	PWRDN	RW	1	Power Down Provides user control over powering down the main oscillator circuit. Value Description 0 Power to main oscillator circuit is enabled. 1 Main Oscillator circuit is powered down.
2	NOXTAL	RW	1	No MOSC/Crystal Connected Provides the user control over the power drawn from the main oscillator circuit. This bit should be set when either crystal or single-ended mode is being used. If the application needs MOSC, this bit should be cleared. Value Description 0 This bit should be cleared when a crystal or oscillator is connected to the OSC0 and OSC1 inputs, regardless of whether or not the MOSC is used or powered down. **Note:** For proper clock functionality when switching to crystal mode, software must clear this bit and set the PWRDN bit in a single write access. 1 This bit should be set when a crystal or external oscillator is not connected to the OSC0 and OSC1 inputs to reduce power consumption.

Source: Modified Page 273 of Texas Instruments M4C1294 Microcontroller Datasheet June 18, 2014

Verfügbare Makros: SYSCTL_MOSCCTL_OSCRNG,..._PWRDN, ... _NOXTAL

Main Oscillator Run Sleep Control Config SYSCTL_RSCLK_CFG_R

Run and Sleep Mode Configuration Register (RSCLKCFG)



Bit/Field	Name	Type	Reset	Description
23:20	OSCSRC	RW	0	Oscillator Source

This field specifies the oscillator source that becomes the oscillator clock (OSCCLK) source, which is used when the PLL is bypassed during run or sleep modes.

Value	Description
0x0	PIOSC is oscillator source
0x1	reserved
0x2	LFIOSC is oscillator source
0x3	MOSC is oscillator source
0x4	Hibernation Module RTC Oscillator (RTCOSC)
0x5-0xFF	reserved

RC-Osc On-Chip approx. 16 MHz

Quartz Osc On-Board precise 25 MHz

Important: When transitioning the system clock configuration to use the MOSC as the fundamental clock source, the PWRDN bit must be set in the MOSCCTL register prior to reselecting the MOSC for proper operation.

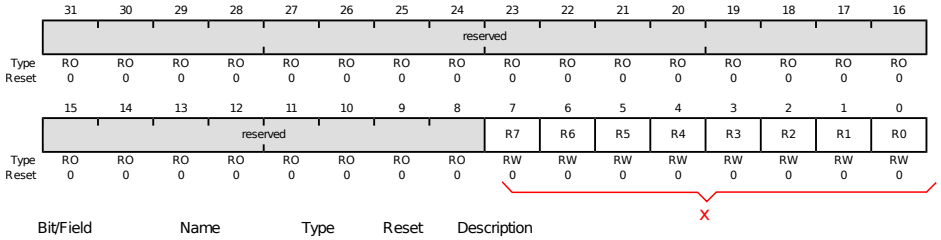
Source: Simplifed from Page 275ff of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Die Taktquelle wird ausgewählt

Verfügbare Makros: SYSCTL_RSCLKCFG_OSCSRC_MOSC,....PIOSC, ...

UART Run Mode Clock Gating Control SYSCTL_RCGCUARTx_R

Universal Asynchronous Receiver/Transmitter Run Mode Clock Gating Control (RCGCUART)



Bit/Field	Name	Type	Reset	Description
31:8	reserved	RO	0	Software should not rely on the value of a reserved bit. To provide compatibility with future products, the value of a reserved bit should be preserved across a read-modify-write operation.
x	Rx	RW	0	UART Module Run Mode Clock Gating Control Value Description 0 UART modulex is disabled. 1 Enable and provide a clock to UART module x in Run mode.

Important: This register should be used to control the clocking for the UART modules.

