Section 6. Serial Communication

Communication Using Serial Interfaces: UART and SPI

1

UNIVERSITY of Rhode Island

Concepts of Serial Communication

- Limitations of Parallel Bus
  - Clock skew becomes a serious issue for high speed and long distance
  - Cost of wire, fewer wires cost less and occupies less space
  - Cross talk between multiple conductors, affecting signal quality
- Serial Communication
  - Sending data a bit at a time, sequentially, over a communication channel
  - Un-clocked, no clock skew problem
  - Fewer wires → low cost, more space for better isolation from surroundings
- Simplex, half-duplex, and full-duplex
  - Simplex: data can be transferred in only one direction
  - Half-duplex: data can be transferred in two directions, but one at a time
  - Full-duplex: data can be transferred in two directions concurrently

2

UNIVERSITY of Rhode Island

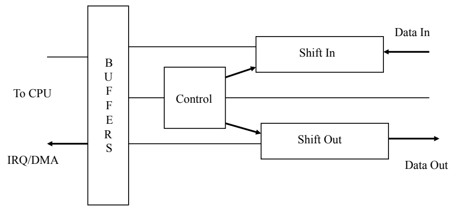
General Procedure and Logic of Serial Communication

- Special signal marks
  - Start bit and stop bit that mark the begin and end of one comm.
- Parity bit
  - Error checking and correction
    - Even parity, odd parity, CRC, Hamming code, etc.
- Baud rate
  - Specifies how fast bit sequence is transmitted
- Buffering for transmitter and receiver

3

UNIVERSITY of Rhode Island

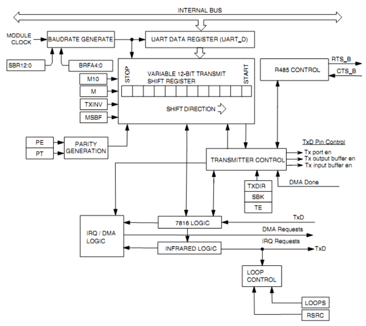
Logic Diagram of Most Serial Interfaces



4

UNIVERSITY of Rhode Island

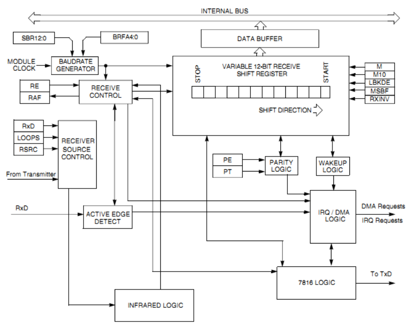
UART Transmitter Block Diagram



5

UNIVERSITY of Rhode Island

UART Receiver Block Diagram



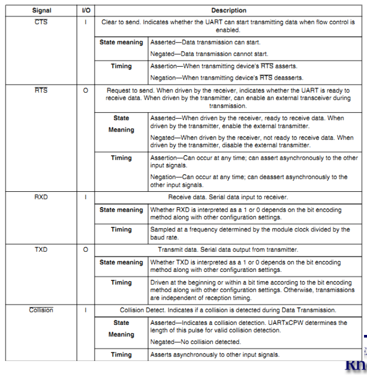
6

UNIVERSITY of Rhode Island

UART Instantiations on Kinetis

UART Instance	ISO-7816 Supported?	FIFOs	Module Clock	Maximum Baud Rate
UART0	Yes	8 entry TxFIFO 8 entry RxFIFO	Core Clock (Max freq = 100 MHz)	6.25 Mbits/sec
UART1	No	8 entry TxFIFO 8 entry RxFIFO	Core Clock (Max freq = 100 MHz)	6.25 Mbits/sec
UART2-UART5	No	No FIFOs (double buffered operation)	Peripheral Clock (Max freq = 50 MHz)	3.13 Mbits/sec

UART Detailed Signal Descriptions



Signal	I/O	Description	
$\overline{\text{CTS}}$	I	Clear to send. Indicates whether the UART can start transmitting data when flow control is enabled.	
		State meaning	Asserted—Data transmission can start. Negated—Data transmission cannot start.
		Timing	Assertion—When transmitting device's $\overline{\text{RTS}}$ asserts. Negation—When transmitting device's $\overline{\text{RTS}}$ deasserts.
$\overline{\text{RTS}}$	O	Request to send. When driven by the receiver, indicates whether the UART is ready to receive data. When driven by the transmitter, can enable an external transceiver during transmission.	
		State Meaning	Asserted—When driven by the receiver, ready to receive data. When driven by the transmitter, enable the external transmitter. Negated—When driven by the receiver, not ready to receive data. When driven by the transmitter, disable the external transmitter.
		Timing	Assertion—Can occur at any time; can assert asynchronously to the other input signals. Negation—Can occur at any time; can deassert asynchronously to the other input signals.
RXD	I	Receive data. Serial data input to receiver.	
		State meaning	Whether RXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Sampled at a frequency determined by the module clock divided by the baud rate.
TXD	O	Transmit data. Serial data output from transmitter.	
		State meaning	Whether TXD is interpreted as a 1 or 0 depends on the bit encoding method along with other configuration settings.
		Timing	Driven at the beginning or within a bit time according to the bit encoding method along with other configuration settings. Otherwise, transmissions are independent of reception timing.
$\overline{\text{Collision}}$	I	Collision Detect. Indicates if a collision is detected during Data Transmission.	
		State Meaning	Asserted—Indicates a collision detection. UARTxCPW determines the length of this pulse for valid collision detection. Negated—No collision detected.
		Timing	Asserts asynchronously to other input signals.

