数字电子技术与微处理器基础

实验指导书

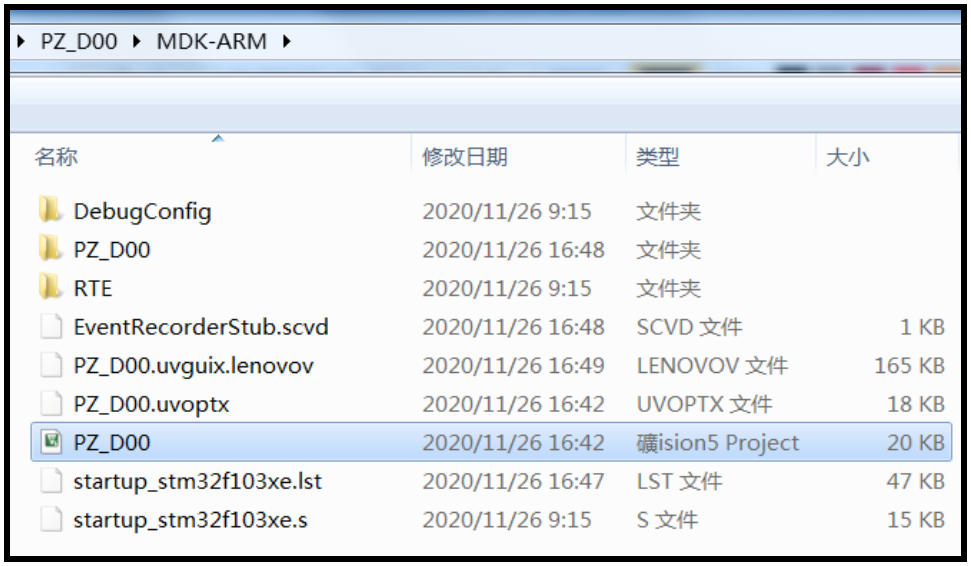
（微处理器部分）

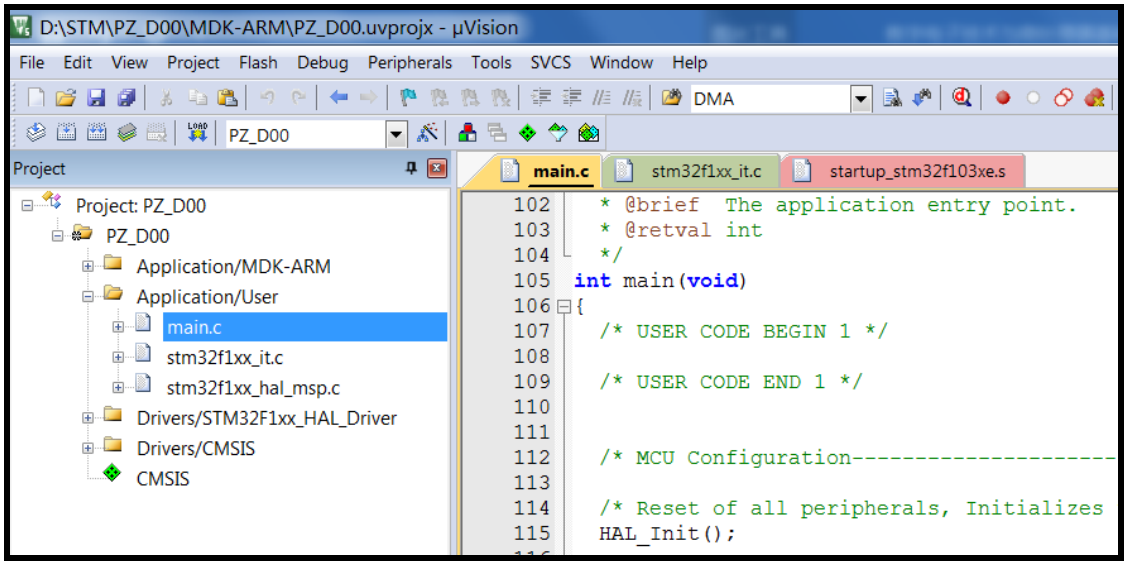
1. 实验教师仅对实验环境及装置异常提供指导与支持。对软件设计及完成办法不予提示、帮助
2. 示例代码仅供参考，不保证满足实验目标。所有代码需自行了解并完成设计。
3. 软、硬件资源需实验前查阅相关资料，如实验板原理图、指导文件。
4. 实验报告须附反映设计、结果内容的屏幕截图。<u>此作为评分重要依据</u>

