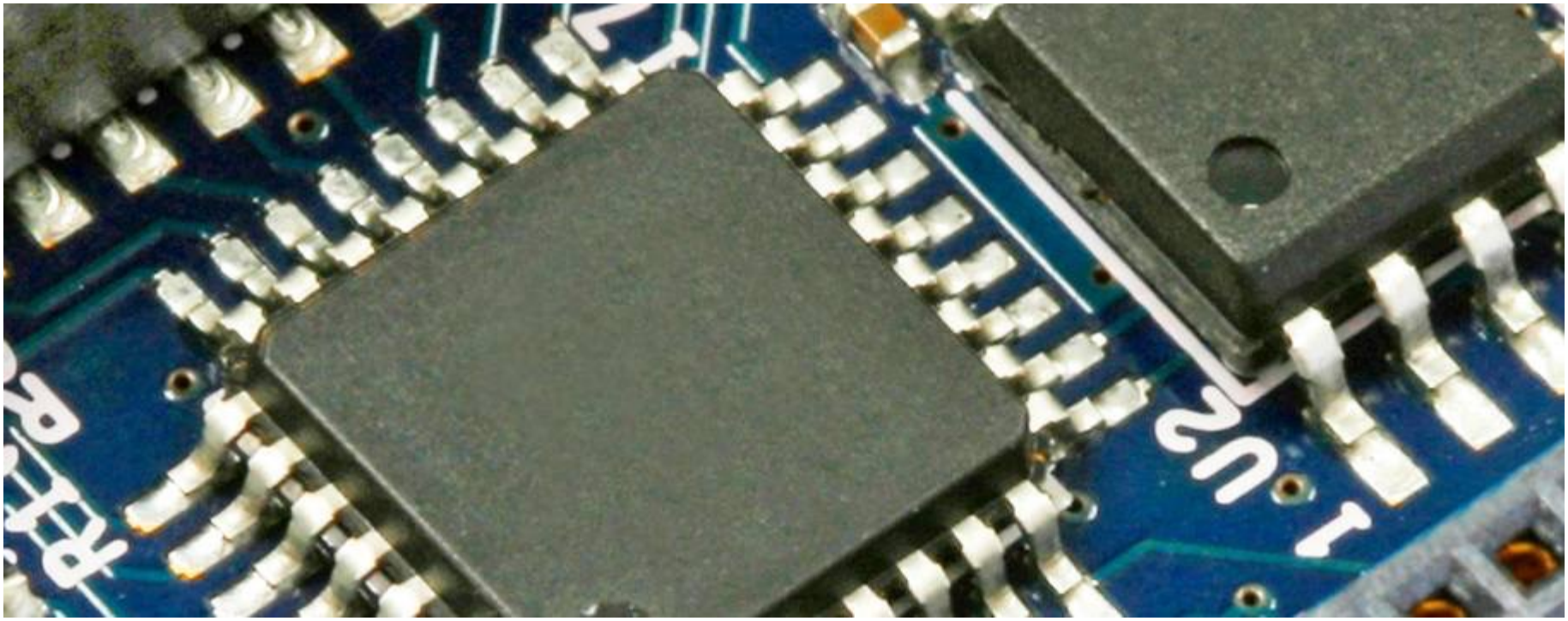


Microcontrolador ESP32

UART (Transreceptor Asíncrono Universal)

UART(Puerto Serie)

