基于 CS8955 的 RDS 解码方案

cheuna@126.com

简介

RDS(Radio Data System)数据广播系统是将数字信号调制到调频波段(87.5~108MHz)并以广播的形式发送的一种信号传输系统。这种传输模式在北美和欧洲得到了采用，而且得到越来越多的其他国家的关注。在北美和欧洲 都有相应的标准，即 RBDS 和 RDS。

传输数据时，数据加载到 57KHz 的副载波频段。副载波经过调制以适应固定的双相编码信号。这个副载波经过抑制以避免调制后的数据在锁相环立体声解码器中产生混叠同时还要与德国同样采用 57KHz 频率作为副载波的 ARI 系统保持兼容。发送的信号以组(group)为单位，每组信息包括 4 块(block),每块信息包括 26 位，分别为 16 位有效信息和 10 位校验信息。整个系统包括 32 种不同的组，分别代表不同的信息。在发送过程中，电台根据需要发送不同的组以及不同的组合以达到发送不同信息的目的。然而 PI、PTY 和 TP 信息在每组信息都有包含。表 1 给出了当前已定义的 RDS 特征。

表 1　RDS 特征

标志	英文注解	中文注解
PI	Program identification	节目识别
PTY	Program type	节目类型
PS	Program service name	节目业务名称
RT	Radio text	广播文本
CT	Clock time and date	时间和日期
AF	Alternative frequencies	替换频率表
TA	Traffic announcement	交通公告识别
TP	Traffic program	交通节目识别
MS	Music/speech switch	音乐/语言切换
DI	Decoder identification	解码器识别
PIN	Programme item number	节目栏信息
EON	Enhance other networks	增强的其他网络信息
TDC	Transparent data channel	透明数据通道
INH	In-house data	内部应用
EWS	Emergency warning systems	紧急报警系统
RP	Radio paging	广播寻呼

信号通过解码器产生时钟和数据信号，然后经过微控制器进行相应的处理还原原始信号。很多解码芯片支持这种功能，例如 BU1923、SAA7579、LA2231 等。一个典型的应用方案如图 1 所示。在本系统中主要完成信号的校验和解码功能，信号通过串口在 PC 机上显示，软件为串口调试助手 V2.2，微控制器采用 MYSON 设计的 CS8955 芯片，由于很多解码芯片具有通用性，在此不具体列举解码芯片，解码芯片的时钟引脚和数据输出引脚分别连接到 CS8955 的 INT0 和 I/O 口。

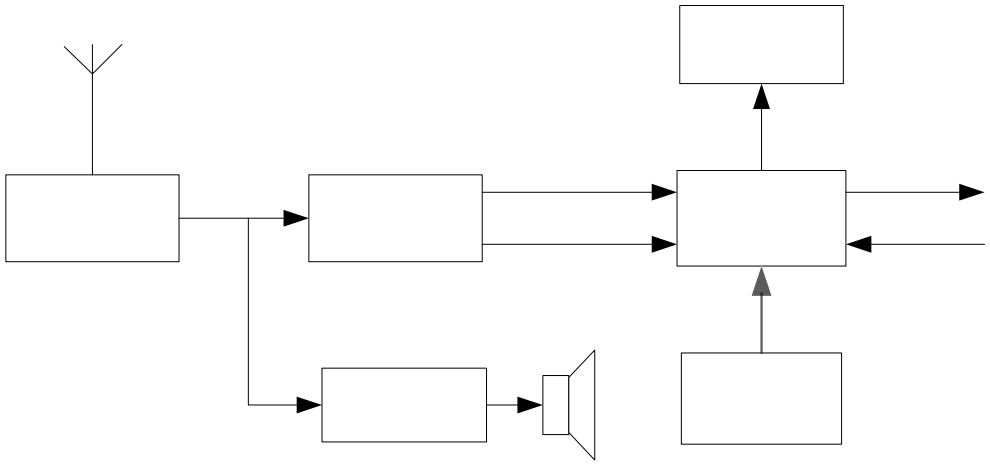


图 1 典型应用框图

RDS 特性

本应用方案支持 PI、PTY、PS、RT、CT、TP、TA、MS、DI、PIN、AF、PTYN 和部分 EON 功能，所有信息通过串口在 PC 机上打印输出。其中 PTY 采用 8 位字符显示，DI 采用自定义的缩写输出（参考文件 rds.h）,CT 采用 RDS 协议规定的显示格式显示，此外还可以显示其他 RDS 特性。EON 信息中 PSON 和 PION 信息可以显示。车载收音机可以根据 EON 数据切换到一个播放本地交通信息的电台上，AF 信息则为让收音机切换到信号最好的频率上。

PI 包含两个字节的内容，主要包括电台所在国家、覆盖区域和服务。它主要用来由微控制器使用但通常并不显示。PI 码的变化意味着接收机重新调整因此所有 RDS 信号将初始化。在本方案中没有显示 PI 信息。

PTY 是由 Block_B 中的 5 位数据指示的当前节目，目前定义了 32 种类型，RDS 和 RBDS 规范中针对 5 位 PTY 码的涵义解释有所不同，具体可以参考 RDS 和 RBDS 规范。在 RDS 协议中规定了 8 字符和 16 字符两种显示格式，当然显示格式也可以自定义。在本方案中采用的是欧洲 RDS 规范的 8 字符显示格式。不同 PTY 码在 RDS 协议中的涵义以及 8 字符和 16 字符显示格式如表 2 所示：

表 2　　PTY 码涵义及显示格式

Number	Code	PTY	8 字符显示	16 字符显示
0	00000	No programme type or undefined	None	None
1	00001	News	News	News
2	00010	Current Affairs	Affairs	Current Affairs
3	00011	Information	Info	Information
4	00100	Sport	Sport	Sport
5	00101	Education	Educate	Education
6	00110	Drama	Drama	Drama
7	00111	Culture	Culture	Culture
8	01000	Science	Science	Science
9	01001	Varied	Varied	Varied Speech
10	01010	Pop Music	Pop M	Pop Music

11	01011	Rock Music	Rock M	Rock Music
12	01100	Easy Listening Music	Easy M	Easy Listening
13	01101	Light classical	Light M	Light Classic M
14	01110	Serious classical	Classic	Serious Classics
15	01111	Other Music	Other M	Other Music
16	10000	Weather	Weather	Weather&Metr
17	10001	Finance	Finance	Finance
18	10010	Children's progarmmes	Children	Children's Progs
19	10011	Social Affairs	Social	Social Affairs
20	10100	Religion	Religion	Religion
21	10101	Phone In	Phone In	Phone In
22	10110	Travel	Travel	Travel&Touring
23	10111	Leisure	Leisure	Leisure&Hobby
24	11000	Jazz Music	Jazz	Jazz Music
25	11001	Country Music	Country	Country Music
26	11010	National Music	Nation M	National Music
27	11011	Oldies Music	Oldies	Oldies Music
28	11100	Folk Music	Folk M	Folk Music
29	11101	Documentary	Document	Documentary
30	11110	Alarm Test	Test	Alarm Test
31	11111	Alarm	Alarm!	Alarm-Alarm!