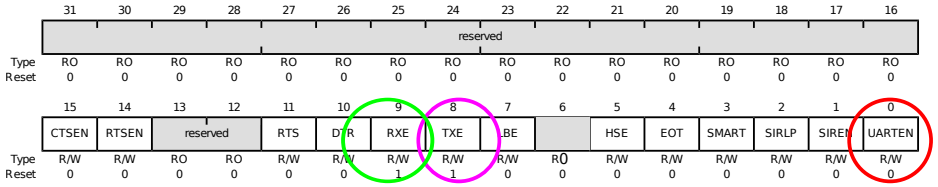
Source: Modifed from Page 388 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Der Takt wird für die UART x an- und abgeschaltet.

Verfügbare Makros: SYSCTL_RCGCUART_Rx

UART Control Register UARTx_CTL_R

UART Control (UARTCTL)



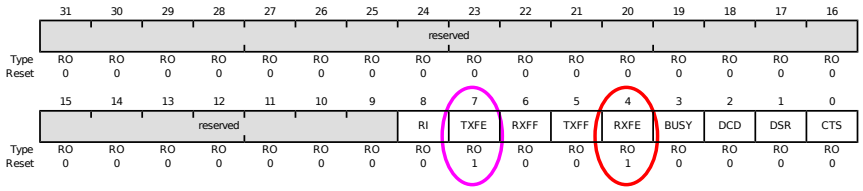
Bit/Field	Name	Type	Reset	Description
9	RXE	R/W	1	UART Receive Enable Value Description 1 The receive section of the UART is enabled 0 The receive section of the UART is disabled ... Note: To enable reception, the UARTEN bit must also be set.
8	TXE	R/W	1	UART Transmit Enable Value Description 1 The transmit section of the UART is enabled. 0 The transmit section of the UART is disabled. ... Note: To enable transmission, the UARTEN bit must also be set.
0	UARTEN	R/W	0	UART Enable Value DescriptionValue 1 The UART is enabled 0 The UART is disabled If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

Source: Simplifed from Page 1188ff of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

UARTEN für UARTx 'enable' und 'disable', Receiver + Transmitter durch Reset 'enabled' gesetzt.

UART Flag Register UARTx_FR_R

UART Flag (UARTFR)



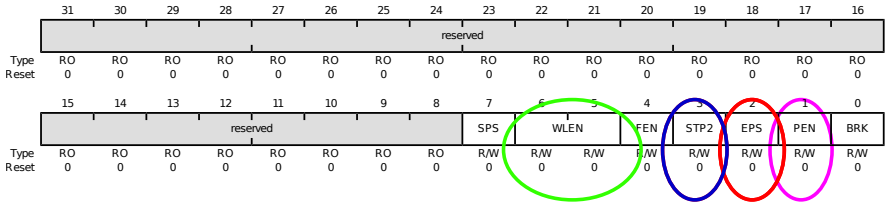
Bit/Field	Name	Type	Reset	Description
7	TXFE	RO	1	UART Transmit FIFO Empty 1 The Transmit FIFO is empty 0 The Transmit FIFO is not empty ...
4	RXFE	RO	1	UART Receive FIFO Empty 1 The Receiver FIFO is empty 0 The Receiver FIFO is not empty

Source: Simplifed from Page 695-696 of Stellaris LM3S9B92 Microcontroller Data Sheet Texas Instrument

Flag TXFE zeigt den freien Sende-FIFO an, Flag RXFE zeigt den freien Empfangs-FIFO an.

UART Line Control Register UARTx_LCRH_R

UART Line Control (UARTLCRH)



Bit/Field	Name	Type	Reset	Description
6:5	WRLEN	R/W	0x0	UART Word Length 0x0 5 Bits (Default), 0x1 6 Bits 0x2 7 Bits 0x3 8 Bits
3	STP2	R/W	0	UART Two Stop Bit Select ... 0 One stop bit ... 1 Two stop bits ...
2	EPS .	R/W	0	UART Even Parity Select ... 0 Odd parity ... generates / checks for an odd number of 1s. 1 Even parity ... generates/checks for an even number of ones in data and parity bits
1	PEN	R/W	0	UART Parity Enable 0 Parity is disabled and no parity bit is added to the frame . 1 Parity checking and generation is enabled.

