Title of script: Serial Communication

Author: Rohan Nayak, Muthu Vangliappan

Keywords: USART, Interrupt, Register

|  |  |
| --- | --- |
| Slides | Narration |
| 01.  Title | * Hello Friends. * Welcome to the spoken tutorial on Firebird V Robotics Research Platform. * This platform is based on Atmega2560 microcontroller which belongs to AVR architecture based microcontroller family. * In this tutorial, we will learn in detail about **Serial Communication in Firebird V robot**.  |  | | --- | |  |   **(press next)** |
| 02.  Agenda | * Now let us see the agenda for discussion in this tutorial.   **(press next)**   * The presentation will start with the introduction to Serial Communication where we will discuss about the needs and ways of Serial Communication.   **(press next)**   * Next we will discuss about the Serial Communication feature available in FireBird V Robot.   **(press next)**   * After that we will discuss about the various Registers used in Serial Communication   **(press next)**   * Then we will discuss about the various interrupts used in Serial communication.   **(press next)**   * Finally with the help of Registers and Interrupts we will implement the C code for Serial Communication.   **(press next)** |
| 03.  What is Serial Communication? | * We can broadly classify communication in two ways : Serial and Parallel.   Serial communication is the process of sending data one bit at a time sequentially over a communication channel. This is in contrast to Parallel communication where several bits are sent as a whole on a link with several parallel channels.  **(press next)** |
| 04.  Needs and ways of Serial Communication | * Next we will discuss about the need of serial communication. It is used to establish communication between PCs, Tablets, and other external devices**(press next)** * Serial communication helps in establishing communication between two or more robots. **(press next)** * Robots and external devices. **(press next)** * Now we will take a look at the ways of serial communication. There are broadly two major classifications based on modes: wired communication and wireless communication**(press next)** * In wired communication, we have USB**(press next)** RS232 **(press next)** etc.**(press next)** * In wireless communication we have ZIGBEE **(press next)** Bluetooth, **(press next)**WiFi etc.   **(press next)** |
| 05.  Inbuilt UART pins of ATmega 2560 | * Now we will discuss about UARTs supported in ATmega 2560 * ATmega 2560 has four inbuilt UARTs namely UART0,UART1,UART2,   UART3.These UARTs are by default configured to various communication modules in Firebird. **(press next)**   * UART0 by default is connected to ZIGBEE module.**(press next)** * UART1 is connected to RS232 serial port **(press next)** * UART2 is connected to FT2232 USB serial port**(press next)** * UART3 is left free in the expansion slot which provides an option to connect any other serial communication module like Bluetooth. **(press next)** * Various UARTs and their corresponding RX and TX pins alongwith the modules are listed in the table. |
| 06.  Types of Registers | * Now coming to the Registers. Here we will see the various registers involved in Serial Communication. They are**(press next)** * UCSRnA-USART control and status register A **(press next)** * UCSRnB-USART control and status register B **(press next)** * UCSRnC-USART control and status register C **(press next)** * UCSRn-USART control and status register B **(press next)** * UBRRnL& UBRRnH –USART baud rate register **(press next)** * UDRn-USART input output register. * Here n represents the UART number. In forthcoming slides we shall discuss about the registers in detail. **(press next)** |
| 07.  USCRnA control and status register A | * Now let us consider the USCRnA register. The purpose of this register is to control and check the status of serial communication. It is an 8 bit Register with each bit having its own importance.**(press next)** * Bit 7 – is the receive complete flag. It is set when there is unread data in the receive buffer UDR & cleared when receive buffer UDR is empty.   If receiver is disabled, the receive buffer is flushed and this bit becomes zero.  