PS 是一个 8 字符的节目业务名称信息，它有电台连续发送 4 个 0A 或 0B 组，每个 8 位字节对应一个字符，具体的映射可以参考规范附录。

RT 是由最多 64 位字符组成的广播文本，它主要给出当前发送的节目的额外信息。在本方案中，RT 内容也是通过串口打印，因此无法滚动播出。当 RT 内容少于 64 字符时，RT 内容以 0x0d 作为结束标志。

CT 数据每分钟发送一次，主要用来给接收机同步时间。在 CT 中修正儒略日（MJD）、协调世界时（UTC）和本地时间变动同时发送。在电路中，微控制器将接收到的 CT 数据转换为习惯的时间格式，如：Time:2007-8-30 2:16 (+0:0)。

AF 主要用于将车载接收机调节到信号最好的频率。AF 功能在本方案中给出了程序代码，但是由于没有 AF 频率信号，没有验证这部分功能。

TA 和 TP 是两个为标志，当接收到的节目包括交通信息时 TP 标志位置 1，当接收到的节目包括交通广播时 TA 标志位置 1。TA 和 TP 的不同组合有不同的含义。在本方案中 TA 和 TP 的位信息通过串口打印给出。微控制器可以根据需要进行进一步的处理。

MS 是一个开关信号，主要用来表明当前播放的节目是音乐节目或者语言节目。听众可以更具此信号调整接收机的音量和音调以达到最佳收听效果。

DI 和 C1、C0 位组合表明当前发送的 DI 位，4 位 DI 表明不同的操作模式和当前是否有动态的 PTY 在切换。DI 与 C1、C0 的映射及 DI 的具体含义如表 3 所示。

表 3　DI 的含义

C1 C0	DI	含义	显示内容
1　1	d0=0	Mono	Mono
	d0=1	Stereo	Stereo
1　0	d1=0	Not Artificial Head	Not Art
	d1=1	Artificial Head	Arti
0　1	d2=0	Not compressed	Not com
	d2=1	Compressed	compressed
0　0	d3=0	Static PTY	static
	d3=1	Indicates that the PTY code on the tuned sevice or referenced in EON variant 13,is dynamically switched	PTY on

PIN 信息主要用来表明播音员公布的节目的开始日期和时间，它由两个字节的数据组成。在本方案作了这方面的工作，当有 PIN 信息将显示显示 PIN 信息，没有时不做显示。

EON 由 14A 组提供，在本方案中实现了显示 PS(ON)和解析 PI(ON 的功能)。

同步与校验

每一组信息由四块组成，每块信息中包括 1 6 位数据位和 10 位同步校验位。在 RDS 和 RBDS 规范给出了两种同步和校验方法，分别对应不同的同步校验矩阵。两个矩阵分别如图 2 和图 3 所示，表 4 和表 5 位矩阵分别对应的同步校验码。

$$G = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&1&1&0&0&1&1&1\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&0&1&1&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&1&0&1&1\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&0&1&1&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&1&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&0&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&0&1&0&1&1&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1&1&0&0&1
\end{bmatrix}$$

图 2　1 6 X26 解码矩阵 G

表 4　图 2 对应的同步校验码

块	二进制	十六进制
A	00 1111 1100	0X00FC
B	01 1001 1000	0X0198
C	01 0110 1000	0X0168
C’	11 0101 0000	0X0350

D	01 1011 0100	0X01B4

1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1
1	0	1	1	0	1	1	1	0	0
0	1	0	1	1	0	1	1	1	0
0	0	1	0	1	1	0	1	1	1
1	0	1	0	0	0	0	1	1	1
1	1	1	0	0	1	1	1	1	1
1	1	0	0	0	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1
1	1	0	1	1	1	0	1	1	0
0	1	1	0	1	1	1	0	1	1
1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	1	1	1	0	0
0	1	1	1	1	0	1	1	1	0
0	0	1	1	1	1	0	1	1	1
1	0	1	0	1	0	0	1	1	1
1	1	1	0	0	0	1	1	1	1
1	1	0	0	0	1	1	0	1	1

图 3　26X10 同步校验矩阵 H

表 5　图 3 对应的同步校验码

块	二进制	十六进制
A	11 1101 1000	0X03D8
B	11 1101 0100	0X03D4
C	10 0101 1100	0X025C
C’	11 1100 1100	0X03CC
D	01 0101 1000	0X0158

在本方案中采用方法 1 即用图 2 提供的矩阵进行同步校验。当系统初始化时必须在每次接收到一位数据后进行一次同步直到同步到一个 A 块。在本方案中只要有一个 A 块同步校验正确后后面的数据每接收到 26 块，当 B、C(C’)、D 也校验正确后该组信息有效，然后根据该组信息所在的组进行不同的处理。在连续接收过程中，如果块数据同步校验失败，将重新开始每接收一位同步一次的过程。在 RDS 中位率为 1187.5Hz 因此在每个中端间隔微控制器有很多工作要做。

在方法 1 中，每次外部中断 0 后中断程序将数据读入，并将数据位和同步校验位分别存入 16 位的 rec_dat 和 10 位的 rec_off 中。同步校验时首先用 rec_dat 与 G 矩阵的每一列相乘然后读取奇偶标志位，这样得到一个 26 位的序

列，将其中的最后 10 与 rec_off 进行异或，将抑或的结果与表 5 给出的同步校验码进行比较。从而判断数据是否正确。需要注意的是每个组之间的校验应该是依次进行的。此外由上面的叙述可以看出，由于G矩阵前面 16 列为单位矩阵，因此在计算中可以忽略 rec_dat 与这 16 的计算过程。

方法 2 采用图 3 提供的同步校验矩阵和表 5 给定的同步校验码。这个计算方法就是直接将接收到的 26 位信息与H矩阵的每一列相乘然后判断结果的奇偶位并保存，这样得到的 10 位校验码与表 5 给定的同步校验码进行比较以判断数据是否正确。这个过程同样需要注意每组同步校验过程应该是依次进行的。譬如如果 B 块校验错误则整个组的信息都当作错误信息被过滤。再次接收到新的数据是将开启新的同步校验过程。

电路

图 4 为本方案采用的电路框图，由于不同的 RDS 解码芯片与 CS8955 连接比较简单。因此没有提供 RDS 解码芯片部分的电路。来自解码芯片的时钟在每个下降沿引起一次外部中断。此时数据位是有用的在 P2. 6 上读入。所有的处理结果通过串口在 PC 机打印输出。

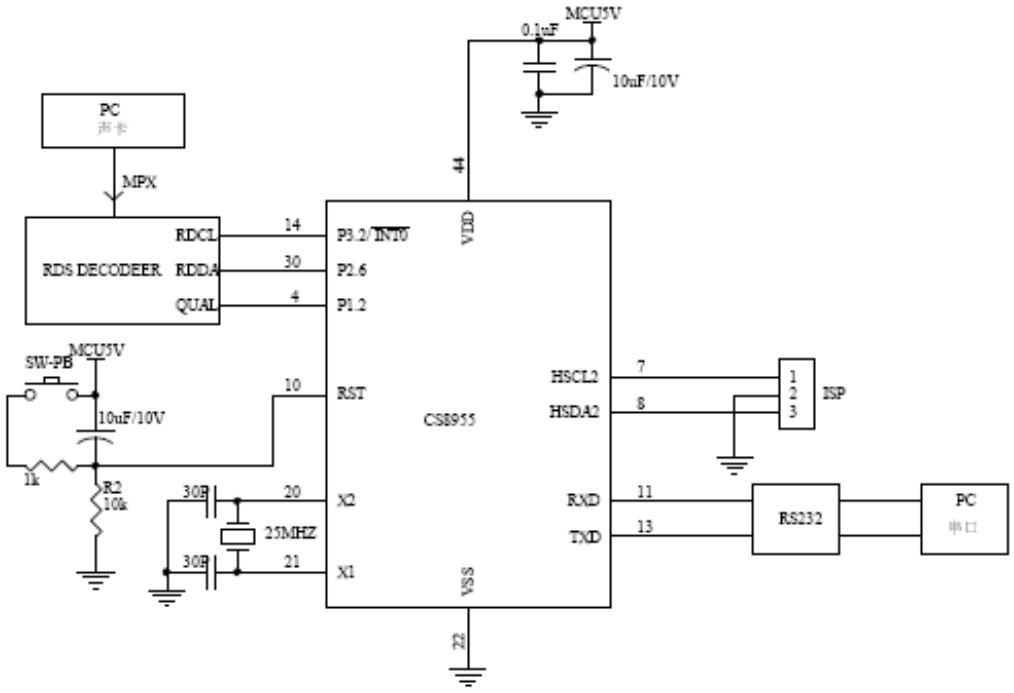


图 4　电路框图

程序设计

在程序设计中 CS8955 初始化完成后等待外部中断，在接收一组正确数据后进行相应的处理，在一组完整的数据转换完成后通过串口在 PC 机上打印输出，同时接收新的中断，转换新的数据为下一次的数据更新作准备。为了在正确传输