UART Modes of Operation

►**UART module supports three main operating modes:**

- **UART mode**
  - **Supported on all 6 UARTs**
- **IrDA mode**
  - **Support on all 6 UARTs**
- **ISO-7816 mode**
  - **Only supported on UART0**

UART Mode Features

- Full-duplex serial communication
- Standard mark/space non-return-to-zero (NRZ) format
- 13-bit baud rate selection with /32 fractional divide, based on module clock frequency
- Programmable data formats
  - 8- or 9-bit data formats including 9-bit with parity
  - Programmable transmitter output and receiver input polarity
  - Ability to select MSB or LSB to be first bit on wire
- 8 entry Rx and Tx FIFOs available on UART0 and UART1
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Address match feature in receiver to reduce address mark wakeup ISR overhead
- Hardware parity generation and checking
- Interfaced to the on-chip DMA
  - DMA Rx and Tx requests from the UART can be used to move data without processor intervention

IrDA Mode Features

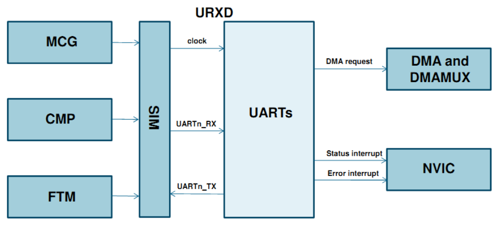
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format
- Programmable narrow pulse transmission and detection
- Support IrDA data rates between 2.4 kbits/s and 115.2 kbits/s
- Several options for changing RX and TX sources for UART0 and UART1 are controlled by the SIM:
  - UARTn_RX input from the pin or from CMP0 or CMP1
  - UARTn_TX output directly from UART or modulated with FTM outputs
  - These options can be used in any UART mode, but are most useful for IrDA applications

►NOTE: Although all UARTs include IrDA features only UART0 and UART1 have the alternate options for the TX and RX sources controlled by the SIM

ISO-7816 Mode Features

- Support for ISO-7816 protocol for interfacing with SIM and smartcards
  - Support of T=0 and T=1 protocols
  - Automatic retransmission of NACK'd packets with programmable retry threshold
  - Support for 11 and 12 ETU transfers
  - Detection of initial packet and automated transfer parameter programming
- Interrupt-driven operation with seven ISO-7816 specific interrupts
  - Wait Time Violated
  - Character Wait Time Violated
  - Block Wait Time Violated
  - Initial Character Detected
  - Transmit Error Threshold Exceeded
  - Receive Error Threshold Exceeded
  - Guard Time Violated

UART interconnect diagram



SIM Configuration for the UART

- SIM controls clock gating for the UARTs (and other modules)
  - SIM must be initialized to enable the clock for any UART before it can be initialized (module registers are not accessible until the clock is enabled).
- SIM control TX and RX sources for UART0 and UART1
  - Default option is a direct connection between the UART and the RX and TX pins (no configuration step is needed to use the default)
  - Alternate options for RX:
    - CMP0
    - CMP1
  - Alternate options for TX:
    - UART TX output modulated with FTM1 channel 0
    - UART TX output modulated with FTM2 channel 0
- If using an alternate source for RX and/or TX, the SIM should be configured to select the source before enabling the receiver and/or transmitter in the UART

UART Operation in Low Power Modes

Low Power Mode	UART Operation	Comments
STOP	Static, wakeup on edge	UART can be a wakeup source
VLPR	Max 125kbps	Frequency is limited in VLPR mode
VLPW	Max 125kbps	Frequency is limited in VLPW mode
VLPS	Static, wakeup on edge	UART can be a wakeup source
LLS	static	UART retains state, but is not active so it cannot be used for wakeup. Several UART signals are muxed with LLWU wakeup sources though, so UART pins could potentially be used for wakeup, just not coming from the UART itself.
VLLSx	OFF	UART does not retain state. Several UART signals are muxed with LLWU wakeup sources though, so UART pins could potentially be used for wakeup, just not coming from the UART itself.

