上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目：肌肉电刺激治疗面瘫中消除伪迹的研究

学生姓名：贾骏
学生学号：5080309067
专　　业：微电子
指导教师：王国兴
学院(系)：微电子

肌肉电刺激治疗面瘫中消除伪迹的研究

摘要

这项研究的主要任务是设计一个在用于治疗面瘫的闭环 FES 系统中排除刺激伪迹的算法。

我们的闭环 FES 系统从面部偏瘫患者正常侧的眼睑采集 EMG 信号，经过处理判断出健康侧眼睑运动状态，然后在瘫痪侧的眼部触发同步的电刺激来恢复其眨眼能力。伪迹排除算法的作用是排除在闭环 FES 系统中由于电刺激产生的伪迹信号对 EMG 信号的影响。

我的研究分两阶段进行，在第一阶段我设计了一个基于 Blanking 的伪迹排除算法，该算法在每次系统进行 FES 刺激之后停止一段时间对 EMG 信号的处理。第二阶段我设计了一个基于数字滤波+Blanking 的算法，并在单片机上实现了 1 阶 IIR 滤波器。

两阶段的算法都通过动物实验验证了效果，最后我们的系统能够在 2 次眨眼/s 的最高判断频率下达到 90%的判断成功率。

。

关键词： FES，刺激伪迹，Blanking，数字滤波

ARTIFACT REMOVAL ALGORITHM ON USING MUSCLE FES FOR TREATING FACIAL PARALYSIS

ABSTRACT

The major task of this research is to develop a stimulus artifact removal algorithm for a closed-loop FES system, which aimed to restore blink function on facial paralyzed.

Our FES system achieves restoration of synchronized blink through processing on EMG signal from eyelid of healthy side in real-time and stimulating the paralyzed eyelid. The artifact removal algorithm works to protect the recording of EMG signal from stimulus artifact.

My research is two-phased. During the first phase, I designed a blanking-based artifact removal algorithm, which stops the blink detection for some time after each FES stimulation. During the second phase, I develop this algorithm into a filtering-blanking-combined algorithm, and realized a 1st order IIR digital filter on the MCU.

Both blanking and filtering-blanking algorithm was tested in animal experiments. Our system finally achieved 90% detection accuracy in the maximum detection rate of 2times/s.

Key words: FES, stimulus artifact, blanking, digital filter

目 录

第一章 绪论

1.1 面神经瘫痪

面神经麻痹，或俗称面瘫，是以面部肌群运动功能障碍为主要特征的一种常见病。面瘫主要由感染、外伤、肿瘤等原因引起，患者面部往往会失去最基本的皱眉、闭眼等功能。

由面瘫引起的最严重的问题之一是失去眨眼能力。由于眨眼是人体通过润湿眼球而保护眼睛的手段，所以失去眨眼能力后，病人眼睛由于长期暴露在干燥空气中，会导致角膜受损，在一些极端的案例中，甚至会致盲。

目前传统的针对面瘫治疗方式主要包括几种：

第一种是通过手术方式，即眼睑缝合术（tarsorrhaphy）。眼睑缝合术的原理是通过手术将病人的眼睑临时性地缝合（图 1-1），但这种手术必将造成不美观的问题、以及视野受到局限[1]。

图 1-1 眼睑缝合术

第二种是通过机械方式，例如在病人瘫痪的眼睑中植入机械微弹簧之类的装置(图 1-2)。通过这种手段，病人可以使瘫痪的眼睑做出周期性的闭合动作。但这类机械方案对手术提出了很高的要求，并且在完成植入后，植入物也需要经常性的调整[2]。

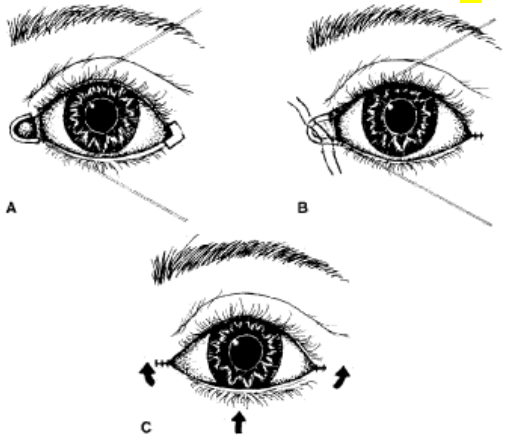

图 1-2 植入式眼睑弹簧(implantable palpebral wire spring)

第三种是功能性电刺激（Functional Eletrical Stimulation, FES）,即通过模仿生物电信号的电流来刺激瘫痪肌肉，使其做出特定的动作（图 1-3）。同其他方案相比，这种方案具有安全性、美观性、动作方式可控的优点，但需要较高的实现成本[3]。

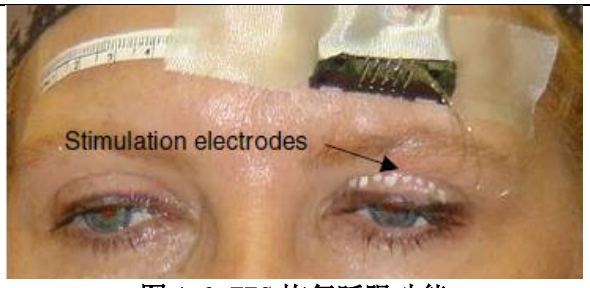

图 1-3 FES 恢复眨眼功能

而本项目的研究内容是在一个 FES 刺激系统中排除刺激伪迹的方法。

1.2 闭环 FES 系统与伪迹原理

面部闭环电刺激的概念在2009的一篇K.F.Chen等人所著的“Closed-loop eyelid reanimation system with real-time blink detection and electrochemical stimulation for facial nerve paralysis”文章中被首次提出（图1-4）[3]。由于在偏瘫患者的治疗中，患者能够控制未受瘫痪影响的一侧肢体活动，研究试图从正常侧采样肌电(EMG)信号，然后通过模数转换、放大、滤波、算法等方式从其中提取某些肌肉动作（如眨眼、闭眼等）的特征信号来控制对另一侧的功能性电刺激，以此达到恢复偏瘫病人两侧眼睑同步眨眼的目的。[4][5][6]

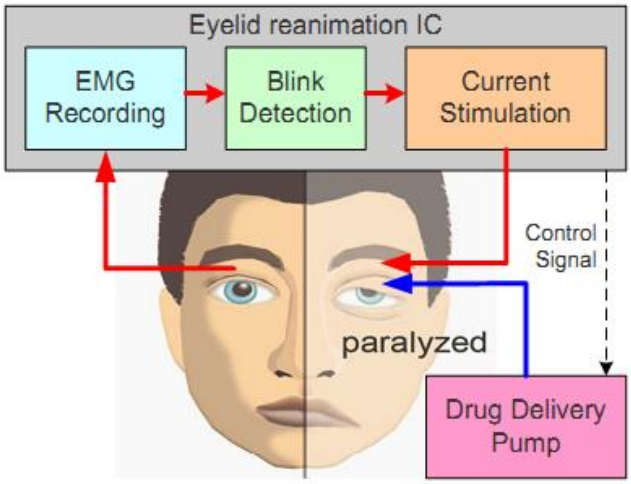

图1-4 由K.F.Chen等人提出的闭环面部FES系统

如何在使用肌电信号控制功能性电刺激时排除刺激伪迹(stimulation artifact)、噪声等的干扰，是上述的完整FES系统中常遇到的一个关键问题。刺激伪迹是指当电刺激、EMG采样同时在相隔一段距离的两块肌肉（或同一块肌肉的不同位置）上进行时，电刺激所造成的干扰信号。由于神经电信号与功能性电刺激信号的幅度相差往往在一个数量级以上，由电刺激产生的刺激伪迹往往会严重干扰、甚至完全覆盖所需要的EMG信号[7]。

本次设计主要针对如何实时排除FES系统中刺激伪迹对眼部EMG采样、分析的干扰，就进行本设计的学生而知，这在伪迹排除技术的研究领域中目前是一个较少得到研究的课题。已知对伪迹的研究主要集中在一种神经电信号（如EEG）中消除其它神经电信号（EOG,EMG等）伪迹的方法[8]；在后置处理（post-process）动作电位(M-wave)时排除刺激伪迹[9]的方法；在采集肢体EMG时实时排除刺激伪迹[10]的方法这几类课题中。

与这些研究相比，眼部采样得到的EMG信号与刺激伪迹幅度更小，且不稳定，这为设计其伪迹消除方法带来了困难。虽然对眼部肌肉的功能性电刺激[11][12]已有十年以上的研究，但基于EMG触发的FES(EMG triggered FES)的实时眼部功能恢复研究还是前沿的课题，仅有的一些研究也还没有达到实际实验的阶段[3]。

第二章 伪迹排除算法研究背景

2.1 研究所采用的闭环 FES 系统

本次毕业设计的前阶段研究中，我们初步实现了一个基于 FES 技术的面瘫患者眼部功能恢复系统。关于这一系统的设计详细资料可参见上海交通大学 07 届毕业生，密西根学院王孟德同学等人的毕业设计报告《Functional Electrical Stimulation For Facial Muscle Rehabilitation》中。

该系统专门针对面部偏瘫患者，由于此类患者面部只有一侧的肌肉群失去控制，而大多数最基本的眼部活动（睁闭眼、眨眼等）是两侧对称的，我们通过采集、并即时处理面瘫患者健康侧的眼部肌电信号(EMG)来判断当下正常眼部肌肉活动情况，然后使用电刺激器对瘫痪侧眼部进行功能性电刺激以恢复患者面部两侧同步的眼睑动作。

系统结构如图所示（图 2-1），系统可分为三部分：EMG 提取、眨眼动作侦测、以及 FES 电刺激器。

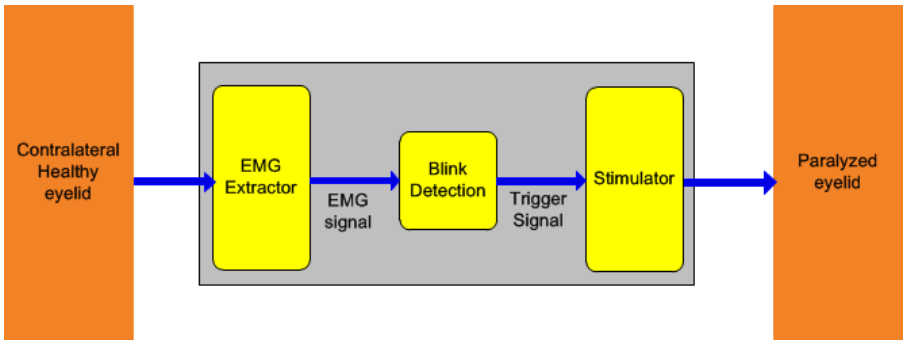

图 2-1 王孟德等人提出的闭环 FES 系统结构

2.1.1 EMG 提取器

首先，使用生物信号电极（Bio-signal electrode）从健康侧眼睑部位记录肌电信号送入 EMG 提取部分，EMG 提取包括前置、后置放大器以及一个 200-500Hz 的带通滤波器。（图 2-2）

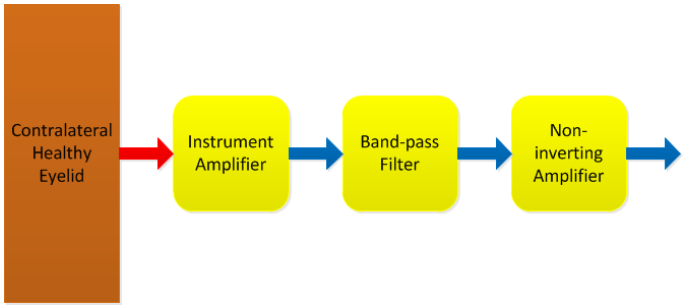

图 2-2 EMG 提取器结构

我们使用 200-500Hz 滤波器，是因为在之前的研究中，我们从兔子的正常侧眼睑采集 EMG 信号进行分析，发现在眼睑的静息状态、和眨眼状态之间，EMG 信号的 200-500Hz 频带

有很大的幅度差别。如果我们把这种差别从原始的 EMG 信号中提取出来，就可以作为进一步判断兔子健康眼睑动作的依据。另外，使用 200-500Hz 的带通滤波器有助于我们滤除 50Hz 的工频干扰。

前置放大器是一个仪表运放，我们利用它的大的 CMRR(common mode rejection ratio)来排除 FES 系统附近的有机体电干扰噪声。而后置运放是一个无反偏(non-inverting)运放，我们用它来做主要的放大级。

通过这一级 EMG 提取器，我们可以从 EMG 信号中提取 200-500Hz 的重要部分，并把 EMG 信号从 200uV 放大到 200mV 的幅度（60dB）。

2.1.2 眨眼侦测器

EMG 信号经过处理后被送入眨眼侦测部分（Blink Detector）。

在这一级中，首先信号使用 ADC 进行模数转换，采样精度是 10bit，采样速度 2kHz。之后，数字化的 EMG 信号被送入 MCU 运行特定的眨眼判断算法，以此根据采到的信号判断正常侧眼睑的当前状态。

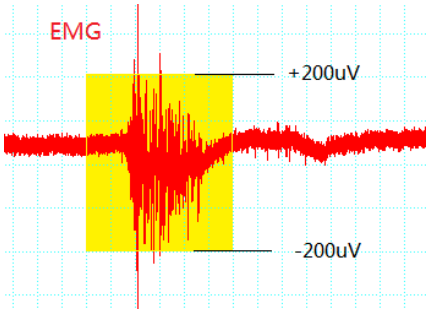

图 2-3 眨眼 EMG 波形

我们所用的判断算法是一个三重阈值算法（triple-threshold algorithm）。由于眨眼的 EMG 信号在幅度上较静息状态更大（图 2-3），并且持续 0.3s 左右的时间，我们使用第一重阈值来判断所采到 EMG 信号的幅度，如果采到的信号幅度大于此阈值，一个计数器变量 Counter 被累加；第二、三重阈值用于检测大幅度 EMG 信号持续的时间长短，具体实现方式就是比较 Counter 变量与第二、三个阈值；通过这样的方式，当算法检测到有超过特定幅度的 EMG 信号出现，并且超过幅度的 EMG 信号持续了较长时间时，就认定健康侧发生了眨眼动作。

当眨眼动作被算法捕捉到时，眨眼侦测级就会触发 FES 刺激器的电刺激。

2.1.3 FES 刺激器

FES 电刺激器是两级的（图 2-4），第一级是一个 10bit 精度的 DAC，这个 DAC 从 MCU 接受描绘刺激波形的数据，并将其转化为电压波形输出。

第二级是电流产生器(current generator)，电流产生器首先通过一个 buffer 来增大刺激器的驱动能力，然后使用运放将 DAC 的电压波形转化为电流波形。使用这样一个电流产生器发出的刺激电流是可控的，可以通过改变加载电阻的阻值来改变刺激电流。使用恒电流刺激而非恒电压刺激的原因是，在 FES 电刺激时，真正决定肌肉动作强度的不是刺激电压，而是刺激电流。

