

1. 一种以帧为基本传输单元的无线电通信系统中用以隐蔽接收机语音解码器中拒收帧的方法，该语音解码器是源滤波器式的，由收到的表示在通信信道上逐帧传输的声频信号参数来控制，所述方法的特征在于：

(a) 根据收到的帧中所含的参数是否表示适当的声频信号而接受或拒收所收到的帧；

(b) 检测已接受的帧主要表示语音抑或背景音；

(c) 若上次接受的帧主要表示语音，则根据第一隐蔽算法更新包含在收到的帧中的参数，若上次接受的帧主要含背景音则根据第二隐蔽算法进行更新包含在收到的帧中的参数，而隐蔽拒收帧。

2. 如权利要求1所述的方法，其特征在于，所述第二隐蔽算法包括将滤波参数和增益参数锁定到其上次接受帧的值上。

3. 如权利要求2所述的方法，其特征在于，当第一预定数量的连续帧遭拒收时，所述第二隐蔽算法降低来自所述语音解码器的声频信号的输出电平。

4. 如权利要求3所述的方法，其特征在于，每一次所述第一预定数量的另一些连续帧遭拒收时，所述第二隐蔽算法反复降低来自所述语音解码器的声频信号的输出电平。

5. 如以上任一权利要求所述的方法，其特征在于，所述第一隐蔽算法将上次接受帧的相应参数内插到预定的从第一拒收帧收到的参数中。

6. 如权利要求5所述的方法，其特征在于，所述第一隐蔽算法将来自上一个拒收帧的相应参数内插到从现行拒收帧收到的预定参数中，

并在至少两个连续帧遭拒收时降低来自所述语音解码器的声频信号的输出电平。

7. 如权利要求6 所述的方法，其特征在于，在另一些第二预定数量的连续帧遭拒收时，所述第一隐蔽算法噪声抑制来自所述语音解码器的输出信号。

8. 如以上任一权利要求所述的方法，其特征在于，所传送的参数表示语音时按第一编码算法编码，表示背景音时则按第二编码算法编码。

9. 如以上任一权利要求所述的方法，其特征在于，所收到的参数表示语音时按第一解码算法解码，表示背景音时则按第二解码算法解码。

10. 一种以帧为基本传输单元的无线电通信系统中的设备，用以隐蔽接收机语音解码器中的拒收帧，所述语音解码器是源滤波器式的，且由收到的表示在通信信道上逐帧传输的声频信号参数所控制，所述设备的特征在于：

(a) 帧接受或拒收装置(20, 28)，用以根据收到的帧中所含的各参数是否被认为是表示适当的声频信号而接受或拒收所收到的帧；

(b) 检测装置(34)，用以检测被接受的帧主要表示语音抑或背景音；

(c) 隐蔽装置(32)，用以 在上次被接受的帧主要表示语音时按第一隐蔽算法(框S) 更新拒收帧中包含的参数，在上次被接受的帧主要表示背景音时按第二隐蔽算法(框B) 进行更新拒收帧中的参数而隐蔽拒收帧。

11. 如权利要求10 所述的设备，其特征在于，用以履行所述第二隐蔽算法的装置(32)，通过将滤波参数和增益参数锁定到其在上次被接受帧的值上履行所述第二隐蔽算法。

12. 如权利要求11所述的设备, 其特征在于, 所述用以履行所述第二隐蔽算法的装置在第一预定数量的连续帧遭拒收时降低来自所述语音解码器(40)的声频信号的输出电平。

13. 如权利要求12所述的设备, 其特征在于, 所述用以履行所述第二隐蔽算法的装置在每次所述第一预定数量的另一些连续帧遭拒收时反复降低来自所述语音解码器的声频信号的输出电平。

14. 如权利要求10-13任一项所述的设备, 其特征在于, 用以履行所述第一隐蔽算法的装置(32), 通过将来自上一次被接受的帧的相应参数内插到第一拒收帧中预定收到的参数中而执行所述第一隐蔽算法。

15. 如权利要求14所述的设备, 其特征在于所述用以履行所述第一隐蔽算法的装置将来自上一个拒收帧的相应参数内插到预定的在现行拒收帧中收到的参数中, 且在至少两个连续帧遭拒收时降低来自所述语音解码器的声频信号的输出电平。

16. 如权利要求15所述的设备, 其特征在于, 所述用以履行所述第一隐蔽算法的装置在第二预定数量的另一些连续帧遭拒收时噪声抑制来自所述语音解码器的输出信号。

17. 如以上任一权利要求所述的设备, 其特征在于传送参数编码装置, 该编码装置在所传送的参数表示语音时按第一编码算法对所传输的参数进行编码, 在所传送的参数表示背景音时按第二隐蔽算法进行编码。

18. 如以上任一权利要求所述的设备, 其特征在于解码装置(38, 40), 该解码装置在所收到的参数表示语音时按第一解码算法对所收到的参数进行解码, 在所收到的参数表示背景音时按第二解码算法对所收到的参数进行解码。