一、实验设备

1. PC 计算机
2. 普中 STM32-F1 开发板(PZ-6806L)

二、打开工程文件(xxx.uvprojx)，进入 KEIL IDE 开发集成环境



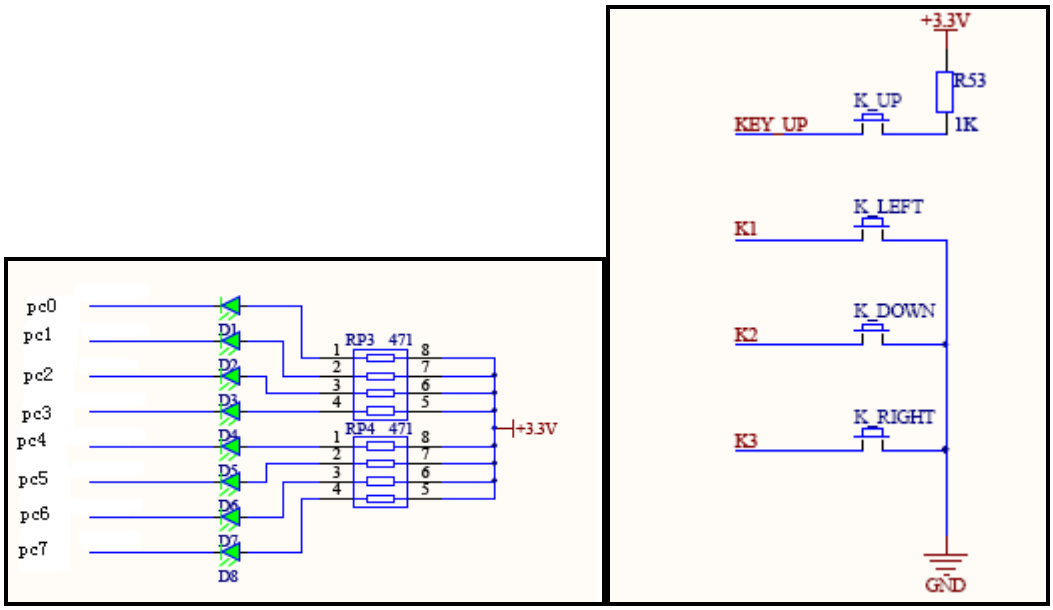


实验一 微处理器应用编程及基本输入/输出实验

一、实验目的

1. 熟练掌握开发环境及 CPU、外设接口、数据的观察、调试等开发方法。
2. 通过 LED、按键，学习、掌握 I/O 的工作原理及编程、应用方法。
3. C 语言、机器指令相结合，观察指令、寄存器,理解、领会微处理器系统工作。

二、实验原理图



a)KEY_UP 按键接 PA0 引脚，K1（PB2）、K2(PB3)、K3(PB4)按键

b)8 个 LED 灯分别接 PC 口(PC0-7)的输出，当 PC 口某位输出为 0 时，相应指示灯即可点亮。

数据相关　　【主要处理数据安排在此数据区，以 memeoy 及 watch 方式观察】

```
//---此程序仅供参考，不提供更多解释。有疑问请查阅资料。进一步内容及设计需自行解决。
struct  staCOMPONENT
{   short     sBUF[16],ADdat[16],   UART_rBUF[8],   mDMA[8],
              mRTC,     Cnt00,      Cnt01,          Cnt02;}md;   //该数据做程序处理、观察的主要方面
```