This flag is used to generate a receive complete interrupt. This bit needs to be set zero in our program. **(press next)**   * Bit 6-is the transmit complete flag. This flag is set when the entire frame in the transmit shift register has been shifted out and no new data is there in transmit buffer. TXCn flag is automatically cleared when transmit complete interrupt is executed or can be cleared by writing one to its bit location. This flag is used to generate a transmit complete interrupt flag. This bit needs to be set to zero in our program. **(press next)** * Bit 5- is called the data empty register. This flag indicates that the transmit buffer is ready to receive new data. If UDERn is one, the buffer is empty and ready to be written. The UDREn flag can generate data register empty interrupt and is set after a reset to indicate that the transmitter is ready. This bit needs to be set to zero in our program. **(press next)** * Bit 4- is called frame error. This flag is valid until the receive buffer is read. This flag is zero when stop bit of received data is one, indicating no error. While writing UCSRnA set this bit to zero. **(press next)** * Bit 3- is the data overrun flag. This bit occurs when receive buffer is full. Always set this bit to zero when writing UCSRnA. **(press next)** * Bit 2- is the parity error flag. This bit is set if next character of receive buffer had a parity error when received. Always set this bit to zero when writing to UCSRnA. **(press next).** * Bit 1- This bit doubles the USART transmission speed. Write this bit to zero when in synchronous operation. Writing this bit to one will reduce the divisor of baud rate from 16 to 8 effectively doubling the transfer rate of asynchronous communication. This bit needs to be set to zero in our program. **(press next)** * Bit 0- is called Multiprocessor communication mode. The transmitter is unaffected by this flag at receiver. When this bit is set the receiver ignores the incoming frames that do not contain address information. This bit needs to be set to zero in our program. **(press next)** * Hence the bit to be uploaded in UCSRnA is 0x00 |
| 08  USCRnB control and status register B | * Now let us consider the USCRnA register. The purpose of this register is to control and check the status of serial communication. It is an 8 bit register **(press next)** * Bit 7- is called RX complete interrupt enable. Writing this bit to one will enable interrupt on RXCn flag. A USART receive complete interrupt will be generated only if the RXCIEn bit is written to one, the global interrupt in SREG is written to one and the RXCn bit in UCSRn is set. This bit needs to be set to one in our program. **(press next)** * Bit 6-is called TX complete interrupt enable. Writing this bit to one enables interrupt on TXCn flag. A USART transmit complete will be generated only if TXCIEn bit is written to one and TXCn bit in UCSRnA is set. This bit needs to be set to zero in our program. **(press next)** * Bit 5- is called USART data register empty interrupt enable n. Writing this bit to one enables interrupt on UDREn flag. A data register empty interrupt is generated only if UDRIEn is written to one and UDREn bit in UCSRnA is set. This bit needs to be set to zero in our program. **(press next)** * Bit 4- is Receive enable n. Writing this bit to one enables USART receiver and the receiver buffer will be flashed invalidating frame error, data overrun and parity error. This bit needs to be set to one in our program. **(press next)** * Bit 3- is the transmitter enable n. Writing this bit to one enables USART transmitter and the transmitter will override normal operation of TxDn pin. This bit needs to be set to one in our program. **(press next)** * Bit 2- gives the character size combined with UCSZn 1:0 bit in UCSRnC, which is explained with figure. This bit needs to be set to zero in our program. * Bit 1- is receive data bit 8n. When operating with serial frames with nine data bits this bit gives the ninth bit from the received character. And must be read before bits from UDRn. This bit needs to be set to zero in our program. * Bit 0- Transmit data bit 8n. This is ninth bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing low bits to UDRn. This bit needs to be set to zero in our program. * Hence the bit to be uploaded in UCSRnA is 0x98   **(press next)** |
| 09  USCRnC control and status register C | * Now let us consider the USCRnC register. The purpose of this register is to control and check the status of serial communication. It is an 8 bit register. **(press next)** * Bit 7 and bit 6 are the USART mode select bits. Details of what each combination does is given in the next slide. For our purpose of asynchronous USART mode they need to be set to zero-zero. **(press next)** * Bit 5 and Bit 4 are parity mode bits. Details of what each combination does is given in next slide but here in our program we shall disable parity by setting this bit to zero-zero.**(press next)** * Bit 3 is stop bit select. In our program we shall select one stop bit by setting this bit to zero. **(press next)** * Bit 2 and bit 1 along with bit 2 of UCSRnB is used to set character size. The details of function of input of different characters is given in next slide but we shall set this to one-one to select 8-bit character size. **(press next)** * Hence the bit to be uploaded in UCSRnA is 0x06. **(press next)** |
