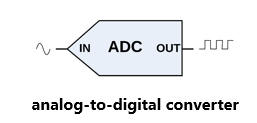
**电子学基础II实验实验报告**

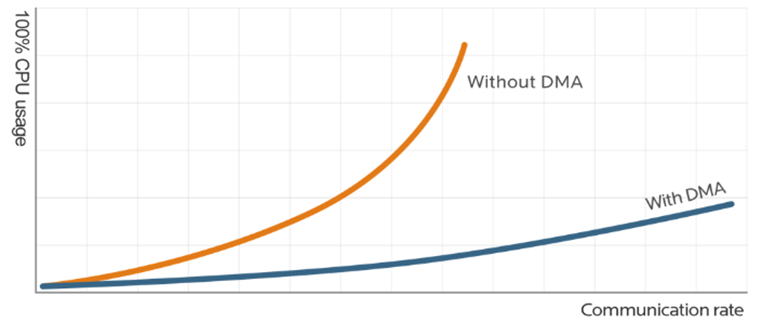
班级编号：0103 Anlan Peng，211180074；Yusong Qi，211180114

1. **实验目标**
2. 利用STM32F429IGTx型号单片机，在Keil软件编译环境下完善代码，使其采集处理第一阶段心电信号放大电路的输出，并在LCD屏幕上良好稳定地显示关键信息。
3. 通过小组合作以及几周的学习，达成以下学习目标：1) 学习掌握单片机相关知识及其工作原理，理解中断、ADC等基础概念；2) 学习使用Keil软件，提高面向功能编写高效代码的能力；3) 提高合理运用各种结构的代码和高效debug的能力；4）体验多人合作完成一个项目，提高与人沟通交流、合作的能力。
4. **实验方案**
5. 实验方案的对比与选择：
6. 数据采集与存储（ADC&数据存储设计）：

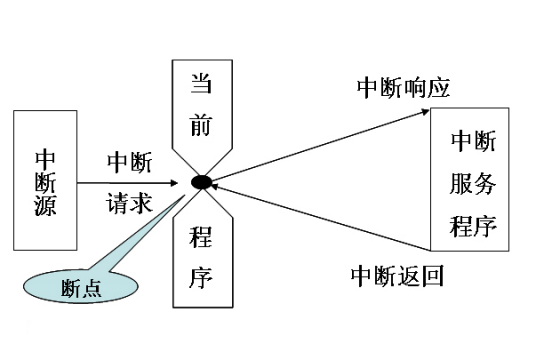
ADC(analog-to-digital converter)指模/数转换器或者模数转换器，是指将连续变化的模拟信号转换为离散的数字信号的器件。考虑到本实验中信号是以模拟信号的形式输入，而单片机的芯片却是基于数字信号的，所以需要一个模数转换器将电压这一模拟量转换成数字信号的组合，所需要的转换器即为ADC。本实验中所使用的12位ADC是一种逐次逼近型模拟数字转换器。它有多达18个通道，可测量16个外部和2个内部信号源。各通道的A/D转换可以单次、连续、扫描或间断模式执行。ADC的结果可以左对齐或右对齐方式存储在16位数据寄存器中。模拟看门狗特性允许应用程序检测输入电压是否超出用户定义的高/低阀值。

ADC的输入时钟是由PCLK2经分频产生。

（图2.1.1 ADC示意图）

DMA(Direct Memory Access)指直接存储器存取，是单片机的一个外设，它的主要功能是用来搬数据，但是不需要占用CPU。DMA的存在使得ADC在传输数据的时候，CPU 可以继续执行当前任务而不会被中断，达到类似“多线程”的效果。

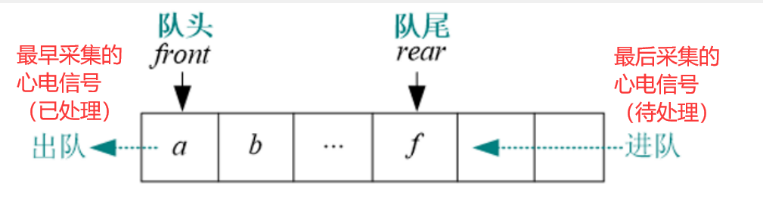
（图2.1.2 DMA在数据传输方面的作用）

中断(Interrupt Request)指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后又返回原被暂停的程序继续运行。STM32F4具备中断响应系统， 支持为数众多的系统异常和外部中断。STM32F4 的每个 IO 都可以作为外部中断的中断输入口，中断控制器支持 22个外部中断/事件请求。每个中断设有状态位，每个中断/事件都有独立的触发和屏蔽设置。

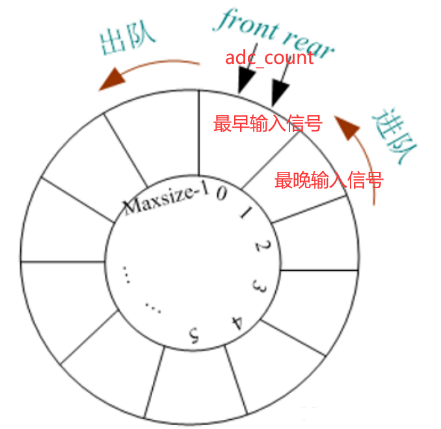
（图2.1.3 “中断”流程的示意图）

综合以上讨论我们在数据采集（ADC）时，有单次、连续、扫描、间断四种模式可供选择；在数据传输时，有DMA传输和通过软件的手段将每次采集到的数据赋值到数组中去两种选择。考虑到本实验是对频率较低的心率信号进行采样，单位时间需要采集和处理的信息量较少，并且根据课程讲解，我们着重了解的是ADC单次的采样模式和软件手段的数据传输，那么在保证单片机性能有充分余量的情况下，我们选择最简单、最熟悉的数据采集和存储手段——单次ADC采样并通过赋值语句将采集到的数据存储到静态数组中去，以期用尽量少并且尽量安全的代码实现目标功能。值得注意的是。此处的“单次ADC采样”并非表明ADC全程只采样一次，而是指在一条语句作用下，ADC仅采样一次。因此，此处还要讨论如何调用ADC采样函数，使ADC通过连续的单次采样，获得输入模拟信号的波形信息。一种方案是将ADC采样设置在主函数中，通过设置有限次数的while循环并加以delay函数控制采样频率，在一次采集够足量数据点后结束采样，再进行后面的数据处理和绘图工作。小组讨论认为这是一种极其粗糙的采样方法，会给程序带来巨大的问题：a）使用delay函数控制采样频率并不可靠——STM32中delay延时函数并不完全准确，而不均匀的采样可能会对波形采集造成一定程度的失真；b）主函数中的数据采集对整体程序运行造成严重延误——以采集到两个完整波形时开始绘图来计（若要计算频率，至少有两个波形），实验采集的心电信号频率最大为2Hz，则采集一次信号至少要等待1s，在这段时间内，单片机几乎什么都不做，意味着波形将保持静态，即LED屏幕绘制的波形刷新率最高只有1Hz，这极大浪费了单片机的性能，并且不能给出连续变化的波形（屏幕上观察到的移动波形实际上移动一次就是一整个屏幕的距离），另外也不能保证所有的信息都被采集并展现在屏幕上（每段连续采样之间还需一段时间用于main函数执行其他语句），而对小组成员而言，准确且不遗漏地采集到心电信号从接入单片机开始每一时刻的信息是贯穿整个程序设计过程的理念，且一直是程序优化的目标（小组成员认为心电信号作为观测人体生命健康的重要信号，每一“点”都有可能显示出被测量者身体存在的健康隐患，都不应该被轻易舍去）；c）对于初学者而言，使用此种方法存储ADC数据还可能犯下将数组声明在主函数中的错误（当然，将存储数据的数组在主函数外部声明为全局变量完全可以有相同的效果，但仍然会有不少同学按照C语言中的编程习惯，将数组在主函数中声明），而这将造成主程序运行时堆栈溢出，引起程序的崩溃（受单片机内存的限制，主函数中使用的数组不允许过大，而对全局变量，则无此限制）。因此本实验采取另一种方案，利用中断的程序设计方法，调用ADC采样函数。该方案通过调节时钟频率、触发频率和软件计数的方式精准控制ADC采样频率，并且ADC采样全程进行，不阻塞主函数运行，保证心电信号的所有信息尽可能完全地被保存下来。对于软件手段传输和储存ADC采样的信息，同样还有一个需要考虑的方面，即存储数据的数据结构，将在下文进行讨论。

1. 储存ADC采样数据的数据结构：

根据基本的逻辑抽象，不难想到最适合本实验数据存储的结构为“队列”结构。采用队列结构，采集到的心电信号将源源不断地按照FIFO(First-in-First-out)的方式输入到内存中，所有的信息都被保留下来，符号小组成员的设计理念。

（图2.1.4 “队列”结构的示意图）

尽管设想很好，这样的数据结构在实际操作时存在巨大问题。对于一般队列，有静态数组和链表两种方式构建。对于数组队列，在输入新的数据时需要将原数组中的每一个元素都向前移位一次，这对于性能有限的单片机是不可取的，对于数据采集设置在中断程序中的采样方式更是灾难性的（中断程序不允许有过于复杂的代码）；而对于链表队列，构造链表所需的成倍空间对于内存有限的单片机同样不可取，并且使用链表对代码的编写也造成了不便。因此，此处考虑使用一种静态数组构成的“循环队列”。该队列有一指针“adc\_count”用于指示队列的队头与队尾，而不是以固定的数组第一位和最后一位作为队头队尾，同时，该队列的“循环”特性亦是通过adc\_count在每次新数据进入时+1后还要对数组元素个数（adc\_num）取模实现的，最终实现的效果是指针adc\_count指示的是当前队列里最早进入的元素的位置，当一个新元素输入时，覆盖在adc\_count所指示的位置，同时adc\_count加1，此即完成了入队和出队的操作，而序列adc\_count到(adc\_count-1)%adc\_num即为一段时间从前往后的心电信号。如此设计既节省了时间又节省了空间，又达成了目标。

（图2.1.5 “循环队列”结构的示意图）

但这样的设计又带来了其他问题，在处理获得的数组时，若简单选取第一个元素作为起始，最后一个元素作为终末，不仅违背了时序逻辑，在adc\_count处还将发生数据的断层（考虑到adc\_count和其前一个元素一个为最早采集的数据，一个为最晚采集的数据，有巨大的时间差）；若以adc\_count为起始，(adc\_count-1)%adc\_num为终末，则在调用数组时还要面临数组下标取模的问题，带来了巨大的不便，为了处理这一问题，小组成员给出了巧妙的方法，将在下文讨论画面刷新的问题时做出进一步讨论。