Example UART Init

1. Enable the clock for the port associated with the UART pins you want to use (SIM_MCGC5)
2. Enable UART pins (PORTx_PCRn)
3. Enable the UART module clock (SIM_MCGCn)
4. Configure the UART control registers for the desired data format
   - Number of data bits (UARTn_C1[M] and UARTn_C4[M10])
   - Parity and parity type (UARTn_C1[PE,PT])
   - MSB or LSB first (UARTn_S2[MSBF])
   - Data polarity (UARTn_S2[RXINV] and UARTn_C3[TXINV])
5. Configure the baud rate (UARTn_BDH and UARTn_BDL)
6. Enable the receiver and/or transmitter (UARTn_C2[RE, TE])

Baud Rate Calculation

- The UART has a 13-bit integer divider and a 5-bit fractional fine adjust counter that are used to generate the UART baud rate.

UART baud rate = UART module clock/ (16 * (SBR[12:0] + BRFD))

- Where BRFD = the BRFA[4:0] field divided by 32

Baud Rate Calculation Examples

The table below gives some baud rate calculation examples using a 50MHz module clock.

SBR	BRFA	BRFD	Rx Clock	Tx Clock	Target Baud Rate	Error (%)
1	0	0	50 MHz	3.125MHz	3125000	0
27	0	0	1851.85 kHz	115740.7 kHz	115200	0.47%
27	4	0.125	1843.32 kHz	115207.4 kHz	115200	0.006%
162	0	0	308.64 kHz	19290.1 kHz	19200	0.47%
162	24	0.75	307.219 kHz	19201.2 kHz	19200	0.006%
325	0	0	153.846 kHz	9615.38 kHz	9600	0.16%
325	17	0.53125	153.595 kHz	9599.69 kHz	9600	0.003%

Polling, Interrupt, or DMA configuration

- The UART can be configured to handle data flow by polling status flags, generating interrupts, or using the DMA.
- Polling is the most CPU intensive, but might make the most sense when handling small messages.
- The UART status interrupt can be used to decrease CPU loading. Status interrupt conditions are:
  - Transmit data empty
  - Transmit complete
  - Idle line (primarily used for multi-drop applications)
  - Receive data full
  - LIN break detect
  - RxD pin active edge (main use is as a CPU wakeup)
  - Initial character detect (used for ISO-7816 mode only)
- The DMA can be used to automatically move receive and/or transmit data to reduce CPU loading even more. DMA requests can be generated on:
  - Transmit data empty
  - Receive data full

UNIVERSITY of Rhode Island

Example ISO-7816 Init

1. Enable the clocks for modules you will be using (SIM_MCGCn) :
   - Port(s) associated with all pins going to the SIM card (UART, GPIO, FTM)
   - UART0
   - FTMn
2. Configure UART, FTM, and GPIO pin functions (PORTx_PCRn). Default states for the GPIOs are:
   - PD – configure as an input for SIM card detect
   - RST – configure as output driving low
   - VPP – configure as an output driving low
3. Configure the UART control registers for an IS0-7816 compatible data format:
   - Configure the UART for single wire mode (UARTn_C1[LOOPS] and UARTn_C1[RSRC])
   - 9-bit mode (8 data bits + even parity) (UARTn_C1[M, PE, and PT])
   - LSB first mode (UARTn_S2[MSBF])
4. Enable ISO-7816 mode (UARTn_C7816[ISO_7816E])
5. Wait for PD signal to go low indicating a SIM card has been plugged in.
6. Drive the VPP GPIO high to enable power to the SIM card.
7. Configure the FTM channel to drive a clock to the SIM card (max 5 MHz).
8. Configure the UART baud rate. Initial baud rate should be 1/372 the frequency of the FTM channel being used as the SIM card clock. (UARTn_BDH and UARTn_BDL)
9. Drive the RST GPIO high to take the SIM card out of reset.
10. Enable the receiver and transmitter (UARTn_C2[RE,TE]).
11. Wait for the initial character to be received.

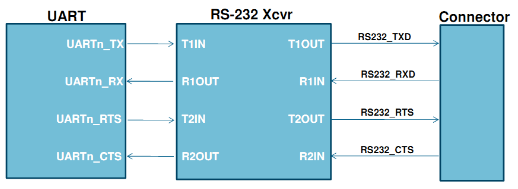
UNIVERSITY of Rhode Island

Note on Clearing Status Flags

- Some of the status and error flags in the UART are cleared by reading the S1 register when the bit is set, then reading the data register.
- When writing code to clear these flags keep in mind that reading the data register while the FIFO is empty will cause the FIFO pointers to get out of sync.
- If you plan to flush the RxFIFO when some error conditions are detected, then we recommend clearing the error flag, THEN flushing the FIFO. This way you don't need to read from the FIFO when it is empty in order to clear the flag.
- If for some reason you need to read from the RxFIFO when it is empty, then flushing the RxFIFO will reinitalize the FIFO pointers.