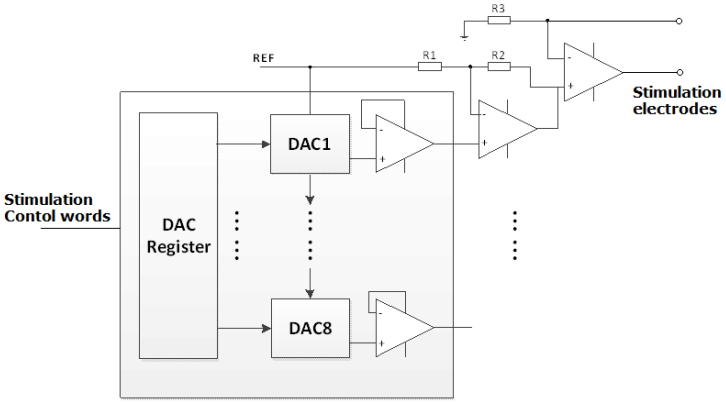

图 2-4　FES 刺激器结构

FES 刺激器进行电刺激波形，如图 2-5 所示的双相脉冲串(bi-phasic pulse train stimulation)，波形频率是 50Hz，幅度是 1mA，每次刺激包含 3-5 个双相脉冲。刺激波形由在本项目中合作的上海市第 9 人民医院的邓思敏博士提出，采用这样的刺激波形可以在保证 FES 刺激效果的前提下减小所需的刺激电流，并且避免对兔子的组织造成损伤。

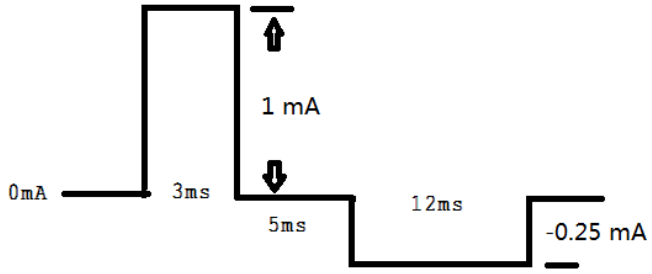

图 2-5 双相脉冲刺激波形

2.1.4 伪迹信号

在系统的临床动物实验中，我们发现偏瘫侧的 FES 电流在采样端产生了刺激伪迹。由于伪迹的最大幅值比 EMG 幅值大了一个数量级以上，EMG 采样受到了严重的干扰，由此限制了面瘫治疗系统的最高刺激频率以及工作的正确率。

2.2 三类伪迹消除算法

在消除神经点信号中实时刺激伪迹的消除算法可分为三大类[7]：伪迹 blanking，伪迹减法，伪迹滤波。

2.2.1 伪迹 blanking

Blanking 算法是指在每次刺激器发生刺激脉冲后完全停止一段时间的 EMG 采样，直到刺激伪迹衰减至不再影响 EMG 的数量级。Blanking 方式消除刺激伪迹可以在硬件或软件上实现，通过硬件实现的伪迹 blanking 电路称为 sample-and-hold（采用-保持）电路[13][14]，通过软件实现伪迹 blanking 需要在信号采集电路中加入微处理器(micro-processor)[15][16]。

软件实现的伪迹 blanking 方式是消除伪迹中最低成本且简单可行的方案，但是这种算法的局限性在于，每次刺激后系统必将丢失一段 EMG 信息，同时系统电刺激的最高频率也

将受到限制。

2.2.2 伪迹减法

伪迹减法（artifact subtraction），这种方法通过在采集到的混合信号中减去理想刺激伪迹的模板(template)得到纯净的 EMG 波形[17-19]。这种伪迹消除方式同样可以在微处理器上实现，并且相比于 Blanking 算法，不会丢失重要的 EMG 信息，也不会产生对刺激频率的限制。

但是，通过这种方式去除伪迹的两个主要问题是：获取理想刺激伪迹的方式、以及伪迹波形是否恒定不变。已有的部分研究考虑到了这些问题，提出通过使用 moving ensemble averaging 的方式动态获得伪迹波形。

2.2.3 伪迹滤波

伪迹滤波（artifact filtering）对混有刺激伪迹的 EMG 信号滤波，利用伪迹信号与 EMG 信号在频域上的不同分布隔除伪迹的影响[20-23]。Artifact filtering 技术通过给信号采集电路添加额外的滤波器实现，考虑到伪迹的频域分布以及幅值并不稳定，大多数系统在滤波时使用自适应滤波器。

滤波算法可以通过软件、硬件的方式实现，具有实时的优点。尽管如此，由于伪迹与 EMG 信号在幅值上的巨大差距，并且二者的频域分布常常相互重叠，滤波方案有不稳定、以及较复杂的设计代价这两个缺点。

第三章　第一阶段：Blanking 算法设计与实现

3.1 Blanking 算法设计

3.1.1 伪迹波形分析

在动物实验中，每次在瘫痪侧做完 FES 电刺激之后，可以在兔子的正常侧眼睑采集到图 3-1 所示的刺激伪迹信号，伪迹信号的第一部分，即 3mV 幅度的部分随着另一侧电刺激的进行同时产生，基本没有延迟。伪迹信号的第二部分，即对 EMG 信号产生的干扰的最主要部分，即长达 1s 的波动尾巴的部分，以 50Hz 的频率进行波动，幅度从一开始的 500mV 左右随时间而逐渐衰减。

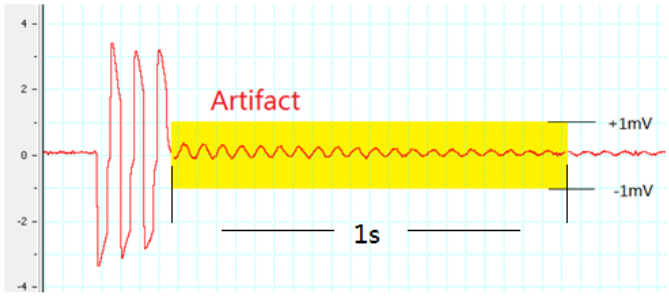

图 3-1 正常侧眼睑采集到刺激伪迹

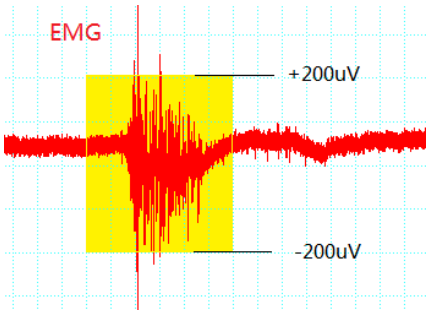

图 3-2 眨眼 EMG 波形

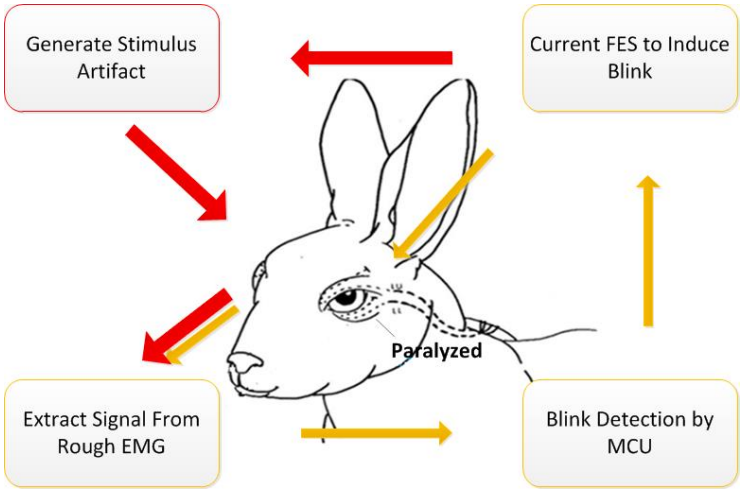

图 3-3 刺激伪迹对 EMG 采样产生了干扰

而 EMG 信号如图 3-2 所示，在健康眼部静息状态下，从眼睑采到的 EMG 信号幅度约在 50mV，当眼睑闭合做出眨眼动作时，幅度上升到了 100-200mV，并持续 0.3s 左右。

由于同 EMG 信号相比，刺激伪迹幅度更大，并且持续时间更长。如果无法成功排除刺激伪迹对 EMG 采样的印象，系统会误认刺激伪迹为另一次的眨眼 EMG，并因此而触发又一次 FES 刺激，新触发的 FES 刺激又会产生新的伪迹信号，由此，一个死循环会在闭环 FES 系统中形成，系统最后会不终止地刺激瘫痪眼睑（图 3-3）。

3.1.2 Blanking 算法设计

在得到如图所示的刺激伪迹信号后，我首先想到的是使用 Blanking 算法来去除伪迹。

在已有的许多 EMG 记录系统中都使用 Blanking 算法来去除伪迹，使用这种算法的系统带有一个采样-保持电路（sample-and-hold circuit），电路的采样运放被一个外来的同步信号所控制（图 3-4），每次有 FES 发出时，同步信号边停止采样运放的工作。

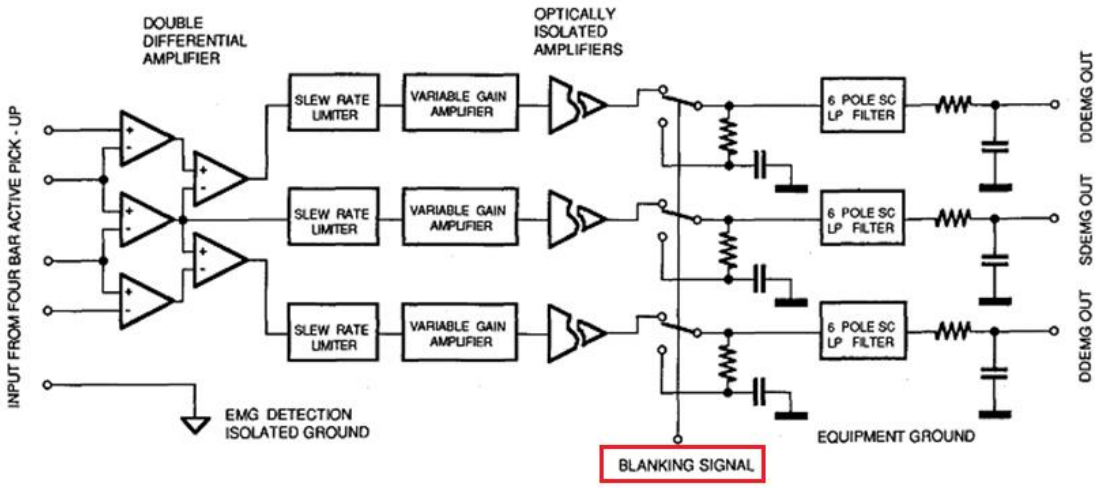

图 3-4 带外来同步信号的 Blanking 电路

这种算法可以应用在我们的面瘫 FES 系统中。由于我们所遇到的刺激伪迹出现在每次 FES 刺激之后，在每次电刺激之后停止一段时间的 EMG 信号采样（在实际实现中，让单片机保持一段时间跑空程序即可）（图 3-5），停止时间的长度可视刺激伪迹的幅度而定，按照我们所采到的信号，可以预计 1-2 秒之后伪迹已衰减到一个不会影响 EMG 信号的幅度，此时，再恢复系统对 EMG 信号的采样。这样就可通过 Blanking 方式排除刺激伪迹的干扰。

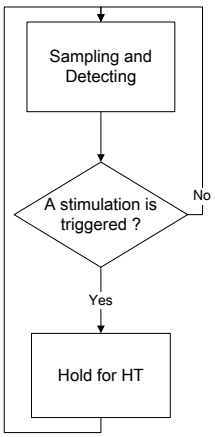

图 3-5 软件 Blanking 算法

我们所涉及的 Blanking 算法中，每次刺激后停止采样的时间长度 hold time(HT)是一个非常重要的参数。这是因为，从一方面来讲，由于每次刺激后都要停止 HT 时间的采样，停止了采样即停止了 FES 电刺激，HT 决定了系统所能达到最高刺激频率，也即通过 FES 系统所能恢复的最高的面瘫病人的眨眼频率。而从另一方面来讲，HT 越长，刺激伪迹就有更长的衰减时间，当经过 HT 后恢复 EMG 采样时伪迹对 EMG 的影响就越小，因此，增加 HT 长度能提高系统的判断准确率。综上所述，HT 的取值重要，并且必须通过实际动物实验来决定。

3.2 Blanking 算法实现

Blanking 算法被实现在系统原有的 Nordic 24LE1 开发板的单片机中。我们选用这块开发板是因为它具备了 51MCU 所需要的定时器、SPI、UART、RC 时钟等，并且还有自带的最高 32kHz 采样频率的 ADC。并且同其他单片机相比，51 单片机有很低的功耗，这些特性都很适合于一个实时、便携系统的要求。

3.2.1 计数器函数

实现 Blanking 算法需要的是一个计时器函数，通过此函数我们可以精确地掌握停止采样时间的长度，这个函数如下：

```
void delay(unsigned int x)
{
unsigned int i,j;
init_Timer();
i=0;
for(i=0;i<x;i++)
{
while(!TF1);
TL1=0x7a;
TH1=0xff;
TF1=0;
}
TR1=0;
return;
}
```

函数中两个变量 TL1/TH1 决定了单片机 Timer 相关的一个寄存器值，当 TL1=0x7a,TH1=0xff, TF1 的值每次从 0 翻转到 1 为 1ms 时间，因此可以通过选择函数参数 x 的值决定计时器计时时间长度是 x ms.