1. 采样率与采样深度：

本实验中采样率由ADC单次采样触发的频率决定，在下文中将讨论其具体计算；实验中采样深度由用于存储ADC采样数据的数组大小决定，按照上文描述，用adc\_num这一变量表示。我们先讨论采样深度，对于画图而言，单片机屏幕的分辨率为240\*240，实际画图区域大概为200\*200，意味着adc\_num最大不超过200，否则多余的点将无法在屏幕上绘制，一定程度上浪费了单片机的资源；接着讨论采样深度，本实验所测频率为1-2Hz的心电信号，为准确测量信号的频率，考虑一种极端情况，数组存储的信息也应至少包含一个完整周期的波形，而采集心电信号一个完整周期所需要的时间为1-0.5s，为了稳妥起见，选择采样深度为1.5s，即最坏情况下单片机也能采集到1.5个周期的波形。综合以上讨论，我们选择200大小的数组，采样深度为1.5s的波形，由此可以计算出我们的采样频率为200/1.5=133.33Hz。值得注意的是，以上讨论只限于有限性能下的最优解，即既不超出单片机画图所能接受的最大点数，又能满足数据处理计算频率所需的最小点数；但事实上我们知道画图和数据处理完全是两件事情，在性能充裕条件下，采样率和采样深度也自然是越高越好；本实验设计之初采用133.33Hz的采样率和200的采样深度，已经基本可以完成所有的任务，但小组成员仍对得到的心率的精度和程序在处理频率在1-2Hz之外的信号的表现感到不满，将在“四、性能优化”继续对这一点进行讨论。

1. 屏幕驱动和画面刷新的设计：

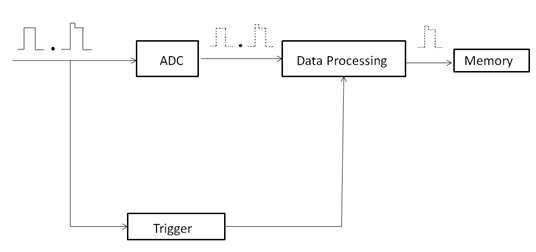
单片机的LCD屏幕由SPI总线驱动。SPI，是一种高速的，全双工，同步的通信总线，并且在芯片的管脚上只占用四根线，节约了芯片的管脚，同时为PCB的布局上节省空间，提供方便。

（图2.1.6 “SPI”通信总线的示意图）

图中SS( Slave Select)表示从设备选择信号线，常称为片选信号线，也称为NSS、CS；SCK (Serial Clock)表示时钟信号线，用于通讯数据同步；MOSI (Master Output，Slave Input)表示主设备输出/从设备输入引脚；MISO(Master Input,，Slave Output)表示主设备输入/从设备输出引脚。考虑到重新构建lcd屏幕的驱动程序较为复杂，本实验直接使用例程“LcdDisplay\_SPI”中的屏幕驱动文件以及其中封装好的划线、打印等绘图函数。

下面考虑画面刷新的问题。在单片机屏幕中需要实时显示的是心电信号的波形和相关的参数。对于需要显示的参数，由于只在固定位置，且封装好的LCD\_ShowString/LCD\_ShowIntNum/LCD\_ShowFloatNum1函数在绘制文字之前会预先设定“背景色”，该背景将能够覆盖掉上次打印的文字，因此无需对画面刷新做额外的考虑。下面重点讨论波形的实时显示。波形的绘制由LCD\_DrawLine函数完成，考虑到波形绘制的位置是不定的，每次绘图并不会自动擦除上一次绘制的图像。这里小组成员一开始打算借鉴在上学期数据结构大作业中使用easyx在屏幕上制作游戏的经验，在每一次画面刷新之前，实行“清屏操作”，即调用LCD\_Fill对整个屏幕刷上背景色，然后立即进行下一次图形的绘制。这种方法在easyx中被证明卓有成效，考虑到电脑CPU处理程序的速度很快，而屏幕的刷新率有限，只要在极短的时间内（快于屏幕的刷新率）完成清屏并立即绘制新图像，就不会有空白帧出现，整个图像将是连续的，不会出现闪烁现象，有较好的观感。但实践证明在单片机该方法并不有效，屏幕往往会有大量的空白时间，出现严重的闪烁现象。小组成员讨论研究认为，这与单片机屏幕刷新的方式有关。单片机的屏幕并不能实现“按帧显示”，即按照一定时间间隔对整个屏幕的内容进行更新，而是只能操纵一个个像素点，按照指令逐个赋上需要的颜色。考虑到单片机性能有限，逐个像素点刷新的方式使得“整体清屏”操作的效率尤其低下，屏幕刷新的延时不可忽略，将出现大量的不完整帧，当LCD屏幕无法及时响应而跳过部分指令时，还会引起画面的混乱。因此应该在尽可能改变较少像素的前提下，实现心电信号波形的更新。小组成员采用的是“画白线”的方法，即保留上一次画线的数组数据，在下一次画线之前，使用上一次的画线数据再次绘制一条同样形状但颜色为背景色的波形，从而达到“擦除”上次图像的效果。但该方案同样带来了新的问题，需要消耗额外的内存储存上一次的画线数据，因此我们希望这个数组可以有更多的用途，最大化空间的利用率。这里我们回到在“（1）储存ADC采样数据的数据结构”中所讨论的有关循环队列对数据读取带来不便，我们构建一个大小为为adc\_num-10的数组adc\_data\_copy，它在每轮while循环中从adc采样直接存储的数组adc\_data中“抄写”下数据，这样做有两个好处——第一个好处是抄写时可以adc\_count所指示元素作为副本的第一个元素，以(adc\_count-1)%adc\_num所指示元素作为副本的最后一个元素，这样在副本中第一个元素即为时间上最早进入的数据，最后一个元素即为时间上最晚进入的数据，这为以后的数据处理提供了极大的便利；第二个好处是副本中的数据在每轮循环抄写完成后一直到下一次抄写之前，都不会发生变化，这就避免了主循环在调用adc采样数组adc\_data的同时，adc还在一边通过中断更新该数组，从而造成数据上的混乱（事实上在副本的抄写过程中也考虑到了这一点，抄写数组实际上是从原数组的(adc\_count+5)%adc位置开始，只抄写adc\_num-10个元素，从而避免在抄写过程中原数组因为adc采样更新，副本抄写到不符合时序的数据）。回到此处画白线的问题上来，副本数组还可以起到另一个作用，它可以为上一次画线数据提供备份。只要在副本数组重新从原数组中抄写数据前，利用该数组画出与上一轮循环所绘制形状一样的白线，即可擦除上一次的图像，在此之后副本继续抄写，然后绘制下一帧的图像，依次类推。综上所述，小组成员利用一个副本数组，巧妙地同时解决了程序设计时面临的两个重大问题。需要补充的是，单片机更新LCD像素点的速度实在有限，即使采用这种方法，依然会有一定程度的闪烁，要想进一步消除闪烁，可以考虑擦去一个点就立即画上新的点，而不是擦去一条200个像素点的线再画上一条200个像素点的线，但这样势必要再增加一个数组辅助存储或大大增加程序的复杂度，故此处不予考虑。

1. 触发的设计：

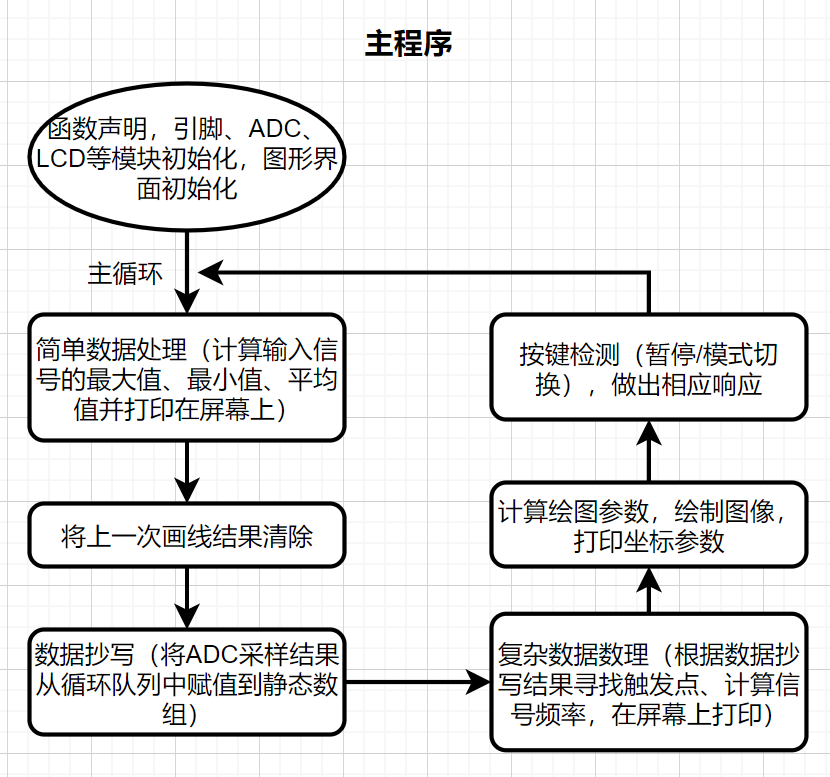
触发(trigger)在不同的背景下往往具有不同的含义，但通俗的来说，“触发”就是隔离出感兴趣的信号。在示波器中触发过程可以有如下描述：

（图2.1.7 示波器触发原理）

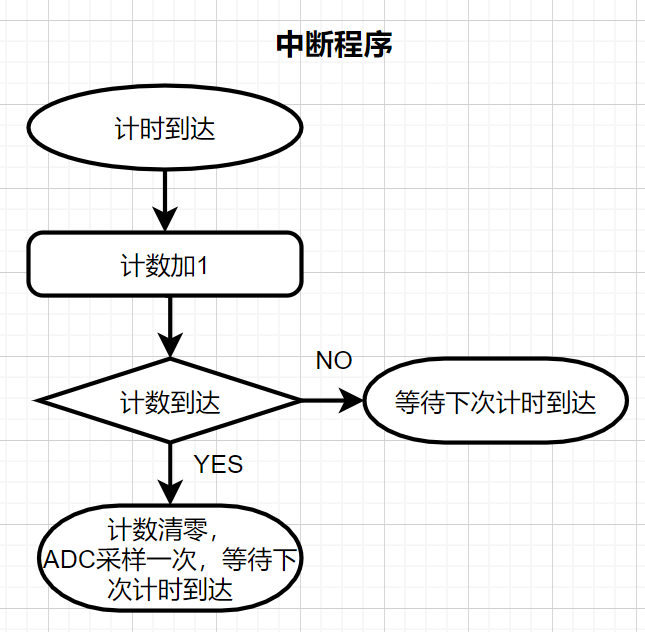
如上图所示，信号从外部进入示波器时会分为两路。第一路信号进入ADC采样量化，然后不断存储到缓存；另外一路信号进入触发器，当这个信号满足触发条件的时候，触发电路会控制示波器继续采样一段时间后停止。在内存中有一部分数据是包括触发点前后一段时间的数据，这个区别于传统的模拟示波器先触发再采样的原理。触发电路将会控制示波器停止采集和在显示。

