UART: Ein Hardware-Kommunikationsprotokoll das universelle asynchrone Empfänger/Sender verstehen

UART, oder Universal Asynchronous Receiver-Transmitter, ist eines der am häufigsten genutzten Kommunikationsprotokolle von Baustein zu Baustein. Dieser Artikel beschreibt, wie man UART als ein Hardware-Kommunikationsprotokoll nutzt, indem man dem Standardverfahren folgt.

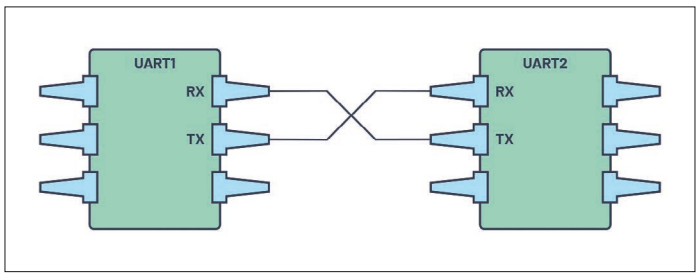

Bild 1: Zwei UARTs kommunizieren direkt miteinander

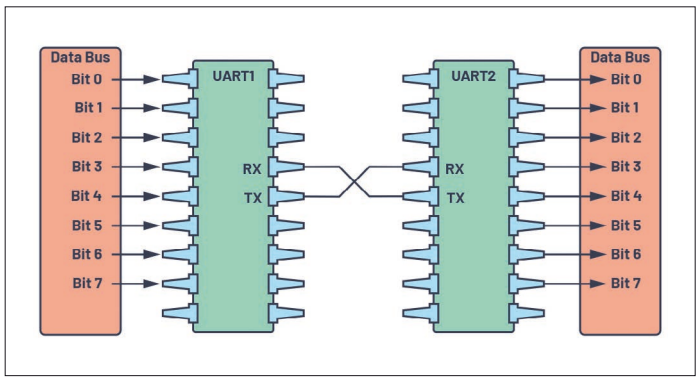

Bild 2: UART mit Datenbus

Sachgemäß konfiguriert kann UART mit vielen unterschiedlichen Arten von seriellen Protokollen arbeiten, die serielle Daten senden und empfangen. In der seriellen Kommunikation werden Daten über eine einzige Leitung oder ein Kabel Bit für Bit übertragen. In der wechselseitigen Kommunikation (Übertragung der Signale in beide Richtungen) nutzt man zwei Leitungen zur erfolgreichen seriellen Datenübertragung. Abhängig von der Anwendung und den Systemanforderungen benötigt die serielle Datenkommunikation weniger Schaltungen und Leitungen, was die Kosten der Implementierung senkt.

In diesem Artikel werden die grundlegenden Prinzipien beim Einsatz von UART untersucht. Dabei liegt der Fokus auf der Übertragung von Datenpaketen, dem Standard-Frame-Protokoll und kundenspezifischen Frame-Protokollen, die Eigenschaften mit einem Mehrwert oder das Einhalten von Sicherheitsvorschriften besonders während der Code-Entwicklung bieten, wenn sie implementiert sind. Während der Produktentwicklung zielt dieses Dokument auch darauf ab, einige grundlegende Schritte zu definieren, wie man ein Datenblatt für die aktuelle Nutzung liest. Am Schluss des Artikels ist es das Ziel, ein besseres Verständnis der Einhaltung von UART-Standards zu gewährleisten, um die Fähigkeiten des Systems und der Anwendung besonders bei der Entwicklung neuer Produkte zu maximieren.

Autoren:
Eric Peña (links),
Senior Firmware Engineer und
Mary Grace Legaspi (rechts),
Firmware Engineer

Analog Devices Inc.
www.analog.com

Richtige Kommunikation

„Das größte Problem der Kommunikation ist die Illusion, dass sie stattgefunden hat“ (George Bernard Shaw)

Das Kommunikationsprotokoll spielt eine große Rolle beim Organisieren der Kommunikation zwischen Bausteinen. Es wird basierend auf den Systemanforderungen auf unterschiedliche Weise entwickelt und diese Protokolle definieren spezielle Regeln zwischen den Bausteinen, um eine erfolgreiche Kommunikation zu erzielen.