3.2.2 Blanking 实现

Blanking 操作紧跟在每次 FES 刺激之后，所以在实现这个算法时，我把它放在了电刺激函数里，如下：

```
while(!ready)
        ;
```

```
        if(counter>=5)          //阈值设置！0xFF 为 3V
        {
                    LED1 = ~LED1;
                //  CS=~CS ;
                //  DIN=~DIN;
                //  SCLK=~SCLK;
                    for (m=0;m<3;m++)
                    {
                        CS=0;
                        delay(1);
                        SPI_DA(0xcf);
                        SPI_DA(0xfc);
                        CS=1;
                        delay(0x1e);
                        CS=0;
                        SPI_DA(0xc8);
                        SPI_DA(0x00);
                        CS=1;
                        delay(0x32);
                        CS=0;
                        SPI_DA(0xc3);
                        SPI_DA(0xfc);
                        CS=1;
                        delay(0x78);
                    }
                    CS=0;
                    /*进行空循环，在此期间 Voltage 由于中断继续采样，但主程
序中对采样数据不做判断*/
                    for( m=0;m<3;m++)
                        delay(1000);
                    ready=0;
                    counter=0;

        }
```

在单片机中的眨眼侦测算法捕捉到眨眼动作时，ready 参数被设为 1，程序跳出无限空循环进入刺激程序，通过调用 SPI_DA 函数，从 SPI 口发送给 DAC 一系列的参数来决定刺激电流的幅度与波形。

在刺激函数的最后，DAC 的使能口 CS 被置 0，刺激器停止工作。之后使用 for 循环调用 delay()函数进行 Blanking 操作，这段时间内，由于 ADC 采样与眨眼侦测都通过定时中断的方式执行，所以 ADC 仍然在不断采样，眨眼侦测算法也在不断运行，ready 位还是有可能被修改为 1，但是由于主程序中在 delay，系统不会做 FES 电刺激。

而在 delay 运算的最后，ready 位被重新改回 0，这表示在 Blanking 时段中的一切 EMG 信号的侦测结果都被清零，算法由此实现了 Blanking 的目的。

3.3 Blanking 算法验证

3.3.1 实验准备

本项目的动物实验与上海市第九人民医院的邓思敏博士、沈国芳老师合作完成。实验所用到的新西兰白兔从中国科学研究动物中心购买，实验的内容得到了上海交通大学医学院附属上海第九人民医院伦理委员会的批准。

首先，在无菌环境下，我们通过截断部分兔子面部的神经完成了单侧面瘫的动物模型，动物实验安排在这 8 周之后。

我们将实验用兔固定在绑兔架上，在健康侧的眼睑植入采样电极，在瘫痪侧眼睑植入刺激电极后，一切准备就绪。（图 3-6）

图 3-6 动物实验的准备

3.3.2 实验设计

为了测试系统是否工作，我们使用手电照射兔子正常侧的眼睑，以引发自然的眨眼，然后再在瘫痪一侧观察瘫痪眼睑的活动情况。

当系统正常工作时，每次随着正常眨眼，瘫痪侧会做出一个由 FES 引起的人工眨眼动作（见图 3-7）。

图 3-7　FES 系统引发的人工眨眼动作

而系统没有正常工作有两种可能情况，第一种是系统没能侦测到兔子做出的眨眼动作，

FES 电刺激没有发生;第二种是系统侦测到了眨眼动作,但是没有成功排除刺激伪迹的影响,FES 电刺激发生后持续不停（即无论正常侧动作如何，每隔 HT 后就刺激一下）。

3.3.3 实验结果

为了评价实时眨眼恢复系统的表现，我们使用高速摄像机记录下了眼睑的活动情况。我们使用了三种 HT 的设置，包括 2s,1s,0.8s，对每种 HT，都进行了 30 次的眨眼恢复尝试。实验结果见下表（表 3-1）

表3-1 Blanking算法实验结果

	Hold Time (HT)		
	2s	1s	0.8s
成功率	93%	83%	<30%

从实验结果来看，对单纯使用 Blanking 的伪迹排除算法，1s 的 HT 是一个明显的分界线，在 HT=1s 时，系统判断的成功率在 80%，算法基本还能成功排除伪迹的干扰，但是当 HT=0.8s 时，系统已几乎无法排除伪迹，这是因为留给刺激伪迹衰减的时间太短，经过 0.8s 衰减的伪迹对 EMG 信号来说还是有太大的幅度。

第四章 第二阶段：Blanking+滤波算法设计与实现

4.1 Blanking+滤波算法设计

在完成 Blanking 算法的设计与实验验证后，我继续考虑改进算法来提升伪迹排除算法的能力。分析伪迹波形，我发现伪迹信号对 EMG 信号产生干扰的最主要部分，即长达 1s 的 waving tail 部分在以 50Hz（即 FES 刺激频率）进行波动。使用生物信号处理软件 LabChart 对该伪迹进行截止频率 100Hz 的理想高通滤波后，waving tail 长度从 1s 缩短到了 10ms 级别（图 4-1,4-2）。

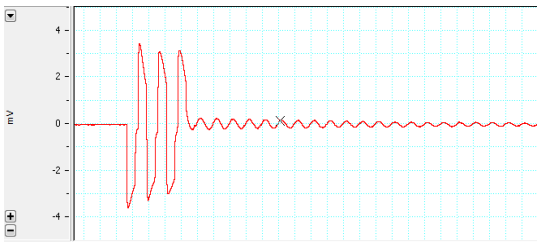

图 4-1 带 waving tail 的伪迹

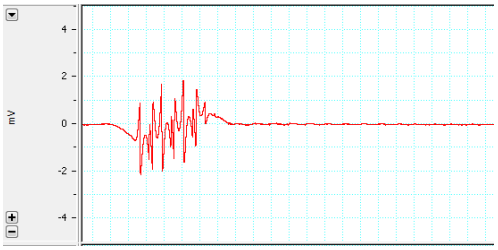

图 4-2 滤波后的伪迹

所以我的新设想是设计一个先滤波，后 Blanking 的算法来排除伪迹的干扰。由于 EMG 信号在 200-500Hz 的频段中，而需要滤除的伪迹在 50Hz，可以通过一个高通滤波器来抑制刺激伪迹中最主要的干扰成分：即 50Hz 的干扰，抑制的效果要视滤波器的性能（最重要的是，滤波器的阻带增益）而定。滤波算法的优点在于它是实时的，不会导致 EMG 信号的丢失。

经过滤波后，伪迹可能从原先的 1s 长缩短至了 0.6s,0.5s 甚至更短，如果此时，将剩下的伪迹通过 Blanking 的方法去除，就能大大缩短 Blanking 所需的时间，进而提升系统的最高刺激频率。

4.2 数字滤波器设计

4.2.1 模拟滤波器与数字滤波器

模拟滤波器的本质，是频率选择电路，用于放大或衰减单一频率成分的信号频谱的一部分。而数字滤波器则是一个离散时间的线性非时变系统，用于完成信号滤波处理功能、并且用有限精度的算法实现。它能改变包含在离散时间信号 x 中的谱信息，使其生成新的离散时间信号 y，采样信号表示为一列数字。数字滤波器本身既可以是用数字硬件装配而成的一台用于完成给定运算的专用数字计算机，也可以是将所需的运算编成的程序，让通用计算机执行。同模拟滤波器相比，数字滤波器具有稳定性高、精度高、灵活性大等突出优点。

由于我们的系统最重要的是目标是实时，以及低的硬件成本，所以选定了使用数字滤波方案，在本系统现有的 51MCU 上通过软件方式实现。

4.2.2 滤波器性能比较

在确立数字滤波器方案之后，我开始了数字滤波器相关知识的学习。数字滤波器可分为 IIR（Infinite impulse response）与 FIR（Finite impulse response）两种。我使用 Matlab 仿真比较了它们的性能。

方案一：1 阶 IIR 滤波器

IIR 型滤波器是一种递归型滤波器，具有反馈，结构简单，系数少，乘法操作少，可以解析控制，效率较高，与模拟滤波器有对应关系。其缺点是要考虑系统稳定性，需要记忆的数据多，易产生溢出。两者相比较，在满足性能要求的前下，IIR 滤波器比相应的 FIR 滤波器具有低得多的阶次。

我首先尝试的方案是一个 1 阶 IIR 滤波器，它的时域函数为:

Y[n]=a1*X[n]+a2*X[n-1]+b*Y[n-1]

其中 X[n]为当前时刻的滤波器输入（即 EMG 信号），Y[n]为当前时刻的滤波器输出，X[n-1],Y[n-1]为上一时刻滤波器的输入与输出，a1,a2,b 为权值系数，它们的取值与采样频率 fs 共同决定了滤波器的通带频率。它们的取值通过 matlab 获得，运行以上语句：

[a,b]=butter(1,0.1,'high');

设计一个巴特沃兹、1 阶、100Hz 截止频率的高通滤波器，得到 matlab 返回的向量[a,b]，带入时域函数即得到所需要的 IIR 滤波器。

为了验证滤波器的效果，我编写了以下的代码：

```
Fs=2000;
n=0:200;
t=n/fs;
x=cos(2*pi*50*t)+cos(2*pi*300*t);
y(1)=0;

for i=2:200
    y(i)=0.86*x(i)-0.86*x(i-1)+0.73*y(i-1);
end
subplot(2,1,1); plot(x);
subplot(2,1,2); plot(y);
```

在代码的前三行首先设定了滤波器的采样频率（fs=2000Hz），仿真时间长度。由于 EMG 信号是 200-500Hz 带通滤波的结果，伪迹在 50Hz，我使用 x=N*cos(2*pi*50*t)+cos(2*pi*300*t)的 50Hz 与 300Hz 混频正弦波模拟实际系统中采集到的混有刺激伪迹的 EMG 信号。系数 N 表示了两种信号之间的幅度差别。

在最开始的仿真中，N 设为 1，即二者幅值相同，滤波后的结果如下图：

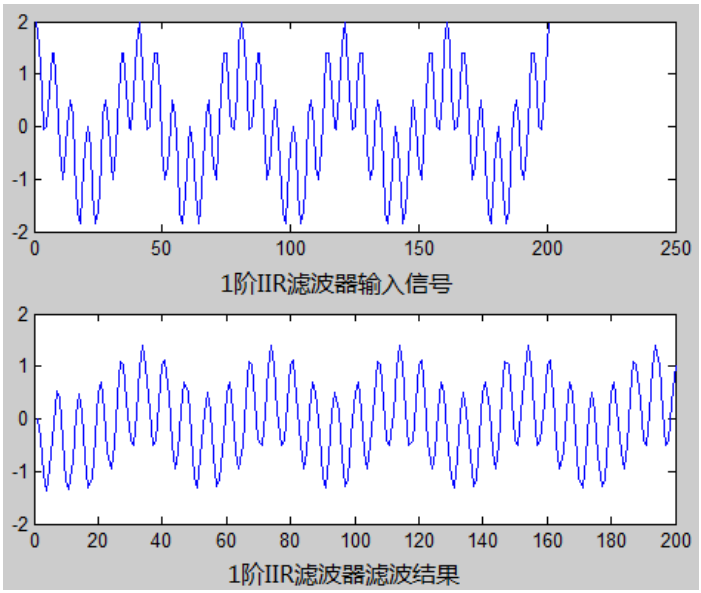

图 4-3 1 阶 IIR 滤波器分隔等幅值比的两信号

从图中可以看出，此一阶滤波器能够成功分隔两种信号。

为了测试滤波器的性能，我将两种信号间的幅值差别继续加大达到 10 倍，滤波结果见下图。

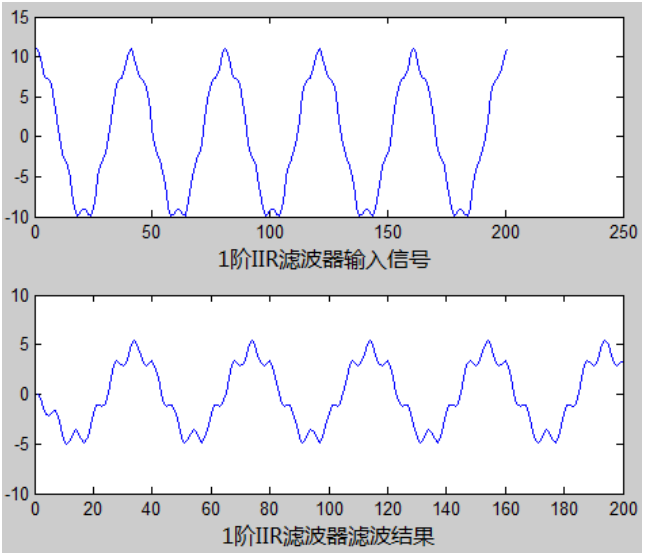

图 4-4 一阶 IIR 无法滤除 10 倍幅度比的 50Hz 信号

可以看出，滤波器已经无法从 300Hz 的正弦波中滤除 50Hz 信号的影响（图 4-4）。

最后，将 LabChart 平台上实际采集到的伪迹信号输入到 1 阶 IIR 滤波器，运行如下代码：

```
x=textread('artifact.txt');
N=1;wc=0.1;
[b,a]=butter(N,wc,'high');
subplot(321);plot(x);title('滤波前信号的波形');
y=filter(b,a,x);
```

```
subplot(323);plot(y);title('滤波后信号的波形');
```

这段代码中，首先使用 textread 函数将伪迹信号（保存在 txt 文件中）存入向量 x 中，使用 butter()函数设计 1 阶滤波器，使用 filter()函数对伪迹进行滤波，最后用 plot 函数观察结果。可见图 4-5。

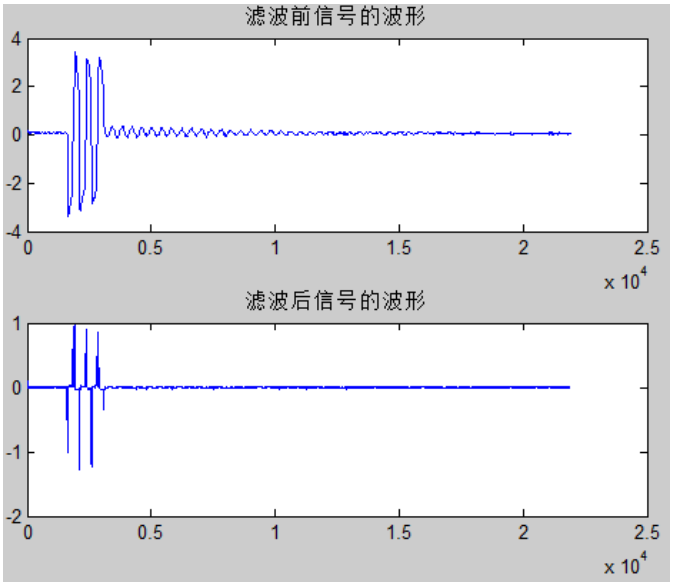

图 4-5 1 阶 IIR 滤波器对实际伪迹信号的滤波效果

从图中看，刺激伪迹的长达 1s 的尾巴被明显的衰减，这定性地反应除了滤波器的效果。但是在滤波器滤除 50Hz 伪迹的同时，EMG 信号也会受到影响，见下图，我将 LabChart 记录到的眨眼时 EMG 信号导入到 Matlab 中，并同样经过了 1 阶 IIR 滤波，可见信号（虽然波形形状未受影响）幅值从约 200mV 衰减到了 50mV。所以具体评价 1 阶 IIR 滤波器的效果，还是要比较经滤波后伪迹的幅度与经滤波后 EMG 信号的幅度而定。

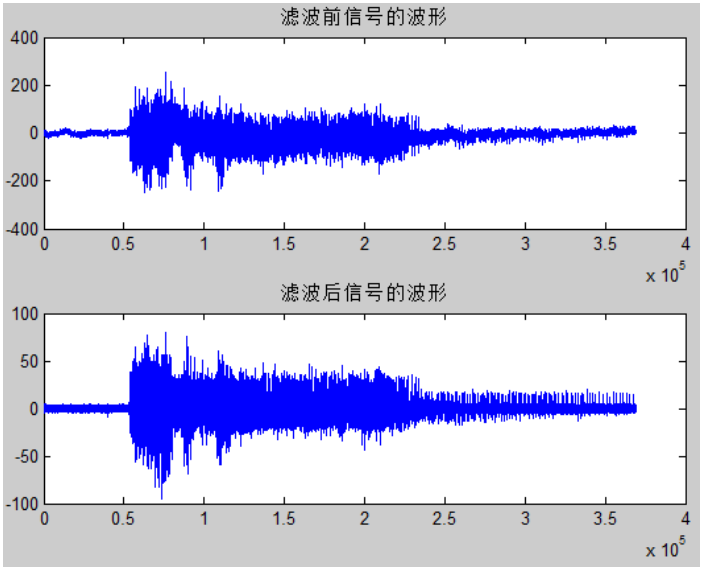

图 4-6 1 阶 IIR 滤波器对实际 EMG 信号的滤波效果

方案二：高阶 IIR 滤波器

高阶 IIR 滤波器同样采用的是递归型结构，因自带有反馈环路而对单位脉冲输入产生无限长响应。数字 IIR 滤波器的结构有直接Ⅰ型、直接Ⅱ型、级联型以及并联型几种等网络结构类型。