Source: Simplifed from Page 1186 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Mit dem 2-Bit-Feld WRLEN und den Bits STP2, EPS, PE wird die jeweilige Protokolloption (z.B. 7E2, 8N1, 8O1) eingestellt.

Konfiguration der Bitrate

Generierung des Transmit-Clocks und des Reveive Clocks durch Teilen des der UART zugeführten Systemtaktes *SysClk* durch den festen Wert 16 und dem Baudrate Divisor *BRD*, der wie folgt berechnet wird:

$$BRD = \frac{SysClk}{16 \cdot \text{gewünschte Bitrate}}$$

Hinweis: BRD ist als Zwischenergebnis eine rationale Zahl (Bruchzahl).

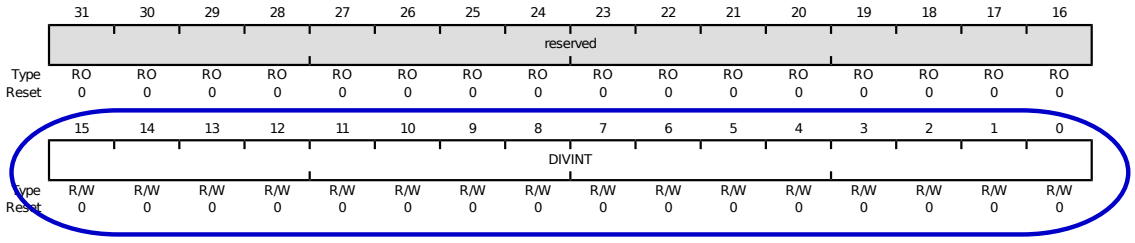
Die UART verfügt über einen Fractional Clock Divider.
Die Bitrate ist mit zwei Registern einstellbar, ein Register für den Divisor-Teil vor dem Komma (16 Bit) und ein Register für den Teil nach dem Komma (6 Bit).

→ Integer Teil: $IBRD = int(BRD)$
Hinweis: int() wirkt 'abschneidend' nicht rundend !

→ Gebrochener Teil: $FBRD = int((BRD - int(BRD)) * 64 + 0.5)$
Der gebrochene Teil wird damit auf einen 6 Bit Ganzzahl-Wert umgerechnet.

UART Integer Baud Rate Divisor Register UARTx_IBRD_R

UART Integer Baud-Rate Divisor (UARTIBRD)



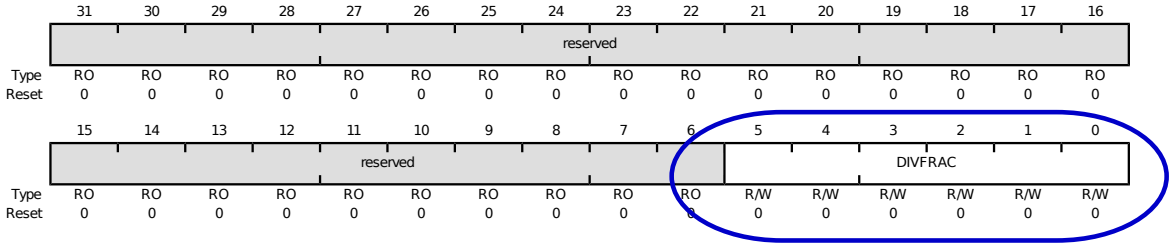
Bit/Field	Name	Type	Reset	Description
15:0	DIVINT	R/W	0x0000	Integer Baud-Rate Divisor

Source: Simplifed from Page 1175 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Mit dem 16-Bit-Feld DIVINT der ganzzahlige Teil des Bitraten-Divisors eingestellt

UART Fractional Baud Rate Divisor Register UARTx_FBRD_R

UART Fractional Baud-Rate Divisor (UARTFBRD)