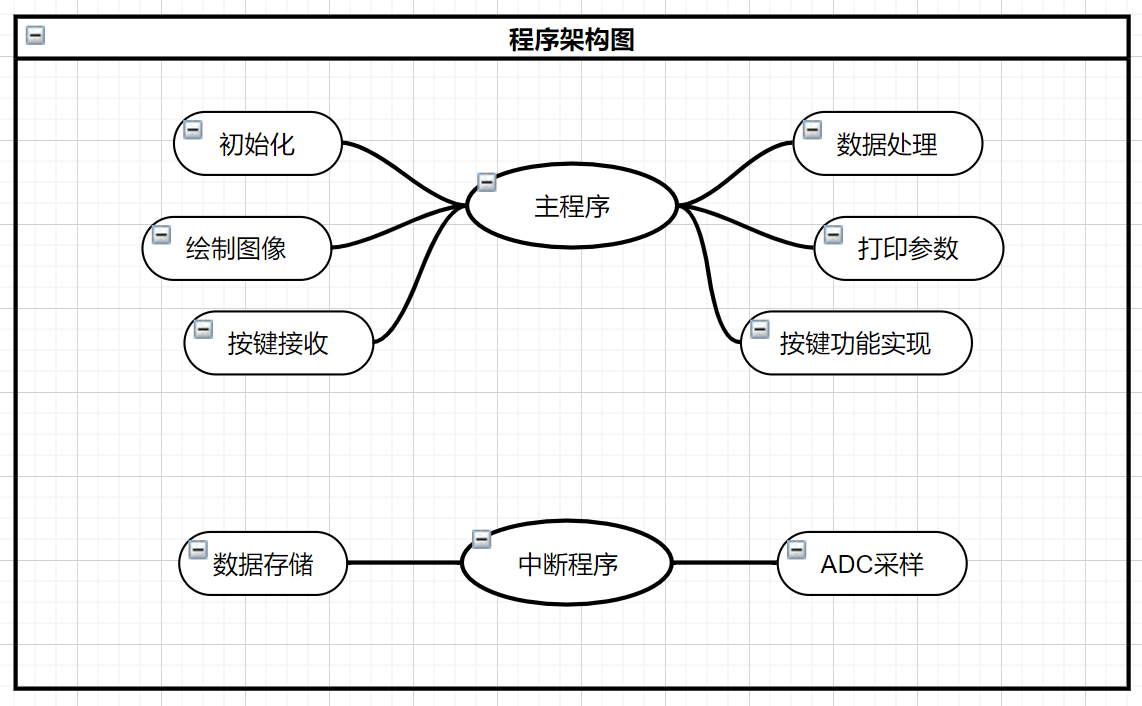
本实验中所要制作的心电信号采集器实际就是一种简易的示波器，但因为缺乏必要的外设，我们并不像示波器从硬件层面设计触发，而是从软件的层面进行设计。我们将“触发”理解为寻找波形的特征点，将触发的过程在“内存”中进行（也就是基于adc采样的数组）。对于寻找到的触发点，我们可以用来计算周期（两个相邻周期的触发点之间的时间间隔即为一个周期），也可以用来确定绘制心电波形的起始点，从而得到稳定不动的心电波形图。因此选择一种稳定的可以分隔开两个相邻波形的触发方式就非常重要。对于心电信号，一种很粗糙的方式是电平触发，即将信号达到某一幅值时作为一个触发点，然后延时一段时间再做同样检测，得到下一周期的触发点，这样的触发方式对触发电平和延时的参数选择有很高的要求，且对于不同的输入信号往往需要设置不同的参数才能正确触发，非常的低效；小组成员采用的是一种上升沿触发的方式，并用下降沿触发分隔相邻两周期的上升沿触发，具体表述为选择一个起始点开始遍历数组，寻找满足a）值比前一个数大，且比后一个数小b）值大于某一指定值（这一值并不一定是一固定值，而是由采样信号平均值、最大值表示的一个值）的数组下标，该下标即对应一个上升沿触发，以类似的方法定理下降沿触发。则整个过程可以表述为，从零处开始寻找到第一个上升沿触发点，然后从此处开始寻找一个下降沿触发点（一个点不可能即能上升沿触发，又能下降沿触发），从找到的下降沿触发点开始寻找第二个上升沿触发点，记得到两个相邻周期的触发点，两点之间即为一个完整的心电波形信号。

1. 实验方案的流程图与架构图：

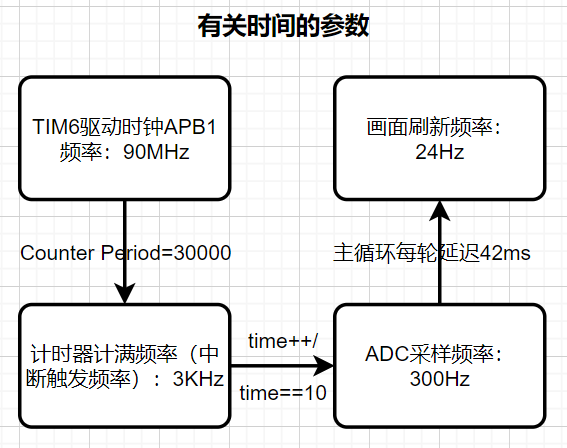
本实验的程序部分主要由主程序（主函数）和中断程序两部分构成，具体流程图和架构图如下：

（图2.2.1 主程序流程图）

（图2.2.2 中断程序流程图）



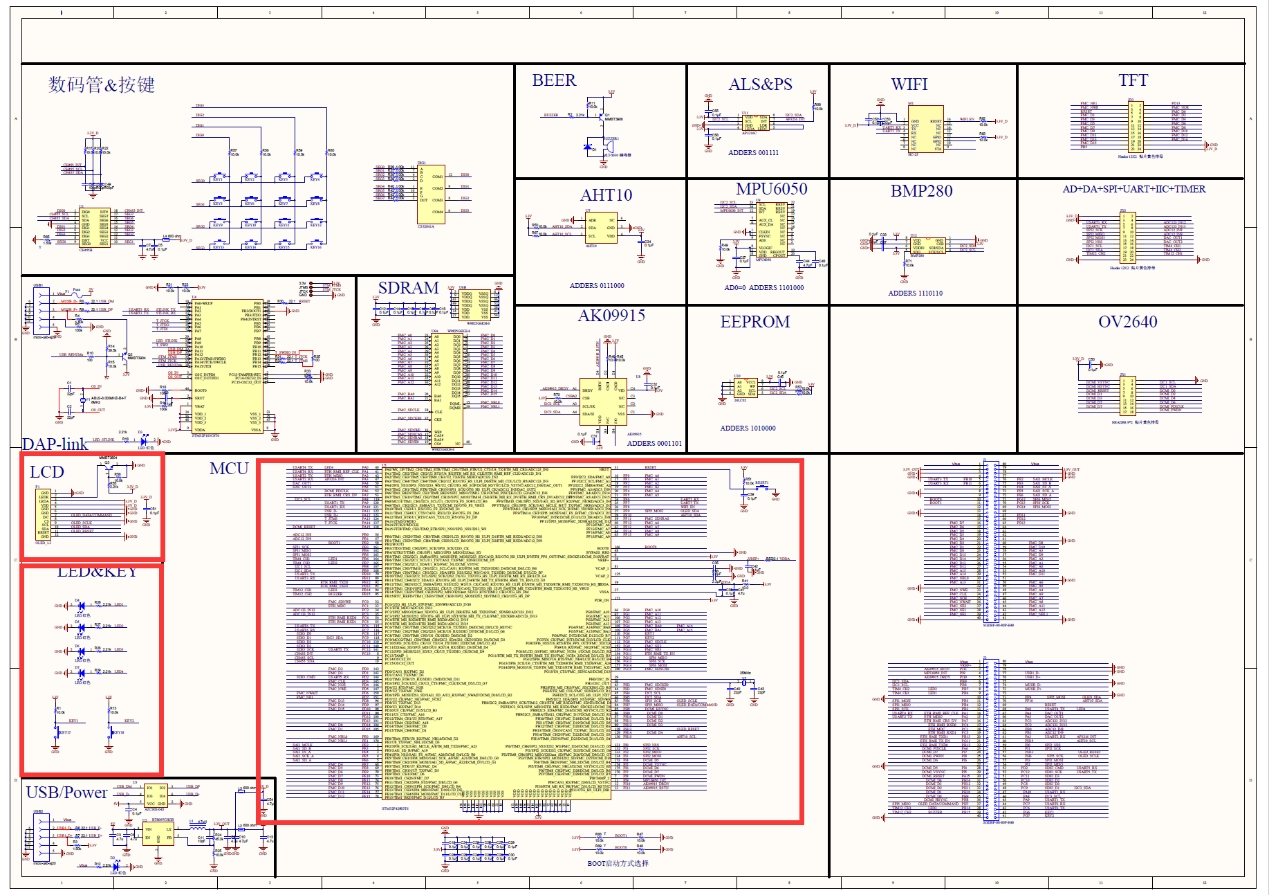
（图2.2.3 实验方案的架构图）

1. 参数计算方案与性能指标评估：
   * 1. 本实验在程序设计方面主要涉及到计算的参数有ADC采样率（采样间隔）、采样深度、画图频率（画面刷新率）、横纵坐标轴（Vertical/Horizontal）坐标数值。
2. ADC采样设置在中断程序中，采样频率由时钟频率、计时器计数周期、程序计数周期共同决定。时钟频率和计时器计数周期可以在CubeMX中进行配置。本实验中所采用通用计时器TIM6由时钟APB1 Timer clocks驱动，其频率为90MHz，对于计数器TIM6，设置其计数周期为30000，即中断程序执行的频率为90MHz/30000=3KHz。另外在每轮中断程序中并不直接对ADC进行采样，而是先执行计数，当计数满10次时，执行一次采样。则采样频率为3KHz/10=300Hz，采样间隔为1/300=3.33ms。具体过程如下图所示：