Eingebettete Systeme, Mikrocontroller und Computer nutzten am häufigsten UART als Protokoll für die Hardwarekommunikation von Baustein zu Baustein. Unter den verfügbaren Kommunikationsprotokollen nutzt UART nur zwei Leitungen für seine Übertragungs- und Empfangsseiten. Obwohl es ein weit verbreitetes Hardwarekommunikationsprotokoll ist, ist es noch immer nicht vollständig optimiert. Eine geeignete Implementierung des Frame-Protokolls wird meist außer Acht gelassen, wenn man die UART-Module im Mikrocontroller nutzt.

Per Definition ist UART ein Hardwarekommunikationsprotokoll, das die asynchrone serielle Kommunikation mit konfigurierbarer Geschwindigkeit nutzt. Asynchron bedeutet, dass es kein Taktsignal gibt, das den ausgehenden Bitstrom vom sendenden Baustein mit der Empfangsseite synchronisiert.

Schnittstelle

Bild 1 zeigt zwei UARTs, die direkt miteinander kommunizieren. Die bei-

Leitungen	2
Geschwindigkeit	9600, 19.200, 38.400, 57.600, 115.200, 230.400, 460.800, 921.600, 1.000.000, 1.500.000
Übertragungsarten	Asynchron
Maximale Anzahl von Mastern	1
Maximale Anzahl von Slaves	1

Tabelle 1: UART-Zusammenfassung

Bild 3: UART-Datenpaket

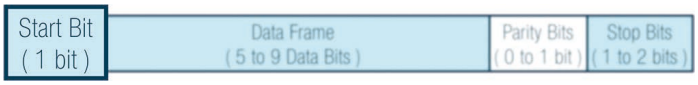

Bild 4: Start-Bit

Bild 5: Datenrahmen

Bild 6: Prüf-Bit

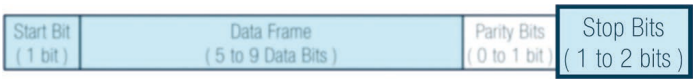

Bild 7: Stopp-Bit

den Signale eines UART-Bausteins werden folgendermaßen bezeichnet:

- Sender (Tx)
- Empfänger (Rx)

Die Hauptaufgabe einer Empfänger-/Sender-Leitung für jeden Baustein ist es, serielle Daten zu übertragen und zu empfangen, um eine serielle Kommunikation zu realisieren.

Bild 2 zeigt einen UART mit Datenbus. Der sendende UART ist mit einem steuernden Datenbus verbunden, der Daten in paralleler Form sendet. Anschließend werden die Daten über die Übertragungsleitung (Kabel) seriell Bit für Bit an den empfangenden UART gesendet. Dieser wandelt dann diese seriellen Daten für das empfangende Bauteil wieder in parallele um. Die UART-Leitungen dienen als Kommunikationsmedium, um Daten untereinander zu senden und zu empfangen. Dabei ist zu beachten, dass ein UART-Baustein einen Sende- und Empfangs-Pin hat, der entweder für das Senden oder das Empfangen dient.

Für UART sowie den Großteil der seriellen Kommunikation muss die Baud-Rate am sendenden als auch empfangenden Baustein gleich eingestellt werden. Dies ist die Rate, mit der Information an einen Kommunikationskanal gesendet wird. Bei einem seriellen Port ist die eingestellte Baud-Rate die maximale Anzahl an Bit pro Sekunde, die übertragen werden kann.

Grunddaten UART

Tabelle 1 fasst zusammen, was man über UART wissen muss.

Das UART-Interface nutzt kein Taktsignal, um die Sende- und Empfangsbausteine miteinander zu synchronisieren, es überträgt die Daten asynchron. Anstelle eines Taktsignals generiert der Sendebaustein basierend auf seinem Taktsignal einen Bitstrom, während der Empfänger sein internes Taktsignal verwendet, um die eintreffenden Daten abzutasten. Die Synchronisation wird dadurch erreicht, dass beide Bausteine dieselbe Baudrate haben. Fehler, die dabei auftreten, können das Timing beim Senden und Empfangen der Daten beeinträchtigen und Diskrepanzen während der Datenübertragung hervorrufen. Die erlaubte Abweichung der Baudrate, bevor das Timing der Bit zu weit auseinanderdriftet, beträgt bis zu 10 %.