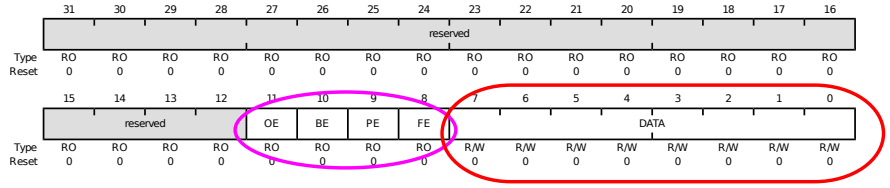
Bit/Field	Name	Type	Reset	Description
5:0	DIVFRAC	R/W	0x00	Fractional Baud-Rate Divisor

Source: Simplifed from Page 1176 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

Mit dem 6-Bit-Feld DIVFRAC wird gebrochene Teil des Bitraten-Divisors eingestellt

UART Data Register UARTx_DR_R

UART Data (UARTDR)



Bit/Field	Name	Type	Reset	Description
11	OE	RO	0	UART Overrun Error New data was received when the FIFO was full, resulting in data loss.
10	BE	RO	0	UART Break Error A break condition has been detected, indicating that the receive data input was held low for longer than a full-word transmission time (defined as start, data, parity, and stopbits).
9	PE	RO	0	UART Parity Error The parity of the received data character does not match the parity defined by bits 2 and 7 of the UART_LCRH_R register.
8	FE	RO	0	UART Framing Error The received character does not have a valid stop bit (a valid stop bit is 1)
7:0	DATA	R/W	0x00	Data Transmitted or Received Data that is to be transmitted via the UART is written to this field. When read, this field contains the data that was received by the UART.

Source: Simplifed from Page 1175 of Texas Instruments TM4C1294 Microcontroller Datasheet June 18, 2014

DATA zum Schreiben und Lesen des FIFO, Fehler-Status-Bits OE,BE,PE und FE nur beim Receiver

Test UART Transmission 1 of 2

```
1  // Test serial transmission via UART6
2  // The bitfield macros are available in the headerfile tm4c1294ncpdt.h
3  // In CCS-Editor: place cursor at the macro and press F3
4
5  #include <stdint.h>
6  #include <stdio.h>
7  #include "inc/tm4c1294ncpdt.h"
8
9  #define BITDATA 0x54                // testdata for transmission
10 #define WAITLOOPLENGTH 2000         // duration of a simple wait loop
11
12
13
14 void main(void){
15     static volatile int i=0;    // static volatile not neccessary by optimization=off
16
17     // switch over to main quartz oscillator at 25MHz
18     // clear MOSC power down, high oscillator range setting, and no crystal present setting
19     SYSCTL_MOSCCTL_R &= ~(SYSCTL_MOSCCTL_OSCRNG | SYSCTL_MOSCCTL_PWRDN |
         SYSCTL_MOSCCTL_NOXTAL);
20     SYSCTL_MOSCCTL_R |= SYSCTL_MOSCCTL_OSCRNG;  // increase the drive strength for MOSC
21     SYSCTL_RSCLKCFG_R = SYSCTL_RSCLKCFG_OSCSRC_MOSC;// set the main oscillator as main clock
         source
22
23     SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R13;   // enable clock for PORTP
24     SYSCTL_RCGCUART_R |= SYSCTL_RCGCUART_R6;    // enable clock for UART6
25
26     // initialize Port P
```


Test UART Transmission 2 of 2

```
26      // initialize Port P
27      GPIO_PORTP_DEN_R |= 0x2;     // enable digital pin function for PP1
28      GPIO_PORTP_DIR_R |= 0x2;     // set PP1 to output
29      GPIO_PORTP_AFSEL_R |= 0x2;  // switch to alternate pin function PP1
30      GPIO_PORTP_PCTL_R |= 0x10;  // select alternate pin function PP1->U6Tx
31
32      // initialize UART6
33      UART6_CTL_R &= ~UART_CTL_UARTEN;     // disable UART6 during initialization
34
35      // initialize bitrate of 115200 bit per second
36      UART6_IBRD_R = 13;  // set DIVINT of BRD
37      UART6_FBRD_R = 36;  // set DIVFRAC of BRD
38
39      UART6_LCRH_R = UART_LCRH_WLEN_8;     // set serial format to 8N1
40      UART6_CTL_R |= UART_CTL_UARTEN| UART_CTL_TXE;   // re-enable UART6 for transmission
41
42      while(1){                            // for ever loop
43          while(!(UART6_FR_R & UART_FR_TXFE)); // wait for flag:  till transmit FIFO empty
44          UART6_DR_R = BITDATA;            // write data to transmit register
45          for (i=0;i<WAITLOOPLENGTH ;i++);// wait loop between frames (only for scope)
46          printf("*\n");                    // output simple activty marks at console
47      }
48  }
```