拒收帧的隐蔽

技术领域

本发明涉及无线电通信系统中拒收帧的隐蔽, 更具体地说, 涉及改进这类系统中声频信号解码的设备和方法。

本发明的背景技术

众所周知, 无线电通信系统中有许多对语音信号进行编码/解码的方法, 而且甚至是经过标准化了的(例如, 美国的IS-54标准和欧洲的GSM标准)。此外, 瑞典专利申请9300290-5中介绍了各种改进背景音编码/解码、主要供数字蜂窝式电话系统用的方法。这两类方法主要是为处理编码器与解码器之间的联系接近理想的情况而设计的, 其意义在于, 经信道解码之后, 只剩下小量的误码或传输误差。但由于这种联系是通过无线电信道建立的, 因而收到的信号可能会含一些误码或传输误差。此外, 各帧除传输信道低劣之外还会由于其它原因而丢失。例如, 在美国数字蜂窝式通信标准IS-54中, 快速相关控制信道(FACCH)是通过从业务信道窃取语音帧形成的(欧洲GSM技术规范中也有类似的信道存在)。在分组交换网络中当分组(帧)丢失或迟到以致不能用于实时产生语音时(各分组可能在发送机与接收机之间取不同的路由)也会出现类似情况。在所有这些情况下, 可能需要修改上述方法。

本发明的目的是提供将拒收帧的隐蔽原理应用到所接收的信号使声频信号的解码更能抗拒传输误差和失帧的影响或对它们不敏感的装置和方法。

发明概述

按照本发明，上述目的是通过权利要求1 所述的方法实现的。

此外，按照本发明，上述目的还通过权利要求10 所述的设备加以实现。

附图简介

结合附图参阅下面的说明可以最清楚地了解到本发明及其其它目的和优点。附图中：

图1 是装有本发明设备的无线电通信系统接收机各相关部分的原理方框图；

图2 是本发明方法的流程图。

最佳实施例的详细描述

为了了解本发明的工作情况，简单复习一下一般数字蜂窝式无线电联系、一般错帧和失帧的隐蔽技术的工作情况，并复习上述瑞典专利申请的算法，还是有用的。

在数字蜂窝式电话系统中，声频信号通常先经过数字化，然后对其运用语音编码算法(参看例如Lawrence R. Rabiner 在IEEE会刊第82卷第2 期第199-228页上发表的题为“语音处理在电信中的应用”(Application of voice processing to telecommunication)的文章)。该算法将语音信号压缩，再将其转换成一系列经量化的参数(通常是

逐帧方式(fram based manner)进行的)。然后用信道编码技术通过加上编码冗余将得出的二进制位加以保护(参看例如G. C. Clark和J. B. Cain写的《数字通信的纠错编码》(Error correction coding for Digital Communication)一书, Plenum出版社1991年出版)。接着调制得到的二进制位流(参看例如J. G. Proakis写的《数字通信》(Digital Communication)一书, McGraw-Hill出版社, 1989年第二版), 并例如应用TDMA(时分多址存取)技术发送出去。在接收机处, 信号经过解调。可能有的时间或多径分散可用各种均衡技术(例如维特比均衡法或判定反馈均衡法)加以抵消(参看例如上述J. G. Proakis的参考文献)。接着用信道解码(参看例如上述G. C. Clark和J. B. Cain的参考文献)对形成语音解码器再建所传输的语音信号所需用的量化参数的二进制位进行解码。从上面的讨论可知, 传输信道受到干扰或失帧会影响再建的语音信号, 从而降低该信号的质量。

虽然信道编码/解码技术能大大减小对干扰的敏感性, 但在数字蜂窝式系统中只应用信道编码通常是不够的。相反, 通常极其普遍的作法是附加地采用所谓误差隐蔽技术, 以便进一步掩蔽残留在语音解码器输入中的可以感觉得出的误码影响。这些技术都依赖于关于传输信道质量的一些信息, 这些信息可在接收端获得或估算出来。当这类信息表明传输信道的质量差时, 误差隐蔽技术就在语音解码器中引发特定的作用, 其目的在于减小误码对再建语音信号的负面影响。误差隐蔽技术的先进程度取决于关于传输信道质量的信息的特性。现在谈谈获取这种信息的一些方法。

关于信道质量的直接信息可通过测定信号强度获取。低值则表明信噪比低, 即可以预料信道质量差。信道编码技术还有其它方面的先进性。其中一种技术是采用冗余信道编码, 例如循环冗余检验(CRC)(参看例如上述G. C. Clark和J. B. Cain的参考文献), 特别是当用冗余码

进行检错时，更是采用这种编码。此外还可以从卷积解码器(采用卷积码的情况下)、解调器、均衡器和/或分组码解码器获取“软”(非二进制量化)信息(参看例如上述J. G. Proakis的参考文献)。经常采用的一种技术是将来自语音编码器的信息二进制位分成不同的类别，各类别有不同的纠错/检错方案，从而反映出了不同二进制位的不同重要性(参看例如电子工业协会1990年的“TR-45全价(full rate)语音编码解码兼容性标准PN-2972”(IS-54))。因此，信息的各部分连同所加的检错/纠错码可用作可能出现在语音帧中误码的指示。