（图2.3.1 有关时间的参数设定图）

1. ADC采样的深度由用于采样数据存储的数组决定。在程序中该数组用float adc\_data[adc\_count]定义，adc\_count为999，则程序最多存储999/300=3.3s的波形信息，对于1-2Hz的心电波形信号，程序至少可以储存3个周期的完整波形
2. 画图频率由相邻两次画图的间隔决定。画图程序只在主函数主循环中执行，且每轮循环仅执行一次。在一般模式下主循环延时为42ms，则近似认为画图频率为1000/42=24Hz；在固定触发模式下该延时延长到210ms，则近似认为画图频率为5Hz。
3. 画图区域设置为200\*200的范围，在两种模式下允许的最大输入电压为1.65V和3.3V，则每个像素点对应的电压值为0.00825V和0.0165V，将垂直坐标轴分为5\*40总计五段，刻度处对应数值为0.66,1.32,1.97,2.64和0.33,0.66,0.98,1.32，单位为V。在两种模式下都选择100个数据点进行绘图，但前者为数组中的连续点，两点间间隔时间为3.33ms，后者为数组中间隔点，两点间间隔时间为6.66ms，则每个像素点之间时间间隔为1.665ms和3.33ms，同样将水平坐标轴分为5\*40总计五段，刻度处对应数值为66,132,198,264和132,264,396,528，单位为ms。
   * 1. 性能指标评估：
4. 根据第一阶段设计的调查资料，心电信号的主要能量集中在0.05-100Hz，则根据奈奎斯特采样定理，单片机采样频率不应该低于200Hz，本方案设计的采样频率为300Hz，足以保留绝大部分心电信息。考虑到测试信号为1-2Hz的心电信号并非一般人体的心电信号，故还要考察在本方案300Hz采样率下单片机能否较好采集较为陡峭的上升沿信息（该处包含丰富的高频分量）：根据观察，心电信号上升沿持续时间约为0.05个周期，则在1-2Hz频率的心电信号输入下，上升沿持续最短时间为25ms，而设置的采样间隔为3.33ms，在整个上升沿期间至少采集到7个点，可以精准地计算出信号的频率和画出圆滑的图像（注意到计算频率时最多有一个点即±3.33ms的误差，以1-2Hz的心电信号为例，测量误差在0.3%-0.6%，具有相当高的精度）。
5. ADC采样深度为3.3s，对于1-2Hz的心电信号，至少可以采集3个以上完整波形，足以用于频率测量与图像绘制。
6. 如前所述，本方案一个重要设计理念是可以观察到信号从输入开始所有时刻的波形，最大可能保留心电信号的全部信息，则希望采集到的每个数据点都至少能在一帧画面中出现，这需要较高的屏幕刷新率。考虑一种极端情况，在屏幕刷新间隔为42ms的情况下，相邻两帧的间隔中ADC重新采集了42/3.33=12.6个数据点，在两种模式下1个数据点占据2个或1个像素点，则对应图像在每一帧中最多向前平移13\*2=26个像素点，而绘图区域总共有200个像素点，即每个数据点都至少在7帧中出现，符合我们的设计理念。
7. **系统设计**

说明：站在初学者的角度，以及为了方便读者阅读本报告的“二、实验方案”部分，系统设计中所涉及到的许多模块的原理以及功能在“二、实验方案”中已作出介绍，在本部分将不再赘述。

STM32F429的引脚图和单片机各模块的引脚图如下图所示。本实验涉及到的模块主要有LCD模块、LED&KEY模块、MCU（微控制单元）模块。

（图3.0.1 STM32F429示意图）

1. MCU模块：

微控制单元(Microcontroller Unit; MCU)，又称单片微型计算机(Single Chip Microcomputer )或者单片机，是把中央处理器(Central Process Unit；CPU)的频率与规格做适当缩减，并将内存(memory)、计数器(Timer)、USB、A/D转换、UART、PLC、DMA等周边接口，甚至LCD驱动电路都整合在单一芯片上，形成芯片级的计算机，为不同的应用场合做不同组合控制。

该部分的配置和设计主要在CubeMX上完成，是对芯片引脚的配置，使单片机能从外部接收信息、处理信息，并驱动外设完成相关的指令。与该部分密切相关的模块有GPIO 配置、ADC采样、程序中断（时钟配置）。

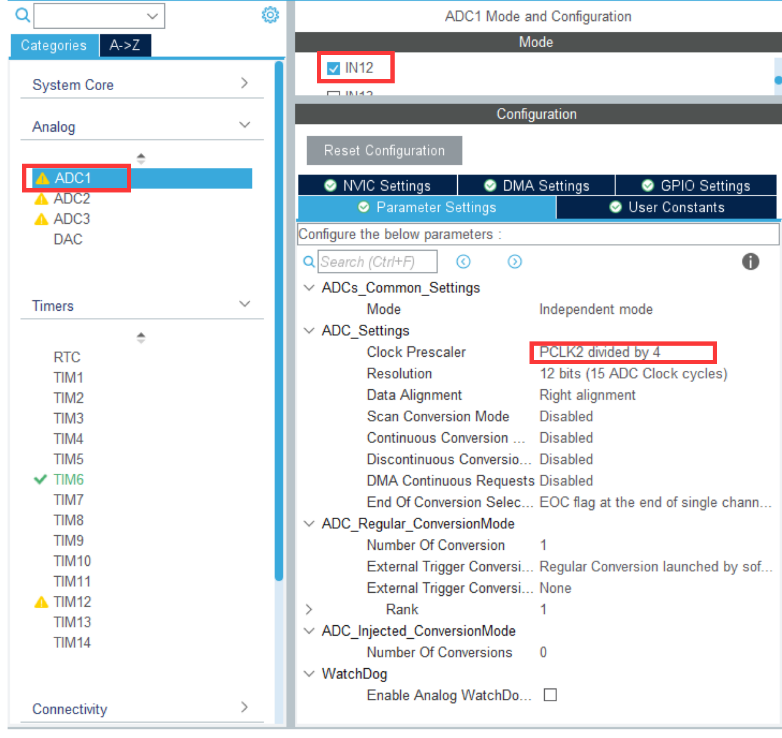
1. GPIO配置：

通用输入输出端口(GPIO)，是STM32 可控制的引脚，STM32 芯片的GPIO 引脚与外部设备连接起来，从而实现与外部通讯、控制以及数据采集的功能。STM32 芯片的GPIO被分成很多组，每组有16 个引脚，所有的GPIO引脚都有基本的输入输出功能。STM32的外设资源丰富，但引脚数量有限，因此不同的外设会共用同一根引脚。引脚一般默认是GPIO管口，而若想采用其作为某个外设引脚接口，需要配置引脚复用功能，设定该引脚为外设服务。

方案中所用于GPIO\_Output的引脚有：PB7（用于驱动LED2，作为指示灯）、PB6（用于驱动LED2，作用同上）、PB14（用于驱动LED3，作用同上）、PA0（用于驱动LED4，也可以接USART串口用于单片机调试）、PH13（接OLED RESET，用于驱动LCD显示屏）、PH7（接OLED DATA/COMMAND，作用同上）。

方案中所用于GPIO\_Input的引脚有：PG6（接KEY1，接收按键信息）、PG7（接KEY2，作用同上）。

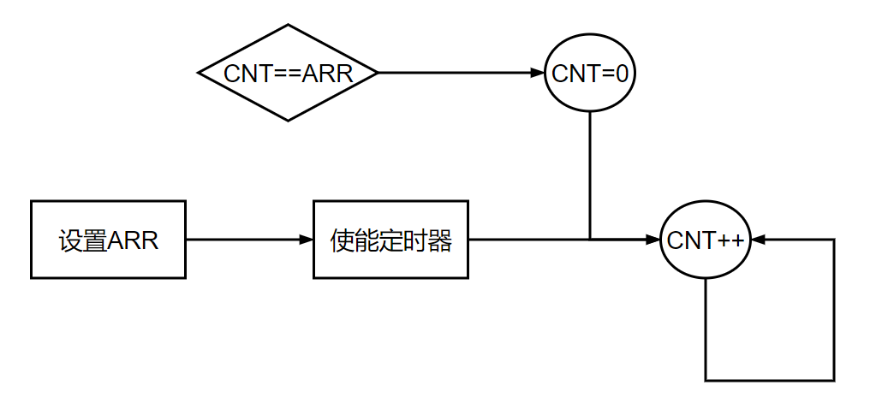
1. ADC采样：

在“二、 1. （1）”部分已对ADC的概念做出解释，此处不再赘述。在本实验中ADC采样选择的通道是ADC1 IN12，其中ADC1由总线PCLK2接时钟APB2驱动，ADC1时钟频率为APB2（频率为90MHz）四分频后的频率22.5MHz。在CubeMX对ADC1的其余配置如下图所示：

