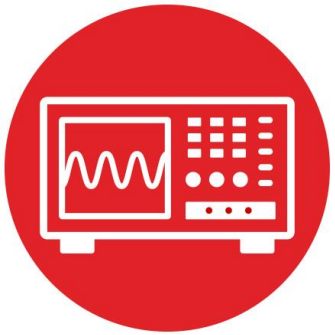

模块 9

实验：UART 通信

实验：UART 通信

9.0 目标

本实验的目的是为 MSP432 上的 UART 开发一个中断驱动的软件驱动程序。在这个模块中，

1. 您将开发先进先出（FIFO）队列来在前台和后台之间传输数据。
2. 您将评估中断 UART 驱动程序的性能。
3. 您将设计，开发和测试可用于机器人系统的命令解释程序。

小知识：复杂系统有很多交织的组件。将数据从一个模块传输到另一个模块需要同步。 FIFO 队列是一种有效的数据流机制，无需紧密耦合两个模块的执行。

9.1 入门

9.1.1 从下面的软件工程起步

浏览下面 2 个工程：
UART（UART 接口的忙等待解决方案）
Lab_UART_TTS（对应视频中语音播报模块程序）
Lab_UART（本实验的入门项目）

9.1.2 参考资料

MSP432P4xx Technical Reference Manual, Timer_A (SLAU356)
MSP432P401R Datasheet, msp432p401m.pdf (SLAS826)
SYN6288 数据手册 语音播报模块数据手册

9.1.3 阅读材料

Volume 1 Sections 4.5, 8.2, 11.3, and 11.4
“Embedded Systems: Introduction to the MSP432 Microcontroller",
或
Volume 2 Sections 3.4, 3.7, 4.9, and 5.6
“Embedded Systems: Real-Time Interfacing to the MSP432 Microcontroller"

9.1.4 本实验所需组件

数量	组件描述	制造商	型号
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	语音播报模块	JiangNiu	

除 LaunchPad 外，您还可以使用任何可用的机器人功能来设计命令解释程序。

9.1.5 所需实验设备

无

9.2 系统设计要求

本实验的第一个目标是开发一个中断驱动的 UART 驱动程序，并使用它来为机器人搭建一个命令解释器。

注：使用 UART 作为调试机制时，执行 **EUSCIA0_OutUDec** 和 **EUSCIA0_OutString** 等函数的时间决定了调试输出的侵入性。使用中断驱动的 UART 驱动程序，如果 FIFO 队列足够大并且输出速率足够低，则 FIFO 永远不会填充满。如果 FIFO 不被填满，则数据不会丢失，并且执行输出功能将会花费很短的时间。

更具体地说，您将开发 UART 串行端口驱动程序所需的两个 FIFO 队列。**TxFifo0** 将数据从主程序输出到 UART ISR，**RxFifo0** 将输入数据从 UART ISR 输出到主程序。您将在头文件 **FIFO0.h** 中找到原型。每个 FIFO 在永久存储器中都有一个缓冲区。**Init** 函数可以将 FIFO 初始化，将其清空。**Put** 函数将数据存储到 FIFO 中，**Get** 函数从 FIFO 中删除数据。FIFO 保留顺序，换句话说，从 FIFO 中删除数据的顺序与放置数据的顺序相匹配。具有 64 个条目的缓冲区可以包含 0 到 63 个数据项。使用 64 个单元时不允许省略了空（无条目）和完整（63 个条目）之间的判断。如果在 **Put** 开始放置时 FIFO 已满，则该函数返回填充满错误值。如果在 **Get** 开始时 FIFO 为空，则该函数返回一个空错误值。

第二个要求是编写一个解释器。当运行像 TExaSdisplay 或 PuTTy 这样的终端模拟器时，输入来源为键盘。随意创建自己的语法和命令列表。例如

If you type	*then the robot does*
Stop	The robot stops
Go	The robot goes straight
Back	The robot backs up
Left	The robot turns left
Right	The robot turns right
Slow	Set duty cycle to 2500
Fast	Set duty cycle to 7500
Sensor	Read and display sensor values