Datenübertragung

Im UART werden die Daten in Form von Paketen übertragen. Das Teil, das den Sender mit dem Empfänger verbindet, übernimmt die Erstellung von seriellen Datenpaketen und steuert die physikalischen Hardware-Leitungen. Ein Datenpaket besteht aus Start-Bit, Datenrahmen, Prüf-Bit und Stopp-Bit(s) (Bild 3).

Start-Bit

Die UART-Übertragungsleitung hat, wenn sie keine Daten übermittelt, normalerweise einen hohen Spannungspegel. Um die Datenübertragung zu starten, zieht der sendende UART die Übertragungsleitung für einen Taktzyklus von high auf low (Bild 4). Wenn der empfangende UART diesen Übergang von high auf low feststellt, beginnt er die Bit im Datenrahmen mit der Frequenz der Baudrate zu lesen.

Datenrahmen

Der Datenrahmen (Bild 5) enthält die aktuellen Daten, die übertragen werden. Diese können fünf bis acht Bit lang sein, wenn ein Prüf-Bit verwendet wird. Wird kein Prüf-Bit genutzt, kann der Datenrahmen neun Bit enthalten. In den meisten Fällen werden die Daten mit dem Bit mit dem niedrigsten Stellenwert zuerst gesendet.

Parität

Die Parität beschreibt, ob eine Zahl geradzahlig oder ungerade ist. Das Prüf-Bit (Bild 6) ist ein Weg für den

