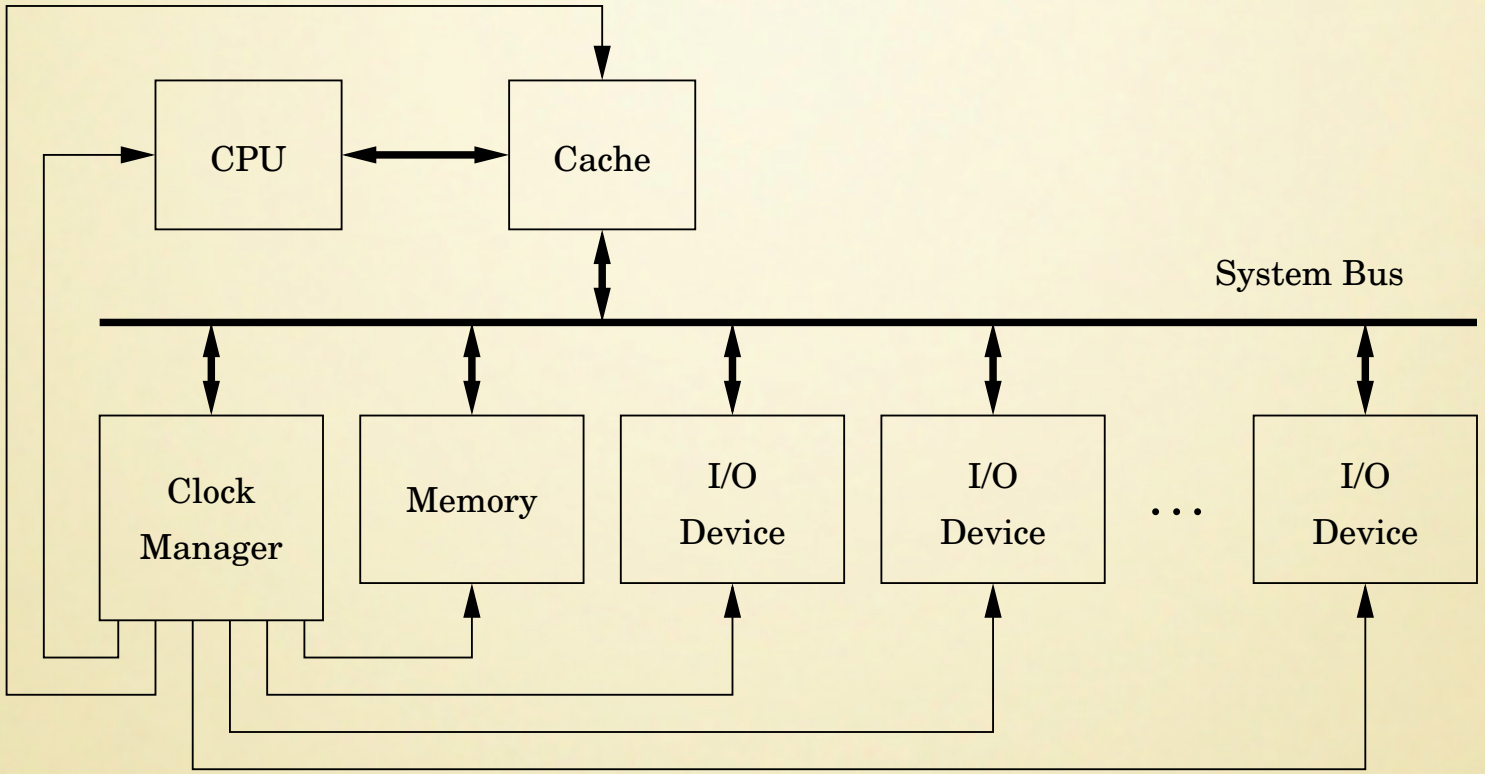
Modern Assembly Language Programming with the ARM processor

Chapter 13: Common System Devices

1. Clock Management

2. Serial Communications

Clock Management



Typical system with a clock management device.

Raspberry Pi Clock Manager

Provides a large number of clock signals to drive various devices.