本实验的第二个目标是基于 UART 通信添加语音播报模块，实现机器人智能语音播报系统。

实验：UART 通信

9.3 命令解释器实验准备

UART 数据通过 USB 调试电缆传输。因此，在本实验中，必须将 USB 电缆从机器人连接到 PC。

9.4 命令解释器实验步骤

9.4.1 开发并测试 FIFO 队列

实现四个 FIFO 函数，用于将数据从前台传输到 UART ISR：**TxFifo0_Init**，**TxFifo0_Put**，**TxFifo0_Get** 和 **TxFifo0_Size**。可以使用 **Program 18_1** 测试这些功能。在此测试中，主程序调用 **Put** 和 ISR 调用 **Get**。数据应按顺序流传输，FIFO 永远不会填满。

```
char WriteData,ReadData;
uint32_t NumSuccess,NumErrors;
void TestFifo(void){char data;
    while(TxFifo0_Get(&data)==FIFOSUCCESS){
        if(ReadData==data){
            ReadData = (ReadData+1)&0x7F; // in sequence
            NumSuccess++;
        }else{
            ReadData = data; // restart
            NumErrors++;
        }
    }
}
uint32_t Size;
int Program18_1(void){  // NumErrors should be zero
    uint32_t i;
    Clock_Init48MHz();
    WriteData = ReadData = 0;
    NumSuccess = NumErrors = 0;
    TxFifo0_Init();
    TimerA1_Init(&TestFifo,43);  // 83us, = 12kHz
    EnableInterrupts();
    while(1){
        Size = Random(); // 0 to 31
        for(i=0;i<Size;i++){
           TxFifo0_Put(WriteData);
           WriteData = (WriteData+1)&0x7F; // in sequence
        }
        Clock_Delay1ms(10);
    }
}
```

注：我们建议您不要维护包含 FIFO 中条目数的计数器。当两个函数在多线程系统中使用时，在 Put 期间递增计数器，在 Get 期间递减计数器将创建一个临界部分。

9.4.2 OutString 的性能评估

本节的目的是比较忙等待和中断驱动程序。在两个系统中，将传输随机大小的字符串。执行 OutString 的时间使用 SysTick 测量。由于两个版本具有相同的 115200 位/秒波特率，因此执行输出的实际时间将相同。但是，您会看到，与忙等待相比，OutString 的中断驱动的执行时间会缩短很多。

编译并运行 **Program18_2**。以 usec 记录 **MaxTIme**。

```
char String[64];
uint32_t MaxTime,First,Elapsed;
int Program18_2(void){      // busy-wait OutString
    uint32_t i;
    DisableInterrupts();
    Clock_Init48MHz();
    UART0_Init();
    WriteData = 'a';
    SysTick_Init();
    MaxTime = 0;
    while(1){
        Size = Random(); // 0 to 31
        for(i=0;i<Size;i++){
            String[i] = WriteData;
            WriteData++;
            if(WriteData == 'z') WriteData = 'a';
        }
        String[i] = 0; // null termination
        First = SysTick->VAL;
        UART0_OutString(String);
        Elapsed = ((First - SysTick->VAL)&0xFFFFFF)/48;
        if(Elapsed >MaxTime){
            MaxTime = Elapsed;
        }
        UART0_OutChar(CR);UART0_OutChar(LF);
        Clock_Delay1ms(100);
    }
}
```

以类似的方式，编译并运行 **Program18_3**。除了使用了中断驱动的 **Outstring** 版本，他们基本上是相同的系统。同样的，记录 **MaxTime**。因为 FIFO 永远不会被填满，所以对 **OutString** 的调用执行得非常快。请注意，在 **FIFO0.c** 中，每次调用 **TxFifo0_Put** 都将测量 FIFO 大小并实现建立一个直方图。该直方图是概率质量函数（PMF），它计算每个 FIFO 大小发生的次数。在调试器中，观察此直方图的内容。您可以使用此测量值来预测 FIFO 中的最大元素数。

