UTFPR - DAELN - CT
Sistemas Embarcados - CSW42
Prof. Douglas Renaux
Raissa A N Higa

Lab 4 - Relatório

Para esse laboratório foi feito um planejamento baseado nos laboratórios anteriores, seguindo os passos:

1-Definição do problema
2-Especificação da solução
3-Estudo da plataforma de HW
4-Estudo da plataforma de SW
5-Design da solução
6-Codificação
7-Teste e Depuração

A definição do problema é criar um projeto que receba dados e converta os caracteres maiúsculos em seus correspondentes minúsculos sem alterar os demais bytes recebidos. Para isso seria necessário utilizar alguma funcionalidade de transmissão e recepção serial, que no caso em questão é através de UART, no formato 8N1 e BAUD RATE 115200bps.

Analisando o manual da placa, verifica-se a existência de uma porta UART0 que opera através da porta USB a qual é conectada ao PC, e os GPIO PortA 0 e 1 possuem função de receptor e transmissor U0Rx e U0Tx respectivamente.

2.3.3 Virtual COM Port

When plugged into a USB host, the ICDI enumerates as both a debugger and a virtual COM port. JP4 and JP5 control the selection of which UART from the TM4C1294NCPDTI is connected to the virtual COM port. In the default configuration, UART0 maps to the virtual COM port of the ICDI. In the CAN jumper configuration, UART2 maps to the virtual COM port of the ICDI.

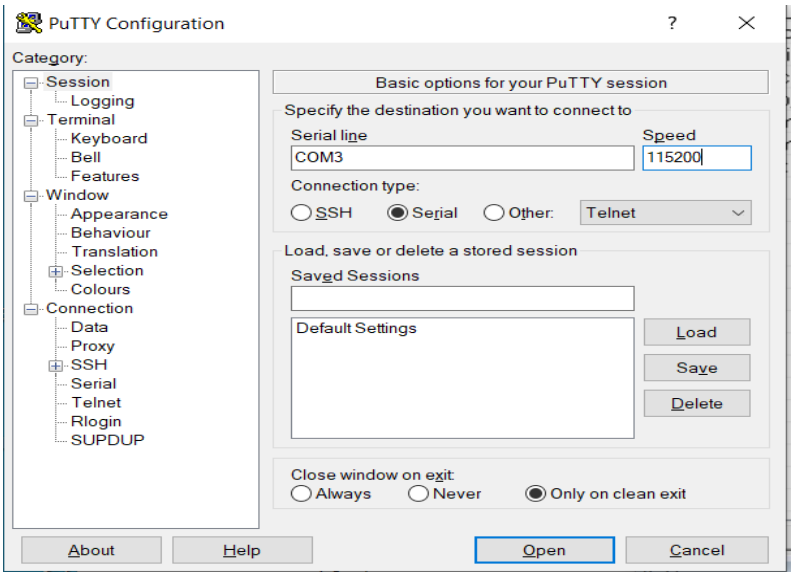
IO	Pin	Analog or Special Function[a]	1
PA0	33	-	U0Rx
PA1	34	-	U0Tx

Para a utilização de UART com o TivaWare, encontram-se as funções de configurações e operações de UARTs que possam ser úteis ao projeto, presentes em uart .h e .c.

- int32_t UARTCharGetNonBlocking (uint32_t ui32Base)
- void UARTCharPut (uint32_t ui32Base, unsigned char ucData)
- bool UARTCharPutNonBlocking (uint32_t ui32Base, unsigned char ucData)
- bool UARTCharsAvail (uint32_t ui32Base)
- uint32_t UARTClockSourceGet (uint32_t ui32Base)
- void UARTClockSourceSet (uint32_t ui32Base, uint32_t ui32Source)
- void UARTConfigGetExpClk (uint32_t ui32Base, uint32_t ui32UARTClk, uint32_t *pui32Baud, uint32_t *pui32Config)
- void UARTConfigSetExpClk (uint32_t ui32Base, uint32_t ui32UARTClk, uint32_t ui32Baud, uint32_t ui32Config)
- void UARTDisable (uint32_t ui32Base)
- void UARTEnable (uint32_t ui32Base)
- uint32_t UARTFlowControlGet (uint32_t ui32Base)
- void UARTFlowControlSet (uint32_t ui32Base, uint32_t ui32Mode)
- void UARTIntClear (uint32_t ui32Base, uint32_t ui32IntFlags)
- void UARTIntDisable (uint32_t ui32Base, uint32_t ui32IntFlags)
- void UARTIntEnable (uint32_t ui32Base, uint32_t ui32IntFlags)
- void UARTIntRegister (uint32_t ui32Base, void (*pfnHandler)(void))
- uint32_t UARTIntStatus (uint32_t ui32Base, bool bMasked)
- void UARTIntUnregister (uint32_t ui32Base)
- void UARTLoopbackEnable (uint32_t ui32Base)
- void UARTModemControlClear (uint32_t ui32Base, uint32_t ui32Control)
- uint32_t UARTModemControlGet (uint32_t ui32Base)

No design da solução, primeiramente é necessário habilitar os periféricos do UART0 e PORTA a serem utilizados e após isso determinar suas configurações de modo e velocidade. Como a solução é baseada em interrupções, também é habilitada a interrupção para o UART0 e implementada sua rotina de tratamento. Nela limpa-se as interrupções do próprio tipo e verifica-se se tem algum caractere recebido na FIFO, coletando seu dado e caso represente um valor entre 65 e 90, significa que é um caractere de uma letra maiúscula(A-Z), segundo a tabela de caracteres representados em ASCII, então se soma 32 ao seu valor, pois em ASCII, (a-z) são representados dos números 97-122, e é transmitido de volta, caso contrário, o dado é retransmitido da forma em que foi recebido.

Para os testes e depurações foi necessário utilizar de um emulador de terminal, no caso o PuTTY, definindo a porta COM mapeada para a placa, velocidade e tipo.



Fontes: Imagens retiradas dos manuais TivaWare™ Peripheral Driver Library USER'S GUIDE, Getting Started with TivaWare™ for C Series e Tiva TM4C1294NCPDT Microcontroller DATA SHEET.