Each clock signal can be driven by one of 16 source clocks.

Number	Name	Frequency	Note
0	GND	0 Hz	Clock is stopped
1	oscillator	19.2 MHz	
2	testdebug0	Unknown	Used for system testing
3	testdebug1	Unknown	Used for system testing
4	PLLA	650 MHz	May not be aviailable
5	PLLC	200 MHz	May not be aviailable
6	PLLD	500 MHz	
7	HDMI auxillary	Unknown	
8–15	GND	0 Hz	Clock is stopped

Raspberry Pi Clock Manager (continued)

Some registers in the clock manager device:

Offset	Name	Description
070_{16}	CM_GP0_CTL	GPIO Clock 0 (GPCLK0) Control
074_{16}	CM_GP0_DIV	GPIO Clock 0 (GPCLK0) Divisor
078_{16}	CM_GP1_CTL	GPIO Clock 1 (GPCLK1) Control
$07c_{16}$	CM_GP1_DIV	GPIO Clock 1 (GPCLK1) Divisor
080_{16}	CM_GP2_CTL	GPIO Clock 2 (GPCLK2) Control
084_{16}	CM_GP2_DIV	GPIO Clock 2 (GPCLK2) Divisor
098_{16}	CM_PCM_CTL	Pulse Code Modulator Clock (PCM_CLK) Control