| 10  Slide 10 | * The outputs of different combinations of some of the bits of UCSRnC are given here. * First bit 7 and bit 6 of UCSRnC called UMSELn and UMSELn0 when set zero-zero selects asynchronous mode of USART. Setting it Zero-one selects synchronous USART mode. It must not be set in one-zero mode as it is reserved And setting it one-one selects Master SPI mode.**(press next)** * Bit 5 and bit 4 of UCSRnC called UPMn and UPMn0 are parity mode selectors. When these bits are set to zero-zero, parity is disabled. They must not be set zero-one. When they are set to one-one, even parity is enabled.**(press next)** * Bit 3 of UCSRnC called USBSn used to set stop bit when set 0 selects bit stop bit and when set 1 selects 2 bit stop bit.**(press next)** * Bit 2 of UCSRnB along with bit 2 and bit 1 of UCSRnC called UCSZn2, UCSZn1, UCSZn0 gives 5 bit, 6 bit, 7 bit, 8 bit, and 9 bit frame sizes by setting these bits to different combinations. |
| 11.  UBRRnL and UBRRnH | * The UBRRnL and UBRRnH registers are used to set the baud rates of the USART port. * For a crystal frequency of 14.7456 MHz, an example to obtain a standard baud rate of 9600 is given below: **(press next)** * So after calculating the baud rate using this formula we can upload the register with the calculated hex value. * Note that while loading UBRR we must first load UBRRH then UBRRL. * In the above example load UBRRH with 0x00H and UBRRL with 0x5FH.**(press next)** |
| 12  UDRn | * Now let us consider the UDRn register which is the USART input output register. **(press next)** * The USART Transmit Data Buffer Register and USART Receive Data Buffer Register share the same I/O address referred to as USART Data Registers or UDR. **(press next)** * The transmit Data Buffer register(TxB) will be the destination for data written to the UDRn Register location. **(press next)** * Reading the UDRn Register location will return the contents of the received data buffer register(RxB). **(press next)** * While transmitting we will write data to this register and while receiving we will receive data from buffer of this register. **(press next)** |
| 13.  Interrupts in Serial Communication  Receive Complete ISR | * Now let us move on to the interrupts in serial communication. First, we shall take a look at the receive complete interrupt. When UART receives eight data bits on receive pin of the microcontroller, RXC flag is set. If RXCIE interrupt is enabled then receive complete interrupt triggers ISR. This ISR then reads valid data from UDR1 and stores it in a separate variable before next character is received and overwritten. It is always recommended to save the data read from UDR1 in a separate variable as next character received will overwrite and destroy the existing data in UDR1. This is the general syntax for the receive complete interrupt subroutine. **(press next)** |
| 14  Data Register empty ISR | * The transmitter side of the UART is double buffered containing UDRn to hold the data written from the program and transmit register to actually transmit parallel data sequentially bit-by-bit on the transmit pin. The data written to UDRn is transferred to transmit register. At this point, the UDRn is available to accept next data word from the program. This sets UDRE flag and if UDRIE interrupt is enabled then UDRn data register empty interrupt triggers ISR. This ISR then loads next data byte to be transmitted into UDRn. **(press next)** |
| 15  Receive Complete ISR | * In case of packet based data communication it is necessary to know when a byte has been completely transmitted out of the microcontroller. The TXC flag is provided to indicate that the transmit register is empty and no new data is waiting to be transmitted. If the transmit register is empty it sets the TXC flag and if TXCIE interrupt is enabled then transmit complete interrupt triggers ISR. This ISR can be used as a confirmation that the byte that was loaded in UDR1 is successfully transmitted out of the microcontroller transmit pin. This interrupt can be used to check if all the bytes in a packet transmission are transmitted successfully. **(press next) (press next)** |
| 16  C code | * Now moving on to the C code for UART initialization for serial communication. * We need to first disable UCSRnB while setting baud rate. * Then we can set UCSRnA and UCSRnC according to the discussion discussed before. * Then set UBRRnL and UBRRnH to set baud rate to a value calculated before * And finally now you can set UCSRnB also to the discussed value**(press next)** |
| 17.  Demonstration | * Now let us move to the practical demonstration of serial communication in various ways. |
| 18.  Thank you | With this we have come to the end of this tutorial. Thank you for listening. For any queries or doubts you can visit <http://qa.e-yantra.org/>  This is XXXXX signing off!! |