快速原型开发工作室

整车控制器软件说明文档

软件模块说明

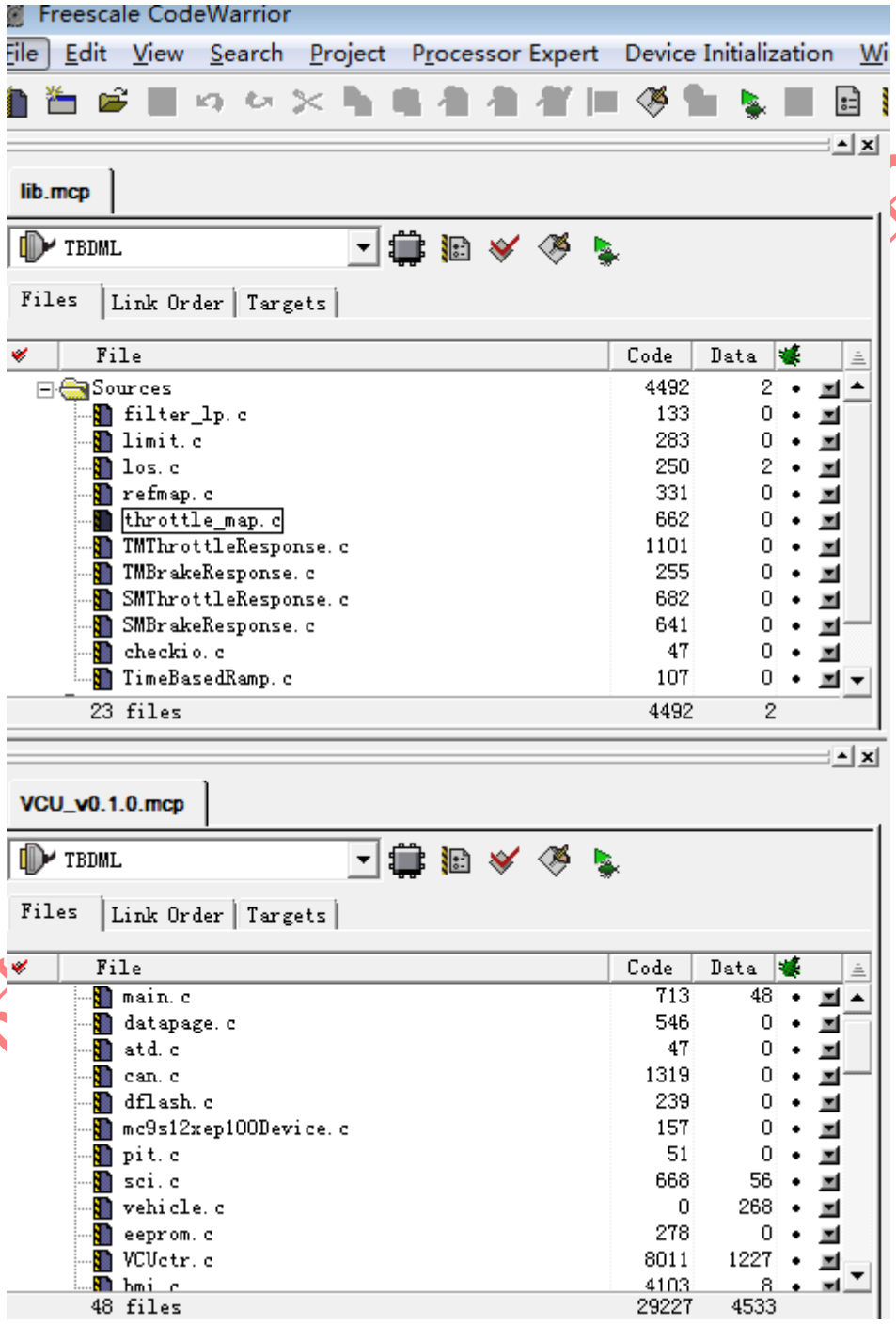
快速原型开发工作室 v0.1.0
2015/6/7

目录

一、软件概述.....2
二、第一部分库函数模块说明.....3
2.1 filter_lp.c.....3
2.2 limit.c.....3
2.3 los.c.....4
2.4 refmap.c.....5
2.5 throttleMap.c.....6
2.6 TMThrottleResponse.c.....7
2.7 TMBrakeResponse.c.....8
2.8 SMThrottleResponse.c.....8
2.9 SMBrakeResponse.c.....9
2.10 checkio.c.....10
2.11 TimeBasedRamp.C.....11
2.12 lib 库说明.....11
三、第二部分工程函数说明.....13
3.1 函数说明.....13
3.2 相关中断说明.....16
3.3 主循环任务说明.....16

一、软件概述