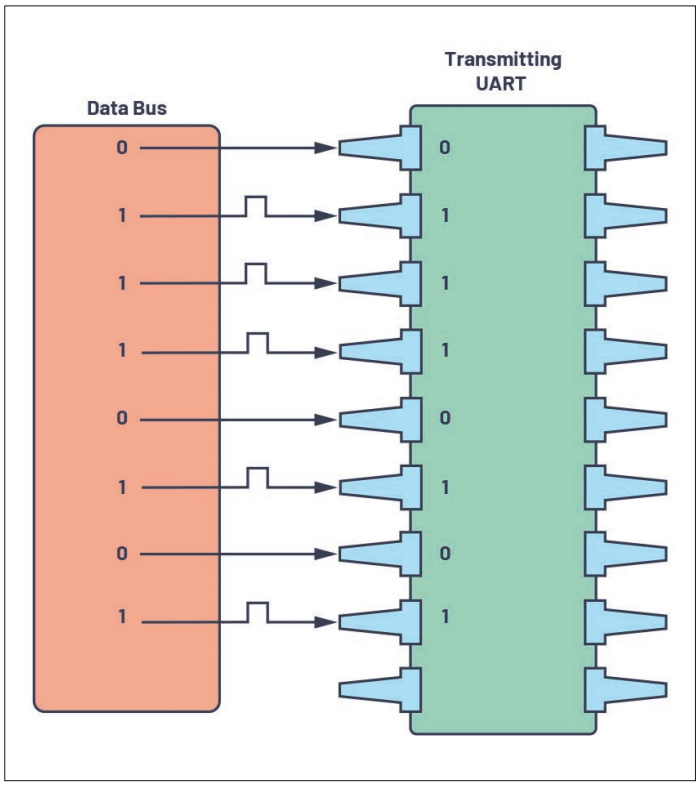

Bild 8: Datenbus für den sendenden UART

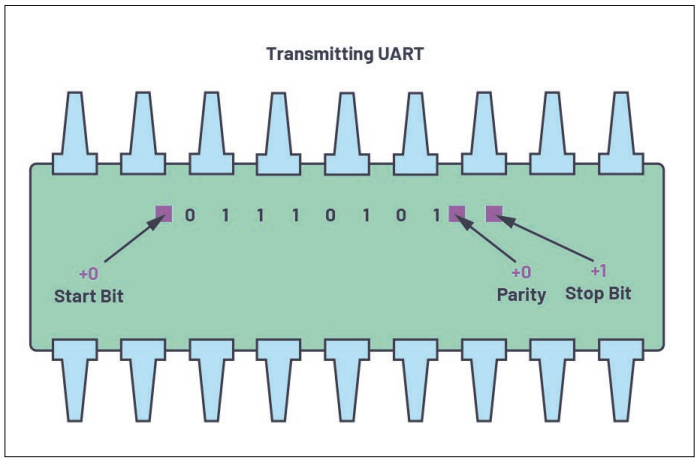

Bild 9: UART-Datenrahmen auf der Tx-Seite

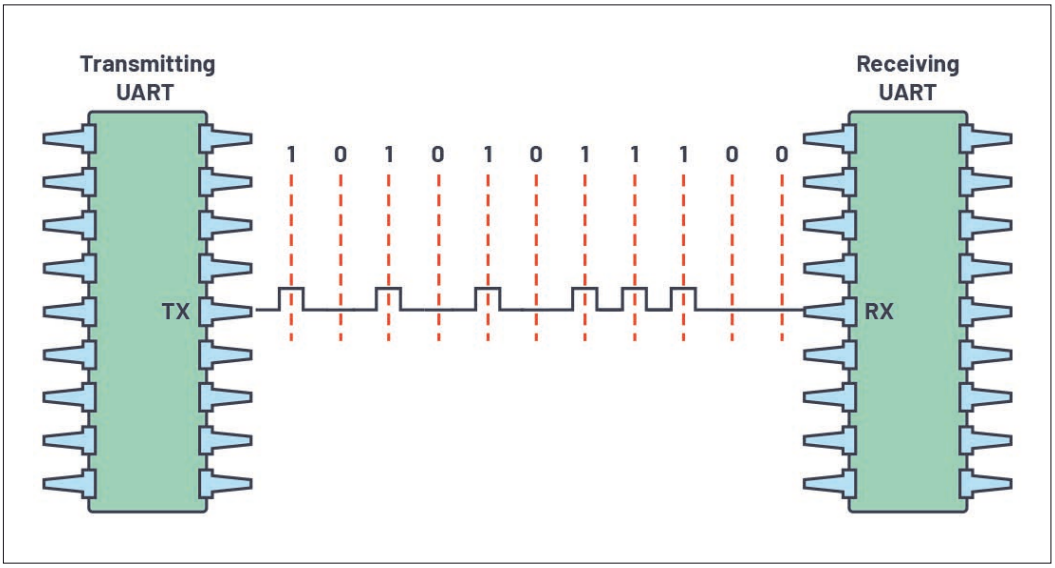

Bild 10: UART-Datenübertragung

empfangenden UART mitzuteilen, ob Daten während der Übertragung verändert wurden. Bit können durch elektromagnetische Strahlung, falsch angepasste Baudraten oder Datenübertagungen über lange Entfernungen geändert werden.

Nachdem der empfangende UART den Datenrahmen gelesen hat, zählt er die Anzahl der Bit mit dem Wert 1 und prüft, ob ihre Gesamtzahl gerade oder ungerade ist. Wenn das Prüf-Bit 0 ist (gerade Parität) sollten die Einsen oder logisch hohen Bits insgesamt eine gerade Anzahl haben. Ist das Prüf-Bit eine 1 (ungerade Parität) sollten die Einer-Bit oder logischen High-Zustände insgesamt eine ungerade Anzahl haben.

Wenn das Prüf-Bit mit den Daten übereinstimmt, weiß der UART, dass die Datenübertragung fehlerfrei war. Ist das Prüf-Bit jedoch eine 0 und die Gesamtanzahl der Bit ist ungerade oder das Prüf-Bit ist eine 1 und die Gesamtanzahl ist gerade, dann weiß der UART, dass sich Bit im Datenrahmen geändert haben.