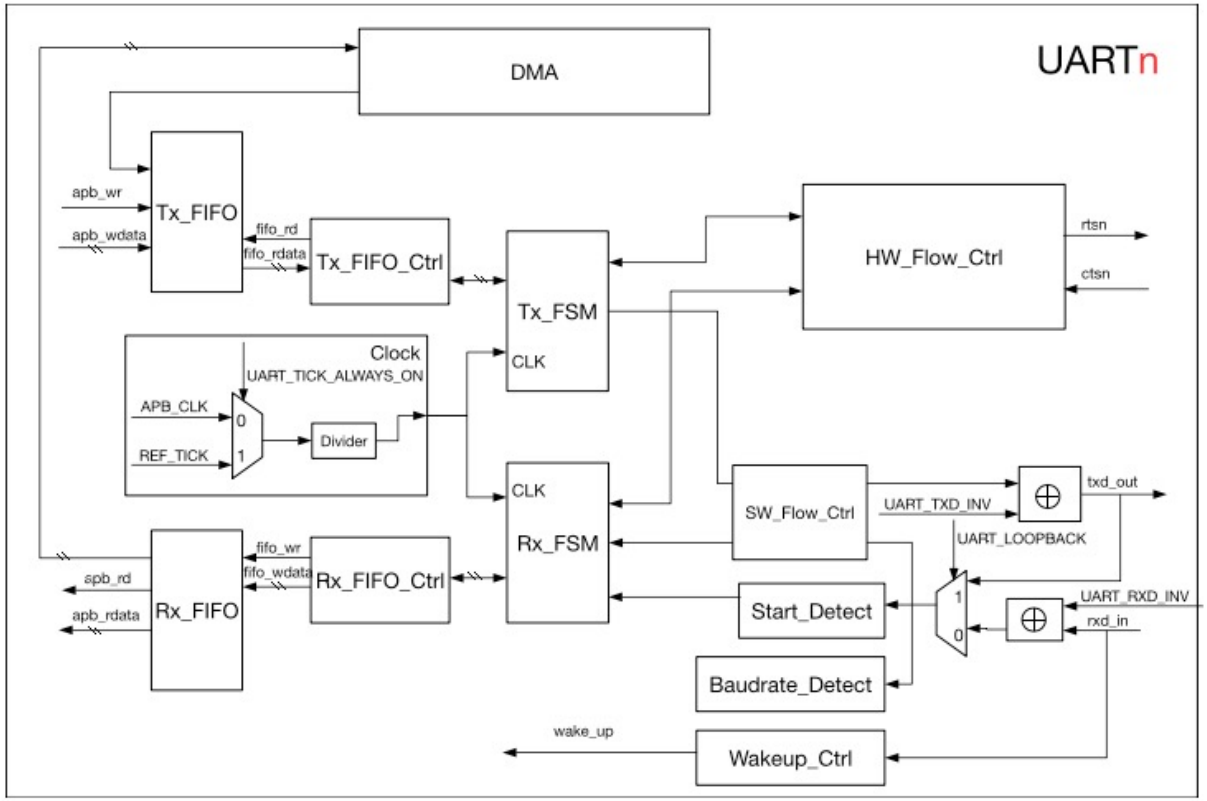
Las aplicaciones embebidas por lo general requieren un método simple de intercambio de datos entre dispositivos que necesitan recursos mínimos del sistema. El receptor/transmisor asíncrono universal (UART) es uno de esos estándares que puede realizar un intercambio de datos full-duplex flexible entre diferentes dispositivos. El ESP32 posee tres controladores UART que son compatibles con dispositivos habilitados para UART de varios fabricantes. El UART del ESP32 puede realizar un **IrDA** (intercambio de datos por infrarrojos) o funcionar como un módem RS-485.

Todos los controladores UART integrados en el ESP32 cuentan con un conjunto de registros idéntico con el fin de facilitar la programación y su flexibilidad. En esta presentación se hará referencia a ellos como **UARTn**, donde n=[0,1,2] y por tanto corresponde a UART0, UART1 y UART2, respectivamente.

Caracteristicas del UART (Puerto Serie)

- Velocidad de baudios programable
- RAM de 1024 × 8 bits compartida por tres FIFO de transmisión y FIFO de recepción UART
- Soporta autocomprobación de velocidad
- Soporta 5, 6, 7, 8 bits de longitud de datos
- Soporta 1, 1.5, 2 y 3 bits de paro (**stop**)
- Soporta bit de paridad
- Compatible con el protocolo RS485
- Compatible con el protocolo IrDA
- Soporta **DMA** para comunicar datos a alta velocidad.
- Soporta despertar el CPU mediante el UART
- Soporta control de flujo de software y hardware