Test UART Receiver 1 of 2

```
1   /// Test serial receiver via UART6
2   // The bitfield macros are available in the headerfile tm4c1294ncpdt.h
3   // In CCS-Editor: place cursor at the macro and press F3
4
5   #include <stdint.h>
6   #include <stdio.h>
7   #include "inc/tm4c1294ncpdt.h"
8
9   #define BUFFERLENGTH   16           // length data buffer for input data
10
11  void main(void){
12      static volatile int i=0;    // static volatile not neccessary by optimization=off
13      char c, buffer[BUFFERLENGTH];
14
15      // switch over to main quartz oscillator at 25MHz
16      // clear MOSC power down, high oscillator range setting, and no crystal present setting
17      SYSCTL_MOSCCTL_R &= ~(SYSCTL_MOSCCTL_OSCRNG | SYSCTL_MOSCCTL_PWRDN |
          SYSCTL_MOSCCTL_NOXTAL);
18      SYSCTL_MOSCCTL_R |= SYSCTL_MOSCCTL_OSCRNG;  // increase the drive strength for MOSC
19      SYSCTL_RSCLKCFG_R = SYSCTL_RSCLKCFG_OSCSRC_MOSC;// set the main oscillator as main clock
          source
20
21      SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_R13;   // enable clock for PORTP
22      SYSCTL_RCGCUART_R |= SYSCTL_RCGCUART_R6;    // enable clock for UART6
23
24      // initialize Port P
25      GPIO_PORTP_DEN_R |= 0x1;    // enable digital pin function for PP0
26      GPIO_PORTP_DIR_R &= ~0x1;   // set PP0 to input
```


```
27      GPIO_PORTP_AFSEL_R |= 0x1;  // switch to alternate pin function PP0
28      GPIO_PORTP_PCTL_R |= 0x1;   // select alternate pin functions PP0->U6Rx
29
30      // initialize UART6
31      UART6_CTL_R &= ~UART_CTL_UARTEN;    // disable UART6 during initialization
32
33      // initialize bitrate of 115200 bit per second
34      UART6_IBRD_R = 13;  // set DIVINT of BRD  int 13
35      UART6_FBRD_R = 36;  // set DIVFRAC of BRD int(36,553)
36      UART6_CTL_R |= UART_CTL_UARTEN |  UART_CTL_RXE ;    // re-enable UART6
37
38      do{
39      while(i < BUFFERLENGTH){            // loop while buffer not full
40
41          while(UART6_FR_R & 0x10);       // wait for flag:  WHILE receive FIFO empty
42                                          // = till a data frame are received
43          c = UART6_DR_R;                   // read byte from UART6 data register
44          if (c == 0x04) break;           // stop loop if "EOT" (End of Tramission) received
45                                          // Console input for EOT = Strg+D
46          buffer[i]=c;                    // Copy byte at buffer index position i
47          i++;                            // read byte counter increment
48      }                                  // repeat receiving
49      buffer[i]=0x00;                   // Set 0 (Zero character) at the and of the string
50                                        // as ususal for character-strings in c
51      // output the received data buffer as string to console
52      printf("\n Content of Data Buffer \n%s\n",buffer);
53      i=0;
54      }
55      while(1); // For ever loop
56  }
```