Stopp-Bit

Um das Ende eines Datenpakets anzuzeigen, treibt der sendende UART die Übertragungsleitung für die Dauer von einem (1) oder zwei (2) Bit von einer geringen (low) auf eine hohe (high) Spannung (Bild 7).

Baudrate	OSR	DIV	DIVM	DIVN
9.600	3	24	3	1.078
115.200	3	4	1	1.563

Tabelle 2: Baudraten-Beispiel 26-MHz-PCLK

Baudrate	OSR	DIV	DIVM	DIVN
9.600	3	17	3	1.078
115.200	3	2	2	348

Tabelle 3: Baudraten-Beispiel 16-MHz-PCLK

Name	Beschreibung
UART_DIV	Baudratenteiler
UART_FIBR	Teil-Baudrate (Fractional Baud Rate)
UART_LCR2	Steuerung der zweiten Leitung (Second Line Control)

Tabelle 4: UART-Register-Beschreibung

Die Schritte der UART-Datenübertragung

Als erstes empfängt der sendende UART parallele Daten vom Datenbus. Bild 8 zeigt einen Datenbus für den sendenden UART. Als zweites fügt der sendende UART Start-, Prüf- und Stopp-Bit(s) in den Datenrahmen ein (Bild 9).

Als dritter Schritt wird das gesamte Datenpaket, angefangen mit dem Start- bis zu den Stopp-Bit(s) seriell vom sendenden an den empfangenden UART übertragen (Bild 10). Der empfangende UART tastet dabei die Datenleitung mit der vorkonfigurierten Baudrate ab.

Viertens löscht der empfangende UART die Start-, Prüf – und Stopp-Bit(s) aus dem Datenrahmen. (Bild 11).

Im fünften und letzten Schritt wandelt der empfangende UART die seriellen Daten wieder in parallele um und sendet sie an den Datenbus auf der Empfangsseite (Bild 12).

Rahmenprotokoll

Eine in UART verfügbare Schlüsseleigenschaft, die jedoch meist nicht voll genutzt wird, ist die Implementierung eines Rahmenprotokolls. Der Hauptnutzen dieses Protokolls und seine Bedeutung sind ein Mehrwert für die Sicherheit und den Schutz jeden Bausteins.

Wenn z. B. zwei Bausteine dasselbe UART-Rahmenprotokoll verwenden, besteht die Möglichkeit, dass, wenn man sie, ohne die Konfiguration zu prüfen, mit demselben UART verbindet, die Bausteine mit unterschiedlichen Pins verbunden werden, was zu einem Fehlverhalten im System führen kann. Andererseits gewährleistet die Implementierung Sicherheit, aufgrund der Notwendigkeit die empfangene Information in Übereinstimmung mit dem entwickelten Rahmenprotokoll zu analysieren. Jedes Rahmenprotokoll ist so entwickelt, dass es eindeutig und sicher ist.

Die Gestaltung eines Rahmenprotokolls erlaubt es den Entwicklern die gewünschten Header und Trailer, inklusive CRC, für unterschiedliche Bausteine einzusetzen. In Bild 13 sind zwei Byte als Teil des Headers eingefügt.

Basierend auf diesem Beispiel, kann man Header, Trailer und CRC, die eindeutig für den Baustein sind, einsetzen.

Header 1 (H1 ist 0xAB) und Header 2 (H2 ist 0xCD)

Der Header ist der eindeutige Identifizierer, der bestimmt, ob man mit dem korrekten Baustein kommuniziert.