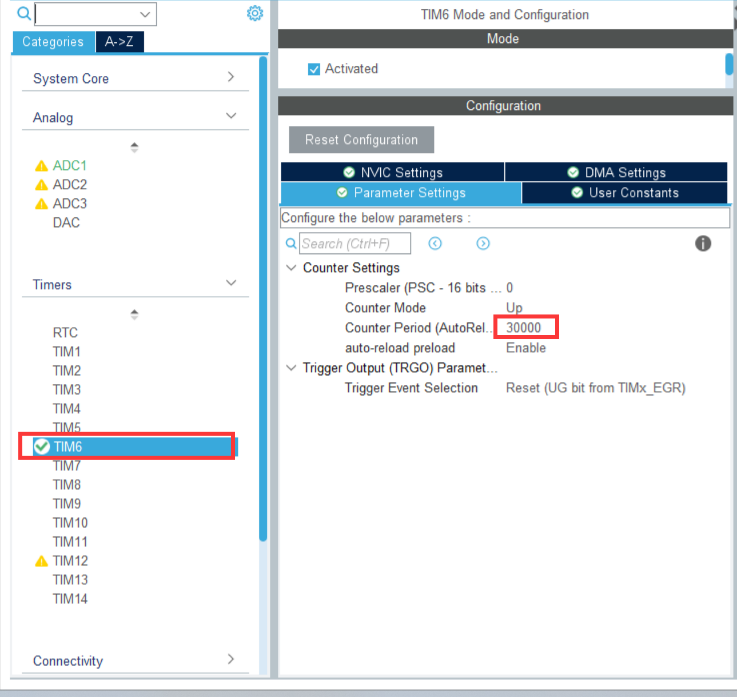
（图3.1.1 ADC配置图）

1. 计时器与时钟配置：

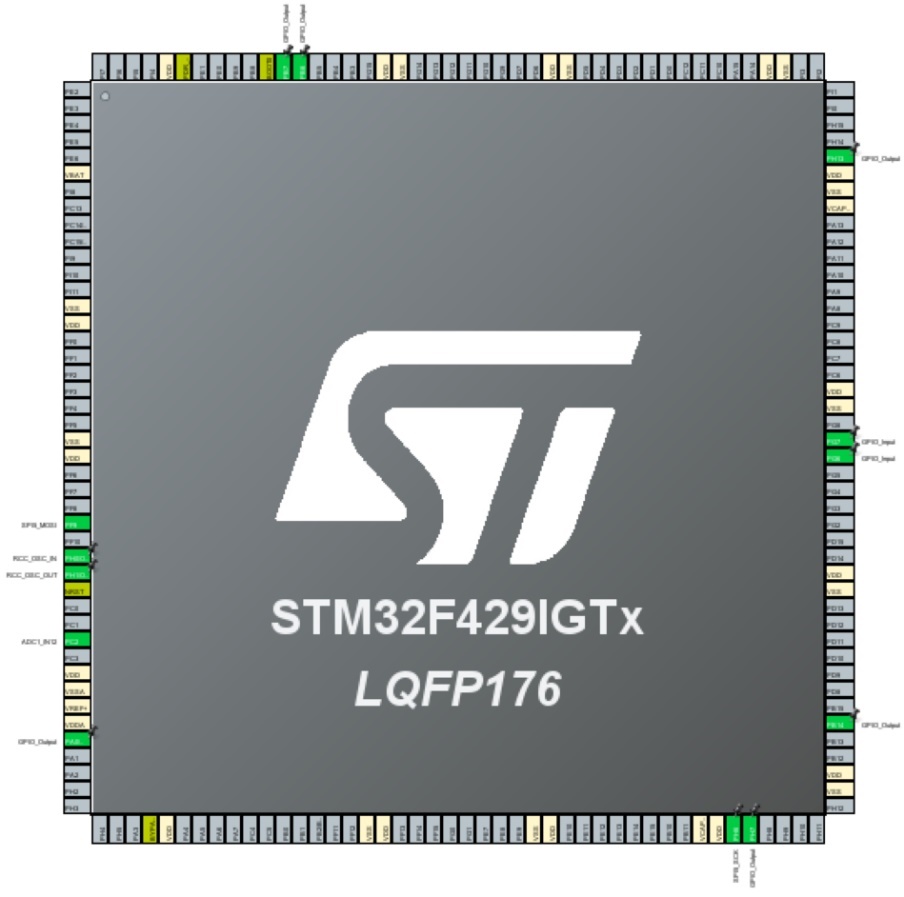
定时器(Timer)的基本功能为计数、定时。比如定时发送USART数据，定时采集AD口。STM32F429具有丰富的定时器资源，有TIME1和 TIME8等高级定时器，也有TIME2~TIME5，TIM9~TIM14等通用定时器，还有TIME6和TIME7等基本定时器。本方案中使用的是基本计时器TIM6，它是一个16 位向上递增的定时器，当在自动重载寄存器(ARR)设置一个数值并使能定时器TIM，计数寄存器(CNT)就从0开始递增，当CNT数值与ARR数值相等就产生一个事件并把CNT清零，完成一个循环过程。如果没有停止定时器就循环执行上述过程。

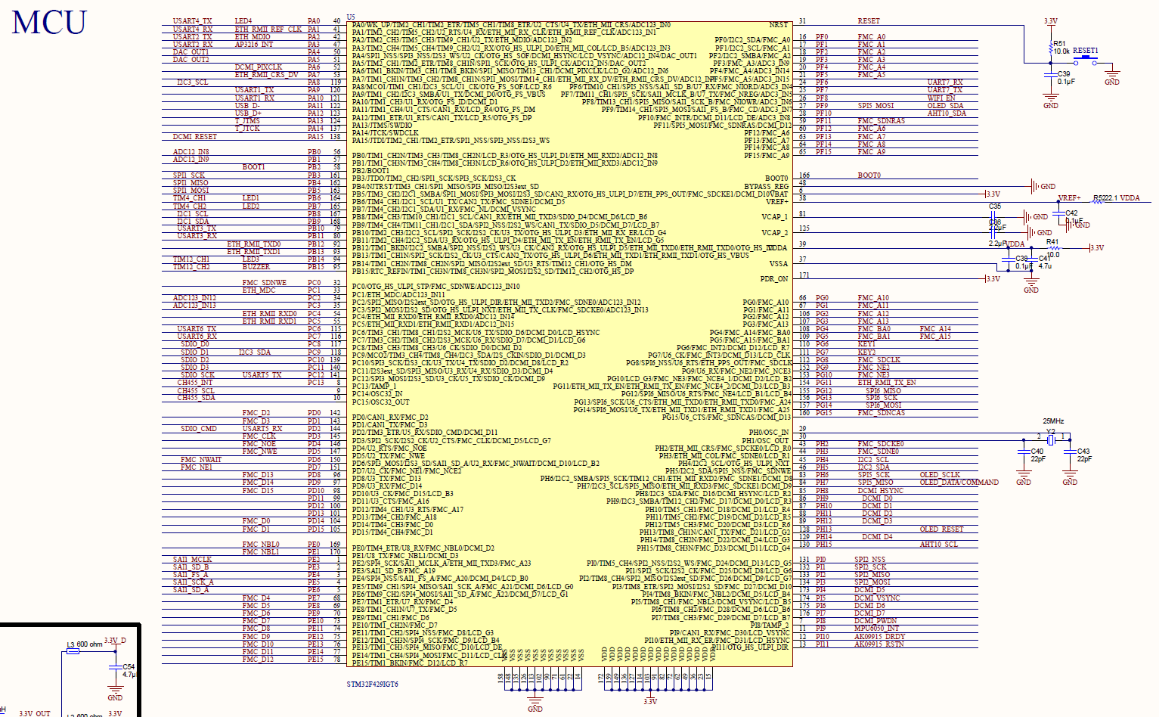
中断的概念已在“二、 1. （1）”中做出介绍，此处不再赘述。

（图3.1.2 计时器工作流程图）

计时器在CubeMX中配置如下所示，它由时钟APB1（频率为90MHz）驱动。有关计数器计数频率和中断程序触发频率的介绍已在“二、 3. （1） A）”作出说明，此处不再赘述。

（图3.1.3 计时器配置图）

下面给出Cube上对MCU的整体配置图并辅以STM32F429的MCU部分的引脚图作为参照，读者可自行查阅或查看“附件”。

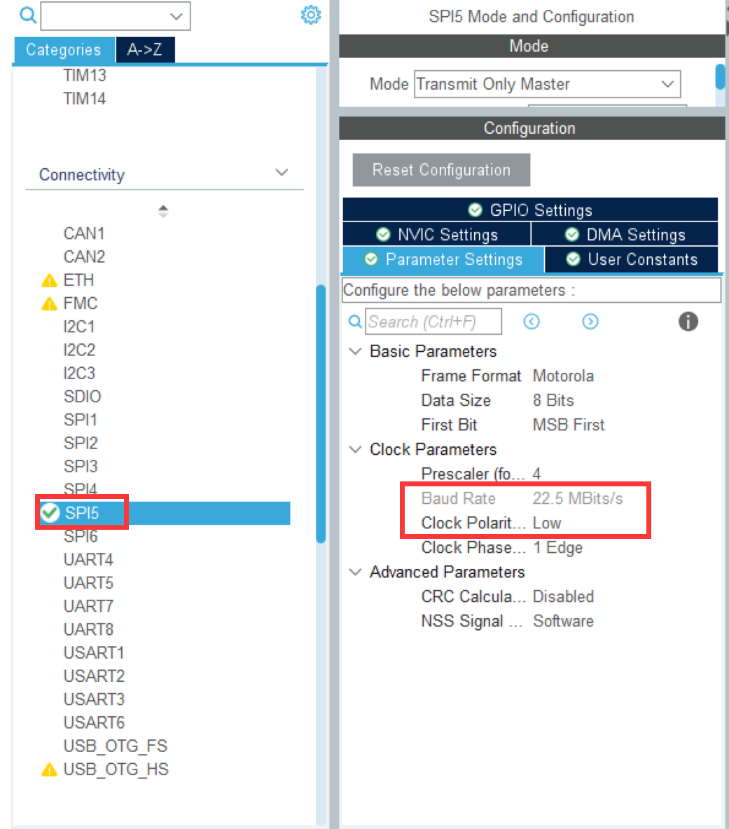
（图3.1.4 CubeMX上MCU配置图）

（图3.1.5 STM32F429的MCU部分的引脚图）

1. LCD模块：

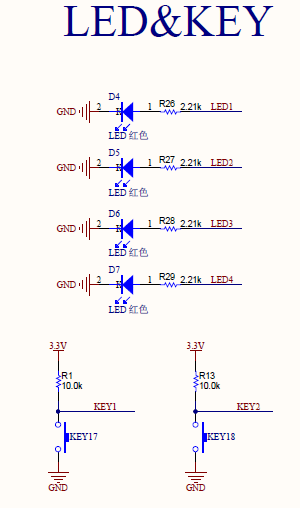
LCD模块的介绍已在“二、 1. （4）”中做出说明，此处不再赘述。

LCD模块与MCU模块之间存在数据传输，传输协议为SPI协议，有关该协议的介绍亦在“二、 1. （4）”中提及。

在CubeMX中的有关配置如下图所示：

（图3.2.1 CubeMX中SPI5的配置图）

1. LED&KEY模块：

该模块结构、功能较为简单，不做过多叙述。有关该模块的配置主要在“MCU模块”的配置中完成，可以参考本部分“1. （1）GPIO配置”。

（图3.3.1 STM32F429中LED&KEY模块的引脚图）

1. **性能优化**

说明：本方案做出了大量的前期准备工作，包括参考资料的收集和对其他同学完成后反馈的调研，因此在程序设计时很多指标已经设置为最优，这使得后期性能优化的空间相对较少，但程序性能已经较为不错。