语音拒收帧的隐蔽

现在简单介绍一下在普通的语音解码器中引用误差隐蔽法掩蔽被认为含有误码的帧的一些技术。当检测出坏帧时，通常采用来自上一个可接受的帧的信息。在坏帧持续若干帧的时间的情况下，这种技术往往与噪声抑制(降低输出电平)结合使用(参看例如电子工业协会1990年的“TR-45全价语音编码解码兼容性标准PN-2972”(IS-54))。这种情况在移动式电话系统中并不罕见，这时若移动速度低，衰减扰动(fading dip)现象就会持续相当长的一段时间。噪声抑制的效果是将干扰掩蔽在重建信号中。特别是避免了响亮的“喀嘞”声。能获取有关所接收的输入二进制位各部分的质量的更详细信息时，就能就可能有的传输误差追寻到语音解码器的某些参数。鉴于所述参数模拟出语音的不同现象，因而可以开发出对各特定参数的物理意义来说最佳的误差隐蔽技术。这方面的具体实例有所谓的音调增益(参看例如T. B. Mindc等人写的“采用长分析帧的低位速率语音编码技术”(Techniques for low bit rate speech coding using long analysis frames)一书，美国明尼阿波利斯ICASSP出版社1993年出版)。在语音

的瞬变时间(transient period)期间, 这个参数有时要求大于1 的值。但这个值相当于不稳定滤波器的模型, 就是说用它有一点危险。具体地说, 应当引用能够每当在参数中检测出可能有的误码时将音调增益限制到小于1 的值的误差隐蔽技术, 这样做比较合适。另一个实例是现代语音编码算法中通常使用的频谱滤波器模型(参看例如上述T. B. Mindc 等人的参考文献)。在该情况下, 误差隐蔽技术可用来在相应的频谱信息中指显示出有误码时避免使用不稳定的滤波器。相反的情况也是能够成立的, 即每当检测出滤波器不稳定时, 可以表明帧是坏的, 因而可以应用误差隐蔽技术。

在例如符合标准IS-54 的美国数字蜂窝式通信系统中, FACCH的帧被窃取时可能会在接收机的语音解码器中引起失帧。语音解码器通过“填入”适当的信息解决这个问题。通常用来自上一帧的相应信息代替失帧。

背景音拒收帧的隐蔽

若解码器对背景音具有所谓反回荡(anti-swirling)作用(这个作用下面将进一步说明), 则若采用上述语音隐蔽方法, 得出的声频信号, 其质量就可能是不可接受的。

如瑞典专利申请9300290-5 中所述的那样, 可按若干方式完成反回荡作用。一个可行的作法是扩大滤波器的带宽。这意味着, 将滤波器的各极移向复平面的原点。另一种可行的方案是低通滤波在瞬时域滤波器的参数。就是说, 通过低通滤波至少某些所述参数可以使滤波参数或其图象随帧的快速变化迅速衰减。这个方法的特殊情况是求出滤波器参数图象(representation)在若干帧的平均值。

因此, 由于背景是按与语音不同的方法编码和/或解码的, 当然语音用的拒收帧隐蔽法用来隐蔽背景音就不能令人满意了。本发明即

通过对语音和背景音采用不同的隐蔽方法解决这个问题的。

考虑到背景信息，现在参照图1说明本发明的内容。图1示出了移动式无线电通信系统中说明本发明需用的各部分。天线收到来自传输信号的信息，在输入线10上将其传送给解调器12。解调器12对收到的信号进行解调，并通过线路14将其传送到均衡器16（例如维特比均衡器），由该均衡器将收到的经解调的信号转换成一股或若干股比特位流，比特位流经线路18传送到第一判定装置20。第一判定装置20确定收到的帧含来自业务信道的二进制位抑或来自快速相关控制信道（FACCH）的二进制位。瑞典专利申请9102611-2介绍了适用的第一判定装置，这里也把该专利申请的内容包括进来，以供参考。若收到的帧含来自业务信道的二进制位，二进制位流就通过线路22传送到信道解码器24。信道解码器24将二进制位流转换成滤波参数流和激励参数流，供语音解码用。另一方面，若收到的帧含来自FACCH的二进制位，二进制位流就不传送给信道解码器24，而是通过线路33通知隐蔽装置32：当前的这个帧不含语音数据。隐蔽装置32可采用微处理器作为状态机，所进行的不同的瞬态过程（transition）将参看图2更详细地加以说明，同时也在本说明书所附附录的PASCAL程序模块中加以说明。

解调器12和均衡器16也分别通过线路50和52将关于收到的二进制位或符号的“软”信息传送给第二判定装置28。上面说过，信道解码器24将二进制位流转换成滤波参数流和激励参数流，供语音解码之用。此外，信道解码器24对各收到的帧的至少若干部分进行循环冗余检验（CRC）解码。这些检验结果通过线路26传送给第二判定装置28。

接收机还装有语音检测器34（也叫做话音活动检测器或VAD）。英国电信PLC公司的专利文献WO 89/08910中介绍了适用的语音检测器。语音检测器34根据所述滤波参数和激励参数确定收到的帧主要含语音抑或含背景音。语音检测器34作出的判断结果经线路36传送给参数调

节器38，供调节收到的滤波参数用(此外还可以在语音检测器34与参数调节器38之间装一个信号鉴别器，确定所收到的表示背景音的信号是否静止)。瑞典专利申请9300290-5中详细介绍了这个调节过程，这里也把该专利申请的内容包括进来以供参考，下面也将讨论这方面。可能经调节的滤波参数和激励参数传送给语音解码器40，解码器40在输出线42上输出声信号。