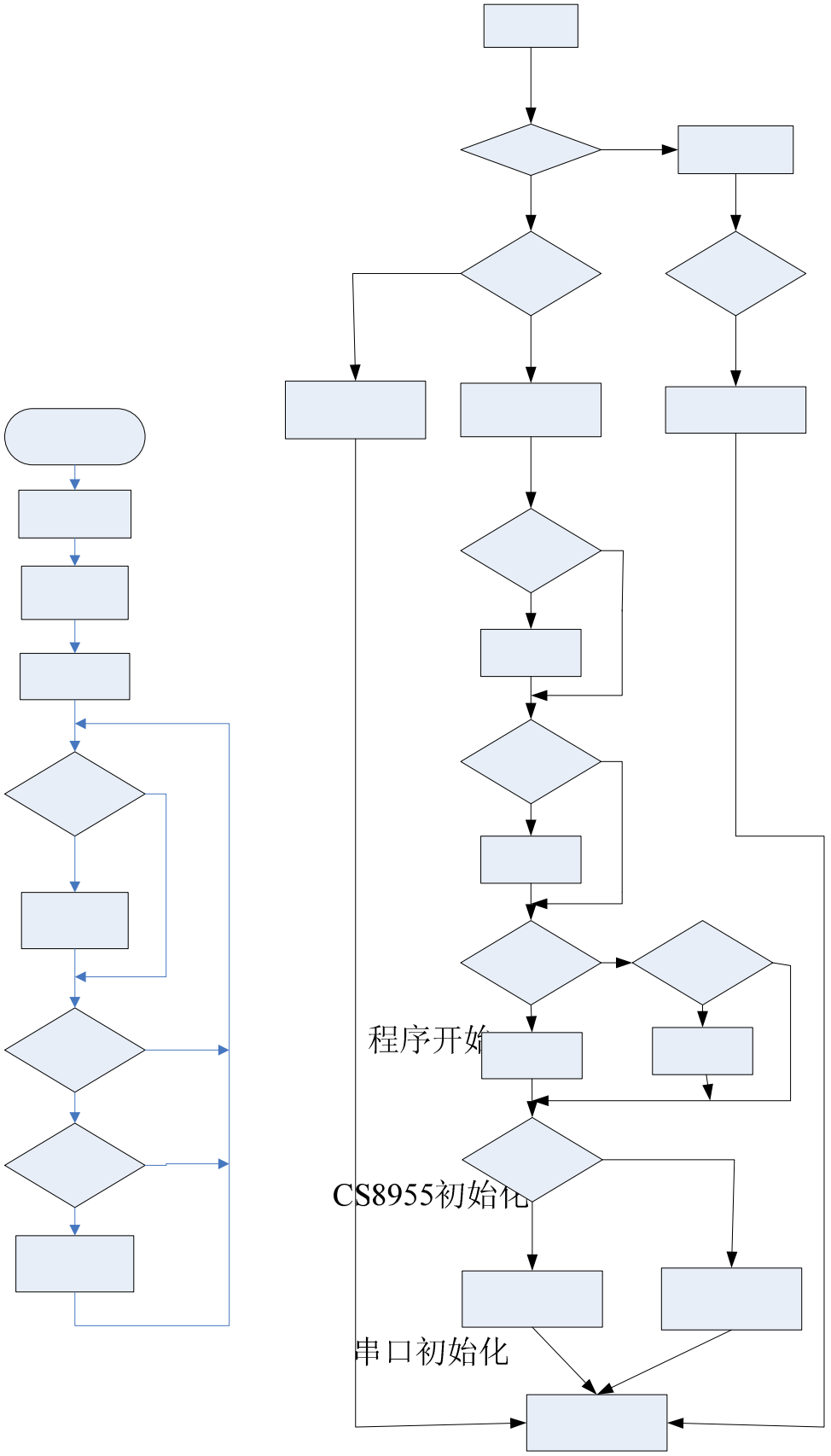


图 5 程序框图

否

数据的同时提供最大的传输速率。在本方案中 CS8955 采用 25MHz 晶振。波特率设置为 19200bps。程序框图分别如图 5 所示。

组信息处理

当接收到一组完整的信息后将信息保存,然后可以在中断间隙完成相应的处理过程。在实际应用中，可以将 AF 和 PI 结合实现更多的功能。

在本方案中，0A、0B、1A、1B、2A、2B、4A、10A、14A 和 15B 的信息进行了相应的处理。表 6 给出了以上各组包含的功能。

表 6 RDS 信息组功能

组号	功能
ALL	PI、PTY、TP
0	TA、DI、MS、PS、AF(0A group only)
1	PIN
2	RT
4A	CT
10A	PTYN
14A	EON
15B	TA、DI、MS

0 组及 15B 组

0A 组和 0B 组的区别仅仅在于 0A 组包含 AF 信息，因此这两个组的处理在程序中放到了一起，只是在公共信息处理完成后再次判断组号，如果是 0A 组则继续完成 AF 功能否则跳出处理程序。表中 0A 组的信息读出后存入 RAM，由于串口显示的局限性，需要等待其他部分信息都处理完成后一起显示，在本方案中则是等待 RT 信息也处理完成后一起显示。15B 的信息与 0 组信息是 0 组信息的一个子集，它主要是为了增加这部分信息的刷新速度。

1 组

1A 和 1B 组的主要区别在于 1B 组在 block3 中重复了 PI，这也是所有 xA 和 xB 组信息的区别。PIN 数据在处理时也是放入 RAM，然后和其他数据一起显示。当没有 PIN 信息时则不显示任何 PIN 提示。PIN 信息主要是为了以后控制其他外设，譬如录音机等。在本方案中完成了日期和时间的处理和显示功能，但是由于信号源的局限性没有实际验证，也没有对相应的外设进行控制的功能。这部分代码仅供用户开发时参考。

2 组

在 2A 组中 RT 信息包含在 block3 和 block4 中，在 2B 组中 block3 中的信息是 PI 码。因此在发送同样长度的 RT 时需要的 2A 和 2B 组的数量是不同的。RT 信息最多包含 64 个字节。当长度小于 64 时以 0x0d 作为结束标志。在程序处理中验证了 2A 组的功能，2B 组的处理代码时在 2A 组的基础上作适当修改。需要注意的是，在实际信息的发送中，2A 和 2B 是不能穿插使用发送同一组信息的， 当发现穿插发送 RT 的情况时，之前接收到的信息必须清零。在本方案中，RT 的显示是在接收到一组完整的信息之后进行的。RT 接收完成之后 RTflag 置 1，通知 CS8955 可以显示完成全部信息的刷新。

4A 组

4A 组的信息包括修正儒略日(MJD)、协调世界时(UTC)以及本地时间差。广播中发送的 UTC 其效果和 GMT 是一样的，因此在 UTC 之后 6 位给出了本地时

间误差。

关于 MJD 的计算 RDS 和 RBDS 上都有详细介绍，下面只介绍年、月、日的计算，关于星期以及其他的一些计算在本方案中没有进行相应的处理，用户可以参考 RDS 和 RBDS 规范。

```
Y'=int[(MJD-15078.2)/365.25]
M'=int{[MJD-14956.1-int(Y'x365.25)]/30.6001}
Day=MJD-14956-int(Y'x365.25)-int(M'x30.6001)
If M'=14 or M'=15
  Then K=1
  Else K=0
Year=Y'+K+1900
Month=M'-1-12K
```