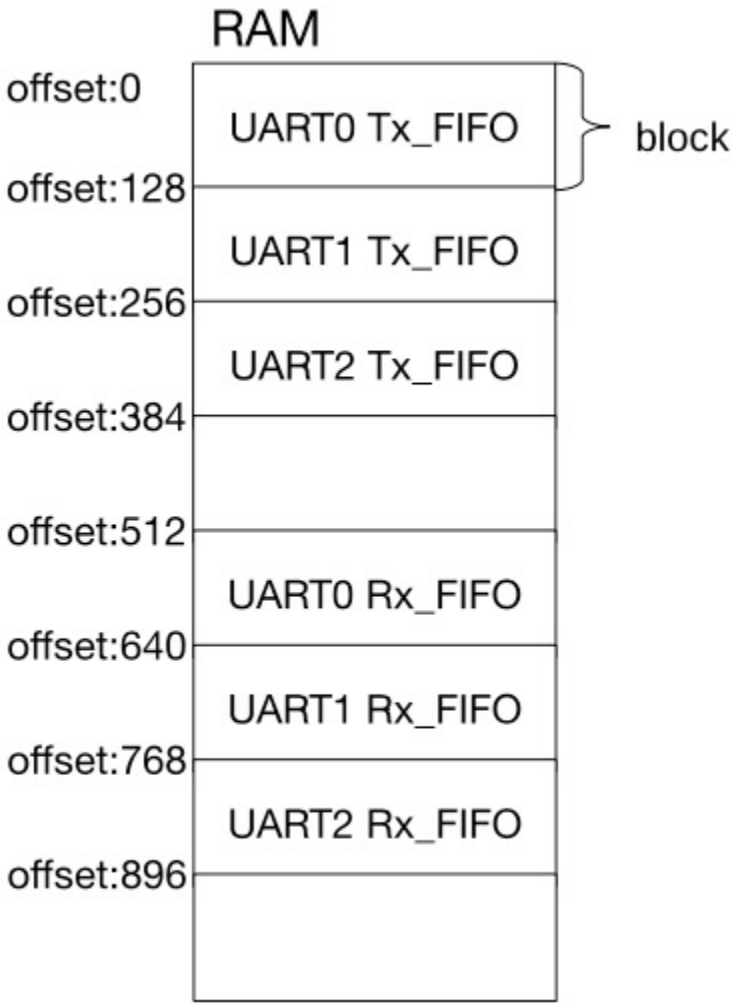
Diagrama de bloques del UART (Puerto Serie)



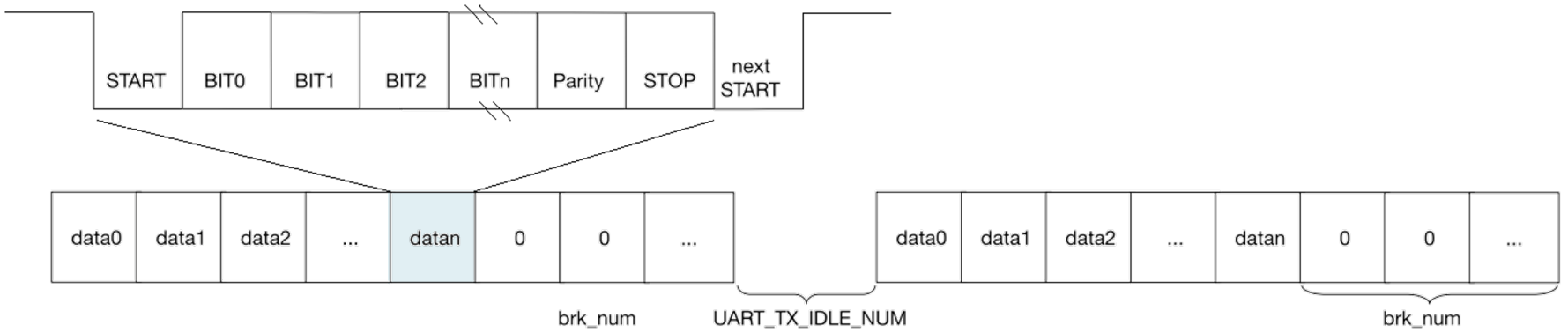
El UART puede derivar su reloj de dos fuentes de oscilación ya sea el APB_CLK de **80 MHz** o el reloj de referencia REF_TICK). La fuente de oscilación seleccionada se divide para generar la señale de reloj que establece la velocidad de operación del UART.

UART_CLKDIV_REG contiene el valor del divisor de reloj en dos partes que son: UART_CLKDIV (**parte entera**) y UART_CLKDIV_FRAG (**parte decimal**).