三、实验程序 【此 GPIO 初始化代码由 CubeMX 生成，可做修改。更多修改建议在 CubeMX 环境进行】

```
static void MX_GPIO_Init(void)   CubeMX
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                          |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_5, GPIO_PIN_SET);

  /*Configure GPIO pins : PE2 PE3 PE4 */
  GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

  /*Configure GPIO pins : PC0 PC1 PC2 PC3 PC4 PC5 PC6 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                          |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : PA0 */
  GPIO_InitStruct.Pin = GPIO_PIN_0;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLUP;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

延时程序：
【该函数用于与延时、间隔、周期有关问题，可根据需要在不同层次安排一些与时间控制相关内容】

```
void Delay(unsigned tDly)
{   short tDi;
     while(tDly--)
     { tDi=4000;
       while(tDi--)
       {   if(!(md.mDMA[6]&0x10))   continue;
                md.mDMA[6]&=~0x10;      md.mDMA[5]++;
              //can do some works, but no stay here more time
       }
     //自行 Begin......u can do something
          md.mRTC=RTC->CNTL;


     //自行 END  ......u can do something
     };
}
//--主要处理部分-------------------------
while (1)
  {
       if(!(GPIOA->IDR&1))  i++;   else
       {   i+=9;                   }
       GPIOC->BSRR|=LEDcd[i%=10]&0xff;
       Delay(500+i*23);          GPIOC->BRR|=0xff;
    //-----------------------------------
       md.sBUF[i]=i*3+x;
       if(i==0)   {   md.sBUF[14]=md.sBUF[15]; md.sBUF[15]=0;}
       md.sBUF[15]+=md.sBUF[i];
    //Begin


    //END
  }
```

四、目标要求：

1. 填充学号至 sBUF，通过 8 段 LED，轮流显示自己学号各位。
2. 按下 UP 键（PA0），倒序（或暂停）显示自己学号
3. 根据学号个位数，调整更新间隔[0.5s+学号个位*0.1s]
4. 对 sBUF 前 10 个数据累加、结果存至 sBUF[15]
5. 最后设断点，在 UP 键按下时，可暂停至断点。

实验二 定时器及中断实验

一、实验目的

1.了解 STM32-F1 系列处理器定时器及定时中断的工作原理及编程方法

2.编写定时中断服务程序，完成周期性工作，并为其他模块提供时间控制。

二、实验内容

设定定时器周期，设计定时中断服务程序。以变量计数器观察其运行。

三、试验程序

```
static void MX_TIM3_Init(void)
{
  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};
  TIM_OC_InitTypeDef sConfigOC = {0};

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 7;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 2499;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim3.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK) Error_Handler();

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
     Error_Handler();
  if (HAL_TIM_PWM_Init(&htim3) != HAL_OK) Error_Handler();
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_OC1;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) !=
HAL_OK) Error_Handler();
  sConfigOC.OCMode = TIM_OCMODE_PWM1;
  sConfigOC.Pulse = 6;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  if (HAL_TIM_PWM_ConfigChannel(&htim3, &sConfigOC, TIM_CHANNEL_2) !=
HAL_OK) Error_Handler();
  /* USER CODE BEGIN TIM3_Init 2 */

  /* USER CODE END TIM3_Init 2 */
  HAL_TIM_MspPostInit(&htim3);

}
void HAL_TIM_MspPostInit(TIM_HandleTypeDef* htim);
```

中断服务程序（参考示例，需根据不同目的要求设计）