其中 MJD 为修正儒略日

Year、 Month、 Day 为当前年月日；

Y'、M'、K 为中间变量。

10A 组

在 10A 组中 block3 和 block4 包含的 PTYN 信息。PTYN 信息包含 8 个字节的信息，它是对当前 PTY 信息的进一步描述。比如当前 PTY 为 SPORT，PTYN 信息为 Football 就是给 PTY 信息给出更具体的信息。PTYN 信息必须在两个 10Ablock 发送完成，当前组发送的是高 4 位字符还是低 4 位字符由 block2 的 LSB(C0)决定。

C0=0 为低四位，C0=1 为高四位。PS、PTYN 和 RT 码与实际字符的映射关系可以参考协议。

14A 组

14A 包含丰富的 EON 内容，当量的信息可以通过 14A 组发送。全部信息的发送完成到用于接收需要长达 2 分钟的时间。在本方案中只是处理其中的一部分功能，即显示 PS(ON)信息，而且未经验证，用户可以参考协议作进一步的开发。

15B 组

15B 组的处理方式与 0 组相同，只是增加 TA、DI 和 MS 的刷新速度。

显示

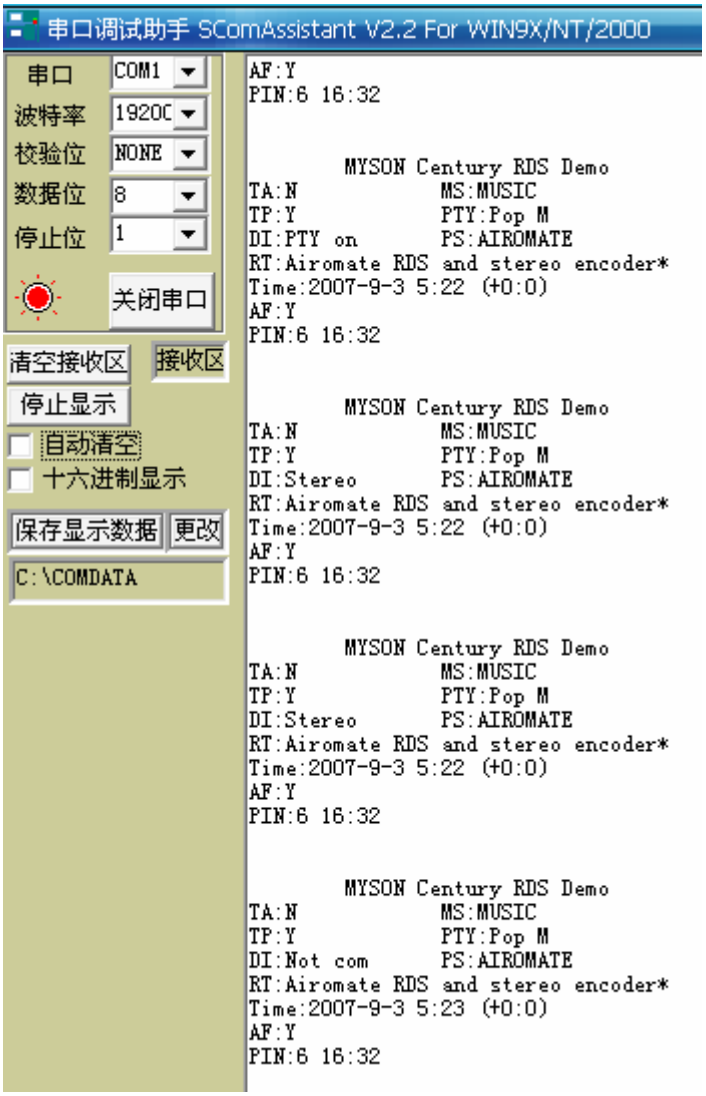
在本方案中所有的信息通过串口输出，为了达到直观的效果，在显示时将所有的内容集中在一起进行刷新。考虑到 CT 每分钟刷新一次，间隔时间相对较长，而 PS 和 RT 刷新时间较短，因此在实际处理中只要完成了对 PS 和 RT 的接收功能就实现一次刷新。

测试结果

Airomate 界面


AIROMATE 2
RDS AF TP TA Music
PS: AIROMATE
RT: id stereo encoder.
PTYN: Pop Music
Time: 2007-9-3 13:24 (+00:00)
Output peak: 15%
L R O
ON/OFF SETUP DISPLAY TA M/S
www.diffusionsoftware.com
Diffusion software

串口显示界面


串口调试助手 SComAssistant V2.2 For WIN9X/NT/2000
串口 COM1
波特率 1920C
校验位 NONE
数据位 8
停止位 1
关闭串口
清空接收区
接收区
停止显示
自动清空
十六进制显示
保存显示数据
更改
C:\COMDATA
AF:Y
PIN:6 16:32
MYSON Century RDS Demo
TA:N MS:MUSIC
TP:Y PTY:Pop M
DI:PTY on PS:AIROMATE
RT:Airomate RDS and stereo encoder*
Time:2007-9-3 5:22 (+0:0)
AF:Y
PIN:6 16:32
MYSON Century RDS Demo
TA:N MS:MUSIC
TP:Y PTY:Pop M
DI:Stereo PS:AIROMATE
RT:Airomate RDS and stereo encoder*
Time:2007-9-3 5:22 (+0:0)
AF:Y
PIN:6 16:32
MYSON Century RDS Demo
TA:N MS:MUSIC
TP:Y PTY:Pop M
DI:Stereo PS:AIROMATE
RT:Airomate RDS and stereo encoder*
Time:2007-9-3 5:22 (+0:0)
AF:Y
PIN:6 16:32
MYSON Century RDS Demo
TA:N MS:MUSIC
TP:Y PTY:Pop M
DI:Not com PS:AIROMATE
RT:Airomate RDS and stereo encoder*
Time:2007-9-3 5:23 (+0:0)
AF:Y
PIN:6 16:32