直接Ⅰ型

高阶 IIR 滤波器的系统函数为：

$$H(z)=\frac{\sum_{r=0}^{M} b_r z^{-r}}{1+\sum_{k=1}^{N} a_k z^{-k}}$$

在时域上，它对应的差分方程是这样的：$y(n)=\sum_{i=0}^{M} b_i x(n-i)-\sum_{i=1}^{N} a_i y(n-i)$

它的信号流程图如下：

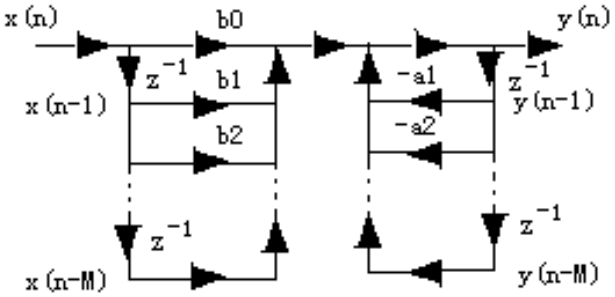

图 4-7 直接Ⅰ型滤波器信号流程图

直接Ⅱ型

IIR 滤波器的系统函数也可以被写成：

$$H(z)=\frac{Y(z)}{X(z)}=\frac{Y(z)W(z)}{W(z)X(z)}=(\sum_{r=0}^{M} b_r z^{-r})\frac{1}{1+\sum_{k=1}^{N} a_k z^{-k}}=H_1(z)H_2(z)$$

它的第一个子系统 $H_1(z)$ 实现了零点，转化为时域为 $y(z)=\sum_{r=0}^{N} b_r w(n-r)$ 。

它的第二个子系统 $H_2(z)$ 实现了极点，转化为时域为 $w(z)=x(z)+\sum_{k=1}^{N} a_k w(n-k)$ 。

它的信号流程图如下：

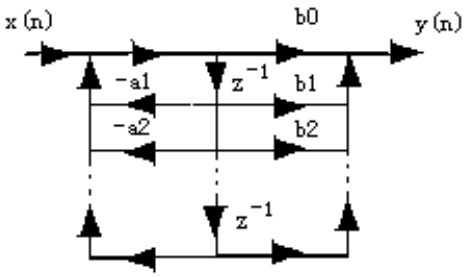

图 4-8 直接 II 型滤波器信号流程图

由于直接 I 型在时域差分方程上有简单的形式，我采用了直接 I 型来设计高阶的 IIR 滤波器。

方案二中，我设计了一个基于切比雪夫函数的 3 阶滤波器，其时域函数为

Y（n）=Σbk*x（n-k）-Σak*y（n-k）。

其中，x(n)为滤波器输入的离散信号数组，y(n)为经过 IIR 滤波后的数组。

在 matlab 中调用 cheby1(n,Rp,fp)函数得到关于该滤波器的各项参数 ak 与 bk.将其带入查分方程来获得 3 阶 IIR 滤波器。具体代码如下：

```
fs=2000;
n=0:200;
t=n/fs;
x=cos(2*pi*50*t)+cos(2*pi*300*t);
y(1)=0;
y(2)=0;
y(3)=0;
for i=4:200
y(i)=0.0000125*x(i)+0.00003751*x(i-1)+0.00003751*x(i-2)+0.0000125*x(i-3)+2.9322*y(i-1)-2.8687*y(i-2)+0.9364*y(i-3);
end
subplot(2,1,1); plot(x);
subplot(2,1,2); plot(y);
```

3 阶的 IIR 滤波器具有比 1 阶 IIR 滤波器更好的滤波性能，我给滤波器输入了幅值相差 5 倍的两个正弦波信号 x=cos(2*pi*50*t)+5*cos(2*pi*300*t)，得到如下滤波结果(图 4-9)，可见 3 阶 IIR 信号能够清晰地分隔两个信号。

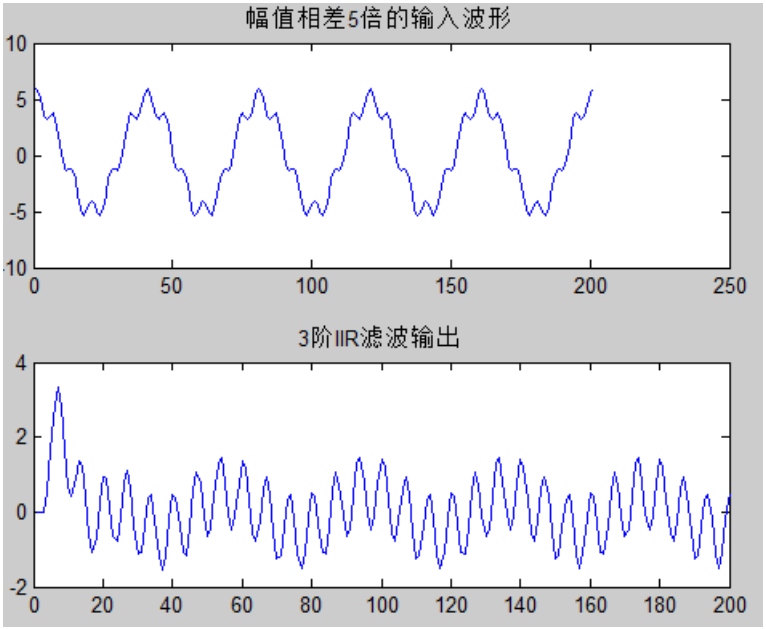

图 4-9 3 阶切比雪夫滤波器能够滤除幅值 5 倍的 50Hz 信号

但是，继续增大代表伪迹信号的 50Hz 正弦波幅度，达到 EMG 信号的 10 倍，此时滤波结果中，50Hz 与 300Hz 的幅度相当，依然无法滤除伪迹信号的影响（见图）。

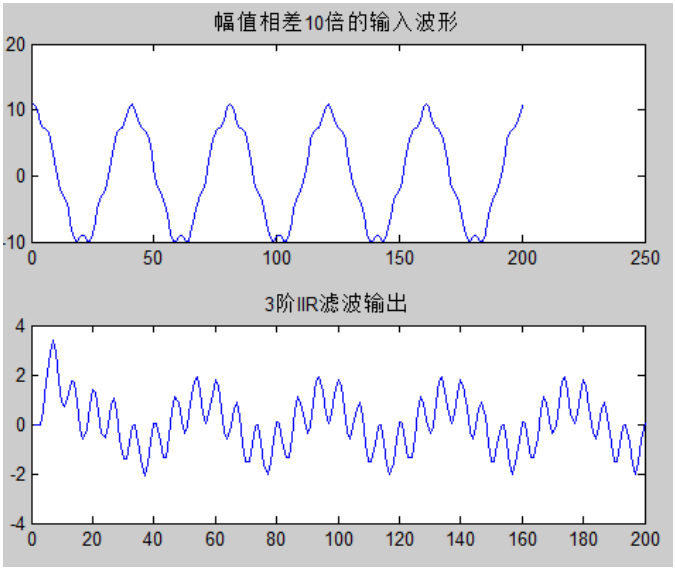

图 4-10 3 阶 IIR 滤波器依然无法滤除幅值 10 倍的 50Hz 信号

方案三：加窗 FIR 滤波器

IIR 滤波器是带有递归的滤波器，其任一时刻的输出都与前若干时刻的输出有关。而 FIR 类滤波器则是非递归型滤波器，它没有反馈项，仅有正馈。也主要是由于这个原因，FIR 滤波器的稳定性比 IIR 滤波器更好、硬件更易实现。但要达到高性能，需要更高的滤波器阶次，更多的系数以及更多的乘法操作。这就意味着，更长的运算时间代价。

FIR 滤波器从设计方法到性能表现都与 IIR 滤波器有较大的差别。FIR 滤波器的设计方法主要有三种：窗函数法、频率采样法和切比雪夫等波纹逼近的最优化设计方法。在这次研究中，我采用了加窗的方法进行设计。

所以，FIR 滤波器的设计关键是窗函数的选择，由于理想滤波器（即滚降因子 a=0，阻带衰减无穷大）的系统函数傅里叶变换（sinc 函数）在时域有无穷长单位脉冲响应，我们对其时域响应加特定窗函数截取其中有限长度的一段，作为实际滤波器的单位脉冲响应。

在设计过程中我考虑了不同窗函数的幅度响应特性：常用的窗函数包括矩形窗、汉宁窗、凯泽窗、布莱克曼窗等，不同窗函数具有不同的时域包络形状，并和窗长一起决定了滤波器的近似过渡带宽、精确过渡带宽、最小阻带衰减三项性能参数。其具体对应关系如下表所示（表 4-1）

表4-1 FIR滤波器窗函数的特性

窗函数名称	近似过渡带宽	精确过渡带宽	最小阻带衰减
矩形窗	4 pi/M	1.8 pi/M	21 dB
汉宁窗	8 pi/M	6.2 pi/M	44 dB
凯泽窗	8 pi/M	6.6 pi/M	53 dB
布莱克曼窗	12 pi/M	11 pi/M	74 dB

按照表格上的数据，我选择了汉宁窗（Hanning Window）进行 FIR 高通滤波器设计，能够提供 40dB 的阻带衰减。窗函数形状、幅频特性见下图（图 4-11）。

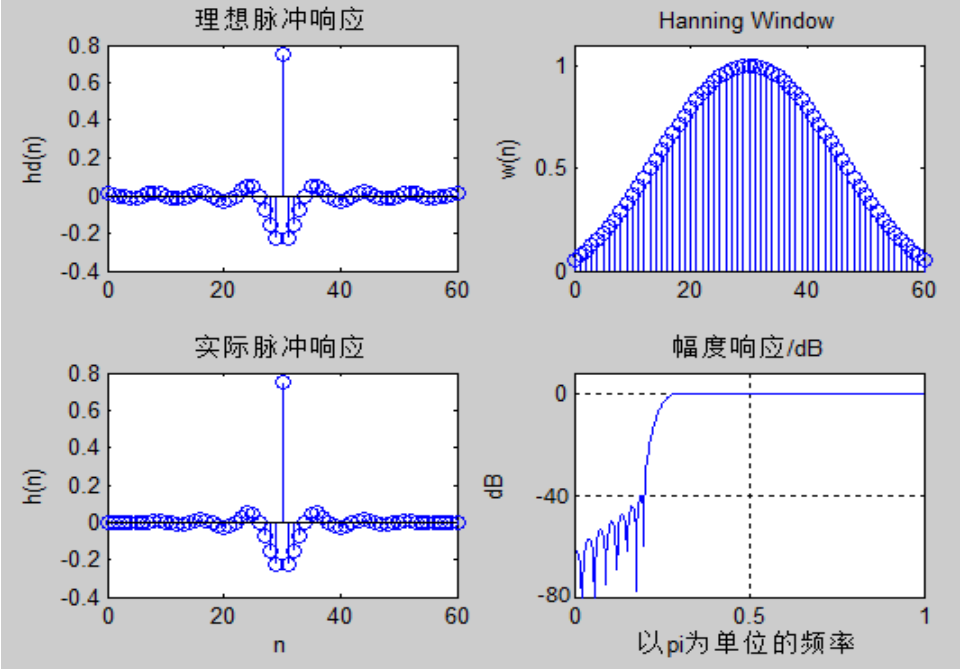

图 4-11 Hanning 窗 FIR 滤波器仿真结果

观察图5中的幅频特性图可见,若调整采样频率,使其通带截止频率(约0.2pi)为300Hz,则 50Hz 伪迹信号处（即 0.05pi 处）衰减能够达到所需的数量级。凯泽窗 FIR 滤波器由此得到验证为可行。

综上所述，三种滤波器方案中，从滤波性能上选择合适的窗函数可以使 FIR 达到最佳的滤波性能，其次高阶 IIR 滤波器，再次 1 阶 IIR 滤波器，但是 IIR 滤波器随着阶数上升性能的提升并不明显。

4.2.3 IIR 与 FIR 滤波器实现成本比较

在确定了两种滤波器的有效性后,我比较了 IIR 和 FIR 滤波器在单片机上使用 C 语言实现的成本——我考虑的方面主要包括，C 代码编写、维护成本；单片机存储成本；以及单片机运算时间成本。这三种成本中，后两者将直接影响到系统的稳定性，以及它对 EMG 信号的响应速度。

IIR 滤波器使用 C 语言实现的成本较小，IIR 滤波器的数学运算通常有两种实现方式。一种是频域法，即利用 FFT 快速运算办法对输入信号进行离散傅立叶变换，分析其频谱，然后根据所希望的频率特性进行滤波，再利用傅立叶反变换恢复出时域信号。这种方法具有较好的频域选择特性和灵活性，并且由于信号频率与所希望的频谱特性是简单的相乘关系，所以它比计算等价的时域卷积要快得多。另一种方法是时域法，这种方法是通过对离散抽样数据做差分数学运算来达到滤波的目的的。

若采用频域法实现滤波运算，IIR 滤波器时域响应为：Y（n）=Σbk*x（n-k）-Σak*y（n-k），对一个 k 阶 IIR 滤波器，其存储成本是用于存储 a,b 系数的 2k 个 double 数据，以及存储 x(n),y(n)的 2k 个 double 数据。每得到一个新的滤波结果的运算成本是 2k 次浮点乘法以及 1 次减法。

相较而言，FIR 滤波器的实现成本要远大于 IIR 滤波器。首先，要得到窗函数就需要复杂的运算，例如凯泽窗函数为，在单片机中做平方、开根号运算都要有较大的代价：

$$\omega(n)=\frac{I_0\left[\beta\sqrt{1-\left[(2n/(M-1))-1\right]^2}\right]}{I_0[\beta]},0\le n\le M-1$$

其次，为了得到 FIR 滤波器的输出结果，需要将输入信号与 FIR 滤波器的时域函数做卷积。FIR 滤波器对输入离散信号做滤波的过程，实际上是将加窗后的 sinc 函数与输入信号数组做不断卷积的过程，如下图所示（图 4-12）。

图 4-12 卷积运算示意图

举例来说，在我 Matlab 仿真时所设计的使用 Hanning Window 的滤波器，其性能与窗长有直接的关系，为了得到 40dB 的阻带衰减需要窗长为至少 40 个点。也就是说，在做卷积运算时，每得到一个新的输出数据，都需要做 40 次浮点乘法运算以及 40 次加法运算。这样大的运算量对单片机是不现实的，由于系统的实时性要求，采样频率为 2KHz，即每 0.5ms 得到一个滤波结果，而单片机的时钟周期约为 10us，在 500us/10us=50 个时钟周期内，无法完成 FIR 卷积这样的复杂运算。