```
void TIM3_IRQHandler(void)
{
  /* USER CODE BEGIN TIM3_IRQn 0 */
    static unsigned short LEDpwm;
    LEDpwm++; LEDpwm%=9800;
    if(LEDpwm<4900)   TIM3->CCR2=          LEDpwm/2+88;
    else                  TIM3->CCR2=4900- LEDpwm/2+88;
    if(LEDpwm%10==1) {   epmd[47]++;       epmd[46]|=0x000f;}
    if(LEDpwm%100==0)                      epmd[46]|=0x00f0;
       //Set a Click Flag

    if(!(GPIOE->IDR&0x08))
           //PE2(down Key) push
    {   if(LEDpwm%8>3)    GPIOB->BSRR|=1<<5;    else   GPIOB->BRR|=1<<5;
    }   //Beep
//ISR user Begin


//ISR user END
    /* USER CODE END TIM3_IRQn 0 */
  HAL_TIM_IRQHandler(&htim3);
  /* USER CODE BEGIN TIM3_IRQn 1 */

  /* USER CODE END TIM3_IRQn 1 */
}
```

四、目标要求

不依靠软件延时 Delay(unsigned tDly)，在主程序实现 1Hz 及 10Hz 周期性简单处理任务（可通过计数变量如 Cntx 观察）。

实验三　ADC 与 DMA 实验

一、实验目的

1.了解 STM32-F1 系列处理器定时器+ADC+DMA 工作原理及使用方法

2.对 ADC 数据进行简单的处理、计算

二、实验内容

通过定时器 3 定时启动 ADC，自 DMA 缓冲区读取 ADC1.1 结果，保存、计算。

三、实验程序

```
static void MX_ADC1_Init(void)
{
  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */
  /** Common config    */
  hadc1.Instance = ADC1;
  hadc1.Init.ScanConvMode = ADC_SCAN_ENABLE;
  hadc1.Init.ContinuousConvMode = DISABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConv = ADC_EXTERNALTRIGCONV_T3_TRGO;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 2;
  if (HAL_ADC_Init(&hadc1) != HAL_OK) Error_Handler();
  /** Configure Regular Channel    */
  sConfig.Channel = ADC_CHANNEL_1;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_13CYCLES_5;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
     Error_Handler();
  /** Configure Regular Channel  */
  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = ADC_REGULAR_RANK_2;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
     Error_Handler();
   /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */
}
```

```
static void MX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Channel1_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);

}
```

中断服务程序：见实验二

```
void TIM3_IRQHandler(void)
```

四、目标要求

1、设定恰当采样率（如 2499+学号个位），以此采样率得到的 ADC 采样结果（PA1 通道），陆续保存至循环缓冲区 md.ADdat[0-7],并在定时中断处理程序计算 8 点数据平均值保存到 ADdat[15]。

2、调整电位器，观察实验结果。

实验四 UART 串行通讯实验

一、实验目的

1.了解 STM32-F1 系列处理器 UART 的工作原理及编程方法

2.对收、发内容进行简单处理。

二、实验内容

实现 UART 收、发功能。对收到的命令做处理，通过 UART 发送相应内容。

三、实验程序

```
//USART1 初始化：
static void MX_USART1_UART_Init(void)
{
  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 9600;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK) Error_Handler();

  /* USER CODE BEGIN USART1_Init 2 */

  /* USER CODE END USART1_Init 2 */

}
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{   md.mDMA[2]++; md.mDMA[2]&=0x7;
    md.UART_rBUF[md.mDMA[2]]=md.mDMA[3];
    HAL_UART_Receive_IT(&huart1,(unsigned char *)&md.mDMA[3],1);
    md.mDMA[4]=md.mDMA[3]+1;
    while(HAL_UART_Transmit(&huart1,(unsigned                    char
*)&md.mDMA[4],1,5000)!=HAL_OK);
}
void HAL_UART_MspInit(UART_HandleTypeDef* huart);
```

四、目标要求：

根据接收的自行约定命令代码，通过 UART 分别发送学号或 ADC 结果（2 字节）,在 PC 串口观察相应内容。

截图参考（实际中自行选定恰当、准确内容，并对重点部分标注）
最后注意附------班级、学号、姓名