注：有一个完整的数学规律叫做**排队论**（Queuing Theory）。该理论的核心是对 FIFO 队列大小数据的收集和解释。

9.4.3 创建第二个 FIFO

完全调试 **TxFifo0** 后，复制/粘贴此代码以实现 **RxFifo0**。**Program 18_4** 可用于测试串行输入和输出。

9.4.4 开发和测试解释器

编写实现解释器的主程序。您可以随意调整命令数量和解释器的确切语法。解释器的目的是帮助解决机器人的挑战。

实现命令解释器的一种方法是创建一个将命令名称映射到命令函数的表。例如，此结构体可以包含字符串和函数指针。

```
typedef struct {
    char CmdName[8];          // name of command
    void (*fnctPt)(void);     // to execute this command
}Cmd_t;
const Cmd_t Table[8]={
{ "Stop",    &doStop},
{ "Go",      &doGo},
{ "Back",    &doBack},
{ "Left",    &doLeft},
{ "Right",   &doFast},
{ "Slow",    &doSlow},
{ "Fast",    &doFast},
{ "Sensor"   &goSensor}};
```

其中 **doStop**，**doGo**，...等是实际执行相关命令的 void-void 函数。解释器通过调用 **EUSCIA0_InString** 读取字符串，然后在表中搜索匹配项。匹配项被找到后，其对应功能将会被执行。

9.5 语音播报系统实验准备

语音播报模块是一款高集成的语音合成模块，可实现中文、英文、数字的语音合成；并且支持命令词或提示音的定制需求。本模块控制简单，本实验将 V1.1 转接板安装在机器人上面，然后将语音播报模块安装在 V1.1 转接板的⑨号位置，连接 MSP432 Launchpad 的 P9.6 和 P9.7 引脚，使用 UART3 通信。引脚连接如下图 1 所示：

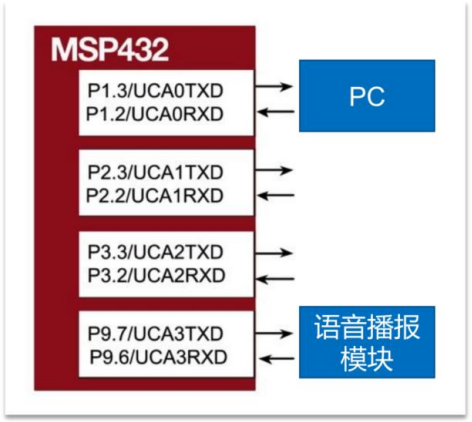

图 1. 语音播报模块引脚连接

9.6 语音播报系统实验步骤

首先定义播报内容的数组，本实验使用 GBK 编码格式的文本播报内容为“匠牛社区”，`char sendData[14]={0xFD, 0x00, 0x0B, 0x01, 0x01, 0xBD, 0xB3, 0xC5, 0xA3, 0xC9, 0xE7, 0xC7, 0xF8, 0x8F};`

帧结构	帧头	数据区长度	数据区			
			命令字	命令参数	待发送文本	异或校验
数据	0xFD	0x00 0x0B	0x01	0x01	匠牛社区 0xBD 0xB3 0xC5 0xA3 0xC9 0xE7 0xC7 0xF8	0x8E
数据帧	0xFD, 0x00, 0x0B, 0x01, 0x01, 0xBD, 0xB3, 0xC5, 0xA3, 0xC9, 0xE7, 0xC7, 0xF8, 0x8F					
说明	播放文本编码格式为“GBK”的文本“匠牛社区”，不带音乐背景					

第二步初始化配置 P9.7 和 P9.6 引脚为 UART 功能，配置 UART3 波特率为 9600,8 位数据位，无奇偶校验位，1 个停止位，初始化函数为：`void UART3_Init(void);`

实验：UART 通信

第三步设置发送数据数组的函数，参数 1 指向要传输的数据数组的指针；参数 2 数据数组大小，函数为：**void UART3_Out_arr(char *pt,int size);**

