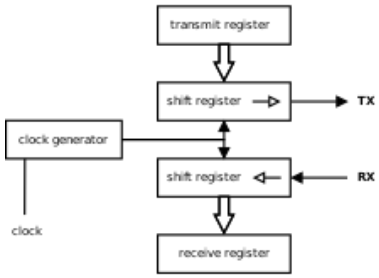
WIKIPEDIA

Universal asynchronous receiver-transmitter

A **universal asynchronous receiver-transmitter** (**UART** /ˈjuːɑːrt/) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. The electric signaling levels and methods are handled by a driver circuit external to the UART. A UART is usually an individual (or part of an) integrated circuit (IC) used for serial communications over a computer or peripheral device serial port. One or more UART peripherals are commonly integrated in microcontroller chips. A related device, the universal synchronous and asynchronous receiver-transmitter (USART) also supports synchronous operation.



Block diagram for a UART

Contents

Transmitting and receiving serial data

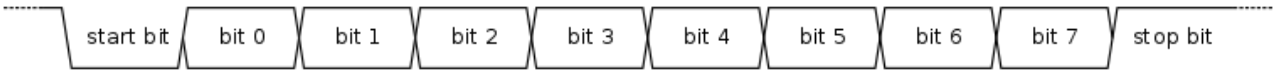
The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits in a sequential fashion.[1] At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

The UART usually does not directly generate or receive the external signals used between different items of equipment. Separate interface devices are used to convert the logic level signals of the UART to and from the external signalling levels, which may be standardized voltage levels, current levels, or other signals.

Communication may be *simplex* (in one direction only, with no provision for the receiving device to send information back to the transmitting device), *full duplex* (both devices send and receive at the same time) or *half duplex* (devices take turns transmitting and receiving).

Data framing



The idle, no data state is high-voltage, or powered. This is a historic legacy from telegraphy, in which the line is held high to show that the line and transmitter are not damaged. Each character is framed as a logic low start bit, data bits, possibly a parity bit and one or more stop bits. In most applications the least significant data bit (the one on the left in this diagram) is transmitted first, but there are exceptions (such as the IBM 2741 printing terminal).

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the **mark** (logic high, i.e., '1') condition and called the stop bit(s). They signal to the receiver that the character is complete. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters.

If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

Receiver

All operations of the UART hardware are controlled by an internal clock signal which runs at a multiple of the data rate, typically 8 or 16 times the bit rate. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid and signals the start of a new character. If not, it is considered a spurious pulse and is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After the required number of bit periods for the character length (5 to 8 bits, typically) have elapsed, the contents of the shift register are made available (in parallel fashion) to the receiving system. The UART will set a flag indicating new data is available, and may also generate a processor interrupt to request that the host processor transfers the received data.

Communicating UARTs have no shared timing system apart from the communication signal. Typically, UARTs resynchronize their internal clocks on each change of the data line that is not considered a spurious pulse. Obtaining timing information in this manner, they reliably receive when the transmitter is sending at a slightly different speed than it should. Simplistic UARTs do not do this, instead they resynchronize on the falling edge of the start bit only, and then read the center of each expected data bit, and this system works if the broadcast data rate is accurate enough to allow the stop bits to be sampled reliably.

It is a standard feature for a UART to store the most recent character while receiving the next. This "double buffering" gives a receiving computer an entire character transmission time to fetch a received character. Many UARTs have a small first-in, first-out (FIFO) buffer memory between the receiver shift register and the host system interface. This allows the host processor even more time to handle an interrupt from the UART and prevents loss of received data at high rates.

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

Transmitter

Transmission operation is simpler as the timing does not have to be determined from the line state, nor is it bound to any fixed timing intervals. As soon as the sending system deposits a character in the shift register (after completion of the previous character), the UART generates a start bit, shifts the required number of data bits out to the line, generates and sends the parity bit (if used), and sends the stop bits. Since full-duplex operation requires characters to be sent and received at the same time, UARTs use two different shift registers for transmitted and received characters. High performance UARTs could contain a transmit FIFO (first in first out) buffer to allow a CPU or DMA controller to deposit multiple characters in a burst into the FIFO rather than have to deposit one character at a time into the FIFO. Since transmission of a single or multiple characters may take a long time relative to CPU speeds, a UART maintains a flag showing busy status so that the host system knows if there is at least one character in the transmit buffer or shift register; "ready for next character(s)" may also be signaled with an interrupt.

Application

Transmitting and receiving UARTs must be set for the same bit speed, character length, parity, and stop bits for proper operation. The receiving UART may detect some mismatched settings and set a "framing error" flag bit for the host system; in exceptional cases the receiving UART will produce an erratic stream of mutilated characters and transfer them to the host system.

Typical serial ports used with personal computers connected to modems use eight data bits, no parity, and one stop bit; for this configuration the number of ASCII characters per second equals the bit rate divided by 10.

Some very low-cost home computers or embedded systems dispense with a UART and use the CPU to sample the state of an input port or directly manipulate an output port for data transmission. While very CPU-intensive (since the CPU timing is critical), the UART chip can thus be omitted, saving money and space. The technique is known as bit-banging.

History

Some early telegraph schemes used variable-length pulses (as in Morse code) and rotating clockwork mechanisms (http://www.railroad-signaling.com/tty/m19/M19_8w.jpg) to transmit alphabetic characters. The first serial communication devices (with fixed-length pulses) were rotating mechanical switches (*commutators*). Various character codes using 5, 6, 7, or 8 data bits became common in teleprinters and later as computer peripherals. The teletypewriter made an excellent general-purpose I/O device for a small computer.