UNIVERSITY of Rhode Island

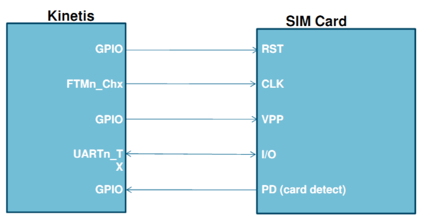
UART RS-232 Hardware



NOTE: RTS and CTS flow control signals are optional.

UNIVERSITY of Rhode Island

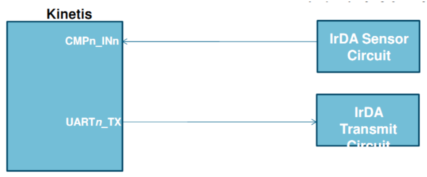
ISO-7816 Hardware Connections



NOTE: UART should be configured for single wire mode. In this mode the UART_TX is used for transmit and receive data.

UNIVERSITY of Rhode Island

IrDA Hardware Connections

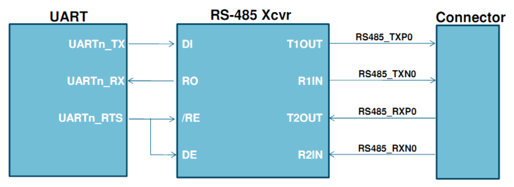


For IrDA mode, the SIM options for alternate TX and RX sources should be enabled. The output of the comparator should be used to drive the UART input, and the UART_TX pin should be the UART output modulated with an FTM channel.

NOTE: The SIM also includes an option to use dual-pad drive strength for the CMT/UART0_TX signal. You might want to use this option for an IrDA interface.

UNIVERSITY of Rhode Island

UART RS-485 Hardware



NOTE: RTS should be configured for TxRTS mode and active high polarity.

GPIOC Features

- 5V tolerant pins on some devices
  - Use open drain connections to interface to other 5V devices
- Pin interrupt and DMA capability
  - Rising/Falling edges and both edge and level
  - Each 32 pin port can generate an interrupt or DMA request
- Digital glitch filter
  - Spurious noise filter
  - Configurable width and clock source
- Hysteresis and configurable pull up/pull down device on all input pins
- Configurable slew rate and drive strength on all output pins
  - Reduced noise on output pins

GPIOC Registers

- **GPIOx_PDOR: Port Data Output Register**
  0 Logic level 0 is driven on pin provided pin is configured for General Purpose Output.
  1 Logic level 1 is driven on pin provided pin is configured for General Purpose Output.
- **GPIOx_PSOR: Port Set Output Register**
  0 Corresponding bit in PDORn does not change.
  1 Corresponding bit in PDORn is set to logic one.
- **GPIOx_PCOR**
  0 Corresponding bit in PDORn does not change.
  1 Corresponding bit in PDORn is set to logic zero.
- **GPIOx_PTOR**
  0 Corresponding bit in PDORn does not change.
  1 Corresponding bit in PDORn is set to the inverse of its existing logic state.
- **GPIOx_PDIR**
  0 Pin logic level is logic zero or is configured for use by digital function.
  1 Pin logic level is logic one.
- **GPIOx_PDDR**
  0 Pin is configured as general purpose input, if configured for the GPIO function
  1 Pin is configured for general purpose output, if configured for the GPIO function

Sample Code

- **Sample code below configures PTB11 and PTC9 as GPIO output pins, PTC5 and PTC13 as IRQ pins.**

```
// enable PTB11, PTC9 as GPIO output pin
PORTC_PCR9 &= ~0x700;
PORTC_PCR9 |= PORT_PCR_MUX(1);
PORTB_PCR11 &= ~0x700;
PORTB_PCR11 |= PORT_PCR_MUX(1);

GPIOB_PDOR |= 0x800; // PTB11, output high
GPIOB_POER |= 0x800;  // PTB11, output
GPIOC_PDOR |= 0x200; // PTC7,8,9, output high
GPIOC_POER |= 0x200;  // PTC7,8,9, output

// init sw6 & sw7, pull-up enable
PORTC_PCR5 = PORT_PCR_IRQC(0x0A)|PORT_PCR_MUX(1)
        |PORT_PCR_PFE_MASK|PORT_PCR_PE_MASK
        |PORT_PCR_PS_MASK;
PORTC_PCR13 = PORT_PCR_IRQC(0x0A)|PORT_PCR_MUX(1)
        |PORT_PCR_PFE_MASK|PORT_PCR_PE_MASK
        |PORT_PCR_PS_MASK;

// enable interrupt
enable_irq(INT_PORTC-16);
```