1. 采样率和采样深度的提升：

在程序设计之初考虑到单片的性能有限，因此在采样频率和数组大小设置方面较为保守，按照其他同学的建议，仅仅将采样率设置为133.33Hz，采样深度也仅有200个数据点。程序整体表现让人满意，但无论是对较低频率的信号，还是对较高频率的信号，在频率测量时都有巨大误差或根本不能检测到触发点。分析原因认为是在输入信号频率较低时，储存深度有限，不能同时储存到相邻两周期的上升沿，故完全无法计算频率；而在输入信号频率较高时，采样频率有限，对于高频分量丰富的上升沿部分，采集的点过于稀疏，跳过了当前上升沿，直到下一个上升沿才被触发，导致频率测量存在巨大误差。

基于以上情况，并考虑到暑期课程曾用单片机写过一个可以测量几KHZ正弦信号的程序，小组认为单片机的性能仍有大量富余，因此将采样频率提高到300Hz，采样深度设置为999个点，根据“二、 3. （2）”的讨论，此时程序对心电信号测量有相当优异的表现。

另外在输入开路或断路时，单片机依然能采集到一些噪声信号并计算出无意义的频率，为了避免误导，程序还设计了“低输入提醒”，在屏幕上打印“Low\_Input”以示当前信号无效。

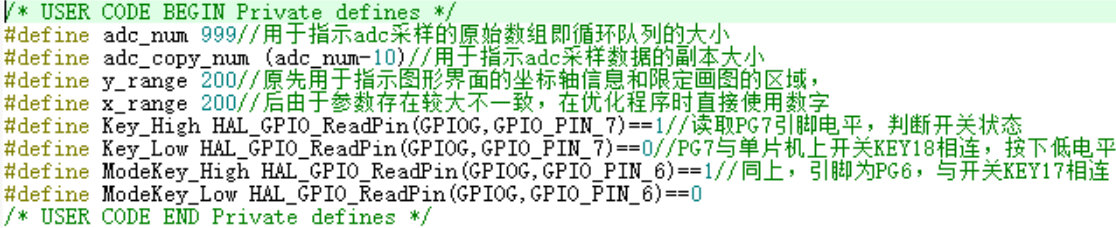
1. 绘图延时与绘图错误的改善：

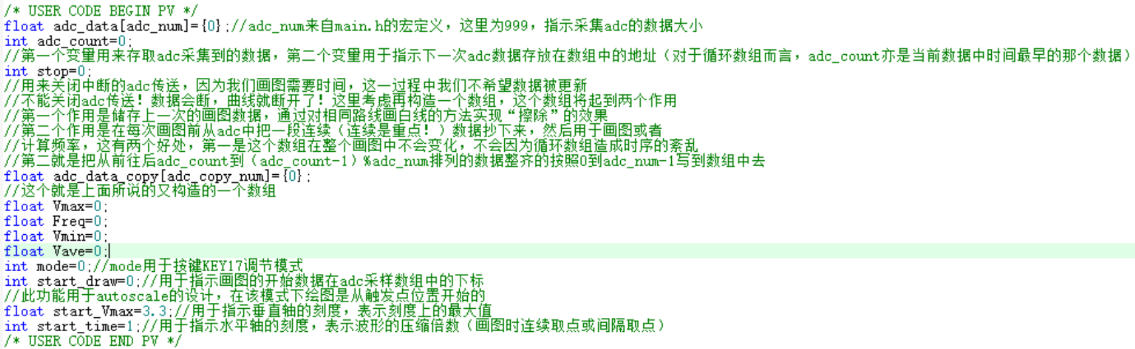
承“1.”，在提高采样深度后又出现新的问题，考虑到每次画图最多只使用200个点，而画图是从时间上由前往后进行，因此在采样深度为999个点3.3s的情况下，LCD屏幕显示的只是0-0.66s的波形，这意味着最新输入的信号至少要经过2.64s后才能在屏幕上有所体现，这表明程序的响应速度很慢，但要改善该问题，只需要将每次开始绘图的点选择在采样数组中较靠后的元素即可，最后确保LCD屏幕显示的是2.64-3.3s的波形（注意不要设置的过于靠后，避免数组访问越界）。

另外发现在“autoscale”设置自动触发的模式下，绘图会出现“擦不干净”的情况，认为是SPI传输数据速度过慢，低于数据产生速度，部分指令未被执行即被跳过，为降低单片机在该情况（有较多计算任务）下的负荷，调低屏幕刷新的频率，增加画图的时间，最终改善了绘图错误的情况。

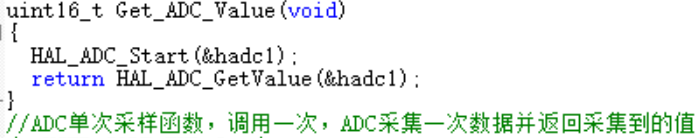
1. **关键代码**

说明：本部分给出程序实现的关键代码。为了赋予代码一个良好的实现风格，程序中使用了一定数量的宏定义和变量，因此在给出关键代码前，先对这些私有变量和宏定义做出说明以便于读者阅读后面的关键代码；另外本部分的文字说明较少，必要的解释已经在代码中以注释的形式给出。