本文档主要针对 VCU 代码进行相关说明，代码基于 Freescale CodeWarrior 进行开发。VCU 使用的芯片为 MC9S12XEP100。VCU 源代码分为两部分，第一部分是相关算法模块库函数，第二部分为主代码工程。如下图所示 1.1 所示。主工程主要调用第一部分生成的库函数。



二、第一部分库函数模块说明

下面主要对各个模块函数进行相关说明。

2.1 filter_lp.c

此函数主要用于低通滤波。输入参数 input；输出参数 output;滤波系数 K2 K3。其中 K3 = 1-K2；低通滤波公式为:
y(k)=Ts/(Rc+Ts)*X(K)+(1-Ts/(Rc+Ts))*y(k-1); u=Ts/(Rc+Ts),fc=1/(2*pi*Rc)
其中 Ts 为采样周期，y 为输出，X 为输入。fc 为截至频率。代码使用公式:
y_out(k+1) = K2*input(k+1)+(1-K2)*y_out(k) 其中 K2= Ts/(Rc+Ts)
代码中运算使用 Q15 格式。

2.2 limit.c

此函数功能主要用于限制电机的力矩值。根据电机的转速设定转矩的限制值。分为 1，2，4，8 倍的转速差值段。主要分为驱动限制和发电限制分别如下图 2.1 和图 2.2 所示。