为了说明本发明的拒收帧隐蔽技术，有必要简单说明一下误码对上述瑞典专利申请9300290-5中所述的所谓反回荡算法的影响。这些影响可以粗略地分为以下几个方面：

1. 用以控制反回荡算法的话音活动或语音检测器34通常是自适应式的(“语音活动检测”建议(“Voice Activity Detection” Recommendation) GSM 06.32, ETSI/GSM, 1991)。这意味着，这里存在在语音活动检测器中采用测出的语音信号或在这里假定应用选用于接收机时采用来自信道解码器经解码的参数而内部自动更新的阈值和相应状态。当输入的参数中有误差时，会产生不恰当更新的阈值或内部状态变量，从而使判断错误。其后果可能是再建的声频信号质量下降。

2. 话音活动或语音检测器34是采用输入的滤波和激励参数同时利用内部更新状态(即旧的输入参数)和先验信息作出其对语音/背景音的判断的。因此误码可能会使接收机中即刻作出错误的判断，从而降低再建声频信号的质量。此外，由于当前的判断也取决于旧的输入参数，因而误码也可能影响未来的判断。

3. 参数调节器38中实际的反回荡作用(主要是频谱低通滤波结合带宽扩展，如上述瑞典专利申请9300290-5中所详述的那样)因误码而受损。其中一个影响是因来自语音或话音活动检测器34(或来自任选的信号鉴别器)的错误判断引起的。在这些情况下，可能开始或停止对低通滤波器的更新过程，从而比起完美信道的情况多了偏差。

另一个影响是出现影响到馈给低通滤波器的频谱信息和带宽扩展的误码时产生的。这两个影响都会使质量下降。

从以上的讨论中可知，拒收帧会导致参数调节器38的不恰当更新。按照本发明，这些问题是通过修正有拒收帧期间的更新过程而减少或解决掉的。现在参照图2更详细地说明这个经修正的更新过程。

在图2的步骤100中，收到了新帧。在步骤102中，判定该帧是否可以接受。这个判定可以由第一判定装置20或第二判定装置28作出，前者拒收不含声频信号的帧，后者拒收含传输误差的声频帧。若经判断：收到的帧是可以接受的，算法就往前进入步骤128，在步骤128中，收到的参数是无需隐蔽误差而使用的。此外使两个超时(即TIMEOUT1和TIMEOUT2)复位。这些超时下面还要进一步说明。

若步骤102判定：收到的信号不能接受，则算法就往前进入步骤104，在步骤104判定上一个(已接受的)帧是否含语音抑或背景音。这一步骤可以由隐蔽装置32执行，因为语音检测器34通过线路48通知隐蔽装置32有关其判定结果。必须指出有一点很重要，判定必须根据上一个(已接受的)帧作出，因为当前的帧已遭拒收。

语 音

若上一个帧在步骤104中经判定为语音帧时，算法就往前进入图2的框S中。在步骤106中，将上一帧中收到的参数内插到某些收到的参数中，即内插到帧能量R0和反射系数中，同时从当前的帧提取剩余的参数。视乎拒收帧中收到的二进制位的质量(如线路26、50、52上的“软”信息所规定的那样)而定，内插过程中在当前帧与上一个帧之间的加权可能有所不同。例如，若现行这个帧确实坏或者业已被“窃取”用作其它用途，则内插对上一个帧产生的加权比现行帧大。另

一方面，一个几乎可以接受的帧在内插过程中会获得更高的加权。1993年12月7日提交的转让给本申请的同一受让人的美国专利申请# 08/162,605介绍了这个过程细节，这里也把该申请的内容包括进来以供参考。EIA/TIA IS-54中第2.2.2.2.3.2节中还介绍了加一种不太复杂的方法。然后在步骤107中用经内插的参数通过线路46控制语音解码器40。

算法往前进入步骤108，在步骤108中收到了一个新帧。步骤110测试该帧是否可以接受。若该帧是可以接受的，算法就往前进入步骤128。若该帧是不可接收的，算法就往前进入步骤112，在步骤112中，上一帧最后计算出的参数都内插到各参数中去。同时，语音解码器的输出电平下降。接着，在步骤114中测试是否超过超时TIMEOUT1。TIMEOUT1一般为120毫秒，这相当于6个帧的时间。若超过TIMEOUT1，在步骤116就噪声抑制来自语音解码器40的输出信号。这之后，算法返回到步骤107。这样，语音的拒收帧隐蔽过程主要包括内插各参数和降低输出电平直到超过超时为止，这之后，输出信号就受到噪声抑制了。

背景音

若在步骤104上一帧中含背景音，算法就往前进入框B。在步骤118，语音解码器40中的滤波系数锁定到其在上一帧的值。这可以通过例如让隐蔽装置32经线路44控制参数调节器38而进行，以保留上一帧的滤波参数。步骤118也将语音解码器的增益参数锁定到其在上一帧的值上。这是由隐蔽装置32通过线路46进行的。增益参数是一些确定选自编码器与解码器的不同代码本的向量之间的混合向量的参数。其余的参数，例如代码本目录、迟延时间等都可直接从现行(拒收的)帧提取。