$09c_{16}$	CM_PCM_DIV	Pulse Code Modulator Clock (PCM_CLK) Divisor
$0a0_{16}$	CM_PWM_CTL	Pulse Modulator Device Clock (PWM_CLK) Control
$0a4_{16}$	CM_PWM_DIV	Pulse Modulator Device Clock (PWM_CLK) Divisor
$0f0_{16}$	CM_UART_CTL	Serial Communications Clock (UART_CLK) Control
$0f4_{16}$	CM_UART_DIV	Serial Communications Clock (UART_CLK) Divisor

pcDuino Clock Signals

Provides a small number of clock signals to drive various devices.

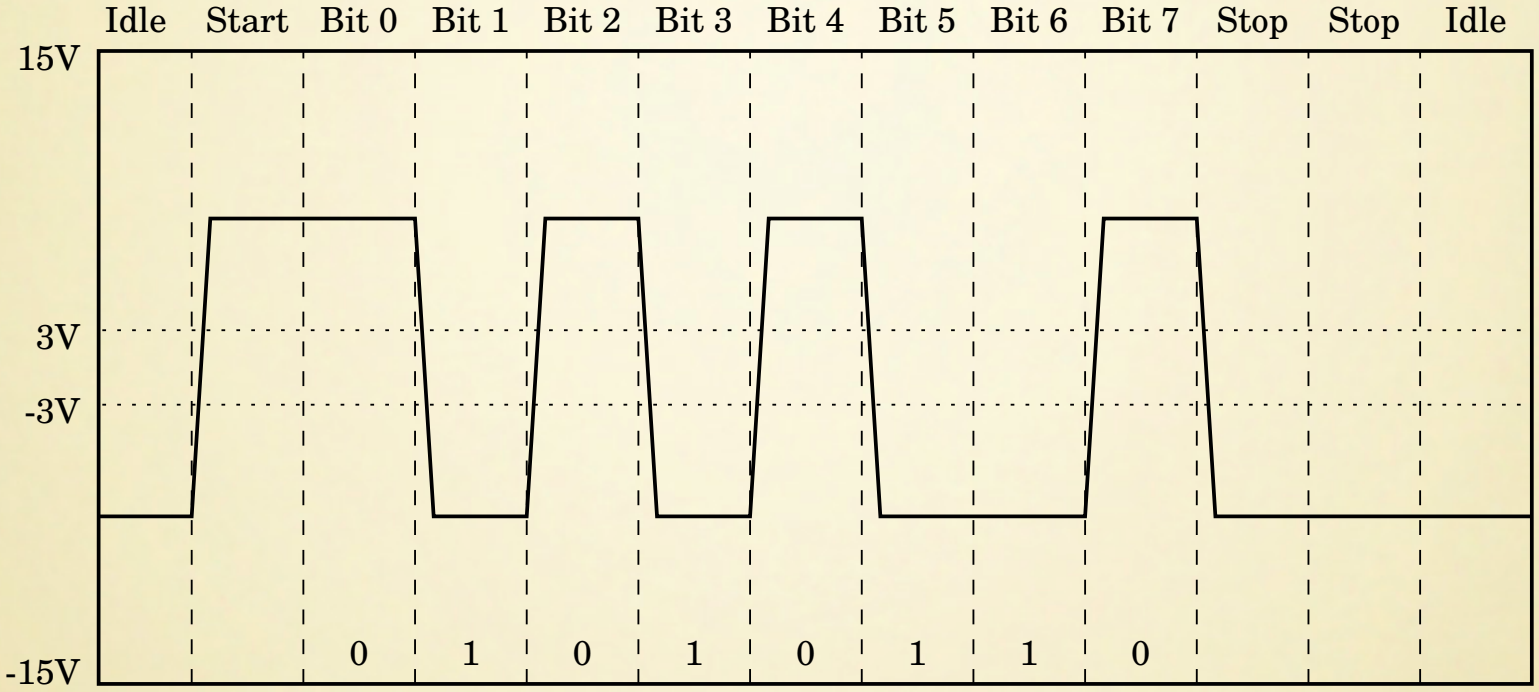
Clock Domain	Modules	Frequency	Description
OSC24M	Most modules	24MHz	Main clock
CPU32_clk	CPU	2KHz – 1.2GHz	Drives CPU
AHB_clk	AHB devices	8KHz – 276MHz	Drives some devices
APB_clk	Peripheral bus	500Hz – 138MHz	Drives some devices
SDRAM_clk	SDRAM	0Hz – 400MHz	Drives SDRAM memory
USB_clk	USB	480MHz	Drives USB devices