Memoria RAM del UART (Puerto Serie)



Trama de datos del UART (Puerto Serie)



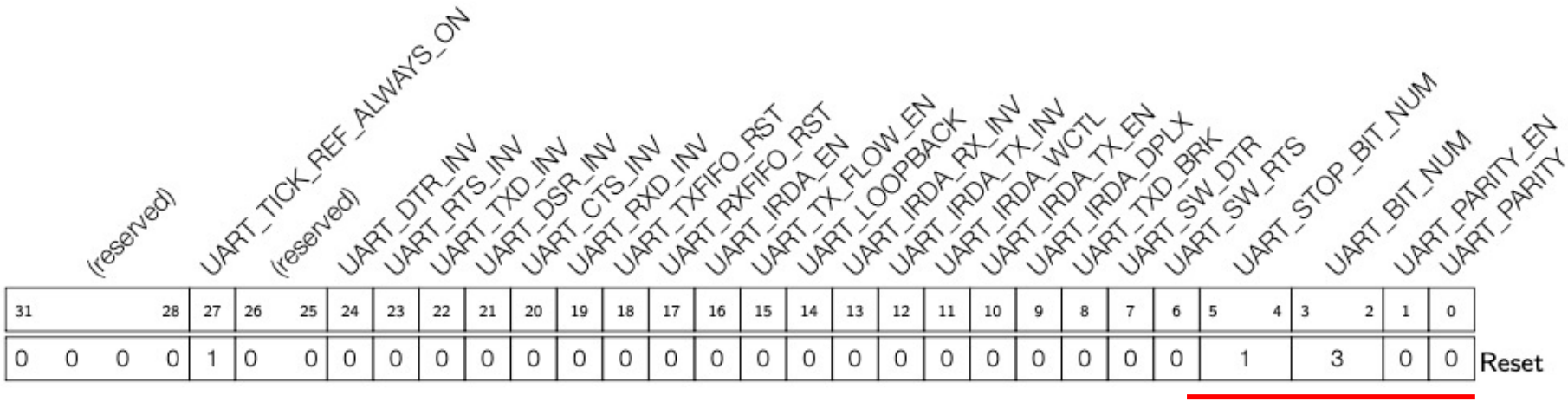
Registros del UART (Puerto Serie)

Name	Description	UART0	UART1	UART2	Acc
Configuration registers					
UART_CONF0_REG	Configuration register 0	0x3FF40020	0x3FF50020	0x3FF6E020	R/W
UART_CONF1_REG	Configuration register 1	0x3FF40024	0x3FF50024	0x3FF6E024	R/W
UART_CLKDIV_REG	Clock divider configuration	0x3FF40014	0x3FF50014	0x3FF6E014	R/W
UART_FLOW_CONF_REG	Software flow-control configuration	0x3FF40034	0x3FF50034	0x3FF6E034	R/W
UART_SWFC_CONF_REG	Software flow-control character configuration	0x3FF4003C	0x3FF5003C	0x3FF6E03C	R/W
UART_SLEEP_CONF_REG	Sleep-mode configuration	0x3FF40038	0x3FF50038	0x3FF6E038	R/W
UART_IDLE_CONF_REG	Frame-end idle configuration	0x3FF40040	0x3FF50040	0x3FF6E040	R/W
UART_RS485_CONF_REG	RS485 mode configuration	0x3FF40044	0x3FF50044	0x3FF6E044	R/W
Status registers					
UART_STATUS_REG	UART status register	0x3FF4001C	0x3FF5001C	0x3FF6E01C	RO

UART_CONF0_REG

Name	Description	UART0	UART1	UART2	Acc
Configuration registers					
UART_CONF0_REG	Configuration register 0	0x3FF40020	0x3FF50020	0x3FF6E020	R/W
UART_CONF1_REG	Configuration register 1	0x3FF40024	0x3FF50024	0x3FF6E024	R/W
UART_CLKDIV_REG	Clock divider configuration	0x3FF40014	0x3FF50014	0x3FF6E014	R/W
UART_FLOW_CONF_REG	Software flow-control configuration	0x3FF40034	0x3FF50034	0x3FF6E034	R/W
UART_SWFC_CONF_REG	Software flow-control character configuration	0x3FF4003C	0x3FF5003C	0x3FF6E03C	R/W
UART_SLEEP_CONF_REG	Sleep-mode configuration	0x3FF40038	0x3FF50038	0x3FF6E038	R/W
UART_IDLE_CONF_REG	Frame-end idle configuration	0x3FF40040	0x3FF50040	0x3FF6E040	R/W
UART_RS485_CONF_REG	RS485 mode configuration	0x3FF40044	0x3FF50044	0x3FF6E044	R/W
Status registers					
UART_STATUS_REG	UART status register	0x3FF4001C	0x3FF5001C	0x3FF6E01C	RO