在步骤119 应用放声用的部分锁定参数之后，算法就往前进入步骤 120，在该步骤收到新帧。步骤122 测试该帧是否可以接受。若帧可以接受，算法就往前进入步骤128。若帧遭拒收，步骤124 就测试是否超过超时TIMEOUT2。TIMEOUT2 一般大约2 秒种，相当于100 个帧的时间。若超过TIMEOUT2，则在步骤126 中输出电平下降。这之后，算法返回步骤119，在步骤119 用预先经锁定和实际收到的(在现行帧中的) 参数组合解码。若未超过TIMEOUT2，算法就返回到步骤119，而不减小输出电平。

框S 与框B 明显的一个区别是TIMEOUT1 比TIMEOUT2 短得多。这样，若连续几个帧都遭拒收，语音信号不久就受到噪声抑制。这是很自然的，因为再也没有其它可靠的语音信息带给收听者。另一方面，背景音的性质就比较稳固定，因此框B 的持续时间可能会久一些而不致对收听者有干扰作用。

更重要的一个不同点在于，框B 中的滤波参数都锁定于其在上一个被接受的帧的值上。鉴于此帧含背景音，因而对该帧施加了反回荡作用。于是，滤波器的带宽扩大了，或者滤波参数(或其表达式) 都经过低通滤波。这样，在某种意义上说，反回荡作用也施加到遭拒收的帧上。

在上述说明中，假设被接受各帧的参数视乎它们是表示语音或背景音而按不同的方式解码。但还有一种方法是用对背景音编码不同的方式在发信机中对语音的各参数进行编码。在这样的实施例中，参数调节器38 可以取消。此外还可以将经修正的编码/ 解码过程分派在发信机与接收机上进行。更详细的情况在上述瑞典专利申请9300290-5 中有介绍。

在一个最佳实施例中，反回荡作用包括求出自相关系数(这些系数是根据收到的反射系数算出的) 和例如上8 个被接受帧的帧能量R0

的平均值。实际滤波系数可以通过在解码器中进行另一次LPC 分析根据这些自相关系数和帧能量RO的平均值计算出来。所有这些反射系数、自相关系数和滤波系数之间的换算都包括在GSM建议6.32 和标准EIA/TIA IS-54中。应用时可以认为这些参数彼此相等。

从步骤128，算法返回到步骤100。这样，若没有帧遭拒收，算法就只在步骤100、102和128三者之间循环。

本说明书所附附录的PASCAL程序模块中详细举例说明了本发明方法的一个最佳实施例。

本技术领域的技术人员都知道，在不脱离本发明在所附权利要求书中所述的精神实质和范围的前提下是可以对本发明进行种种修正和更改的。

附录

```

PROCEDURE FLbadFrameMasking(
    ZFLcrcError      : Boolean;
    ZFLsp            : Boolean;
    ZFLdvccError     : Boolean;
    ZFLfacchCrc      : Boolean;
    ZFLestimatedBer  : Integer;
    ZFLsoftQual      : Integer;
    VAR ZFLbadQuality : Boolean;
    VAR ZFLnoiseShift : Integer;
    VAR ZFLframeData  : integerFrameDataType;
    VAR ZFLsubframeData : integerSubframeDataType);

VAR
    i          : Integer;
    a,b,c      : Integer;
    z          : Integer;
    lowQual    : Boolean;

BEGIN

    IF ZFLfacchCrc THEN
        ZFLcrcError := True;

    IF ZFLdvccError AND ( ZFLsoftQual < 2000 ) THEN
        ZFLcrcError := True;

    IF ZFLsoftQual < lowSoftTreshold THEN
        ZFLcrcError := True;

    lowQual:=False;
    IF ZFLsoftQual < highSoftTreshold THEN
        lowQual := True;
    IF ZFLsoftQual < mediumSoftTreshold THEN
        z := 6
    ELSE z := 13;

    IF lowQual OR ZFLcrcError THEN
        ZFLbadQuality := True
    ELSE
        ZFLbadQuality := False;

    IF ZFLcrcError THEN
    BEGIN
        IF FLcrcErrorState < 1000 THEN
            FLcrcErrorState := FLcrcErrorState + 1;
        CASE FLcrcErrorState OF
            1,2          : ;
            3            : BEGIN
                IF ZFLsp THEN BEGIN
                    IF FLoldFrameData[1] >= 2 THEN
                        FLoldFrameData[1] := FLoldFrameData[1] - 1
                    ELSE
                        FLoldFrameData[1] := 0;
                    END;
                END;
            5            : BEGIN
                IF ZFLsp THEN BEGIN
                    IF FLoldFrameData[1] >= 2 THEN
                        FLoldFrameData[1] := FLoldFrameData[1] - 1
                    ELSE
                        FLoldFrameData[1] := 0;
                    END;
                END;
            7            : BEGIN