除此之外，使用 c 语言实现实时输入的卷积函数本身就需要较复杂的代码、以及较大的存储空间需求。以上这些成本都对单片机的性能提出了较苛刻的需求。

综上所述，在部分关键性能类似的情况下， IIR 滤波器成本远低于凯泽窗 FIR 滤波器，代码编写简单、运算量小、且所需存储空间小，更适宜于 51 单片机平台的实现。

再比较方案一、二，当伪迹与 EMG 幅度相当时，两种方案都能够滤除刺激伪迹的影响，但当伪迹继续增大，达到 EMG 的 5-10 倍时，两种方案都无法完全排除伪迹的干扰。

考虑实际采集到的信号波形，伪迹 waving tail 部分一开始的幅度大约在 0.5mV，而 EMG 在眨眼动作时 0.2mV、在静息状态 0.1mV，二者相差的倍数在 2-5 倍之间，所以要完全排除伪迹的干扰，IIR 滤波器可能还无法办到。

两种 IIR 方案都可以办到的是衰减刺激伪迹信号，并且衰减的程度并没有太大的差别，考虑到随着阶数升高，需要更多的运算时间成本，1 阶 IIR 滤波器是个更好的选择。

4.3 Blanking+滤波算法实现

根据以上的分析，我最终选择了 1 阶 IIR 滤波器。在完成算法的设计后，该算法被实现在了已有 FES 系统的单片机中。

```
For ( j = 1 ; j > 0 ; j -- )
{
    X_in [ j ] = X_in [ j – 1 ] ;
    Y_out [ j ] = Y_out [ j – 1 ] ;
}
X_in [ 0 ] = ADCDATH ;
Filt_out = 78 * X_in [ 0 ] -78 * X_in[ 1 ] + 55 * Y_out [ 1 ] ;
```

```
Y_out [ 0 ] = Filt_out / 100 ;

If ( Y_out [ 0 ] > Volthreshold_pos )
    Counter ++ ;
```

滤波算法是与眨眼侦测算法互相嵌入在一起的，整个滤波+侦测的流程图见图 4-13 所示。滤波算法只有在非 Blanking 期间才会运行，在程序中，我使用了两个数组 int x_in [2], y_out [2], 分别存储滤波器的最近两个时刻的输入、输出结果，使用 int 类型、而不是 float 类型是因为，实际的电压值是(x_in [I] / 0xFF) * 3V。

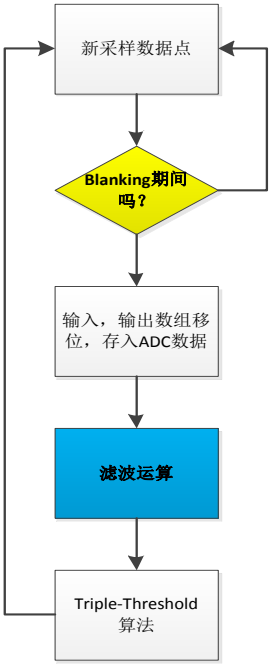

图 4-13 Blanking+滤波流程图

Filt_out = 78 * X_in [0] -78 * X_in[1] + 55 * Y_out [1]；这一句做 1 阶 IIR 的滤波运算，使用 78,55 等系数之后再将结果除以 100，是因为单片机做浮点乘法的速度比整形乘法要慢很多，这是一种常用的技巧。

4.4 Blanking+滤波算法验证

第二阶段的动物实验共进行了 1 次，实验日期在 2012 年 4 月份进行，同上海市第 9 人民医院的邓思敏合作进行。

实验的准备与实验的方法同第一阶段是一样的，我们通过手术截断面部神经完成了面部偏瘫的兔子模型，然后使用手电照射瘫痪的眼部引发眨眼动作，观察瘫痪侧的眼部动作。在试验中，使用了滤波+Blanking 的算法，评价算法优劣的参数是 Blanking 的时间长 HT。首先我们将 HT 设为 2s，调整参数直到系统能够正常工作。

之后，我们把 HT 设为了 0.5s，即每次 FES 刺激之后停止 0.5s 的眨眼侦测，我们一共尝试了两组眨眼的恢复，每组包含连续的 10 次手电筒照射引发的眨眼动作，每两次照射之间的时间间隔约为 1-2s，两组的成功率都在 90%，即除了 1 次眨眼恢复失败（失败时，系统没有能够侦测到正常侧眼睑的眨眼动作），其余 9 次都成功。

与之前使用单纯 Blanking 的伪迹排除方式相比，（到 Blanking 时间长减小到 0.8 时，

成功率已经在 30%以下）滤波+Blanking 的效果要好很多。

第五章 总结

5.1 问题整理

5.1.1 IIR 滤波器所参考的模拟滤波器类型

由于高阶 IIR 数字滤波器的设计借鉴于已成熟的模拟滤波器设计方法，考虑选用何种的模拟滤波器对 IIR 滤波器的性能也会有影响。

常用的模拟滤波器有通、阻带都单调衰减的巴特沃兹(Butterworth filter)滤波器、通带是等波动逼近阻带单调衰减的切比雪夫（Chebyshev filter）滤波器、以及通阻带都是等波动逼近的椭圆滤波器。

在 Matlab 上做性能仿真时我设计的是巴特沃兹型滤波器，这是因为不清楚各种模拟滤波器之间区别时，巴特沃兹滤波器是最常用的一种，这种滤波器通带阻带的袋内特性最为平坦，也有较好的截止特性以及相位特性，总的来说，易于得到符合设计值得特性。

而考虑其他几种模拟滤波器：切比雪夫滤波器通带内有等波纹起伏，有非常好的截止特性，但相位特性与群延迟特性不好。贝塞尔滤波器通带内的延时特性最为平坦，能够无失真地传送方波、三角波等频谱很宽的信号，但截止特性就比较差。

考虑到我们的设计中，最重要的阻带的衰减特性，事实上切比雪夫滤波器应该是一个更好的选择。尤其是在我们的 FES 系统中，无论对滤除伪迹还是保留 EMG 信号的精度要求都不高，侦测眨眼算法判断 EMG 的依据是 EMG 信号的幅度以及持续的时间长度，所以滤波器通带是否失真、截止特性如何并不是特别重要的因素。

5.1.2 非线性相位响应问题

在查阅资料学习 FIR/IIR 滤波器设计时，我注意到大多数参考资料都会提到这两种滤波器之间最重要的区别之一在于相位响应特性——IIR 滤波器为非线性相位滤波器，FIR 为线性相位滤波器， IIR 滤波器会导致通过滤波的信号不同频率分量之间相位差与频率不成正比。在一些精密图像信号处理场合，这点区别会导致信号失真。

但在本次设计中，考虑到系统对信号具体波形是否失真扭曲等并不看重，我认为使用相位非线性的 IIR 滤波器并不会有问题。最终的动物试验中，IIR 滤波器成功地配合眨眼侦测算法判别出了兔子的眨眼动作，这也证明了我的想法，非线性响应问题并没有影响到系统的工作。

5.1.3 高通滤波器与低通滤波器

在设计滤波器的过程间，我犯了一个因经验不足而导致的错误，虽然系统需要的是一个能将 50Hz 信号滤除在 300Hz 信号之外的高通滤波器，在项目初期，我的整个设计过程中一直在使用 FIR、IIR 的低通滤波器来验证性能。这是因为，我想当然地以为可以用一个全通滤波器（即原信号）减去一个低通滤波的结果来获得高通滤波的效果，这个假想在之后 Matlab 的仿真中被证明为错误的。

具体原因是因为全通-低通=高通是频域的运算，而时域的相减运算并不等价于频域的相减运算，再深究原因，可能是信号经过低通滤波后相位已产生了变化，再用原信号相减就有了一个相位差，不再能得到原本的高频信号。

虽然如此，同型号、同阶数的高通与低通滤波器性能应该是相近的，如果低通滤波器能够滤除 10 倍的高频信号，高通滤波器应该也能滤除 10 倍的低频信号。所以当我低通滤波器验证了滤波器性能后，我又设计了高通滤波器并仿真，结果符合我的设想。

5.2 展望

本次的毕设研究内容，实际上是我们关于眨眼 FES 恢复系统两年的研究中的一个环节。回顾整个研究过程，我们最初实现的对兔子健康侧眨眼动作的判断功能，当时我们从兔子健康侧提取 EMG 信号并将信号送入单片机做判断，如果系统认为有眨眼动作发生，就点亮 LED 灯作为标志。再之后我们完成了 FES 刺激器的设计，并通过 SPI 口将单片机与刺激器连接了起来，这样就能够在系统判断出眨眼动作时触发 FES 电刺激。完成这一步后，我们在动物实验中第一次遇到了刺激伪迹的问题，而本次毕设的研究就开始于这个时候。截止到撰写报告的今天为止，我们所设计的整个系统已经具备能够抑制伪迹、保证一定成功率下恢复兔子眨眼功能的能力。但是，从这一步到我们研究项目的最终目标——一个有效、稳定、安全、便携的 FES 眨眼恢复器产品还有很长的路要走，我整理了以下几点，我认为如果要开展下一步研究最重要的研究内容。

5.2.1 从手电眨眼到自然眨眼

在验证系统功能的动物实验中，我们使用手电筒、从很近的距离照射兔子正常侧的眼部以引发眨眼动作，但事实上，由手电筒照射所引起的眨眼动作同一个完全自然、出自本能的眨眼动作相比，在同一位置所采集到的 EMG 信号肯定是不同的。我们一开始发现这一点是在我们的动物实验过程中，我们发现，当我们用手电筒照射的方式引发眨眼动作时，系统能够检测到这一动作，并触发同步的 FES 电刺激；但是偶尔，当兔子的健康眼部自发地做出眨眼动作时，系统对这一动作没有任何的反应。

于是，我们另外设计了实验来观察兔子健康眼睑自发眨眼时所采集到的 EMG 信号。如图 5-1 所示，实验是这样设计的，我们通过两种方式同时观察兔子健康眼睑的活动情况，一种是使用电极采集 EMG 信号，一种是在眼睑上悬挂测力仪器，这种仪器能够将绳子上的张力转化为电信号，并同 EMG 信号一起显示在 LabChart 上。

图 5-1 张力器与 EMG 记录同时进行

在放置好仪器、和电极之后，我们安静地等候兔子自发做出眨眼动作，就得到了图中所示的波形，蓝色波形出现波动的时候，正是兔子做出自发眨眼动作的时候，而红色波形就是自发眨眼所产生的 EMG 波形。

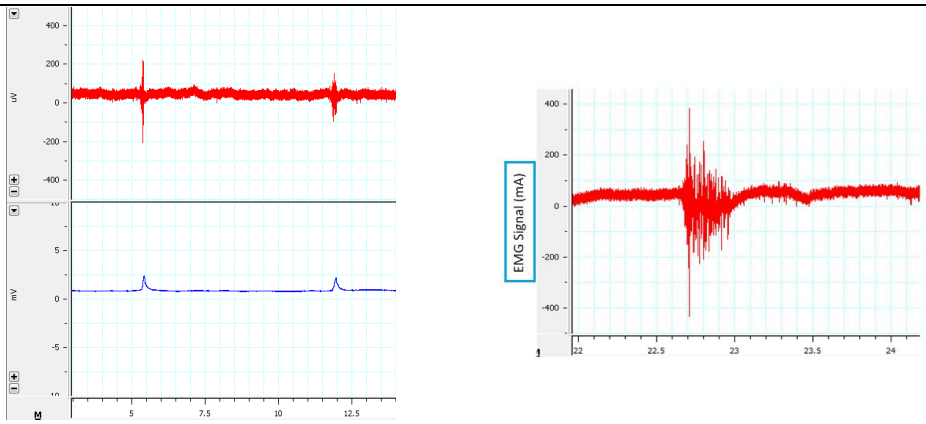

图 5-2 自发眨眼 EMG 与手电眨眼 EMG 信号相比有较大区别

这种波形同手电引发眨眼 EMG 相比（见图 5-2）有两方面非常大的区别：1.波形持续时间要更短，一个手电引发扎眼的 EMG 信号大约持续 0.3-0.5s 的时间，而一个自发眨眼 EMG 信号仅仅持续 0.1-0.3s 的时间。2.波形幅度更小，手电引发眨眼的 EMG 信号幅度在 150-300uV 之间，而自发眨眼 EMG 信号的幅度大约是其一半。

所以要让我们的 FES 系统从 demo 版的只能检测手电引发的大幅度眨眼 EMG 信号进化到能够检测自发眨眼的小幅度 EMG 信号，必须要做出三个方面的改进：

一方面，为了检测这种自发眨眼 EMG 信号，必须要设计更好的眨眼侦测算法，尤其考虑到静息状态下 EMG 信号就已经有了 70-80uV 的幅度，如果自发眨眼 EMG 信号最小情况下只有 100 多 uV，那么设计高准确率的眨眼侦测算法肯定会有相当的难度。

第二方面，很显然，眨眼 EMG 的幅度、持续时间是由我们做出眨眼动作时的肌肉状态决定的。而自发眨眼 EMG 幅度如此小肯定是因为第一、自发眨眼动作的力度很小，第二、自发眨眼动作活动的肌肉同手电引发眨眼时活动的肌肉不同，所以我们应该优化采样 EMG 的位置，来试图采到更大幅度的 EMG 信号。

第三方面，自发眨眼 EMG 信号更小，也意味着伪迹消除算法需要有更好的表现，才能将伪迹信号隔离在 EMG 信号之外。

5.2.2 从眨眼动作到更丰富的眼部动作

虽然我们的 FES 系统目前仅仅实现的是面部偏瘫患者两侧同步眨眼的功能，但长久的说，应该设计一个能够恢复病人瘫痪眼睑闭合能力的系统。

我认为，闭眼能力的恢复应该分成两个方面来看：

短期闭眼动作、以及长期闭眼动作：

短期闭眼动作指的是几秒钟的持续闭眼动作，这样的动作不仅有助于保护眼睛、而且丰富了面瘫病人的面部表情、刺激肌肉以防止瘫痪肌肉因为长时间没有运动而萎缩。短期闭眼动作恢复又分两个方面，第一方面是健康眼睑短期眨眼动作的检测，这方面我们已经做过了实验，在实验中我们用手电筒光线照射兔子健康侧的眼睛、并持续约数秒的时间，所采得到的 EMG 波形如图所示。可见在眼睑闭合的这段时间里，EMG 信号持续在一个较大的幅度，这就如同一个眨眼 EMG 信号从 0.3 秒被拉宽到了 6,7 秒一样，所以按照原来的检测算法，也可以检测出短期闭眼的动作。另一方面是引发短期闭眼动作的 FES 刺激，根据与我们合作的九院邓博士的建议，可以通过高频（10-50Hz）的眨眼刺激来实现短期闭眼刺激。短期闭眼中最重要的问题应该是刺激伪迹，因为刺激的最高频率是由伪迹排除算法中的参数 HT 来决定的，目前我们的 HT 为 0.5s，能达到的最高刺激频率为 2Hz，如果要达到 10Hz 的刺激频率，需要提高伪迹排除算法的性能，将 HT 减小为 0.1s 以下，甚至直接排除刺激伪迹的干扰。