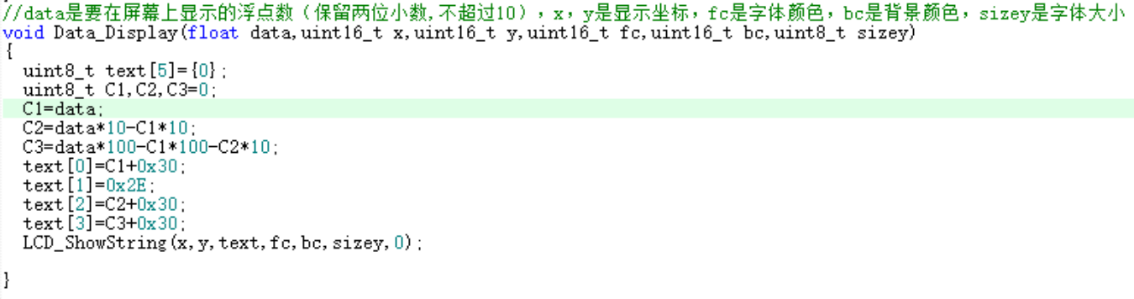
1. 宏定义与变量设置：

（图5.1.1 程序中用到的私有宏定义）

（图5.1.2 程序中用到的私有变量（都以全局变量形式给出））

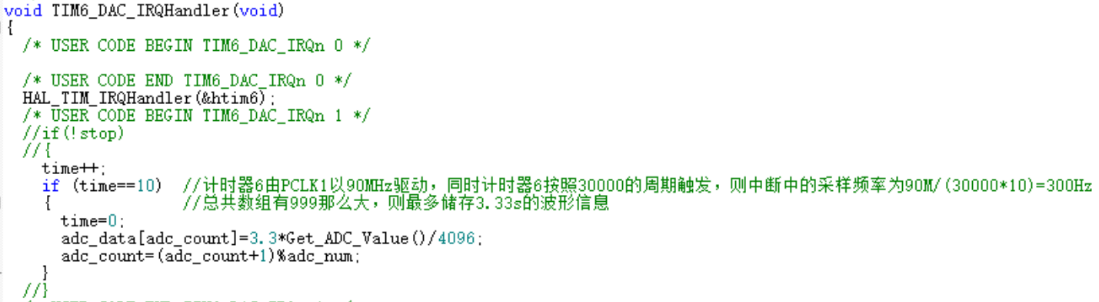
1. 自定义函数：

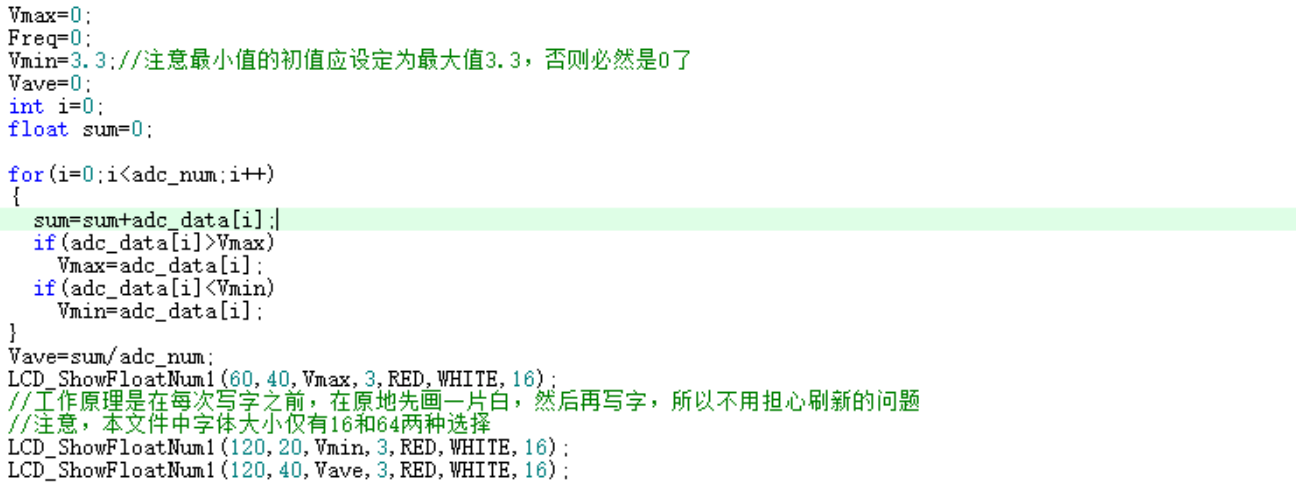
（图5.2.1 ADC单次采样函数）

（图5.2.2 触发点寻找函数）

（图5.2.3 打印浮点数函数）

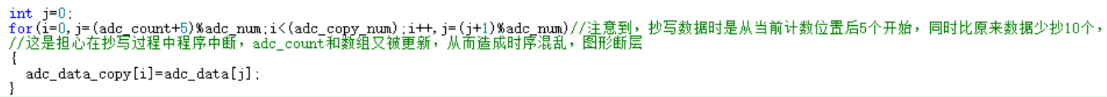
（图5.2.4 绘制坐标轴函数）

1. 中断程序（ADC采样与存储）：

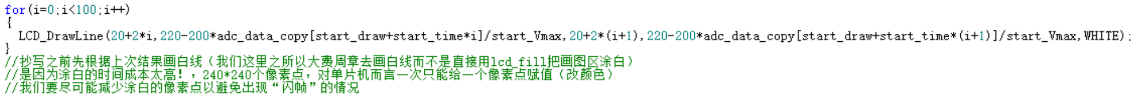
（图5.3.1 中断程序与ADC采样函数）

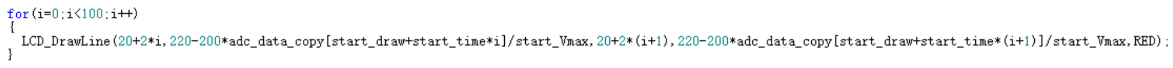
1. 简单数据处理：

（图5.4.1 简单数据处理，计算最大值最小值与平均值并打印在屏幕上）

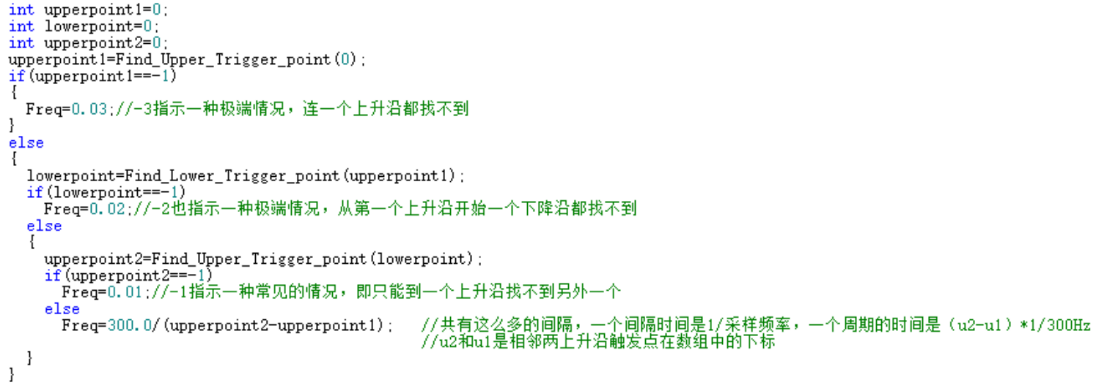
1. 数据抄写：

（图5.5.1 ADC采样数据副本的建立）

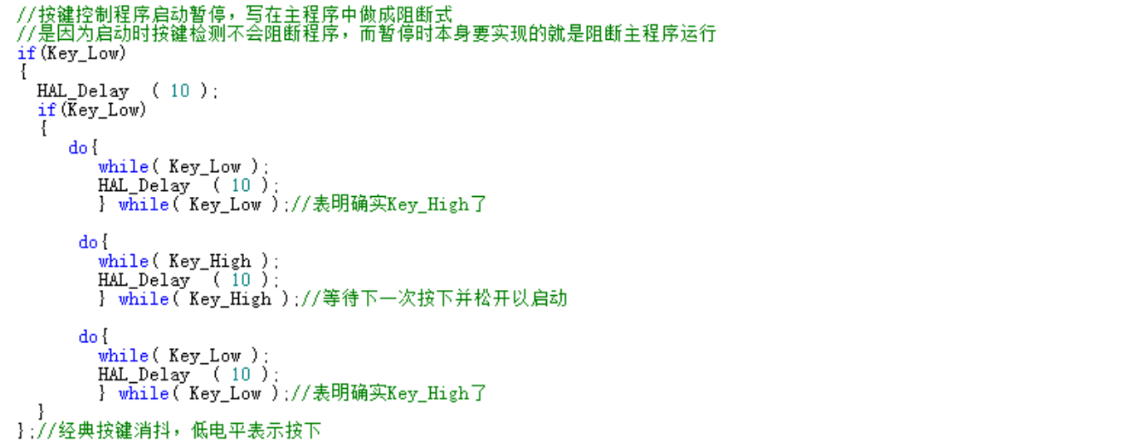
1. 图像的擦除和绘制：

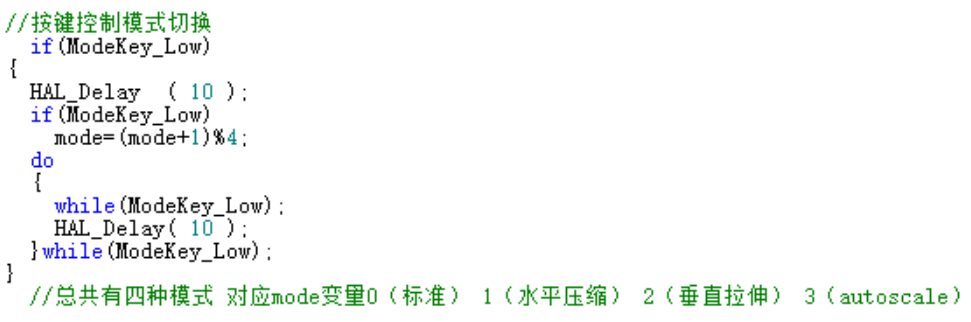
（图5.6.1 擦除上次画线结果）

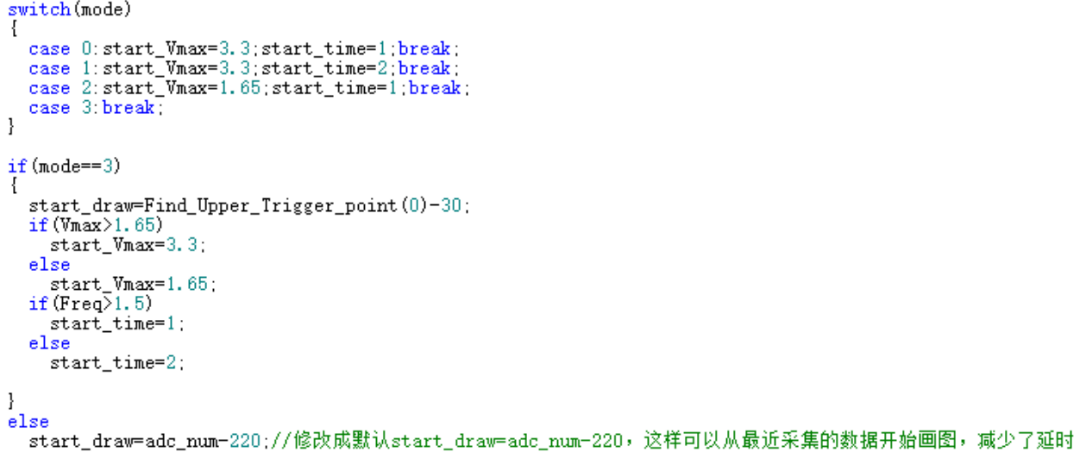
（图5.6.2 重新绘制图像）

1. 复杂数据处理：

（图5.7.1 利用触发点求输入信号的频率）

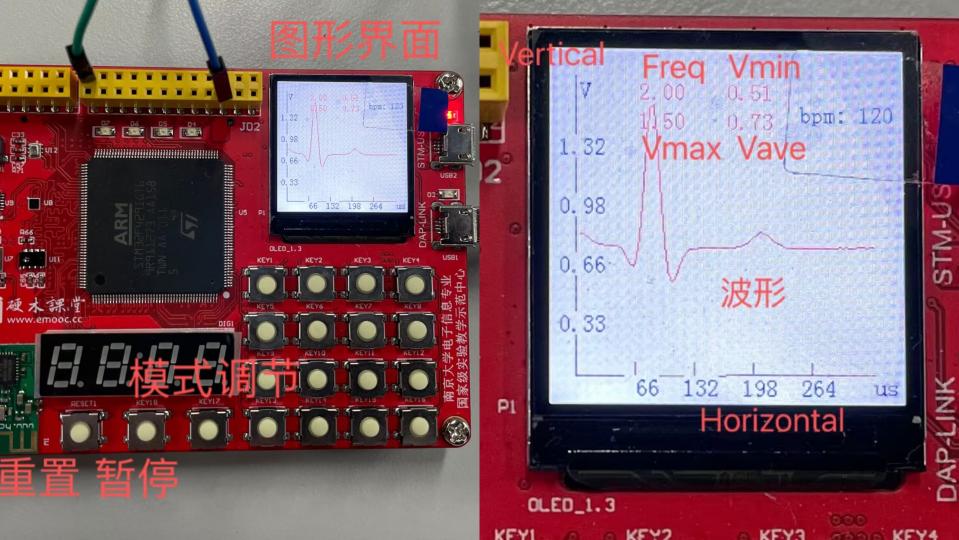
1. 按键信息处理：

（图5.8.1 按键控制程序启停）

（图5.8.2 按键控制显示模式切换）

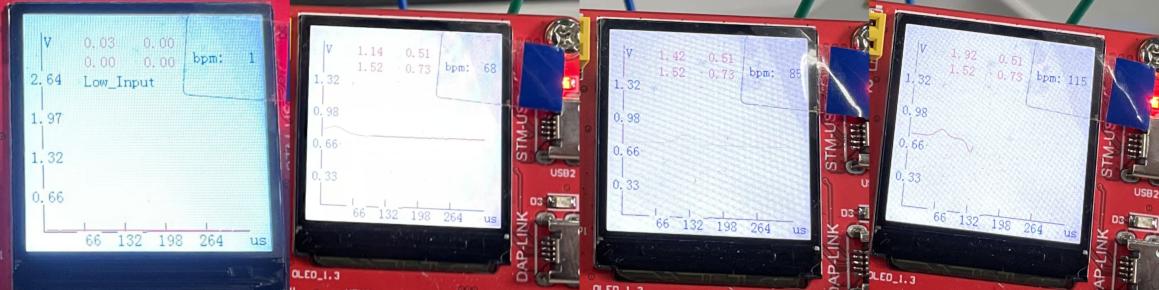
（图5.8.3 四种模式的具体实现）

1. **测试方案与结果**

由于以下测试全部由单片机自身的LCD界面显示，故在最先标明显示界面中各个数值含义。

（图6.0.1 界面介绍）

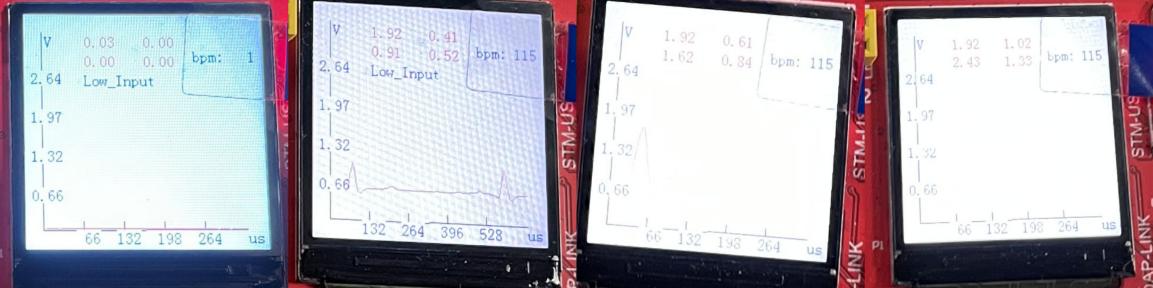
1. 频率测试：

 设置信号源波形为Cardiac，高电平为1.5V，低电平为0.5V，分别取频率为1.14Hz，1.42Hz，1.92Hz，观察LCD屏如下图。左一为无输入情况，目的是形成对照，消除其他因素的影响。其余为测试结果，可以看出，频率处显示分别为1.14，1.42，1.92（带有信号发生器参数的图片参见附件，可以证明小组的测试数据是完全真实的），心率分别为68BPM，85BPM，115BPM，完全符合输入预期，表明小组设计的程序频率测量较为精准，因为受限于屏幕中只显示三位有效数字，如果考虑误差，取最坏情况±0.01Hz，则可能取到的最大误差也不超过1%，精度很高，这与设计程序时的预期是一致的。