UART_CONF0_REG



31–28	27	26–25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5–4	3–2	1	0	
(reserved)	UART_TICK_REF_ALWAYS_ON	(reserved)	UART_DTR_INV	UART_RTS_INV	UART_TXD_INV	UART_DSR_INV	UART_CTS_INV	UART_RXD_INV	UART_TXFIFO_RST	UART_RXFIFO_RST	UART_IRDA_EN	UART_TX_FLOW_EN	UART_LOOPBACK	UART_IRDA_RX_INV	UART_IRDA_TX_INV	UART_IRDA_WCTL	UART_IRDA_TX_EN	UART_IRDA_DPLX	UART_TXD_BRK	UART_SW_DTR	UART_SW_RTS	UART_STOP_BIT_NUM	UART_BIT_NUM	UART_PARITY_EN	UART_PARITY	
0 0 0 0	1	0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	Reset

UART_STOP_BIT_NUM: Usado para establecer longitud del **stop bit**;

1: 1 bit
2: 1.5 bits.
3: 2 bits (?)

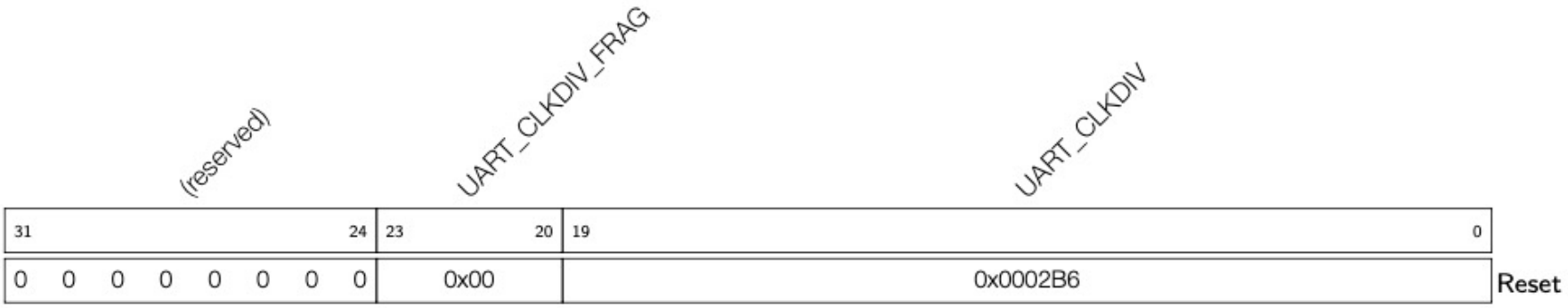
UART_BIT_NUM: Usado para establecer la **longitud de datos**;

0: 5 bits,
1: 6 bits,
2: 7 bits,
3: 8 bits.

UART_PARITY_EN: Usado para activar el **uso de paridad**; 0: Desactivar
1: Activar

UART_PARITY: Usado para establecer el **modo de paridad** a verificar; 0: Par
1: Impar

UART_CLKDIV_REG



UART_CLKDIV_FRAG: La parte **decimal** del factor divisor de frecuencia (4 bits).

UART_CLKDIV: La parte **entera** del factor divisor de frecuencia (20 bits máximo);

Ejemplo: Si se requiere operar el UART a la velocidad de **115200** Baud se tiene entonces

$$Divisor = \frac{80\,000\,000}{115200} = 694.444$$

Parte entera: 694 (2B6 hex)
Parte decimal: 0.444 (7 hex)

0.444_{10} a Hex:

0.444 * 2 = **0**.888
0.888 * 2 = **1**.776
0.776 * 2 = **1**.552
0.552 * 2 = **1**.104

0 1 1 1 = 7h

UART_CLKDIV_REG

Una camino más simple usando escalamiento y división entera.

Ejemplo: Si se requiere operar el UART a la velocidad de **115200** Baud se tiene entonces

$$Divisor = \left[\frac{80\,000\,000 * \mathbf{16}}{115200}\right] = 11111$$

$$1111_{10} = 2B67\ h$$

Parte entera

Parte decimal

Nota: Multiplicar por 16 es equivalente a un corrimiento de 4 bits (un nibble)