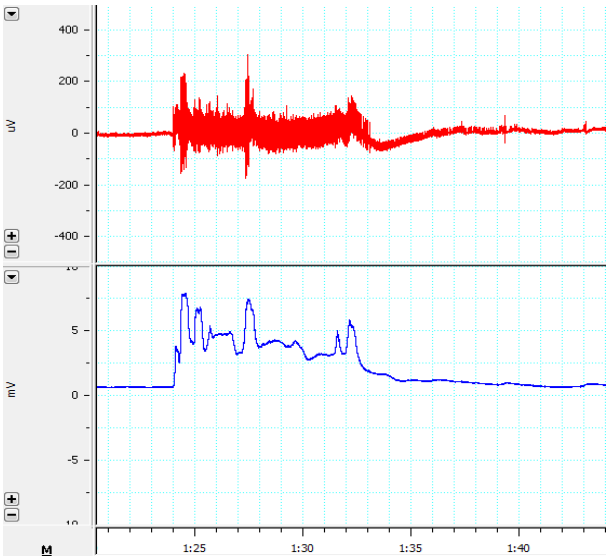

图 5-3 短期持续闭眼的 EMG 波形

长期闭眼动作，特别指的是我们在夜晚睡觉的时候，如果健康侧的眼睛长达 7，8 个小时地闭合，瘫痪眼睛该如何处理？我首先想到的是像短期闭眼 FES 一样，不停用电刺激使其闭合，但长时间不停息地使用电刺激很可能会带来疼痛、甚至损伤组织。而且，我们睡觉时闭眼与平时的闭眼有本质上的区别，睡觉闭眼时，我们眼部的肌肉是松弛不用力的。所以，我认为长时间的闭眼不应该通过 FES 达到，而应通过某种机械的方式，比如用线将眼睑固定在一起。

5.2.3 从兔子到人体

如果我们的 FES 系统要应用实际的人体上，有两件事是必须做得，第一件是优化 FES 刺激方式，第二件是改变 EMG 处理方式。

优化 FES 的刺激方式：我们目前使用的 FES 刺激，波形为 50Hz,1mA 的双相脉冲串，但是据我们的观察，在动物试验中对兔子施加这种刺激时，兔子的眼睑闭合非常用力，甚至连耳朵也会随着 FES 刺激而抽搐，这样的刺激如果施加到人体上肯定会引起人体的不适。
刺激方案的优化目标，应是用尽可能小的刺激电流来获得自然的眨眼动作。为了达到这个目标，我阅读了一些现存的关于眼部 FES 的研究论文。

眼部 FES 的研究是从 1977 年开始的，最早的一些研究，都旨在验证通过眼部 FES 引发人工眨眼动作的可行性，这些研究在兔子、狗等动物上做实验，但并没有优化它们的刺激方案。

在 2007 年发表的研究[24]，第一次对 FES 眼部刺激的效果做了量化的、较详尽的研究，它考察了刺激通道数（stimulation channel）、脉冲串数量、刺激幅度等因素对刺激效果的影响，该研究指出通过使用多通道刺激、脉冲串刺激的方式，可以再较小的刺激电流下达到较大的动作幅度（见图 5-4）。

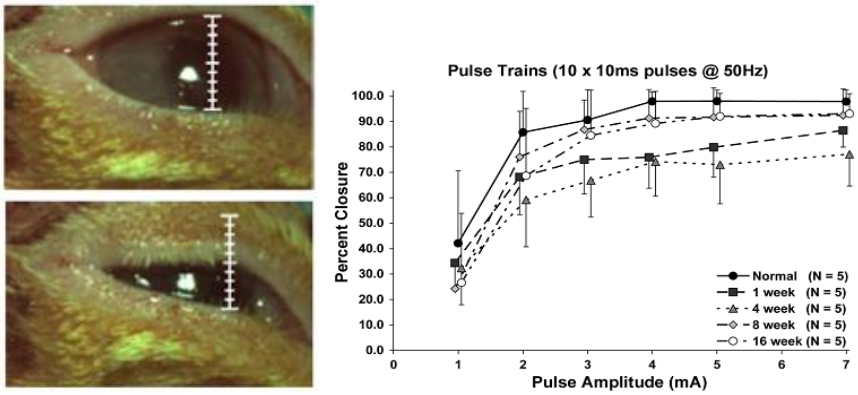

图 5-4 2007 年，对引发眨眼的 FES 刺激方式的量化研究

另外一篇对我们优化电刺激方式有参考价值的文章，是发表在 2009 年的[25]一文。截止到这篇文章为止，眼部 FES 已经在动物上做过好几次实验，可行性以及刺激方案也有了充分的学习，而这篇文章是第一次在实际的面瘫患者身上做 FES 恢复眨眼功能的研究。这篇文章考察了不同的刺激方式，包括表面刺激、interferential 刺激等，衡量这些刺激方式的指标包括两个：一个是病人的疼痛级别、另一个是眼睑的闭合程度，并且在文章最后，他们指出 interferential 的刺激方式可以在引起较小疼痛感觉的情况下达到更好的刺激效果。

改变 EMG 处理方式：随着 EMG 信号采样方式的改变、采样对象的改变，眨眼动作所产生的 EMG 信号特征肯定会有变化，同时，刺激所产生的刺激伪迹信号也会有变化，这两种信号的变化可能发生在幅度、频带上，但我认为总的来说，他们的形状不会有大的改变，所以，适当改进当前在兔子实验中使用的眨眼判断算法，就能够移植的人体上使用。

5.2.4 从实验样本到自适应系统

要将我们目前的 demo 发展成为一个能够使用在不同面瘫患者身上的产品，还必须要处理不同使用者的差别问题。

我们在动物实验中就已经遇到了这一问题，在我们前后的若干次实验中，我们发现最开始都要重新调整上次实验中已经调整到最佳的一些参数，比如模拟放大器的放大倍数、眨眼判断算法中的几项变量等等。这是因为，每次实验我们使用了不同的兔子、并且电极植入位置、深浅也有变化，这就导致了 EMG 信号的变化。

可以预见，我们的系统如果要用在真正的病人身上，由于每个病人皮肤的特性不同，而且佩戴仪器的方式不同，电信号肯定也会有区别。要解决这个问题，像动物实验一样每次都调整系统，显然是不现实的。最优可能的解决方法，应该是把我们系统中的一些部分改成具有自适应特性的，比如眨眼判断算法。事实上，在 2009 年的一篇研究中[3]，已经有了这方面的尝试。这篇文章同样设计了一个用于恢复偏瘫患者眨眼功能的闭环 FES 系统，但是他们的研究中没有将刺激电流真正地传入瘫痪眼睑中，而是传入了一个外接的电阻。他们系统中的眨眼判断算法就有自适应的功能，在每次系统开始工作前的一段时间里，算法会采样 EMG 信号，判断其幅值、并针对它修改自己的几项系数。

我们的系统可以借鉴这次研究的思路，设计自己的自适应系统。

5.2.5 从板级 demo 到便携仪器

一个眨眼恢复器产品肯定是便携的，面瘫患者不需要长期地卧床休养，而是要把仪器带在身边，能够随时随地地工作。

要把我们的 FES 系统做成便携式的：从电路上来说，可能要把 EMG 采集电路和 FES 刺激器都做成集成芯片，运行眨眼侦测算法以及伪迹排除算法的 MCU 用什么来代替，是用数字电

路、还是用现成的 IP 核，这要视成本而定。从外型上来说，可能系统可以做成类似眼睛的型状（如图 5-5），这样比较美观。

图 5-5 便携的 FES 刺激系统

我认为，系统便携化后肯定会遇到一个比较重要的问题，就是运动伪迹。就是说，采样电极随着病人的行动会和皮肤、组织有相对的运动，这样的运动就会造成 EMG 信号中混入运动伪迹信号，需要另外学习运动伪迹的特性，以及如何排除它的方法。

5.3 总结

本研究学习了在 FES 闭环系统中出现的刺激伪迹的研究现状以及常用消除方法，并设计了一种基于 IIR 数字滤波和伪迹 Blanking 算法的伪迹消除算法，通过该算法，一个用于治疗面部偏瘫、恢复眨眼功能的闭环 FES 系统成功地抑制了刺激伪迹的干扰。

文章介绍了 FES 的背景、刺激伪迹的背景、以及现有的三种排除伪迹方法：伪迹 Blanking、伪迹减法、以及伪迹滤波的研究状况。

之后文章描述了本研究所针对的一个、没有带伪迹排除功能的用于恢复偏瘫患者眨眼功能的闭环 FES 系统，它的三个部分：EMG 提取器、眨眼侦测器、FES 刺激器，以及这个系统在动物实验中所遇到的刺激伪迹问题。

之后，文章描述了我们所采用的两种伪迹排除技术：Blanking、滤波的原理，我们的设计过程，以及最后的实现方式。毕设的进行分两个阶段完成，第一阶段中，我设计了一个单纯基于 Blanking 算法的伪迹排除方法，然后通过动物实验验证了算法的可行性、并观察了它的效果。第二阶段，我引入了数字滤波的方案，使原来的排除算法变成了一个先滤波再 Blanking 的方法，在引入数字滤波方案时，我学习了数字滤波器相关的一些知识，并通过 Matlab 学习了 FIR/IIR 滤波器的滤波效果，我提出了三种滤波方案：分别是 1 阶巴特沃兹 IIR 高通滤波器、3 阶巴特沃兹 IIR 高通滤波器、以及加汉宁窗的 FIR 高通滤波器，并基于性能、成本的分析选择了 1 阶 IIR 滤波器方案，这个方案被实现在 MCU 中，并通过第二阶段的动物实验验证了可行性。

最后，我整理、回顾了整个毕设过程中我遇到的一些值得思考的问题，然后对系统提出了 7 点发展展望，如果我们这样一个基于 PCB、单片机的 demo 系统最后要发展成一个成熟的产品的话，这 5 点将是必须研究的内容。

参考文献

[1] May M. Gold weight and wire spring implants as alternatives to tarsorrhaphy. Arch Otolaryngol Head Neck Surg 1987; 113:656–660.

[2] McNeill J, Oh Y. An improved palpebral spring for the management of paralytic lagophthalmos. Ophthalmology 1991; 98:715–719.

[3] K. Chen, T. C. Chen, K. Cockerham, and W. Liu, "Closed-loop Eyelid Reanimation System with Real-time Blink Detection and Electrochemical Stimulation for Facial Nerve Paralysis", IEEE International Symposium on Circuits and systems, 2009, May 2009.

[4] P. Prinz, P. M. Zeman, etc; Feature extraction through wavelet de-noising of surface EMG signals for the purpose of mouse click emulation; Canadian Conference on Electrical and Computer Engineering, 2006

[5] Im, J. J, Rho D. H., etc; Extraction of parameters from EMG signals for the biofeedback electrical stimulation, 24th Annual Conference of the Biomedical Engineering Society, 2002.

[6] Seok-pil Lee, Jung-sub Kim, etc.; An enhanced feature extraction algorithm for EMG pattern classification; IEEE Transactions on Rehabilitation Engineering, 1996.

[7] Thierry Keller and Milos R. Popovic; Real-time stimulation artifact removal in EMG signals for neuroprosthesis control applications; IFESS'2000 Conference, 2000.

[8] R. J. Croft and R. J. Barry; Removal of ocular artifact from the EEG: a review; Neurophysiol Clinic, 2000.

[9] Derek T. O'Keeffe, Gerard M. Lyons, etc.; Stimulus artifact removal using a software-based two-stage peak detection algorithm; Journal of Neuroscience Methods, 2001.

[10] Anne E. Hines, Patrick E. Crago, etc.; Stimulus artifact removal in EMG from muscles adjacent to muscles; Journal of Neuroscience Methods, 1996.

[11] Nicholas A. Sachs, Eli L. Chang, etc; Electrical Stimulation of the Paralyzed Orbicularis Oculi in Rabbit; IEEE Transaction on Neural Systems and Rehabilitation Engineering, 2007.

[12] Daniel McDonnall and K. Shane Guillory, etc.; Restoration of blink in facial paralysis patients using FES; IEEE EMBS Conference on Neural Engineering, 2009.

[13] J. A. Freeman, "An electronic stimulus artifact suppressor," EEG and Clinic. Neurophysiol., vol. 31, pp. 170-2, 1971.

[14] J. Minzly, J. Mizrahi, N. Hakim, and A. Liberson, "Stimulus artefact suppressor for EMG recording during FES by a constant-current stimulator," Med. Biol. Eng. Comput, vol. 31, pp. 72-5, 1993.

[15] Mandrile F, Farina D, Pozzo M, Merletti R. "Stimulation artifact in Surface EMG Signal: Effect of the Stimulation Waveform", Detection System, and Current Amplitude Using Hybrid Stimulation Technique

[16] Knaflitz M, Merletti R. "Suppression of Simulation Artifacts from Myoelectric-Evoked Potential Recordings," IEEE Trans Biomed Eng. 1988 Sep;35(9):758-63.

[17] K. C. McGill, K. L. Cummins, L. J. Dorfman, and B. B. Berlizot, "On the nature and elimination of stimulus artifact in nerve signals evoked and recorded using surface electrodes," IEEE Trans. Biomed. Eng., vol. 29, pp. 129-37, 1982.

[18] T. Blogg and W. D. Reid, "A digital technique for stimulus artifact reduction," EEG and

Clinic. Neurophysiol., vol. 76, pp. 557-61, 1990.

[19] T. Wichmann, “A digital averaging method for removal of stimulus artifacts in neurophysiologic experiments,” J. Neurosci. Method., vol. 98, pp. 57-62, 2000.

[20] M. Solomonow, R. Baratta, T. Miwa, H. Shoji, and R. D. Ambrosia, “A technique for recording the EMG of electrically stimulated skeletal muscle,” Orthopedics, vol. 8, pp. 492-5, 1985.

[21] C. M. Epstein, “A simple artifact-rejection preamplifier for clinical neurophysiology,” Am. J. EEG Techn., vol. 35, pp. 64-71, 1995.

[22] F. Del Pozo and J. M. R. Delgado, “Hybrid stimulator for chronic experiments,” IEEE Trans. Biomed. Eng , vol. BME-25, pp. 92-4, 1978.

[23] V. Parsa, P. Parker, and R. Scott, “Convergence characteristics of two algorithms in non-linear stimulus artefact cancellation for electrically evoked potential enhancement,” Med. Biol. Eng. Comput., vol. 36, pp. 202-14, 1998.

[24] N.A. Sachs, E.L. Chang, N. Vyas, B.N. Sorensen, and J.D. Weiland, “Electrical stimulation of the paralyzed orbicularis oculi in rabbit,” IEEE Trans Neural Syst Rehabil Eng, vol. 15, pp. 67-75, 2007.