```

```

IF ZFLsp THEN BEGIN
  IF FLoldFrameData[1] >= 2 THEN
    FLoldFrameData[1] := FLoldFrameData[1] - 1
  ELSE
    FLoldFrameData[1] := 0;
END;
END;

OTHERWISE
  FLstate6Flag := True;
  IF ZFLsp THEN BEGIN
    IF FLoldFrameData[1] >= 4 THEN
      FLoldFrameData[1] := FLoldFrameData[1] - 1
    ELSE
      FLoldFrameData[1] := 4;

    END ELSE BEGIN
      IF FLCrcErrorState > 50 THEN BEGIN
        IF FLoldFrameData[1] >= 4 THEN
          FLoldFrameData[1] := FLoldFrameData[1] - 1
        ELSE
          FLoldFrameData[1] := 4;

        END;
      END;

    END;
  ZFLframeData := FLoldFrameData;
  a := ZFLsubframeData[1,1] + 19;
  b := FLoldSubframeData[1,1] + 19;

  IF Abs(a - b) < 8 THEN
    c := Round(0.5*(a + b))
  ELSE IF Abs(a - 2*b) < 8 THEN
    c := Round(0.5*(a/2 + b))
  ELSE IF Abs(2*a - b) < 8 THEN
    c := Round(0.5*(2*a + b))
  ELSE
    c := a;
  IF c > 146 THEN
    c := 146
  ELSE IF c < 19 THEN
    c := 19;
  ZFLsubframeData[1,1] := c - 19;

  a := ZFLsubframeData[2,1] + 19;
  b := FLoldSubframeData[2,1] + 19;
  IF Abs(a - b) < 8 THEN
    c := Round(0.5*(a + b))
  ELSE IF Abs(a - 2*b) < 8 THEN
    c := Round(0.5*(a/2 + b))
  ELSE IF Abs(2*a - b) < 8 THEN
    c := Round(0.5*(2*a + b))
  ELSE
    c := a;
  IF c > 146 THEN
    c := 146
  ELSE IF c < 19 THEN
    c := 19;
  ZFLsubframeData[2,1] := c - 19;

  a := ZFLsubframeData[3,1] + 19;
  b := FLoldSubframeData[3,1] + 19;
  IF Abs(a - b) < 8 THEN
    c := Round(0.5*(a + b))
  ELSE IF Abs(a - 2*b) < 8 THEN
    c := Round(0.5*(a/2 + b))

```

```

ELSE IF Abs(2*a - b) < 8 THEN
  c := Round(0.5*(2*a + b))
ELSE
  c := a;
  IF c > 146 THEN
    c := 146
  ELSE IF c < 19 THEN
    c := 19;
  ZFLsubframeData[3,1] := c - 19;

  a := ZFLsubframeData[4,1] + 19;
  b := FLoldSubframeData[4,1] + 19;
  IF Abs(a - b) < 8 THEN
    c := Round(0.5*(a + b))
  ELSE IF Abs(a - 2*b) < 8 THEN
    c := Round(0.5*(a/2 + b))
  ELSE IF Abs(2*a - b) < 8 THEN
    c := Round(0.5*(2*a + b))
  ELSE
    c := a;
    IF c > 146 THEN
      c := 146
    ELSE IF c < 19 THEN
      c := 19;
    ZFLsubframeData[4,1] := c - 19;

END
ELSE
BEGIN
  IF lowqual THEN BEGIN
    ZFlFrameData[1] :=
      Round(((16-z)*FlOldFrameData[1] + z*ZFlFrameData[1])/16);
    ZFlFrameData[2] :=
      Round(((16-z)*FlOldFrameData[2] + z*ZFlFrameData[2])/16);
    ZFlFrameData[3] :=
      Round(((16-z)*FlOldFrameData[3] + z*ZFlFrameData[3])/16);
    ZFlFrameData[4] :=
      Round(((16-z)*FlOldFrameData[4] + z*ZFlFrameData[4])/16);
    ZFlFrameData[5] :=
      Round(((16-z)*FlOldFrameData[5] + z*ZFlFrameData[5])/16);
    ZFlFrameData[6] :=
      Round(((16-z)*FlOldFrameData[6] + z*ZFlFrameData[6])/16);
    ZFlFrameData[7] :=
      Round(((16-z)*FlOldFrameData[7] + z*ZFlFrameData[7])/16);
    ZFlFrameData[8] :=
      Round(((16-z)*FlOldFrameData[8] + z*ZFlFrameData[8])/16);
    ZFlFrameData[9] :=
      Round(((16-z)*FlOldFrameData[9] + z*ZFlFrameData[9])/16);
    ZFlFrameData[10] :=
      Round(((16-z)*FlOldFrameData[10] + z*ZFlFrameData[10])/16);
    ZFlFrameData[11] :=
      Round(((16-z)*FlOldFrameData[11] + z*ZFlFrameData[11])/16);
  END;
  IF NOT FLstate6Flag THEN
    FLCrcErrorState := 0;
    FLstate6Flag := False;
    FLoldFrameData := ZFLframeData;
    FLoldsubframeData := ZFLsubframeData;
  END;

  IF ZFLCrcError THEN BEGIN
    ZFLsubframeData[1,4] := FLoldsubframeData[1,4];
    ZFLsubframeData[2,4] := FLoldsubframeData[2,4];
    ZFLsubframeData[3,4] := FLoldsubframeData[3,4];
    ZFLsubframeData[4,4] := FLoldsubframeData[4,4];
  END;