Serial Communications

Universal Asynchronous Receiver/Transmitter (UART)

- "Universal" indicates that the device is highly configurable and flexible.
- "Asynchronous" means that a receiver and transmitter can communicate without a synchronizing signal.
- To transfer a group of bits, called a *data frame*, the transmitter typically first sends a *start bit*.
- After each group of data bits, the transmitter will return the signal to the low state and keep it there for some minimum period called the *stop bits*. (typically the time it would take to send two bits of data)
- The *stop bits* allow the receiver to have some time to process the received byte and prepare for the next start bit.

Transmission

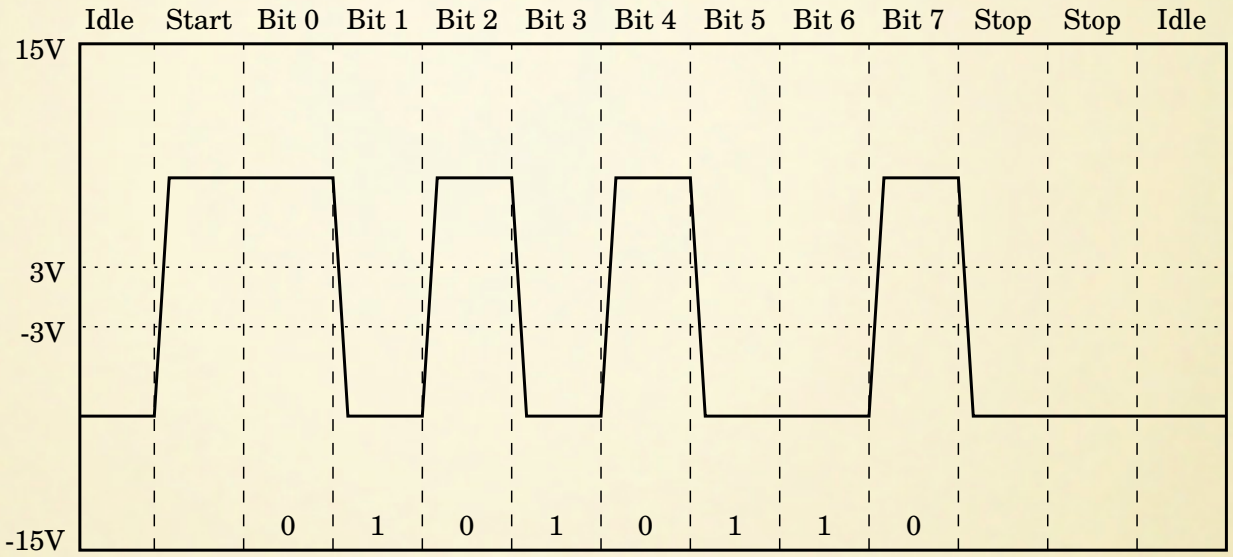


Waveform of a UART transmitting 56_{16} (the ASCII 'V' character).

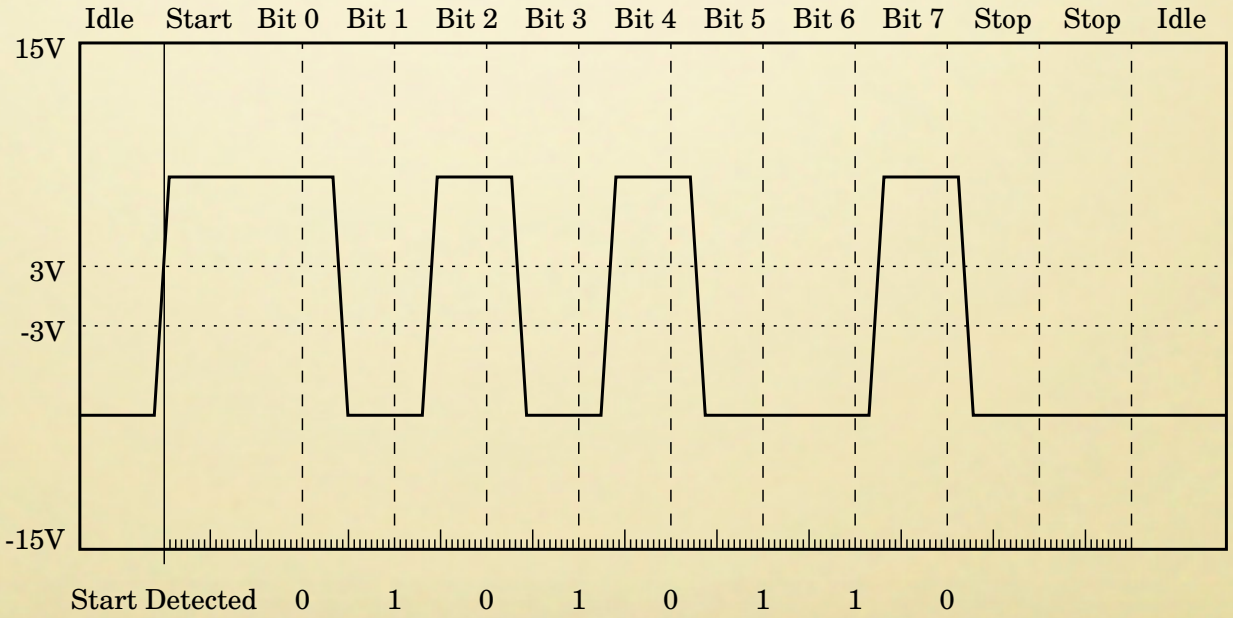
- The UART enters the idle state only if there is not another byte immediately ready to send.
- If the transmitter has another byte to send, then the start bit can begin at the end of the second stop bit.