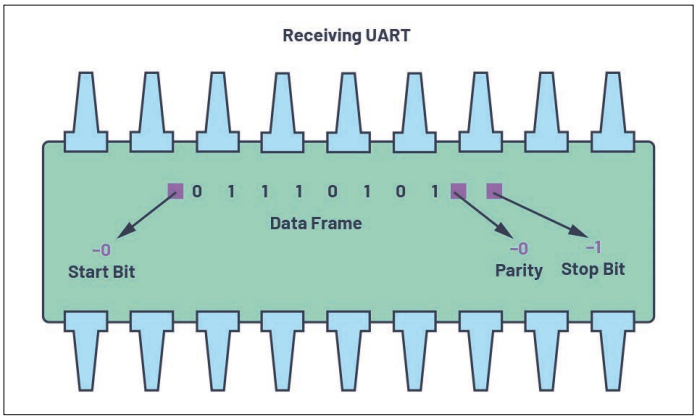

Bild 11: Der UART-Datenrahmen auf der Rx-Seite

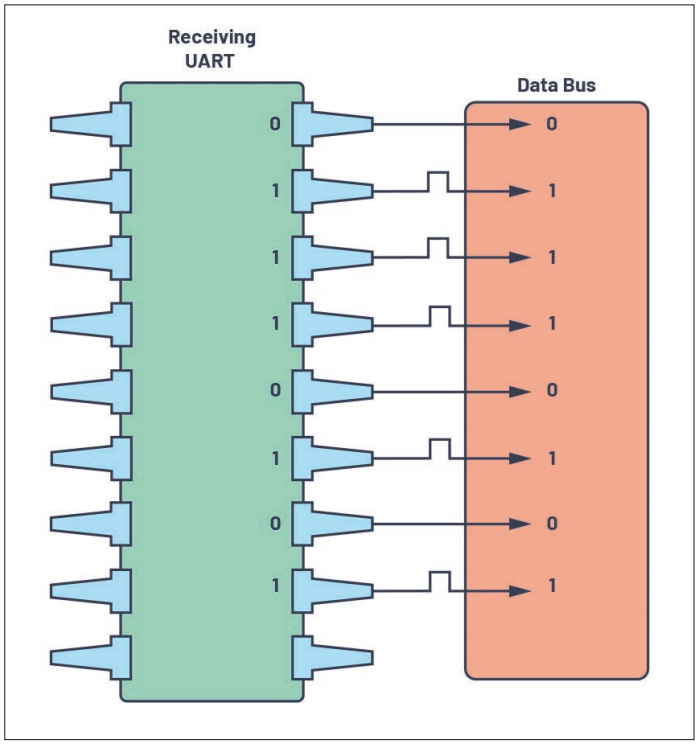

Bild 12: Datenübertragung vom UART zum Datenbus

Befehlsauswahl (Command Selection, CMD)

Sie hängt von der Liste der Befehle ab, die entwickelt wurden, um die Kommunikation zwischen zwei Bausteinen zu ermöglichen.

Datenlänge (DL) pro Befehl

Die Datenlänge basiert auf dem gewählten Befehl. Abhängig vom gewählten Befehl kann man die Länge der Daten maximieren, so dass sie entsprechend der Auswahl variieren kann. In diesem Fall kann die Datenlänge variiert werden.

Daten n (Variierende Daten)

Die Daten sind der Inhalt, der von den Bausteinen übertragen wird.

Trailer 1 (T1 ist 0xE1) und Trailer 2 (T2 ist 0xE2)

Trailer sind Daten, die hinzugefügt werden, nachdem die Übertragung beendet ist. Wie die Header können sie eindeutig identifiziert werden.

Zyklische Redundanzprüfung (CRC)

Die zyklische Redundanzprüfung (Cycling Redundancy Checking) ist ein zusätzlicher Modus zur Fehlererkennung, um unbeabsichtigte Änderungen in den Rohdaten zu entdecken. Der CRC-Wert des sendenden Bausteins muss immer gleich der CRC-Berechnung auf der Empfangseite sein.

Es ist empfehlenswert Sicherheit durch das Implementieren von Rahmenprotokollen für jeden UART-Baustein einzufügen. Das Rahmenprotokoll benötigt sowohl auf den sendenden als auch empfangenden Bausteinen identische Konfigurationen.

UART-Betrieb

Wenn man ein beliebiges Hardware-Kommunikationsprotokoll einsetzt, dann ist es Voraussetzung, das Datenblatt und das Hardware-Referenzhandbuch zu studieren.

Hier folgen die Schritte, die man dabei einhalten sollte.