```

```

END;    { FLbadFrameMasking }

[GLOBAL]

PROCEDURE FLspdFrame(
    FLCrcError           : Boolean;
    FLfacchCrc          : Boolean;
    FLEstimatedBer      : Integer;
    FLsoftQual          : Integer;
    FLframeData         : integerFrameDataType;
    VAR FLsubframeData  : integerSubframeDataType;
    VAR FLbadQuality    : Boolean;
    VAR FLnoiseShift    : Integer;
    VAR FLlrcPres       : realArray10Type;
    VAR FLlaPres        : realArray10Type;
    VAR FLetaCurr       : realArray10Type;
    VAR FLCapRqCurr     : Real;
    VAR FLsp            : Boolean;
    VAR FLaPostPres     : realArray10Type;
    VAR FLmyUse         : Real);

VAR
    i                    : Integer;

BEGIN

    FLbadFrameMasking(
        FLCrcError,      (* IN *)
        FLsp,           (* IN *)
        FLdvccError,    (* IN *)
        FLfacchCrc,     (* IN *)
        FLEstimatedBer, (* IN *)
        FLsoftQual,     (* IN *)
        FLbadQuality,   (* OUT *)
        FLnoiseShift,   (* OUT *)
        FLframeData,    (* IN/OUT *)
        FLsubframeData); (* IN/OUT *)

    FLgetLTPLags(
        FLsubframeData, { IN }
        FLltpLags);     { OUT }

    FLframeDemux(
        FLframeData,    (* IN *)
        FLCapR0,        (* OUT *)
        FLCapLPC );    (* OUT *)

    FLdecodeFrameEnergy(
        FLCapR0,        (* IN *)
        FLCapRqCurr ); (* OUT *)

    FLdecodeLPCcodewords(
        FLCapLPC,      (* IN *)
        FLreflCurr );  (* OUT *)

    FLlrc2acf_special_pas(
        FLCapRqCurr,   { IN }
        nrCoeff,       { IN }
        FLreflCurr,    { IN }
        FLacfVad,      { OUT }
        FLalphaCurr);  { OUT }

    FOR i := 1 TO nrCoeff DO
        FLalphaCurrNy[i] := FLnyWeight[i] * FLalphaCurr[i];

    FLcalculateACF(
        FLalphaCurrNy, (* IN *)
        FLacfW);      (* OUT *)

    FLpostCoeffCalculation(
        FLacfW,        (* IN *)
        FLetaCurr );   (* OUT *)

    IF( NOT (FLCrcError)) AND NOT noSwirling THEN BEGIN

```

```

FLvadStart (          FLacfVad,          { In
                     FLacfOld,          { In/Out
                     FLav0);           { Out
                                     }

FLadaptiveFilterEnergy(  FLrVad,          { In
                       FLacfOld,        { In
                       FLmarginFactor,  { In
                       FLmargin,        { Out
                       FLpVad);         { Out
                                     }

FLacfAveraging(        FLacfOld,          { In
                       FLav0,           { In/Out
                       FLav1,           { Out
                       FLaver);         { Out
                                     }

FLreflFrameCalculation2( FLav1,          { In
                       nrOfAcfLagsUsed, { In
                       FLrcTemp);       { Out
                                     }

FLrc2acf_special_pas(  FLCapRqCurr,    { IN
                       nrOfAcfLagsUsed, { IN
                       FLrcTemp,        { IN
                       FLacfDummy,      { OUT
                       FLalphaTemp);    { OUT
                                     }

FLaav1[0] := 1.0;
FOR i := 1 TO nrOfAcfLagsUsed DO
  FLaav1[i] := FLalphaTemp[i];

FLpredComputation(     FLaav1,          { In
                       FLrav1);         { Out
                                     }

FLspectralComparison(  FLav0,          { In
                       FLrav1,          { In
                       FLdmDiffMinThresh, { In
                       FLdmDiffMaxThresh, { In
                       FLlastDm,         { In/Out
                       FLstat);         { Out
                                     }

FLvadThresh(          FLacfVad,          { In
                     FLrav1,          { In
                     FLstat,          { In
                     FLptch,          { In
                     FLpvad,          { In
                     FLmargin,        { In
                     FLlac,           { In
                     FLadp,           { In
                     FLstatCount,     { In/Out
                     FLadaptCount,    { In/Out
                     FLthvad,         { In/Out
                     FLrvad);         { In/Out
                                     }

FLdecision(           FLpvad,          { In
                     FLthvad,          { In
                     FLburstCount,    { In/Out
                     FLhangCount,     { In/Out
                     FLvad);          { Out
                                     }

FLperiodicityDetection( FLltpLags,        { In
                       FLlthresh,      { In
                       FLnthresh,      { In
                       FLoldLtp,       { In/Out
                       FLoLdLagCount,  { In/Out
                       FLveryOldLagCount, { Out
                       FLptch);        { Out
                                     }