第四步设置获取串口输入的函数，函数为：**char UART3_InChar(void)**；

实现播报“匠牛社区”功能的 main 函数具体实现为：

```
void main(void)
{
    char recData;
    char sendData[14]={0xFD, 0x00, 0x0B, 0x01, 0x01, 0xBD,
0xB3, 0xC5, 0xA3, 0xC9, 0xE7, 0xC7, 0xF8, 0x8F}; //GBK 编码
格式

    Clock_Init48MHz();  // makes SMCLK=12 MHz
    UART0_Initprintf(); // initialize UART and printf
    UART3_Init();
    while(1){
       //MSP432向语音播报模块发送“匠牛社区”
       UART3_Out_arr(sendData,14);

       recData = UART3_InChar();
       //返回0x4E 表明语音播报模块仍在合成播音中；   返回0x4F
表明文本被正确接收且合成播音完毕，语音播报模块处于空闲状态；
0x41 表明接收成功；
       printf("== %02x ==\n ",recData);
       Clock_Delay1ms(2000);
    }
}
```

注：UART3 的初始化函数、发送数据函数、接收数据函数具体实现位于 UART3.c 文件。

9.7 疑难解答

没有串行输出：

- 运行 UART 项目。它的输出为 115200 bps。
- 有两个与 MSP432 关联的 COM 端口，使用较小的串口号。

无法打开 MSP432 的 COM 端口：

- 检查设备管理器以获取 COM 端口号。
- 有时 CCS 会打开 COM 端口，阻止 TExaSdisplay 或 PuTTy 访问。关闭 CCS，拔出 MSP432，再插入 MSP432，启动 TExaSdisplay 或 PuTTy，打开 COM 端口，然后启动 CCS。

TxFifo 或 RxFifo 偶尔会丢失数据：

- 确保 FIFO 正确处理 Get 上的空和 Put 上被填满的情况。
- 确保 **Put** 和 **Get** 没有被写入相同的共享全局。这将使用临界区。 **Get** 可以写入 **Put** 读取的全局。 或是将 **Put** 写入 **Get** 读取的全局。

Program 18_4 不起作用：

- 重新测试 FIFO 队列。

9.8 请思考

在本节中，我们列出了完成本实验后要考虑的思考问题。这些问题旨在测试您对本实验中概念的理解。此模块的目标是让您了解 FIFO 队列及其在线程之间流数据的用法。

- 思考使用 **EUSCIA0_OutString** 的输出通道。如果 **TxFifo0** 通常为空，这意味着什么？
- 思考使用 **EUSCIA0_OutString** 的输出通道。如果 **TxFifo0** 通常为满，这意味着什么？
- 思考使用 **EUSCIA0_InString** 的输入通道。如果 **RxFifo0** 通常为空，这意味着什么？
- 考虑使用 **EUSCIA0_InString** 的输入通道。如果 **RxFifo0** 通常为满，这意味着什么？
- 假设您使用 FIFO 队列在线程之间传输数据。您可以定期测量 FIFO 大小并计算 FIFO 平均大小。设 **N** 是 FIFO 中元素的平均数（以字符为单位）。假设您已知**λ**，数据发送的平均速率（以字符/秒为单位）。使用**利特尔法则**（Little’s Law）估计平均响应时间，即数据在队列中等待发送的时间。

9.9 其它挑战

在本节中，我们列出了您可以执行的其它活动，以进一步探索此模块的概念。例如，

- 运行有直方图和没有直方图的 Program 18_3，以评估维护直方图所需的开销。

- 以第二种方式实现 TxFifo0（例如，指针和索引）。使用 Program 18_3 估计两种方法的相对速度。
- 了解卡恩进程网络（KPN）。

9.10 接下来是哪些模块？

完成本单元后，您就可以解决任何机器人设计挑战。如果您希望扩展机器人以包括无线通信，您有两种选择：

模块 13）添加 WiFi 功能。
模块 15）添加蓝牙功能。

9.11 您应该已经学会

在本节中，我们将回顾您应该在本单元中学到的重要概念：

- 理解 FIFO 队列如何允许您在复杂系统上的线程之间传输数据。
- 了解 PMF 如何用于描述队列的行为。
- 了解如何使用 FIFO 队列，以便队列永远不会变满。