[25] D. McDonnall, K.S. Guillory, and M.D. Gossman, “Restoration of blink in facial paralysis patients using FES,” Neural Engineering,January 2009, pp. 76-79.

谢辞

在这里我要感谢所有在毕设过程中给予我帮助的老师、学长、同学。

感谢王老师的指导，这次的毕业设计是从大四下学期开始的，做了一个学期的时间。但是我加入 BiCasl 实验室，并在王老师的指导下进行面瘫治疗器的研究，是在 10 年年底的时候就开始的。王老师从来没有在我们具体的项目进行过程中有过太多干涉，按照他的话来说，他更希望提供给我们一个平台、一个方向，而在这个平台上如何表示则是学生的自由。而另一方面，他却在其他各个方面给予我们影响，他对学生演讲、报告机会的重视，举办“生物电子”主题的实验室征文比赛，对技术文档格式的严格要求，等等。对我来说，他提供了一个窗口，透过它，我们可以超越自己当前单调的学生生活、看到更远的职业前景，然后再回过头，对自己当前的工作意义有更深刻的理解。

感谢实验室的学长，易欣、王孟德、张伟峰，在项目进行的过程中给予我的建议，在平时日常琐事上给我的帮助。感谢父母、同学在整个项目的过程中给予我的理解与支持。通过这 1 年多的实验室生活，我已有了不少的收获，但依然觉得自己不够成熟，希望在美国的求学生涯中继续努力！

附录

51 单片机代码

```
#include <reg24le1.h>
#include <stdint.h>
#include "API.h"

#define INTERRUPT_RFIRQ  9
#define INTERRUPT_TOIRQ  3
#define INTERRUPT_TICK   13

#define TX_ADR_WIDTH    5                    // 5 bytes TX(RX) address
width
#define TX_PLOAD_WIDTH  20                   // 20 bytes TX payload
#define    PIN32

#ifdef     PIN32

sbit LED3 = P0^1;                              // 1/0=light/dark
sbit LED2 = P0^2;                              // 1/0=light/dark
sbit LED1 = P0^0;                              // 1/0=light/dark

sbit   SCLK    =P0^3;                          // DA 时钟
sbit   CS      =P1^1;                          // DA 使能
sbit   DIN    =P0^7;                         // DA 数据
sbit    Save_Control   =P1^6;


#define p0dir 0x70   //引角方向寄存器
#define p1dir 0xbD   //引角方向寄存器
#endif
typedef unsigned char uchar;
typedef unsigned char uint;

uint8_t const TX_ADDRESS[TX_ADR_WIDTH]  = {0x34,0x43,0x10,0x10,0x01};
// Define a static TX address

uint8_t data rx_buf[TX_PLOAD_WIDTH];
uint8_t data tx_buf[TX_PLOAD_WIDTH];

uint8_t bdata sta;
sbit   RX_DR  =sta^6;          //       有新数据进入时产生中断写 1 清中
```

```
断
sbit   TX_DS  =sta^5;          //      当发送完一数据产生中断
sbit   MAX_RT =sta^4;          //

uint8_t J1,J2,J3,J4;
uint16_t num=0;
uint8_t flag=0;


//int Voltage[10];                  // 电压
int Volthreshold_pos;
int Volthreshold_neg;
int counter=0;
int sample=0;
int ready=0;

int Filt_out;
int X_in[2];
int Y_out[2];
unsigned char j=0;

unsigned char initial=1;                          // 是否为初始化
unsigned char n=0;                              // Voltage Counter
unsigned char m=0;
unsigned char fire=0;

/*************************************************/
void init_Timer ( void )
{
    TR1 = 0 ;          // stop timer 1
    TMOD = 0x 10 ;      // timer 1 works in mode 2: reload
    TL1 = 0x 7a ;     // 1 time unit overflow
    TH1 = 0x ff ;     // 1 time unit overflow
//  ET1 = 1 ;
    TR1 = 1 ;
}

void delay ( unsigned int x )
{
    unsigned int I , j ;
    I = 0 ;
    For ( i=0; i<x; I ++ )
    {
       J = 108 ;
```

```
            ;
        While ( j-- ) ;
    }
}

/*void delay ( unsigned int x )
{
    unsigned int i;
    init_Timer ( ) ;
    I = 0 ;
    For ( I = 0; I < x; I ++ )
    {
        while(! TF1 ) ;
        TL1 = 0x 7a ;
        TH1 = 0x ff ;
        TF1 = 0 ;
    }
    TR1 = 0 ;
    Return ;
}   */

void delay1 ( unsigned int x )
{
    unsigned int i,j;
    i=0;
    for ( I = 0 ; I < x ; I ++ )
    {
        J = 5 ;
            ;
        While ( j -- ) ;
    }
}

/*************************************************/
uint8_t SPI_RW ( uint8_t value )
//这个是一个可以把数据通过 SPI 写入一个寄存器的小子程序
{
  SPIRDAT = value;

  while( ! ( SPIRSTAT & 0x02 ) ) ;  // wait for byte transfer finished

  return SPIRDAT;              // return SPI read value
}
/*************************************************/
```

```
uint8_t SPI_RW_Reg ( uint8_t reg, uint8_t value )
//把特定的数据写入一个特定的寄存器。且写两次数据就是写入一个特定地方
{                                                                    //如
果是写一次寄存器在写一次 0 的话就是读取寄存器
    uint8_t status;

    RFCSN = 0;                          // CSN low, init SPI transaction
    status = SPI_RW ( reg ) ;              // select register
    SPI_RW ( value ) ;                     // ..and write value to it..
    RFCSN = 1;                          // CSN high again

    Return ( status ) ;                    //  return  nRF24L01  status
byte
}
/*************************************************/
uint8_t SPI_Read ( uint8_t reg )
{

    uint8_t reg_val;                //把数据通过 spi 读出来
    RFCSN = 0;                          //  CSN  low,  initialize  SPI
communication...
    SPI_RW ( reg ) ;                       // Select  register  to  read
from..
    reg_val = SPI_RW ( 0 ) ;               // ..then read registervalue
    RFCSN = 1;                          //  CSN  high,  terminate  SPI
communication

    Return ( reg_val ) ;                   // return register value

}
/*************************************************/
//读取一串数据
uint8_t SPI_Read_Buf (uint8_t reg, uint8_t *pBuf, uint8_t bytes)
{
    uint8_t status,byte_ctr;

    RFCSN = 0;                          //  Set  CSN  low,  init  SPI
tranaction
    status = SPI_RW ( reg ) ;              // Select  register  to  write
to and read status byte//根据之前的规则先要写一次数据选寄存器

    for ( byte_ctr = 0 ; byte_ctr < bytes ; byte_ctr ++ )
        pBuf [ byte_ctr ] = SPI_RW ( 0 ) ;     // Perform SPI_RW to read
byte from nRF24L01
```

```
    RFCSN = 1;                              // Set CSN high again

    Return ( status ) ;                          // return nRF24L01 status
byte
}
/**************************************************/
//写入一串数据
uint8_t SPI_Write_Buf ( uint8_t reg , uint8_t *pBuf , uint8_t bytes )
{
    uint8_t status , byte_ctr ;

    RFCSN = 0 ;                            //  Set  CSN  low,  init  SPI
tranaction
    status = SPI_RW ( reg ) ;              // Select register to write
to and read status byte
    for ( byte_ctr = 0 ; byte_ctr < bytes ; byte_ctr ++ ) // then write
all byte in buffer ( * pBuf )
        SPI_RW ( * pBuf++ ) ;
    RFCSN = 1;                             // Set CSN high again
    return(status);                        // return nRF24L01 status byte
}

/**************************************************/
void RX_Mode ( void )
{
    RFCE = 0 ;
    SPI_RW_Reg ( WRITE_REG + CONFIG, 0x0f ) ;           // Set PWR_UP
bit, enable CRC (2 bytes) & Prim:RX. RX_DR enabled..
    RFCE = 1;                                    // Set CE pin high to
enable RX device
}
/**************************************************/
void TX_Mode ( void )
{
    RFCE = 0 ;
    SPI_RW_Reg ( WRITE_REG + CONFIG, 0x0e ) ;              //     Set
PWR_UP bit, enable CRC ( 2 bytes ) & Prim:TX. MAX_RT & TX_DS enabled..
    SPI_Write_Buf ( WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH ) ;  // Writes
data to TX payload
    RFCE = 1 ;
}
/**************************************************/
void rf_init ( void )
```

```
{
    SPI_Write_Buf ( WRITE_REG + TX_ADDR , TX_ADDRESS, TX_ADR_WIDTH ) ;
    // Writes TX_Address to nRF24L01
    SPI_Write_Buf ( WRITE_REG + RX_ADDR_P0 , TX_ADDRESS, TX_ADR_WIDTH ) ;
    // RX_Addr0 same as TX_Adr for Auto.Ack

    SPI_RW_Reg ( WRITE_REG + EN_AA, 0x01);        //             Enable
Auto.Ack:Pipe0
    SPI_RW_Reg ( WRITE_REG + EN_RXADDR, 0x01);    // Enable Pipe0
    SPI_RW_Reg ( WRITE_REG + SETUP_RETR, 0x1a);   // 500us + 86us, 10
retrans...
    SPI_RW_Reg(WRITE_REG + RF_CH, 40);            // Select RF channel
40
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);       //       TX_PWR:0dBm,
Datarate:1Mbps, LNA:HCURR
    SPI_RW_Reg(WRITE_REG + RX_PW_P0, TX_PLOAD_WIDTH);    // Select same
RX payload width as TX Payload width
/*
    SPI_Write_Buf(WRITE_REG  +  TX_ADDR,  TX_ADDRESS,  TX_ADR_WIDTH);
    // Writes TX_Address to nRF24L01
    SPI_Write_Buf(WRITE_REG  +  RX_ADDR_P0,  TX_ADDRESS,  TX_ADR_WIDTH);
    // RX_Addr0 same as TX_Adr for Auto.Ack

    SPI_RW_Reg(WRITE_REG + EN_AA, 0x01);          //             Enable
Auto.Ack:Pipe0
    SPI_RW_Reg(WRITE_REG + EN_RXADDR, 0x01);      // Enable Pipe0
    SPI_RW_Reg(WRITE_REG + SETUP_RETR, 0x1a);     // 500us + 86us, 10
retrans...
    SPI_RW_Reg(WRITE_REG + RF_CH, 40);            // Select RF channel
40
    SPI_RW_Reg(WRITE_REG + RF_SETUP, 0x07);       //       TX_PWR:0dBm,
Datarate:1Mbps, LNA:HCURR
    SPI_RW_Reg(WRITE_REG + RX_PW_P0, TX_PLOAD_WIDTH);    // Select same
RX payload width as TX Payload width

*/
}
/**************************************************/     //读取传输的
主要数据
void RF_IRQ(void) interrupt INTERRUPT_RFIRQ
{
    sta=SPI_Read(STATUS);                         // read register
STATUS's value     //这里非常重要就是从这里把 STATUS 的值传到了 sta
```

```
    if(RX_DR)                                   //  if  receive  data
ready  (RX_DR)  interrupt
    {
        SPI_Read_Buf(RD_RX_PLOAD,rx_buf,TX_PLOAD_WIDTH);// read receive
payload from RX_FIFO buffer
        SPI_RW_Reg(FLUSH_RX,0);                          //接收完数据就把
接收寄存器清理
    }

    if(MAX_RT)
        SPI_RW_Reg(FLUSH_TX,0);
    if(TX_DS)
        SPI_RW_Reg(FLUSH_TX,0);

    SPI_RW_Reg(WRITE_REG+STATUS,0x70); // clear RX_DR or TX_DS or MAX_RT
interrupt flag//这个命令是用来写状态和命令寄存器专用，并且写 1 来清中断
}
/*****************************************************/
void uart_init(void)
{
//  ES0 = 0;                            // Disable UART0 interrupt while
initializing         //关闭了串口中断
    REN0 = 1;                           // Enable receiver


    SM0 = 0;                            // Mode 1..
              //SM0 与 SM1 来确定是 10 位异步收发，波特率由定时器控制
    SM1 = 1;                        // ..8 bit variable baud rate

    PCON |= 0x80;                       // SMOD = 1
              //PCON 的 SMOD 被置 1，所有工作方式的波特率加倍
    WDCON |= 0x80;                     // Select internal baud rate generator

    SORELL = 0xe6;                      // 19.2K(998)  0x03e6
                    //设置波特率
    SORELH = 0x03;
    TI0 = 0;
   RI0 = 0;
   TR1 = 1;


}
```

```
/*****************************************************/


void uart_putchar(uint8_t x)
{

    SOBUF=x;
   while (!TI0);
   TI0=0;


}
/*****************************************************/
uint SPI_DA(uint uchar)
{
   uint bit_ctr;
   for(bit_ctr=0;bit_ctr<8;bit_ctr++)                              //
output 8-bit
   {

       DIN = (uchar & 0x80);
       delay1(5);                                    // output 'uchar',
MSB to MOSI
       uchar = (uchar << 1);                                  // shift
next bit into MSB..
       SCLK = 1;                                                // Set
SCK high..
        delay(1);                                  // delay   ?1000
       SCLK = 0;
       delay(1);                                          //    ..then
set SCK low again
   }
    return(uchar);                                            //
return read uchar
}
/*****************************************************/
void CheckButtons(uint8_t k1,uint8_t k2,uint8_t k3,uint8_t k4)

{


       tx_buf[0]=k1;
       tx_buf[1]=k2;
       tx_buf[2]=k3;
```

```
        tx_buf[3]=k4;
        TX_Mode();                  // set TX Mode and transmitting
        while (!(TX_DS|MAX_RT)); //要么发送完毕要么就是重发的次数超过
了预定的次数
        if (TX_DS)
        {
            LED1=LED2=LED3=1;     // turn off LED
        }
        sta = 0;
        delay(300);
        RX_Mode();

}

/*************************************************
void T0_time( ) interrupt INTERRUPT_T0IRQ
{
//   uint16_t num;
//   num=0;
     TH0=(65536-400)/256;
     TL0=(65536-400)%256;
     num++;
     if(num==10000)
        {
            num=0;
           flag=1;
        }
}

/*************************************************/

void adc_init(void)
{
    ADCCON1 = 0x11;                                  // 选择 AIN7，参考电压为
VDD
    ADCCON2 = 0x02;                                  // 单端输入，单步模式
    ADCCON3 = 0x40;                                  // 采样数据 8bit,左对齐
}

void rtc2_init(void)
{
    CLKLFCTRL=0x01;                              // 使用 32K RC 时钟
    RTC2CMP0=0xA3;                               // 100hz 频率
    RTC2CMP1=0x00;
```

```
    RTC2CON=0x07;                          // 比较模式，启动 RTC2
    WUIRQ=1;                               // 使能 TICK 中断
}

void adc_convert(void)
{
    ADCCON1 = ADCCON1 | 0x80;                          // 启动 ADC
    while ((ADCCON1 && 0x40) == 0x40);                 // 等待转换结束
}

void voltage_swap ( void )
{
/* unsigned int m = 0 ;
    for ( m = 1 ; m < 10 ; m ++ )
    {
        Voltage [ m - 1 ] = voltage [ m ] ;
    }
    Voltage [ 9 ] = ADCDATH ;
    VolMax = voltage [ 0 ] ;
    VolMin = voltage [ 0 ] ;
    for (n = 1 ; n < 10 ; n ++ )
    {
    if ( voltage [ n ] > VolMax )
    VolMax = voltage [ n ] ;
    if ( voltage [ n ] < VolMin )
    VolMin = voltage [ n ] ;
*/
    For ( j = 1 ; j > 0 ; j -- )
    {
        X_in [ j ] = X_in [ j - 1 ] ;
        Y_out [ j ] = Y_out [ j - 1] ;
    }
    X_in [ 0 ] = ADCDATH ;
    Filt_out = 78 * X_in [ 0 ] - 78 * X_in [ 1 ] + 55 * Y_out [ 1 ] ;

    Y_out [ 0 ] = Filt_out / 100 ;

    If ( Y_out [ 0 ] > Volthreshold_pos )
        Counter ++ ;
}

void RTC2_IRQ ( void ) interrupt INTERRUPT_TICK
{
        adc_convert ( ) ;
```

```
        sample ++ ;
        if ( sample < 25 )
          voltage_swap ( ) ;
        else
          {
             Ready = 1 ;
            Sample = 0 ;
          }
}

void DAC_initial ( void )
{
    CS = 0 ;
    Delay ( 1 ) ;
    SPI_DA ( 0x ff ) ;
    SPI_DA ( 0x ff ) ;
    CS = 1 ;
    Delay ( 1 ) ;
    CS = 0 ;
    SPI_DA ( 0x c7 ) ;
    SPI_DA ( 0x E4 ) ;
    CS = 1 ;
}

void main ( void )
{
//  uint8_t i,j;

    P0DIR = p0dir;                                  // Output: P0.0 -
P0.2, Input: P0.3 - P0.5
    P1DIR = p1dir;                                  // Output: P0.0 -
P0.2, Input: P0.3 - P0.5
    P2DIR = 0x00;

    P0CON = 0x00;                                 // All general I/O
    P1CON = 0x00;                                    //  All  general
I/O
    P2CON = 0x00;                                 // All general I/O

    for (j=0;j<2;j++)
    {
    X_in [ j ] = 0 ;
    Y_out [ j ] = 0 ;
    }
```

```
    LED1 = LED2 = LED3 = 0 ;        // turn on LED
    Delay ( 1000 ) ;
    LED1 = LED2 = LED3 = 1 ;    // turn off LED
                            // Radio + SPI setup
    RFCE = 0;                                   // Radio chip enable low
    RFCKEN = 1;                                  // Radio clk enable
    RF = 1;                                      // Radio IRQ enable
    Save_Control = 0 ;
    uart_init ( ) ;
//初始化串口
    rf_init ( ) ;

    Volthreshold_pos = 0x 07 ;

//  TMOD = 0x 01 ;
//  TH0 = (65536-400) / 256 ;
//  TL0 = (65536-400) % 256 ;
                    //   初始化 24L01

    EA = 1 ;                                        // Enable global IRQ
     RX_Mode ( ) ;
    adc_init ( ) ;
    rtc2_init ( ) ;
    DAC_initial ( ) ;

//  ET0 = 0 ;
//  TR0 = 0 ;

    While ( 1 )
    {
            while( ! ready )
           ;

              If ( counter > = 5 )
              {
//                if ( fire == 0 )
//                {
                    LED1 = ~LED1;
                //  CS =~CS ;
                //  DIN =~DIN;
                //  SCLK =~SCLK;
                    for ( m = 0 ; m < 5 ; m ++ )
```

```
                    {
                        CS = 0 ;
                        Delay ( 1 ) ;
                            SPI_DA ( 0x cf ) ;
                        SPI_DA ( 0x fc ) ;
                        CS = 1 ;
                        Delay ( 0x 1e ) ;
                        CS = 0 ;
                        SPI_DA ( 0x c8 ) ;
                        SPI_DA ( 0x 00 ) ;
                        CS = 1 ;
                        Delay ( 0x 32 ) ;
                        CS = 0 ;
                        SPI_DA ( 0x c3 ) ;
                        SPI_DA ( 0x fc ) ;
                        CS = 1 ;
                        Delay ( 0x 78 ) ;
                    }
                    CS = 0 ;
                    SPI_DA ( 0x c7 ) ;
                    SPI_DA ( 0x E4 ) ;
                    CS = 1 ;
                    // fire = 1 ;
                     Delay ( 5000 ) ;
                    Ready = 0 ;
                    Counter = 0 ;
//                  }

//              else
//              {
//                  fire = 0 ;

//              }
    }

      /*  if ( RX_DR )                          // finish received
        {
            LED2 = ~ LED2 ;
            high_time = rx_buf [ 2 ] * 10 ;
            gap_time = rx_buf [ 3 ] * 10 ;
            low_time = rx_buf [ 6 ] * 10 ;


          */
```

```
/*          for ( I = 0 , I < 10 , I ++ )
            {

                Sta = 0 ;
                CS = 0 ;
                 Delay ( 1 ) ;
                  SPI_DA ( rx_buf [ j ++ ] ) ;
//                SPI_DA ( rx_buf [ j+1 ] ) ;
                CS = 1 ;
                Delay ( 1 ) ;
//              j+ = 2 ;
            }
*/
/*              sta = 0 ;
                CS = 0 ;
                 Delay ( 1 ) ;
                  SPI_DA ( rx_buf [0] ) ;
                  SPI_DA ( rx_buf [1] ) ;
                CS = 1 ;
                Delay ( 1 ) ;
                ET0 = 1 ;
                TR0 = 1 ;
                While ( flag == 0 ) ;
                ET0 = 0 ;
                TR0 = 0 ;
                CS = 0 ;
                 Delay ( 1 ) ;
                  SPI_DA ( rx_buf [ 2 ] ) ;
                  SPI_DA ( rx_buf [ 3 ] ) ;
                CS = 1 ;
                Delay ( 1 ) ;
                ET0 = 1 ;
                TR0 = 1 ;
                While ( flag == 0 ) ;
*/
            // sta=0;

                /*while ( ! RX_DR )
                {
                    CS = 0 ;
                    Delay ( 1 ) ;
                    SPI_DA ( rx_buf [ 0 ] ) ;
                    SPI_DA ( rx_buf [ 1 ] ) ;
                    CS = 1 ;
```

```
                Delay ( high_time ) ;
                CS = 0 ;
                SPI_DA ( 0x c8 ) ;
                SPI_DA ( 0x 00 ) ;
                CS = 1 ;
                Delay ( gap_time ) ;
                CS = 0 ;
                SPI_DA ( rx_buf [ 4 ] ) ;
                SPI_DA ( rx_buf [ 5 ] ) ;
                CS = 1 ;
                Delay ( low_time ) ;
            }   */




/*          if ( rx_buf [ 0 ] == 1 )
                LED1 = 0 ;
            if ( rx_buf [ 0 ] == 2 )
               LED2 = 0 ;
             if ( rx_buf [ 0 ] == 3 )
               LED 3 = 0 ;

            if ( rx_buf [ 1 ] == 1 )
               LED1 = 0 ;
            if ( rx_buf [ 1 ] == 2 )
               LED2 = 0 ;
            if ( rx_buf [ 1 ] == 3 )
               LED3 = 0 ;
        }

           Delay ( 1000 ) ;
           LED1 = LED2 = LED3 = 1 ;      */
//   }
}
}
```