附录

```
/* main.c
** Authors : chen_ch3
** Myson Century An RDS Decoder using the CS8955
** This demo program show how to decode the RDS signal using
** CS8955
*/

#include "MTV515.h"
#include "stdio.h"
#include "absacc.h"
#include "RDS.h"

main()
{
 /*Initialize CS8955*/
 MCU_Ini();
 /*Initialize UART*/
 UART_Ini();
 /*Initialize INT*/
 INT_Ini();

 /*wait for INT0-interrupt*/
 while(1)
 {
    P1_0=~P1_0;
  //P1_0=~P1_0;
  /*deal with different groups*/
  if(groupflag)
  {
    //printf("MYSON Century RDS Demo\n");
    /*clear groupflag for the next groupflag*/
    groupflag=0;

    //GROUP_ALL_done();
    /*get the grouptype and deal with it*/
    grouptype=bblock>>11;
    grouptype&=0x1f;
    switch(grouptype)
    {
      /* 0A,0B group process
     ** show the TA DI MS PS function
     ** AF function is reserved
```

```
*/
 case 0:
 case 1:
          GROUP0_done();
           break;
 /* 1A,1B group process
** show the PIN function
*/
 case 2:
 case 3:
           GROUP1_done();
          break;
/* 2A group process
** show the RT function broadcasting by 2A groups
*/
 case 4:
           GROUP2A_done();
          break;
/* 2B group process
** show the RT function broadcasting by 2B groups
*/
case 5:
          GROUP2B_done();
          break;
/* 4A group process
** show the CT function
*/
 case 8:
          GROUP4A_done();
           break;
/* 10A group process
** show the PTYN function
*/
case 20:
          GROUP10A_done();
          break;
/* 14A group process
** show the PS(ON)&PI(ON) function
** just a example of EON
*/
 case 28:
            GROUP14A_done();
           break;
 /* 15B group process done
```

```
     ** show TA DI MS function
     */
      case 31:
                 GROUP15B_done();
                break;
      default:
                 break;
     }

   }
   else    if(PSNflag&RTflag)
   {
    RTflag=0;
    PSNflag=0;
    printf("\tMYSON Century RDS Demo\n");
    //rtcnt=rt_num+4;
    textcnt=0;
    if(TAflag)
    printf("TA:Y\t\t");
    else
    printf("TA:N\t\t");
    if(MSflag)
    printf("MS:MUSIC\n");
    else
    printf("MS:SPEECH\n");
    GROUP_ALL_done();
    printf("DI:%s\t",DIM[DI_num]);
    printf("PS:");
    display_cov(PSNlast,8);
    printf("\n");
    printf("RT:");
    display_cov(RT,rtcnt);
    //printf("%d,%d,%d\n",bblock,cblock,dblock);
    printf("\n");
    if(timeflag)
    {
     if(loflag)
      printf("Time:%d-%d-%d
%d:%d(-%d:%d)\n",Year,Month,Day,Hour,Minute,loHour,loMinute);
      else
      printf("Time:%d-%d-%d
%d:%d(+%d:%d)\n",Year,Month,Day,Hour,Minute,loHour,loMinute);
    }
    else
```

```
    printf("Time:\tWait\n");
    if(AFflag)
    printf("AF:Y\n");
    if(PINflag)
    printf("PIN:%d %d:%d\n",PIN_Day,PIN_Hour,PIN_Minute);
    printf("\n\n");
   }
  }
}

/*Initialize CS8955*/
void MCU_Ini()
{
 PADMOD3=0x7f;              //P3_0,P3_1 为 RXD,TXD 功能
}

void UART_Ini()
{

 PCON|=0x80;             //SMOD=1;
 TMOD=0x22;              //T1 工作于定时器模式,工作模式 2
 TH1=0xf9;              //19200bps 波特率
 TH1=0xf9;
 SCON=0x50;              //串口工作方式 1,立即启动串口
 TR1=1;                  //启动定时器 1.
 TI=1;
 printf("UART is ok\n");

}

/*INT Initialize*/
void INT_Ini()
{

 IT0=1;                 //INT0 occurs when low level
 EX0=1;                  //enable INT0 interrupt
 EA=1;                   //enable all interrupt
}

/* INT0 interrupt function
** occur frequency: 57KHz/48=1187.5
** rec_dat saves the 16 bits data
** rec_off saves the 10 bits offset_bit
*/
```

```
void RDS_EX0() interrupt 0
{P1_1=~P1_1;
  /*save rec_dat*/
  rec_dat<<=1;
  /*the MSB of rec_off is high level*/
  if(rec_off&0x0200)
  rec_dat|=0x0001;

  /*save rec_off*/
  rec_off<<=1;
  /*the data is high level*/
  if(P2_0==1)
  rec_off|=0x0001;
  rec_off&=0x03ff;

  bitcnt++;
  /*no A_block is synchronized*/
  if(!synflag)
  {
    crc=0;

    CRC_check();
    crc^=rec_off;
    if(crc==offset_A)
    {
      synlvl=1;
      bitcnt=0;
      synflag=1;
      ablock=rec_dat;
    }
  }

  /* a A_block is synchronized
  ** synchronize the last 3 blocks
  ** or get a new A_block
  */
  else if(bitcnt==26)
  {
    bitcnt=0;
    crc=0;
    CRC_check();
    crc^=rec_off;
    if((crc==offset_A)&&(synlvl==0))
    {//P1_3=~P1_3;
```

```
     synlvl=1;
     ablock=rec_dat;
    }
    else if((crc=offset_B)&&(synlvl==1))
    {//P1_3=~P1_3;
     synlvl=2;
     bblock=rec_dat;
    }
    else if((crc=offset_C)&&(synlvl==2))
    {//P1_3=~P1_3;
     synlvl=3;
     cblock=rec_dat;
    }
    else if((crc=offset_C2)&&(synlvl==2))
    {//P1_3=~P1_3;
     synlvl=3;
     cblock=rec_dat;
    }
    else if((crc=offset_D)&&(synlvl==3))
    {P1_3=~P1_3;
     synlvl=0;
     dblock=rec_dat;
     groupflag=1;
    }

    /*no available group,synchronize again*/
    else
    {
     P2_1=~P2_1;
     synflag=0;
     groupflag=0;
     synlvl=0;
    }
   }
}
```

```
/*calculate the CRC bits
**     |1 0  0  0  0  000  0000000000001110111|
**     |0 1                        01011100111|
**     |0   1                      01110101111|
**     |0     1                    01100001011|
**     |0       1                  01101011001|
**     |0         1                01101110000|
**     |0           1              00110111000|
**     |0             1            00011011100|
** G = |0               1           00001101110|=|E|
**     |0                 1         00000110111|
**     |0                   1       01011000111|
**     |0                     1     01110111111|
**     |0                       1   01100000011|
**     |0                         1 01101011101|
**     |0                          10110111001 0|
**     |0                           10110111001|

offset_X=P(rec_dat&E)^rec_off
P(rec_dat&e)be caculated here
*/

void CRC_check()
{
 bit pority;
 unsigned int tempH,tempL;

 tempH=rec_dat/0x100;
 tempL=rec_dat%256;

 //C9
 ACC=tempH&0x7c;
 pority=P;
 ACC=tempL&0x3e;
 pority^=P;
 if(pority)
 crc|=0x200;

 //C8
 ACC=tempH&0x3e;
 pority=P;
 ACC=tempL&0x1f;
 pority^=P;
 if(pority)
```

```
crc |=0x100;

//C7
ACC=tempH&0x63;
pority=P;
ACC=tempL&0x31;
pority^=P;
if(pority)
crc |=0x080;

//C6
ACC=tempH&0xcd;
pority=P;
ACC=tempL&0xa6;
pority^=P;
if(pority)
crc |=0x040;

//C5
ACC=tempH&0xe6;
pority=P;
ACC=tempL&0xd3;
pority^=P;
if(pority)