Reception with Clock Mismatch

Transmitter



Receiver



Raspberry Pi UART0

The PL011 UART is similar to industry standard 16550A UART

Offset	Name	Description
00_{16}	UART_DR	Data Register
04_{16}	UART_RSRECR	Receive Status Register/Error Clear Register
18_{16}	UART_ FR	Flag register
20_{16}	UART_ILPR	not in use
24_{16}	UART_IBRD	Integer Baud rate divisor
28_{16}	UART_FBRD	Fractional Baud rate divisor
$2c_{16}$	UART_LCRH	Line Control register
30_{16}	UART_CR	Control register
34_{16}	UART_IFLS	Interupt FIFO Level Select Register
38_{16}	UART_IMSC	Interupt Mask Set Clear Register
$3c_{16}$	UART_RIS	Raw Interupt Status Register
40_{16}	UART_MIS	Masked Interupt Status Register
44_{16}	UART_ICR	Interupt Clear Register
48_{16}	UART_DMACR	DMA Control Register
80_{16}	UART_ITCR	Test Control register
84_{16}	UART_ITIP	Integration test input reg
88_{16}	UART_ITOP	Integration test output reg
$8c_{16}$	UART_TDR	Test Data reg

Data Register

Read to receive, write to transmit.
Bits 8-11 give status for the byte received.

Bit	Name	Description	Values
7–0	DATA	Data	Read: Last data received Write: Data byte to transmit
8	FE	Framing error	0: No error 1: The received character did not have a valid stop bit
9	PE	Parity error	0: No error 1: The received character did not have the correct parity, as set in the EPS and SPS bits of the Line Control Register (UART_LCRH)
10	BE	Break error	0: No error 1: A break condition was detected. The data input line was held low for longer than the time it would take to receive a complete byte, including the start and stop bits.
11	OE	Overrun error	0: No error 1: Data was not read quickly enough, and one or more bytes were overwritten in the input buffer
31–12	-	Not used	Write as zero, read as don't care

Calculating the Baud Rate Divisor

$$BAUDDIV = \frac{UARTCLK}{16 \times Baudrate}$$

- *UARTCLK* is the frequency of the UART_CLK that is configured in the Clock Manager device. The default value is 3 MHz.
- *BAUDDIV* should be calculated as a U(16,6) fixed point number.
- *BAUDDIV* is stored in two registers
  - UART_IBRD holds the integer part and
  - UART_FBRD holds the fractional part.

pcDuino UART

The pcDuino includes eight UART devices that are fully compatible with the 16550A UART

UART addresses:

Name	Address
UART0	0x01C28000
UART1	0x01C28400
UART2	0x01C28800
UART3	0x01C28C00
UART4	0x01C29000
UART5	0x01C29400
UART6	0x01C29800
UART7	0x01C29C00

pcDuino UART Registers

Register Name	Offset	Description
UART_RBR	0x00	UART Receive Buffer Register
UART_THR	0x00	UART Transmit Holding Register
UART_DLL	0x00	UART Divisor Latch Low Register
UART_DLH	0x04	UART Divisor Latch High Register
UART_IER	0x04	UART Interrupt Enable Register
UART_IIR	0x08	UART Interrupt Identity Register
UART_FCR	0x08	UART FIFO Control Register
UART_LCR	0x0C	UART Line Control Register
UART_MCR	0x10	UART Modem Control Register
UART_LSR	0x14	UART Line Status Register
UART_MSR	0x18	UART Modem Status Register
UART_SCH	0x1C	UART Scratch Register
UART_USR	0x7C	UART Status Register
UART_TFL	0x80	UART Transmit FIFO Level
UART_RFL	0x84	UART_RFL
UART_HALT	0xA4	UART Halt TX Register

Setting the BAUD Rate

The baud rate is set using a 16-bit Baud Rate Divisor

$$BAUDDIV = \frac{sclk}{16 \times Baudrate}$$

- *sclk* is the frequency of the UART serial clock, which is configured by the Clock Manager device. The default frequency of the clock is 24 MHz.
- *BAUDDIV* should be calculated as a sixteen bit unsigned integer.
- *BAUDDIV* is stored in two registers.
  - UART_DLL holds the least significant eight bits, and
  - UART_DLH holds the most significant eight bits.

Note that for high baud rates, it may not be possible to get exactly the rate desired.

For example, a baud rate of 115200, would require a divisor of $13.02083\overline{3}$

A divisor of 13 gives a baud rate of $\frac{24000000}{16 \times 13} = 115384.615385$, or about 0.16% faster than desired.

Although slightly fast, it is well within the tolerance for RS232 communication.