```

```

FLhangHandler(
    maxSpFrames,
    maxSpHlang,
    FLvad,
    FLelapsedFrames,
    FLspHangover,
    FLspeechDtx,
    FLsp);
    { In
      In
      In
      In/Out
      In/Out
      In/Out
      Out }

END;

FLweightinga_in(
    FLalphaCurr,
    FLaPostPres);
    { IN }
    { OUT }

IF (FLsp OR (FLcapR0 <= FLcapR0Thresh) OR noswirling) AND
( FLcrcErrorState < 3 ) THEN
BEGIN
    FLexpandinga (
        FLalphaTemp,
        FLreflCurr,
        FLfirstSp,
        FLaPostPres,
        FLaPres,
        FLrcPres,
        FLmyUse,
        FLfilterFilterState,
        FLfilterPostState);
        { IN }
        { IN }
        { IN }
        { IN }
        { OUT }

END
ELSE
BEGIN
    FLfiltpost (
        FLalphaCurr,
        FLfilterPostState,
        FLmyUse,
        FLaPostPres);
        { IN }
        { IN/OUT }
        { OUT }
        { IN/OUT }

    FLstepdn_unstable_special_pas2 ( FLaPostPres,
        FLrcTemp,
        FLunstable);
        { IN }
        { OUT }
        { OUT }

    FLpresweight (
        FLunstable,
        FLalphaCurr,
        FLaPostPres);
        { IN }
        { IN }
        { OUT }

    FLcalculateACF (
        FLaPostPres,
        FLACFw);
        { IN }
        { OUT }

    FLpostCoeffCalculation (
        FLACFw,
        FLetaCurr);
        { IN }
        { OUT }

    FLreflFrameCalculation2(
        FLaver,
        nrOfAcflagsUsed,
        FLrcPres);
        { In }
        { In }
        { Out }

    FLrc2acf_special_pas(
        FLcapRqCurr,
        nrOfAcflagsUsed,
        FLrcPres,
        FLacfDummy,
        FLaPres);
        { IN }
        { IN }
        { IN }
        { OUT }
        { OUT }

END;

FLfirstsp := (FLsp AND FLfirstsp);

END; (* FLspdFrame *)

```

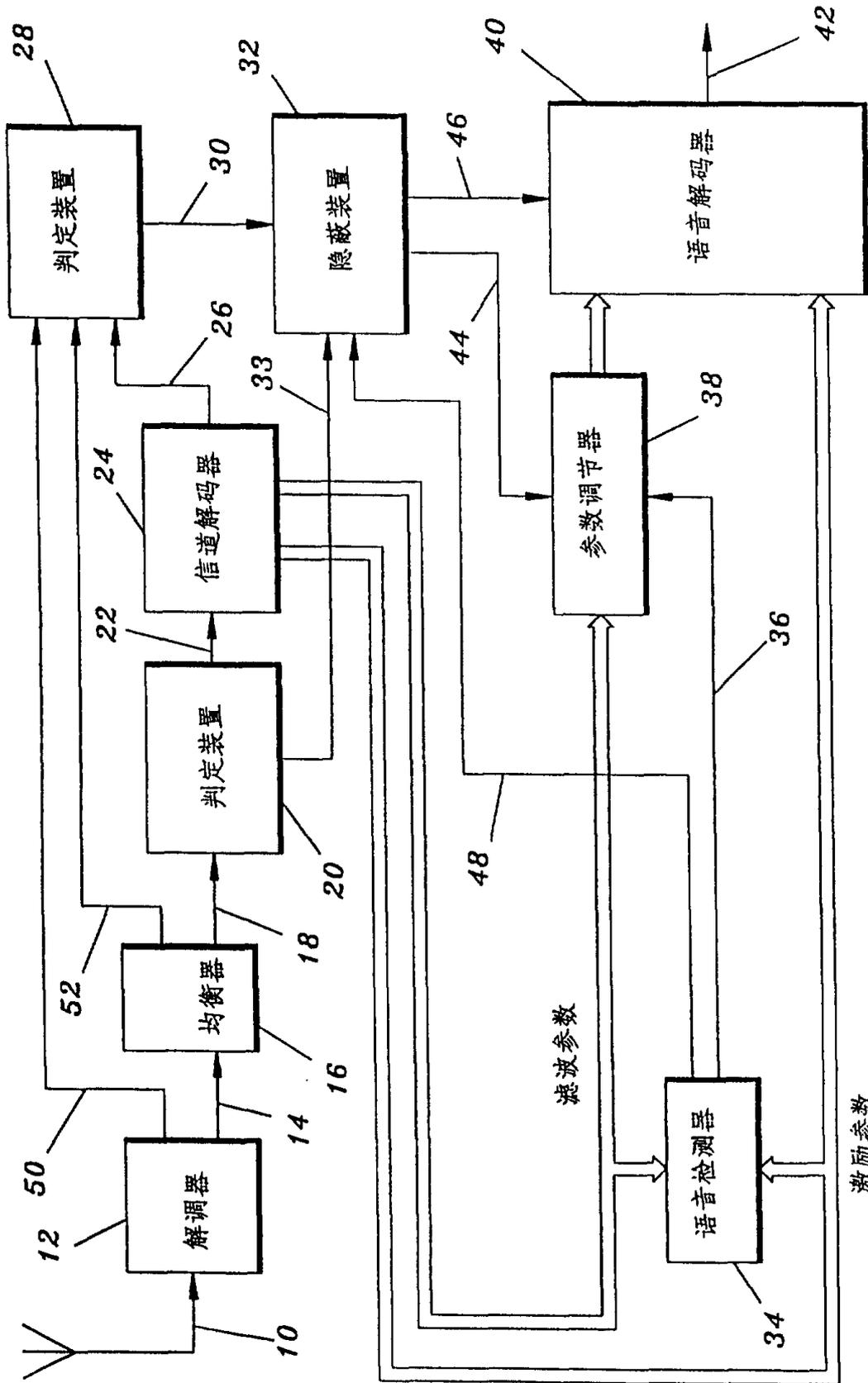


图 1