crc |=0x020;

//C4
ACC=tempH&0x8f;
pority=P;
ACC=tempL&0x57;
pority^=P;
if(pority)
crc |=0x010;

//C3
ACC=tempH&0x3b;
pority=P;
ACC=tempL&0x95;
pority^=P;
if(pority)
crc |=0x008;

//C2
ACC=tempH&0xe1;
```

```
  pority=P;
  ACC=tempL&0xf4;
  pority^=P;
  if(pority)
  crc|=0x004;

  //C1
  ACC=tempH&0xf0;
  pority=P;
  ACC=tempL&0xfa;
  pority^=P;
  if(pority)
  crc|=0x002;

  //C0
  ACC=tempH&0xf8;
  pority=P;
  ACC=tempL&0x7d;
  pority^=P;
  if(pority)
  crc|=0x001;
  P1_2=~P1_2;

}

/*update PI by ablock*/
void Update_PI()
{
  if(ablock!=PI)
  PI=ablock;
}

/*if there is a TP message, set TP
**defined by bblock
*/
void Update_TP()
{
  if(bblock&0x0400)
  {
    TPflag=1;
    printf("TP:Y\t\t");
  }
  else
  {
```

```
   TPflag=0;
   printf("TP:N\t\t");
  }
}

void Update_PTY()
{
  PTY_cnt=bblock>>5;
  PTY_cnt&=0x1f;
  printf("PTY:%s\n",PTY[PTY_cnt]);
}

/* show the common function in all group
** TP,PTY,PI
*/
void GROUP_ALL_done(void)
{
  Update_PI();
  Update_TP();
  Update_PTY();
}




void GROUP0_done(void)
{
  int i;
  /*DI process*/
  unsigned char temp0b;
  DI_num=bblock&0x0007;
  temp0b=DI_num>>2;
  DI_num<<=1;
  DI_num|=temp0b;
  DI_num&=0x07;
  // printf("DI:%s\t",DIM[dc]);

  /*TA& M/S process*/
  temp0b=bblock>>4;
  if(temp0b&0x0001)
  TAflag=1;
  //printf("TA:Y\t");     //TA is broadcasting now
  else
  TAflag=0;
```

```
//printf("TA:N\t");
temp0b=bblock>>3;
if(temp0b&0x0001)
MSflag=1;
//printf("M/S:MUSIC\n"); //Music is broadcasting now
else
MSflag=0;
//printf("M/S:SPEECH\n");//Speech is broadcasting now
/*PS process*/

temp0b=bblock&0x0003;
if((temp0b==0)&&(pslvl==0))
{
 PSN[0]=dblock>>8;
 PSN[0]&=0xff;
 PSN[1]=dblock&0xff;
 pslvl=1;
}
else if((temp0b==1)&&(pslvl==1))
{
 PSN[2]=dblock>>8;
 PSN[2]&=0xff;
 PSN[3]=dblock&0xff;
 pslvl=2;
}
else if((temp0b==2)&&(pslvl==2))
{
 PSN[4]=dblock>>8;
 PSN[4]&=0xff;
 PSN[5]=dblock&0xff;
 pslvl=3;
}
else if((temp0b==3)&&(pslvl==3))
{
 PSN[6]=dblock>>8;
 PSN[6]&=0xff;
 PSN[7]=dblock&0xff;
 PSNflag=1;
 for(i=0;i<8;i++)
 {
  if(PSNlast[i]!=PSN[i])
  PSNlast[i]=PSN[i];
  else
  ;
```

```
    }
    //display_cov(PSN,8);
    //printf("\n");
  }
  else
  {
    pslvl=0;
  }

  /* 0A group process contains AF information
  ** method A
  */
  if(!grouptype)
  {
    AFH=cblock>>8;
    AFH&=0xff;
    AFL=cblock&0xff;
    if(AFH==224)
    AFflag=0;
    else if((AFH>224)&&(AFH<250))
    {
      AFflag=1;
      AF_num=AFH-224;
      AF_arr=0;
      AF[AF_arr]=AFL;
      AF_arr=1;
    }
    else if((AF_arr<AF_max)
                &&(AFH>0)&&(AFL<205)
              &&(AFH>0)&&(AFL<205))
    {
      AF[AF_arr]=AFH;
      AF_arr++;
      AF[AF_arr]=AFL;
      AF_arr++;
      if(AF_arr==AF_max-1)
      AFflag=0;
    }
    else
    AFflag=0;
  }
}

/*show the PIN function*/
```

```
void GROUP1_done()
{
 PIN_Minute=dblock&0x3f;
 PIN_Hour=dblock>>6;
 PIN_Hour&=0x3f;
 PIN_Day=dblock>>11;
 PIN_Day&=0x1f;
 if((PIN_Day!=0)
       &&(PIN_Day<31)
      &&(PIN_Hour>=0)
      &&(PIN_Hour<32)
      &&(PIN_Minute>=0)
      &&(PIN_Minute<60))
 PINflag=1;
 else
 PINflag=0;
 //printf("PIN:%d %d:%d\n",PIN_Day,PIN_Hour,PIN_Minute);
}


/*display Radio Text*/
void GROUP2A_done()
{
 int rt_num;
 rtaddr=bblock&0x000f;
 if(rtaddr==0)
 textcnt=0;
 else
 textcnt++;
 if(textcnt==rtaddr)
 {
  rt_num=rtaddr*4;
  RT[rt_num]=cblock>>8;
  RT[rt_num]&=0xff;
  RT[rt_num+1]=cblock&0xff;

  RT[rt_num+2]=dblock>>8;
  RT[rt_num+2]&=0xff;
  RT[rt_num+3]=dblock&0xff;
//   printf("%d,%d,\n",RT[rt_num],RT[rt_num+1]);
  if(((RT[rt_num]&0xff)==0x0d)
      ||((RT[rt_num+1]&0xff)==0x0d)
     ||((RT[rt_num+2]&0xff)==0x0d)
     ||((RT[rt_num+3]&0xff)==0x0d)
```

```
      | |((rt_num+3)==0x3e))
    {
     RTflag=1;
     rtcnt=rt_num+4;
    }
    else if(textcnt>=16)
    {
     textcnt=0;
    }
  }
}

void GROUP2B_done(void)
{
 int rt_num;
 P1_3=0;
 rtaddr=bblock&0x000f;
 if(rtaddr==0)
 textcnt=0;
 else
 textcnt++;
 if(textcnt==rtaddr)
 {
  rt_num=rtaddr*2;
  RT[rt_num]=dblock>>8;
  RT[rt_num]&=0xff;
  RT[rt_num+1]=dblock&0xff;

  if(((RT[rt_num]&0xff)==0x0d)
      | |((RT[rt_num+1]&0xff)==0x0d)
     | |((rt_num+1)==0x3e))
  {
   rtcnt=rt_num+2;
   textcnt=0;
   printf("RTB:");
   display_cov(RT,rtcnt);
   printf("\n");
  }
  else if(textcnt>=32)
  {
   textcnt=0;
  }
 }
}
```

```
/*display MJD Day and time
** the start of MJD time is 1858-11-17 00:00:00
** source from
**http://baike.baidu.com/view/37497.htm
*/
void GROUP4A_done()
{
 unchar temp_h;
 looffset=dblock&0x001f;
 loHour=looffset/2;
 loMinute=looffset%2;
 if(loMinute)
 loMinute=30;
 else
 loMinute=0;
 if(dblock&0x0020)
 loflag=1;
 else
 loflag=0;
 Minute=dblock>>6;
 Minute&=0x3f;
 Hour=dblock>>12;
 Hour&=0x0f;
 if(cblock&0x01)
 Hour&=0x01;
 temp_h=(unchar)(cblock<<5);
 temp_h&=0x20;
 Hour|=temp_h;
 MJD=((unsigned long)cblock)>>1;
 if((bblock&0x03)==0x03)
 MJD|=0x018000;
 else if((bblock&0x03)==0x02)
 MJD|=0x01000;
 else if((bblock&0x03)==0x01)
 MJD|=0x08000;
 MJD_YMD(MJD);
 if((Month>0)&&(Month<13)
       &&(Day>0)&&(Day<32)
      &&(Hour>=0)&&(Hour<24)
      &&(Minute>0)&&(Minute<60)
      &&(loHour>=0)&&(loHour<24)
      &&(loMinute>=0)&&(loMinute<60))
 timeflag=1;
```

```
  else
  timeflag=0;
  /*delay();
    //    printf("%d,%d,%d\n",bblock,cblock,dblock);
  if(loflag)
  printf("Time1:%d-%d-%d                                                    %d:%d
(-%d:%d)\n",Year,Month,Day,Hour,Minute,loHour,loMinute);
  else
  printf("Time0:%d-%d-%d                                                    %d:%d
(+%d:%d)\n",Year,Month,Day,Hour,Minute,loHour,loMinute);
*/
}

/*Function: void MJD_YMD(unsigned long MJD_num)
**Purpuse: convert 17-bit MJD code to normal format
**source rbds1998(STD.).pdf
**yy=(unsigned long)[(MJD_num-15078.2)/365.25]
**mm=(unsigned                    long){[MJD_num-14956.1-(unsigned
long)(yy*365.25)]/30.6001}
**D=MJD_num-14956-(unsigned                long)(yy*365.25)-(unsigned
long)(mm*30.6001)
**if mm=14 or mm=15,then k=1,else k=0
**yy,mm,k,temp isintermediate variables
*/
void MJD_YMD(unsigned long MJD_num)
{
  int k;
  unsigned long yy,mm,temp;
  yy=(unsigned long)(MJD_num-15078.2);
  yy=yy/365.25;
  Year=yy;
  yy=(unsigned long)(yy*365.25);
  mm=(unsigned long)(MJD_num-14956.1-yy);
  mm=mm/30.6001;
  temp=(unsigned long)(mm*30.6001);