1. Prüfen Sie die Schnittstelle des Bausteins im Datenblatt (Bild 14).
2. Prüfen Sie die UART-Adresse im Speicherabbild (Bild 15).
3. Überprüfen Sie die speziellen Details für den UART-PORT, wie die Betriebsart, die Länge der Daten, das Prüf- und die Stopp-Bit(s). Details eines Beispiel-UART-Ports findet man im Datenblatt: UART-Port.

Die Beispiel-MCU hat einen Vollduplex-UART-Port, der vollständig kompatibel mit den Standard-PC-UARTs ist. Der UART-Port bietet ein vereinfachtes UART-Interface zu weiterer Peripherie oder Host-Bausteinen und unterstützt vollduplex, DMA und den asynchronen Transfer von seriellen Daten. Der UART-Port unterstützt fünf bis acht Datenbit und keine, geradzahlige oder ungeradzahlige Parität. Ein Rahmen wird von eineinhalb oder zwei Stopp-Bit abgeschlossen.

4. Prüfen Sie die Details der Betriebsart des UARTs, einschließlich der Baudraten-Berechnung. Die Baudrate wird wie in der folgenden Beispielformel dargestellt. konfiguriert. Diese Formel ist vom jeweiligen Mikrocontroller abhängig.

Details des Betriebs des Beispiel-UARTs:

- 5 bis 8 Datenbit
- 1, 2, oder 1½ Stopp-Bit
- Keine, geradzahlige oder ungeradzahlige Parität
- Programmierbare Überabtastrate von 4, 8, 16 und 32
- Baudrate = PCLK/((M + N/2048) × 2^{OSR+2} × DIV

mit

OSR (Überabtastrate)
UART_LCR2.OSR = 0 bis 3
DIV (Baudratenteiler)
UART_DIV = 1 bis 65535
M (DIVM Teil-Baudrate M)
UART_FBR.DIVM = 1 bis 3
N (DIVM Teil-Baudrate M)
UART_FBR.DIVN = 0 bis 2047

5. Stellen Sie für die Baudrate sicher, dass überprüft wird, welcher periphere Takt (PCLK) verwendet wird. In diesem Beispiel sind ein 26-MHz- und ein 16-MHz-PCLK verfügbar. Beachten Sie dabei, dass OSR, DIV, DIVM und DIVN vom jeweiligen Baustein abhängen.

Baudraten-Beispiel, basierend auf dem 26-MHz-PCLK (Tabelle 2)

Baudraten-Beispiel, basierend auf dem 16-MHz-PCLK (Tabelle 3)

6. Checken Sie im nächsten Schritt die detaillierten Register für die UART-Konfiguration. Dazu betrachtet man die Parameter zur Berechnung der Baudrate, wie UART_LCR2, UART_DIV und UART_FBR. Tabelle 4 führt zu einem bestimmten Register, um dies abzudecken.

UART-Register-Beschreibung

7. Überpfüren Sie in jedem Register die Details und ersetzen Sie die Werte zum Berechnen der Baudrate. Danach wird die Implementierung des UARTs gestartet.

Header 1	Header 2	Command	Data Length	Data n… Data n+1	Trailer 1	Trailer 2	Cyclic Redundancy Checking

Bild 13: Beispiel eines UART Rahmenprotokolls

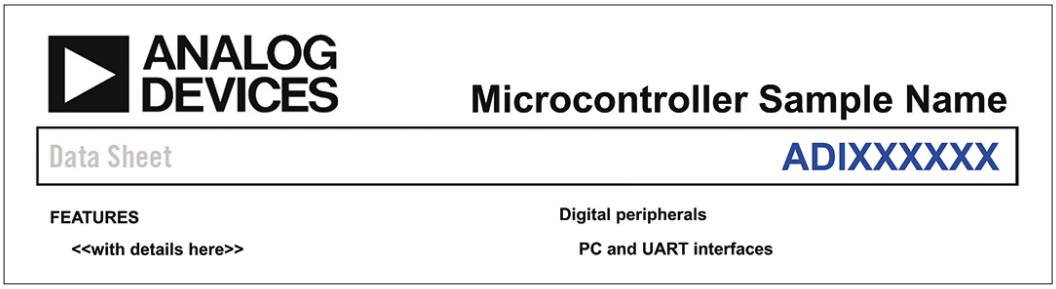

Bild 14: Datenblatt eines Mikrocontrollers

Warum ist dies wichtig?