Gordon Bell of DEC designed the first UART, occupying an entire circuit board called a *line unit*, for the PDP series of computers beginning with the PDP-1.[2][3] According to Bell, the main innovation of the UART was its use of sampling to convert the signal into the digital domain, allowing more reliable timing than previous circuits that used analog timing devices with manually adjusted potentiometers.[4] To reduce the cost of wiring, backplane and other components, these computers also pioneered flow control using XON and XOFF characters rather than hardware wires.

DEC condensed the line unit design into an early single-chip UART for their own use.[2] Western Digital developed this into the first widely available single-chip UART, the WD1402A, around 1971. This was an early example of a medium-scale integrated circuit. Another popular chip was the SCN2651 from the Signetics 2650 family.

An example of an early 1980s UART was the National Semiconductor 8250 used in the original IBM PC's Asynchronous Communications Adapter card.[5] In the 1990s, newer UARTs were developed with on-chip buffers. This allowed higher transmission speed without data loss and without requiring such frequent attention from the computer. For example, the popular National Semiconductor 16550 has a 16-byte FIFO, and spawned many variants, including the *16C550, 16C650, 16C750, and 16C850*.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

Depending on the manufacturer, different terms are used to identify devices that perform the UART functions. Intel called their 8251 device a "Programmable Communication Interface". MOS Technology 6551 was known under the name "Asynchronous Communications Interface Adapter" (ACIA). The term "Serial Communications Interface" (SCI) was first used at Motorola around 1975 to refer to their start-stop asynchronous serial interface device, which others were calling a UART. Zilog manufactured a number of Serial Communication Controllers or SCCs.

Starting in the 2000s, most IBM PC compatible computers removed their external RS-232 COM ports and used USB ports that provided superior bandwidth performance. For users who still need RS-232 serial ports, external USB-to-UART bridges are now commonly used. They combine the hardware cables and a chip to do the USB and UART conversion. FTDI is one supplier of these chips.[6] Note that an operating system might not have the drivers for the chip installed by default (E.g. Windows and MacOS do not have drivers for CH340 and Silicon Labs 210x) thus preventing identification of the USB device. Although RS-232 ports are no longer available to users on the outside of most computers, many internal processors and microprocessors have UARTs built into their chips to give hardware designers the ability to interface with other chips/devices that use RS-232 for their default interface.

Structure

A UART usually contains the following components:

- a clock generator, usually a multiple of the bit rate to allow sampling in the middle of a bit period
- input and output shift registers
- transmit/receive control
- read/write control logic
- transmit/receive buffers (optional)
- system data bus buffer (optional)
- First-in, first-out (FIFO) buffer memory (optional)
- Signals needed by a third party DMA controller (optional)
- Integrated bus mastering DMA controller (optional)

Special transceiver conditions

Overrun error

An "overrun error" occurs when the receiver cannot process the character that just came in before the next one arrives. Various devices have different amounts of buffer space to hold received characters. The CPU or DMA controller must service the UART in order to remove characters from the input buffer. If the CPU or DMA controller does not service the UART quickly enough and the buffer becomes full, an Overrun Error will occur, and incoming characters will be lost.

Underrun error

An "underrun error" occurs when the UART transmitter has completed sending a character and the transmit buffer is empty. In asynchronous modes this is treated as an indication that no data remains to be transmitted, rather than an error, since additional stop bits can be appended. This error indication is commonly found in USARTs, since an underrun is more serious in synchronous systems.

Framing error

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

A UART will detect a *framing error* when it does not see a "stop" bit at the expected "stop" bit time. As the "start" bit is used to identify the beginning of an incoming character, its timing is a reference for the remaining bits. If the data line is not in the expected state (high) when the "stop" bit is expected (according to the number of data and parity bits for which the UART is set), the UART will signal a framing error. A "break" condition on the line is also signaled as a framing error.

Parity error

A *parity error* occurs when the parity of the number of one-bits disagrees with that specified by the parity bit. Use of a parity bit is optional, so this error will only occur if parity-checking has been enabled.

Break condition

A *break condition* occurs when the receiver input is at the "space" (logic low, i.e., '0') level for longer than some duration of time, typically, for more than a character time. This is not necessarily an error, but appears to the receiver as a character of all zero-bits with a framing error. The term "break" derives from current loop signaling, which was the traditional signaling used for teletypewriters. The "spacing" condition of a current loop line is indicated by no current flowing, and a very long period of no current flowing is often caused by a break or other fault in the line.

Some equipment will deliberately transmit the "space" level for longer than a character as an attention signal. When signaling rates are mismatched, no meaningful characters can be sent, but a long "break" signal can be a useful way to get the attention of a mismatched receiver to do something (such as resetting itself). Computer systems can use the long "break" level as a request to change the signaling rate, to support dial-in access at multiple signaling rates. The DMX512 protocol uses the break condition to signal the start of a new packet.

UART models

A dual UART, or *DUART*, combines two UARTs into a single chip. Similarly, a quadruple UART or *QUART*, combines four UARTs into one package, such as the NXP 28L194. An octal UART or *OCTART* combines eight UARTs into one package, such as the Exar XR16L788 or the NXP SCC2698.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.