  Day=MJD_num-14956-yy-temp;
  if((mm==14)|(mm==15))
  k=1;
  else
  k=0;
  Year+=k+1900;
  Month=k*12;
  Month=mm-1-Month;
}
```

```
void GROUP10A_done()
{
 if(((bblock&0x0001)==0))
 {
  PTYN[0]=cblock>>8;
  PTYN[0]&=0xff;
  PTYN[1]=cblock&0xff;
  PTYN[2]=dblock>>8;
  PTYN[2]&=0xff;
  PTYN[3]=dblock&0xff;
  PTYNlvl=1;
 }
 else if((bblock&0x0001==1)&&(PTYNlvl==1))
 {
  PTYN[4]=cblock>>8;
  PTYN[4]&=0xff;
  PTYN[5]=cblock&0xff;
  PTYN[6]=dblock>>8;
  PTYN[6]&=0xff;
  PTYN[7]=dblock&0xff;
  display_cov(PTYN,8);
  PTYNlvl=0;
 }
 else
 PTYNlvl=0;
}

void GROUP14A_done(void)
{
 /*DI process*/
 unsigned char dc,temp0b;
 grouptype=bblock>>11;
 grouptype&=0x001f;

 P0=0xfc;
 dc=bblock&0x0007;
 temp0b=dc>>2;
 dc<<=1;
 dc|=temp0b;
 dc&=0x07;
 //printf("DI:    %s\n",DIM[dc]);

 /*PSON process*/
```

```
  temp0b=bblock&0x0003;
  if((temp0b==0)&&(pslvl==0))
  {
    PSON[0]=cblock>>8;
    PSON[0]&=0xff;
    PSON[1]=cblock&0xff;
    PSONlvl=1;
  }
  else if((temp0b==1)&&(pslvl==1))
  {
    PSON[2]=cblock>>8;
    PSON[2]&=0xff;
    PSON[3]=cblock&0xff;
    PSONlvl=2;
  }
  else if((temp0b==2)&&(pslvl==2))
  {
    PSON[4]=cblock>>8;
    PSON[4]&=0xff;
    PSON[5]=cblock&0xff;
    PSONlvl=3;
  }
  else if((temp0b==3)&&(pslvl==3))
  {
    PSON[6]=cblock>>8;
    PSON[6]&=0xff;
    PSON[7]=cblock&0xff;
     printf("PSON:");
     display_cov(PSON,8);
     printf("\n");
  }
  else
  {
    PSONlvl=0;
  }

}


 void GROUP15B_done()
 {
  /*Dl process*/
  unchar dc,temp0b;
```

```
  dc=bblock&0x0007;
  temp0b=dc>>2;
  dc<<=1;
  dc|=temp0b;
  dc&=0x07;

  /*TA& M/S process*/
  temp0b=bblock>>4;
  if(temp0b&0x0001)
  TAflag=1;
  else
  TAflag=0;
  temp0b=bblock>>3;
  if(temp0b&0x0001)
  MSflag=1;
  else
  MSflag=0;
}

void display_cov(unchar *text,int num)
{
  int i;
  unchar *temp_txt;

  temp_txt=text;

  for(i=0;i<num;i++)
  {
    if(*(temp_txt+i)==32)
    printf("%s",Disp_code[0]);
    else if(*(temp_txt+i)==39)
    printf("%s",Disp_code[1]);
    else if((*(temp_txt+i)>0x2b)&&(*(temp_txt+i)<0x3a))
    {
      *(temp_txt+i)-=42;
      printf("%s",Disp_code[*(temp_txt+i)]);
    }
    else if((*(temp_txt+i)>0x40)&&(*(temp_txt+i)<0x5b))
    {
      *(temp_txt+i)-=49;
      printf("%s",Disp_code[*(temp_txt+i)]);
    }
      else if((*(temp_txt+i)>0x60)&&(*(temp_txt+i)<0x7b))
    {
```

```
     *(temp_txt+i)-=55;
     printf("%s",Disp_code[*(temp_txt+i)]);
    }
    else
    printf(" ");
  }
}


/* filename RDS.h
** Author     chen_ch3
*/

#define offset_A    0x0fc
#define offset_B    0x198
#define offset_C    0x168
#define offset_C2 0x350
#define offset_D    0x1b4
#define RT_max       64
#define AF_max       25

#define unchar unsigned char
#define unint unsigned int

unint rec_dat;          //16bit data
unint rec_off;          //10bit offset
bit synflag,groupflag;             //1 ,syn success;

unint crc;
unchar DI_num;
bit TAflag,MSflag,TPflag;
unchar blockcnt,bitcnt;
unint ablock,bblock,cblock,dblock;
unchar PS_lvl;
unchar synlvl,pslvl,PSONlvl;
bit PINflag;
unint PI,PTY_cnt,PIN,PIN_Day,PIN_Hour,PIN_Minute;
bit AFflag;
unchar AFH,AFL,AF_num;
unchar AF[AF_max],AF_arr;
bit PSNflag,RTflag;
unsigned long MJD;
bit timeflag;
unint Year,Month,Day,Hour,Minute,loHour,loMinute,looffset;
```

```
bit loflag,loflag1,timeflag;
unchar PSN[8],PSNlast[8];
unchar RT[RT_max];
unchar PSON[8];
unchar PTYN[8],PTYNlvl;
unint PION;
unchar rtcnt,textcnt,rtaddr;
unchar grouptype;
code char *PTY[32]={ "None ",         //PTY=0 no program type
                     "News ",         //      1 News
                      "Affairs",      //      2 Current Affairs
                     "Info",          //      3 Information
                     "Sport",         //      4 Sport
                     "Educate",       //      5 Education
                     "Drama",         //      6 Drama
                     "Culture",       //      7 Culture
                     "Science",       //      8 Science
                     "Varied",        //      9 Varied
                     "Pop M",         //      10 Pop Music
                     "Rock M",        //      11 Rock Music
                     "Easy M",        //      12 Easy Listening Music
                     "Light M",       //      13 Light Classical
                     "Classic",       //      14 Serious Classical
                     "Other M",       //      15 Other Music
                        "Weather",    //      16 Weather
                        "Finance",    //      17 Finance
                        "Children",   //      18 Children's Programmes
                        "Social",     //      19 Social Affairs
                        "Religion",   //      20 Religion
                        "Phone in",   //      21 Phone In
                        "Travel",     //      22 Travel
                        "Leisure",    //      23 Leisure
                        "Jazz",       //      24 Jazz Music
                        "Country",    //      25 Country Music
                        "Nation M",   //      26 National Music
                        "Oldies",     //      27 Oldies Music
                        "Folk M",     //      28 Folk Music
                        "Document",   //      29 Documentary
                        "Test",       //      30 Alarm Test
                        "Alarm!"      //      31 Alarm
                    };
code unchar *Disp_code[]={
                            " ",
                            """,
```

```
                                ",","-","*","/",
                                  "0","1","2","3","4","5","6","7","8","9",
                                "A","B","C","D","E","F","G","H","I","J",
                                "K","L","M","N","O","P","Q","R","S","T",
                                "U","V","W","X","Y","Z",
                               "a","b","c","d","e","f","g","h","i","j",
                               "k","l","m","n","o","p","q","r","s","t",
                               "u","v","w","x","y","z"
                    };
code char *DIM[8]={
                        "Static",         //d3=0Static PTY
                      "PTY on ",        //d3=1PTY on the service
                      "Not com",        //d2=0 Not compressed