ARTIFACT REMOVAL ALGORITHM ON USING MUSCLE FES FOR TREATING FACIAL PARALYSIS

This research developed a stimulus artifact removal algorithm for a closed-loop FES system, which aimed to restore blink function on facial paralyzed.

Functional-electrical stimulation (FES) is one possible way of restoring blink and other functions in these patients. In previous research, a blink restoration system for uni-lateral facial paralyzed patients is designed. The system achieves restoration of synchronized blink through processing on EMG signal from eyelid of healthy side in real-time and stimulating the paralyzed eyelid.

Such FES system can be divided into 3 blocks.

The first block is an EMG extractor. EMG signal is firstly recorded using electrodes and then amplified and filtered. Since an obvious fluctuation could be observed during a voluntary blink in the frequency band of 200-500Hz. Thus, a filtering circuit consisting of an instrumental amplifier followed by a 4th order butter-worth 200-500Hz band-pass filter was designed for filtering noise and extracting the defining characteristics of blink motion.

The second block is a blink detector. The EMG signal is first digitized using a 10 bit ADC with sampling rate of 2 kHz/s. Then, three kinds of thresholds are applied to the sampled EMG signal one after another. Within a window length of 30 data, once the number of data surpassing threshold 1 in amplitude, which is recorded by the counter Data_counter, exceeds threshold 2, and such window appears three time3 in series, a blink will be confirmed. Then, MCU will trigger one-time stimulation to the paralyzed eyelid by sending a set of parameters to the stimulator.

The third block, stimulator, consists of a 10-bit voltage-output DAC and a current generator, which is based on a voltage-to-current converter implemented by two operational amplifiers and resistors. The stimulation parameters received from MCU are decoded and supplied to the DAC register to generate stimulation patterns. The output of DAC is connected with a stimulus generator to create bipolar charge-balanced constant current pulses.

However, during previous study, once the reanimation system starts to deliver stimulation current to the paralyzed eyelid, the artifact signal generated by stimulation will keep contaminating the EMG signal recorded in the detector. This artifact has amplitude of several hundred uV, decaying over time. Without removing such signal from natural EMG, the detector may take the artifact as a blink signal, and trigger stimulation, which will generate another artifact, resulting in a deadlock where the system keeps stimulating without stopping. In previous studies, several solutions based on artifact blanking and filtering technique have been proposed. Among them, adaptive filtering technique is mainly used in the case of nerve stimulation. Blanking based system usually requires additional circuit. Software based systems need complex computation and some of them are not real-time. To develop a system with real-time artifact removal, a new method with simple algorithm and low hardware cost is much desired.

Algorithms used to remove artifacts from EMG signal can be divided into 3 kinds: artifact blanking, artifact subtraction, and artifact filtering.

Artifact blanking means the EMG recording was stopped for a certain period after every time FES stimulation is triggered. After the holding time, when EMG extraction is stopped, the stimulation artifact will decay to an amplitude so low which will not interfere with the EMG recording. Blanking algorithm can be implemented in software or hardware ways. Hardware used to implement blanking algorithm is called sample-and-hold circuits. Software realizing such blanking algorithm can be implemented in MCU.

Software artifact blanking way is to eliminate the artifact at the lowest cost and simple and feasible solution, but the limitations of this algorithm is that each stimulus after the system is bound to lose a period of EMG information will also be the highest frequency of the system electrical stimulation restricted.

Artifact subtraction, in this method, the stimulus artifact was subtracted in the mixed-signal acquisition to less ideal template to get pure EMG waveforms. This artifact elimination way can also be implemented in the microprocessor, and compared to Blanking algorithm, this algorithm will not lose important information of the EMG. Thus, it does not produce the restriction of the stimulation frequency.

However, the two main issues of using this way to remove artifacts are: how to obtain the ideal way of the stimulus artifact, and will the artifact waveform remain constant during each stimulation. The existing part of the study took into account these problems, the proposed dynamic artifact waveform by using the moving ensemble averaging.

Artifact filtering method filters stimulus artifact in the EMG signal. This method take advantage of the different distribution of the artifact signal and EMG signal in the frequency domain filters to remove artifacts. Artifact filtering technology can be realized by adding additional filters to the signal acquisition circuit implementation, taking into account the artifact frequency distribution and the amplitude is not stable , most systems use the adaptive filter when the filter.

Filtering algorithm can be implemented by way of software, hardware , and have advantages in real-time processing. Nevertheless, due to the artifact and EMG signals in a huge gap between the amplitude and the frequency domain distribution often overlap , the filtering scheme of instability , as well as more complex design costs for these two shortcomings .

At the first phase of my research, I designed an artifact removal algorithm based on artifact blanking method. This algorithm is used in our system for two reasons: effectiveness and simplicity. The EMG detection is stopped for a holding time (HT) after each stimulation until the artifact signal decays to a level low enough not to interfere with the blink detection algorithm. The parameter HT in this algorithm is the major parameter of the whole system. Since its value will directly determine the maximum detectable blink frequency of the healthy eye. Since the detection time, which is of the order of 10ms, is negligible comparing to HT, the maximum detectable blink frequency is 1/HT times/s. Thus, the longer the HT, less superimposed the stimulation artifact can be to the original EMG. But on the other side, lower the maximum blink rate the system can achieve.

At the second phase of my research, I designed an filtering-blanking-combined artifact removal algorithm. I used filtering algorithm in my design because, according to my observation,

the most important part of the stimulus artifact signal is the waving tail, which lasts for about 1s, and contains the main frequency distribution in 50Hz. Because the EMG signal is mainly from 200-500 Hz, if we designed a high-pass filter, which filter the 50Hz signal from EMG, the artifact will be therefore, suppressed. During my design, different kinds of digital filters, IIR and FIR filters was carefully weighted. First I brought up 3 potential solutions for the system: 1st order IIR filter, 3th order IIR filter, and windowed FIR filter. Then, I tested their performance in matlab, and weighted the cost of realizing them in MCU. Finally I chosen the solution of 1st order IIR filter. And such filter is implemented in MCU.

To test my algorithms. Animal experiments were done.

A unilateral facial paralyzed rabbit model was constructed by resecting part of the facial nerve trunk. All the surgeries were accomplished under sterile conditions. The rabbits were purchased from the Animal Center of Chinese Science Academy. The protocol for the use of the rabbits in this study was approved by the Independent Ethics Committee of Shanghai Ninth People's Hospital affiliated to Shanghai Jiao Tong University, School of Medicine.

Eight weeks post the construction of the animal model, a test of the restoration system was implemented. At the beginning of the experiment, the stimulating electrodes were implanted into the orbicularis oculi of the paralyzed side, and the recording electrodes were implanted into the unaffected side. To test the functionality of the closed loop stimulating system, a flashlight was shined to the unaffected side to induce a natural blink. It has been observed that the stimulation system works as expected and synchronized blinks were achieved. The latency between the light-induced blink and the FES-induced blink was not distinguishable through eyes.

To evaluate the effectiveness of the real-time blink restoration, the response of the paralyzed side eyelid to the electrical stimulation was recorded with a high-speed digital video camera. 30 times of stimulated blink were tried with different HT. Each attempt was considered successful if an artificial blink with an over 50% eyelid closure, and an indistinguishable latency to contralateral natural blink, as shown in Fig. 9, was observed. The restoration accuracy is defined as the ratio of the number of successful attempt to the total trials of 30. The result is that the algorithm helps the FES system to achieve the restoration accuracy of 90% at the maximum blink detection rate of 2times/s.