Die Vertrautheit mit dem UART-Kommunikationsprotokoll ist vorteilhaft, um robuste und qualitativ hochwertige Produkte zu entwickeln. Wenn man weiß, wie man Daten über nur zwei Leitungen überträgt, aber auch, wie man ein ganzes Datenpaket oder Informationen sendet, hilft dies sicherzustellen, dass die Daten auch fehlerfrei gesendet und empfangen werden. Da UART das am häufigsten eingesetzte Hardware-Kommunikationsprotokoll ist, kann dieses Wissen eine hohe Designflexibilität in künftigen Entwicklungen ermöglichen.

Anwendungen

Man kann UART in vielen Applikationen einsetzen, beispielsweise:

- Debugging: Das frühe Erkennen von Systemfehlern während der Entwicklung ist wichtig. Der Einsatz von UART kann in diesem Szenario durch das Empfangen von System-Meldungen helfen.
- Nachverfolgen der Funktion in der Fertigung: Protokolle (Logs) sind in der Fertigung sehr wichtig. Sie bestimmen den Ablauf, indem sie Bediener informieren, was in der Fertigungslinie vorgeht.
- Kunden- oder Client-Updates: Softwareaktualisierungen sind extrem wichtig. Um ein komplettes System zu erhalten, ist es wichtig, eine vollständige, dynamische Hardware mit einer aktualisierungsfähigen Software einzusetzen.
- Testen/Verifikation: Das Verifizieren der Produkte bevor sie den Herstellungsprozess verlassen, hilft dabei, eine hohe Produktqualität sicherzustellen.

Referenzen

- "Basics of UART Communication." Electronics Hub, Juli 2017.
- Campbell, Scott. "Basics of UART Communication." Circuit Basics. Keim, Robert.
- "Back to Basics: The Universal Asynchronous Receiver/ Transmitter." All About Circuits, Dezember 2016.
- "What Is UART Protocol? UART Communication Explained." Arrow. ◀

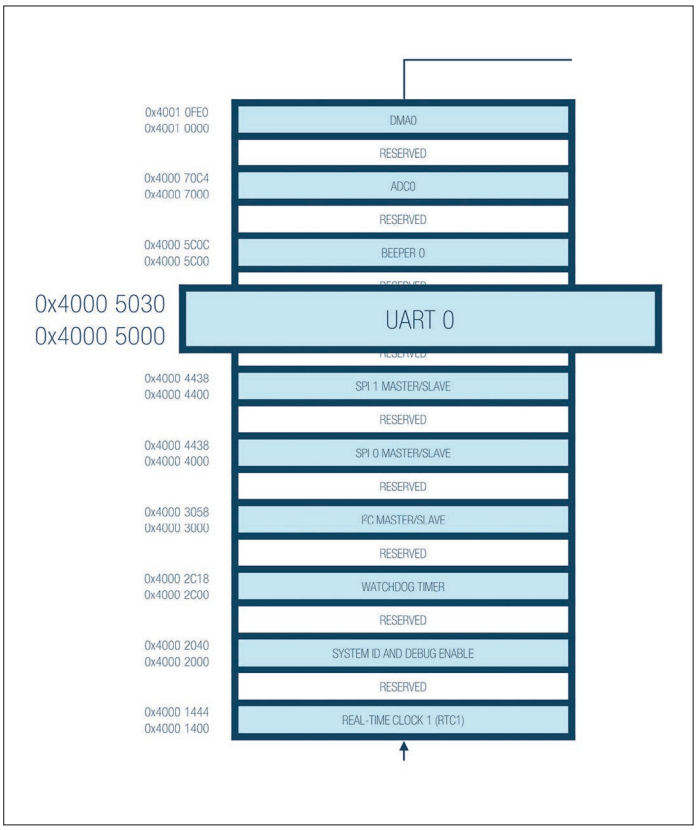

Bild 15: Speicherabbild eines Mikrocontrollers