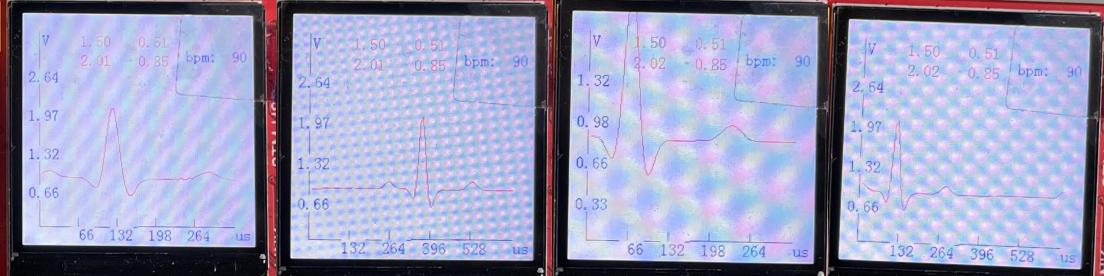
（图6.1.1 频率测试）

1. 幅度测试：

固定频率为1.92Hz，调节高低电平并取三组：0.9/0.4V，1.6/0.6V，2.4/1.0V，观察LCD屏如下图。左一为无输入情况，目的是形成对照，消除其他因素的影响。其余为测试结果，可以看出，最大值最小值十分接近于信号源输出信号的高低电平，并且误差控制在2%以内，这进一步说明无输出情况只是例外并不会影响整体性能。

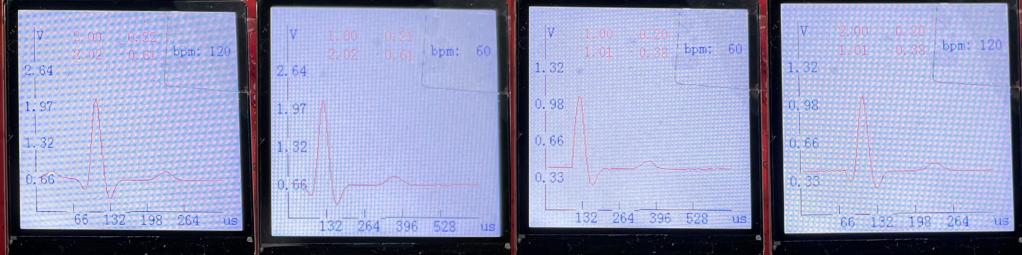
 （图6.2.1 幅度测试）

1. 显示效果模式调节（坐标尺度变化）

取频率1.5Hz，高低电平为2/0.5V的标准波形，验证测试小组自主添加的模式调节模块，显示如下图。前三图中改变横纵坐标刻度以达到不同显示效果；最后一图是我们自主添加的autoscale模式，将输入波形自行转化为最佳的显示模式，同时具有稳定触发，显示相对静止的波形。

（图6.3.1 显示效果模式调节）

1. autoscale模式测试

取两种频率和两组高低电平两两组合（频率1/2Hz;高电平1/2V）成四组测试波形，测试自主添加的autoscale模块，如下图。可以看到，无论怎样的波形经过autoscale模块后都能较为美观地显示在LCD屏上，便于使用者更加清晰地观测心电信号以及其他信号的波形。

（图6.4.1 autoscale模式测试）

1. **实验小结**

（Anlan Peng）

不得不承认，这门课带给了我很大的惊喜，每一个阶段的体验对我来说都是前所未有的。如果说第一阶段只是在形式上的创新，内容仍然是对上学期电子学基础实验I的延续，那么第二阶段则是在第一阶段实验的形式上，对内容又做出了巨大的创新。而单片机的学习一方面让我们第一学期学习的C语言终于派上了用场，弥补了我们电子学生在硬件和软件两个方面学习时的割裂；另一方面单片机也正呼应了当下计算机行业的一个热门方向——嵌入式开发，为同学们以后的发展方向奠定了基础。

关于本阶段实验，我们小组的实验流程和第一阶段有所区别。在第一阶段我们只做了较少的调研，而将大部分时间用于实际电路的调试，甚至最终设计了两块板子。并且在整个实验过程中我们都比大部分同学进展要快，同时我们也希望通过在实践中积累经验，然后尽可能得做出性能较好的电路。但在实际操作中，我们发现这样的方法并不够好，我们大部分时间处于一种孤军奋战的状态，这使我们在面对问题时几乎没有什么参照，也缺乏解决问题行之有效的思路。虽然我们最后做出了性能很不错的电路，但时间成本过于高昂。因此在第二阶段，我们决定降缓实验速度，在前期做出了大量调研工作，并且将更多精力放在和其他同学的交流上。根据很多同学前期设计程序的反馈，我们在方案设计之初就避免了很多可能带来错误的情况，同时在参数设定时都选取了最优的状态。我们发现这样操作大大降低了我们实践时的难度，一方面我们遇到的问题减少了很多，另一方面在解决问题的时候我们有其他同学的方案可以作为参照，不至于完全没有头绪。最后的结果就是我们几乎只用了一天就完成了这个采集程序的雏形，然后又用了一天进行了少量的优化（因为一开始性能就很不错）和创新功能的实现，最后得到的程序性能同样让人满意。在这里我们要感谢所有向我们小组提出过帮助的同学，他们有贺英博老师，葛唐翔老师，张羽老师（他们都是我们21级的同学，但我们希望敬称他们一声老师）。

最后还想谈谈这一阶段实验带给我的一个独特感受。虽然说我们的实验只是做一个心电信号的采集，但其实在本质上我们做的就是一个简易的示波器。如果说心电信号和单片机我们确实不熟悉，那示波器我们没理由感到陌生。但话说回来，都说单片机是电子工程师的“眼镜”，我们电子学生就真的了解示波器吗？就我个人而言，示波器中用到的很多术语，比如采样率、采样深度、触发、死区时间、auto稳定触发等等，我们可能只知道概念，却很难有很深刻的理解。但当我们真正去设计一个示波器（或者具体来说只是心电信号采集器），不得不面对这些让人莫名其妙的参数设计的时候，就不可避免地去对他们有一个非常深入的了解。这让我想到一种哲学理论——“知行合一”，对于我们而言，这门课程就是一个将理论和实践结合起来的良好平台。虽然我们还是难免吐槽我们目前掌握的理论知识的水平和真正能实践的水平仍有一条“鸿沟”需要跨越，但能力的培养和知识的学习从来都是从无到有，从少到多的，我们也只好“痛并快乐着”吧。

（Yusong Qi）

在第二阶段开始之前，我并未接触过单片机方面的知识。从初期相关软件的准备就能感受到此阶段将会异常艰难。碍于期中考试的影响，前几次课程其实没有太放心上，只有零星一点时间观看学习一下教学视频。后来跟着视频一点点熟悉流程，一点点摸索，熟悉CubeMX和Keil软件的操作“先配置功能引脚，再编写代码实现具体功能”。从LED亮灭到流水灯，再到按键控制，PWM波控制亮度，最后到心电信号相关参数计算和波形绘制。在这个过程中，我确实学到了一些关于单片机的知识，一定程度上培养了我的自主学习能力。因此前两次的实验完成地较为顺利并且效果不错。后期老师提供的教程内容相对较少，而自己所掌握的不足以支撑最终要求的实现，我于是就在网络上搜索资料，了解到时钟、中断、ADC、采样等概念，并在和队友Anlan Peng同学的交流之中逐渐搭建起整体框架，以便最后能够利用单片机实实在在地完成一个项目。

但是在这过程中也遇到不少困难。其中有一点是让我们在屏幕中显示参数并实时刷新，最初采用的方案是用clear函数清空屏幕内容再显示新的数值，但效果并不好。问题在于受限于单片机性能，每次只能刷新一个像素点，而整个屏幕却有240\*240个像素点，所以效果不好是必然结果。而后我们改进方案，采用显示数值，之后用白色显示数值将其覆盖以达到刷新效果，实际稳定显示符合预期。之后的绘制波形也就直接采用相同方法。这让我明白实现功能的代码不能只是实现功能，还要兼顾实际效果和空间、时间复杂度，让其效率最大化。在和队友的讨论后，磕磕绊绊，最终算是达成验收指标，并且还添加了我们小组特色的功能（不过这些大多是队友的功劳）。

最后感谢队友Anlan Peng在实验过程中的付出和帮助，感谢实验中心的所有老师。这次经历让我真正意义上接触到了单片机，为以后对单片机的学习提升奠定了基础。

1. **参考资料及文献**
2. 《软件 PPT》 电子系统初级设计 普源精电
3. 《[循环队列详解](https://blog.csdn.net/weixin_55858644/article/details/122313475)》 CSDN
4. 《单片机第3次课 PPT》 詹洪陈
5. 《[触发原理及应用](https://zhuanlan.zhihu.com/p/343484489)》 知乎
6. 《[MCU微控制单元](https://baike.baidu.com/item/MCU/62773?fr=aladdin)》 科普中国 科学百科
7. 《STM32F4xx中文参考手册》 意法半导体
8. 《STM32F429-开发板原理图》 意法半导体
9. **附录**

链接：<https://pan.baidu.com/s/14sGmZng5U8wKO7u70HIDVA>

提取码：0103

（注：百度网盘无法上传部分程序开发文件，因此完整的程序文件需在压缩包中解压后查看）