                      "Compressed",    //d2=1 Compressed
                      "Not Art",        //d1=0 Not Artificial Head
                      "Arti",           //d1=1 Artificial Head
                        "Mono",           //d0=0 Mono
                      "Stereo"          //d0=1 Stereo
                     };

/*function list*/
void MCU_Ini(void);
void INT_Ini(void);
void UART_Ini(void);
void CRC_check(void);
void GROUP_ALL_done(void);
void Update_PI(void);
void Update_TP(void);
void Update_PTY(void);
void GROUP0_done(void);
void GROUP1_done(void);
void GROUP2A_done(void);
void GROUP2B_done(void);
void GROUP4A_done(void);
void GROUP10A_done(void);
void GROUP14A_done(void);
void GROUP15B_done(void);
void display_cov(unchar *text,int num);
void MJD_YMD(unsigned long MJD_num);
```

```
/*---------------------------------------------------------------------------
cs8955.h by chen_ch3

Header file for generic 80C51 and 80C31 microcontroller.
Copyright (c) 1988-2002 Keil Elektronik GmbH and Keil Software, Inc.
All rights reserved.
---------------------------------------------------------------------------*/

#ifndef __cs8955_H__
#define __cs8955_H__

//#define XBYTE unsigned char
/*    BYTE Register    */
sfr P0      = 0x80;
sfr P1      = 0x90;
sfr P2      = 0xA0;
sfr P3      = 0xB0;
sfr PSW     = 0xD0;
sfr ACC     = 0xE0;
sfr B       = 0xF0;
sfr SP      = 0x81;
sfr DPL     = 0x82;
sfr DPH     = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0     = 0x8A;
sfr TL1     = 0x8B;
sfr TH0     = 0x8C;
sfr TH1     = 0x8D;
sfr IE      = 0xA8;
sfr IP      = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

/*P0*/
sbit P0_0=0x80;
sbit P0_1=0x81;
sbit P0_2=0x82;
sbit P0_3=0x83;
sbit P0_4=0x84;
sbit P0_5=0x85;
```

```
sbit P0_6=0x86;
sbit P0_7=0x87;


/*P1*/
sbit P1_0=0x90;
sbit P1_1=0x91;
sbit P1_2=0x92;
sbit P1_3=0x93;
sbit P1_4=0x94;
sbit P1_5=0x95;
sbit P1_6=0x96;
sbit P1_7=0x97;

/*P2*/
sbit P2_0=0xa0;
sbit P2_1=0xa1;
sbit P2_2=0xa2;
sbit P2_3=0xa3;
sbit P2_4=0xa4;
sbit P2_5=0xa5;
sbit P2_6=0xa6;
sbit P2_7=0xa7;

/*P3*/
sbit P3_0=0xb0;
sbit P3_1=0xb1;
sbit P3_2=0xb2;
sbit P3_3=0xb3;
sbit P3_4=0xb4;
sbit P3_5=0xb5;
sbit P3_6=0xb6;
sbit P3_7=0xb7;

/*t2*/

sfr T2CON  = 0xC8;
sfr T2MOD  = 0xC9;
sfr RCAP2L = 0xCA;
sfr RCAP2H = 0xCB;

/*  BIT Register  */
/*  PSW  */
sbit CY  = 0xD7;
```

```
sbit AC     = 0xD6;
sbit F0     = 0xD5;
sbit RS1    = 0xD4;
sbit RS0    = 0xD3;
sbit OV     = 0xD2;
sbit P      = 0xD0;

/*  TCON  */
sbit TF1    = 0x8F;
sbit TR1    = 0x8E;
sbit TF0    = 0x8D;
sbit TR0    = 0x8C;
sbit IE1    = 0x8B;
sbit IT1    = 0x8A;
sbit IE0    = 0x89;
sbit IT0    = 0x88;

/*  IE    */
sbit EA     = 0xAF;
sbit ES     = 0xAC;
sbit ET1    = 0xAB;
sbit EX1    = 0xAA;
sbit ET0    = 0xA9;
sbit EX0    = 0xA8;

/*  IP    */
sbit PS     = 0xBC;
sbit PT1    = 0xBB;
sbit PX1    = 0xBA;
sbit PT0    = 0xB9;
sbit PX0    = 0xB8;

/*  P3  */
sbit RD     = 0xB7;
sbit WR     = 0xB6;
sbit T1     = 0xB5;
sbit T0     = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD    = 0xB1;
sbit RXD    = 0xB0;

/*  SCON  */
sbit SM0    = 0x9F;
```

```
sbit SM1    = 0x9E;
sbit SM2    = 0x9D;
sbit REN    = 0x9C;
sbit TB8    = 0x9B;
sbit RB8    = 0x9A;
sbit TI     = 0x99;
sbit RI     = 0x98;

/*----------------------------
        MTV515 EXT-SFR(XFR)
         0X0F00~~~~~0X0FFF
----------------------------*/
#define I2CCTR    XBYTE[0X0F00]     //I2C interface status/control register
#define I2CSTUS XBYTE[0X0F01]     //I2C interface status register
#define INTFLG    XBYTE[0X0F03]     //Interrupt flag
#define INTEN     XBYTE[0X0F04]     //Interrupt enable
#define  MBUf      XBYTE[0X0F05]     //Master I2C receive/transmit data
buffer
#define RCBBUF    XBYTE[0X0F08]     //Slave B I2C receive buffer(R)
#define TXBBUF    XBYTE[0X0F08]     //Slave B I2C Transmit buffer(W)
#define SLVBADR XBYTE[0X0F09]     //ENSlvB    &    Slave B I2C Address

#define CTRSLVB    XBYTE[0X0F0A]    //(R)
                                    //(W)

/* ISP */
#define ISPSLV    XBYTE[0X0F0B]     //ISP Slave address
#define ISPEN     XBYTE[0X0F0C]     //Write 93h to enable ISP Mode


/* ADC */
#define ADC       XBYTE[0X0F10]     //W
#define ADC0      XBYTE[0X0F10]     //ADC0 Convert result(R)
#define ADC1      XBYTE[0X0F11]     //ADC1 Convert result(R)
#define ADC2      XBYTE[0X0F12]     //ADC2 Convert result(R)
#define ADC3      XBYTE[0X0F13]     //ADC3 Convert result(R)
#define WDT       XBYTE[0X0F18]     //Watchdog Timer control register

#define DA0       XBYTE[OXOF20]     //Pulse width of PWM DAC0
#define DA1       XBYTE[0X0F21]     //Pulse width of PWM DAC1
#define DA2       XBYTE[0X0F22]     //Pulse width of PWM DAC2
#define DA3       XBYTE[0X0F23]     //Pulse width of PWM DAC3
#define DA4       XBYTE[0X0F24]     //Pulse width of PWM DAC4
#define DA5       XBYTE[0X0F25]     //Pulse width of PWM DAC5
```

```
#define DA5FC     XBYTE[0X0F2E]     //
#define DAFC      XBYTE[0X0F2F]     //
#define PADMOD1 XBYTE[0X0F50]     //Pin&ADi mode control register1
#define PADMOD2 XBYTE[0X0F51]     //Pin&ADi mode control register1
#define PADMOD3 XBYTE[0X0F52]     //Pin&ADi mode control register2
#define SEL       XBYTE[0X0F55]
#define OPTION   XBYTE[0X0F56]    //Chip option configuration ,reset
for "0"
#define P4        XBYTE[0X0F57]
#define P40       XBYTE[0X0F58]    //P4.0 pin
#define P41       XBYTE[0X0F59]    //P4.1 pin
#define P42       XBYTE[0X0F5A]    //P4.2 pin
#define P43       XBYTE[0X0F5B]    //P4.3 pin
#define P1E       XBYTE[0X0F5D]    //Bit1~4 for P1.x or DAx

#define SLV2ADR XBYTE[0X0F87]    //ENSlvA2   &   Slave A2 I2c address

#define ETCTR     XBYTE[0X0F88]
#define ETMOD1    XBYTE[0X0F89]
#define ETMOD2    XBYTE[0X0F8A]
#define ETMOD3    XBYTE[0X0F8B]
#define ETMOD4    XBYTE[0X0F8C]
#define ETMOD5    XBYTE[0X0F8D]
#define ETMOD6    XBYTE[0X0F8E]

#define I2CSTUS2 XBYTE[0X0F91]
#define INTFLG0 XBYTE[0X0F93]     // different status between R&W
#define INTEN2    XBYTE[0X0F94]
#define RCBBUF2 XBYTE[0X0F98]     //Slave B receive buffer(R)
#define TXBBUF2 XBYTE[0X0F98]     //Slave B transmit buffer(W)
#define SLVB2ADR XBYTE[0X0F99]   //ENSlvB2      &      Slave B I2C
address

#define CTRSLVB2 XBYTE[0X0F9A]
#define EINTIPEN XBYTE[0X0FF4]

#endif
```