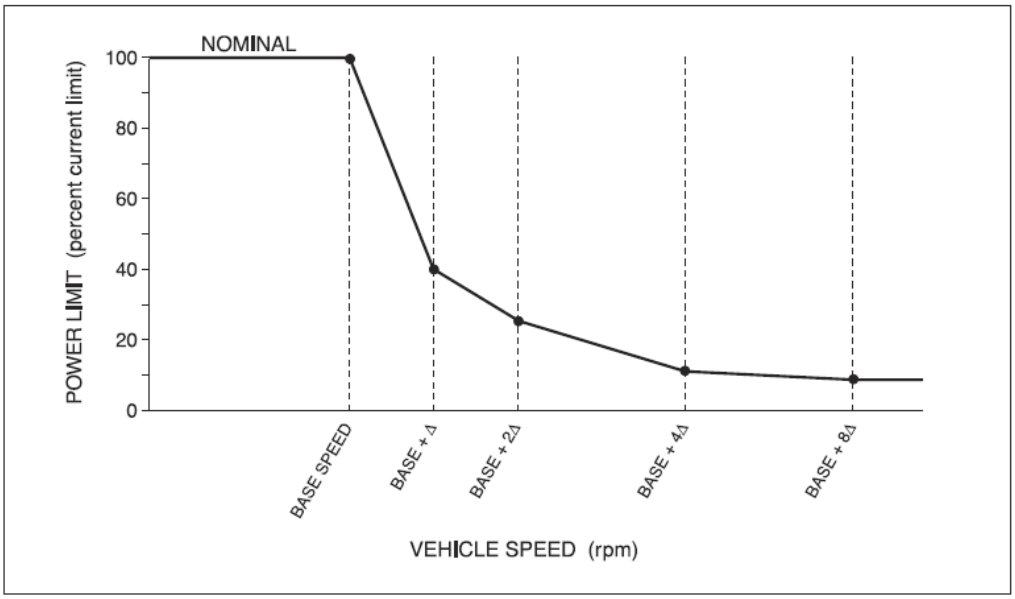


图 2.1 驱动力矩限制

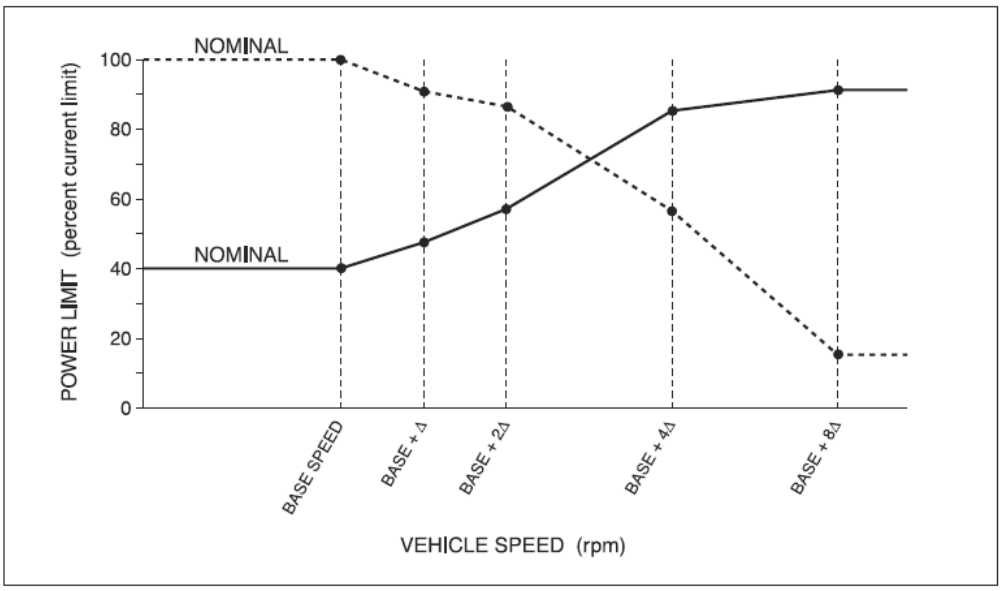


图 2.2　发电力矩限制

代码中重点参数主要有：

```
word        GenSpeed[7];     //发电转速表
word        GenTorque[7];    //发电力矩表
word        EleSpeed[7];     //电动转速表
word        EleTorque[7];    //电动力矩表
int         SpeedFbk;        //当前转速输入值
int         Coeff;           //输出值，表示的输出百分数
word*       SpeedPtr;        //内部转速表指针
word*       TorquePtr;       //内部力矩表指针

word        SpeedCnt   :4;    //  记录转速表位置
word        Mode       :1;//0-电动模式;1-发电模式
```

代码运算使用 Q15 格式。

2.3 los.c

此函数主要用于过电压，低电压，电机控制器过温，电机过温减载功能。在警告点和最大点中间使用使用线性减载功能。例如在过电压减载中当电压值超过设定的警告值，力矩输出就会相应的减少，直到电压值超过设置的最大点，力矩输出值就会为零。在各个减载功能使能的情况下，最终使用的减载系数为最小的那个。减载值不能突变。

代码中的主要参数说明：

```
LOS_STATUS          Cmp;        //用于判断是正向还是负向减载
LOS_STATUS          En;         //用于判断减载功能是否使能
LOS_IQVALUE         Input;      //采样值的输入
```

```
LOS_IQVALUE     Upper;      //设置的最大值
LOS_IQVALUE     Down;       //设置的最小值
LOS_IQVALUE     Delta_value;//内部变量
LOS_IQVALUE     Output;     //用于储存各减载功能的减载系数
word            Coeff;      //最终输出的减载系数
word            CoeffStep;  //用于减载值改变的的步长
word            CoeffOld;   //上一次的减载系数
```

代码运算使用 Q15 格式。

2.4 refmap.c

此函数的功能主要实现踏板百分数对应的输出百分数计算。如下图 2.3 所示。同样刹车踏板使用的也是此函数。

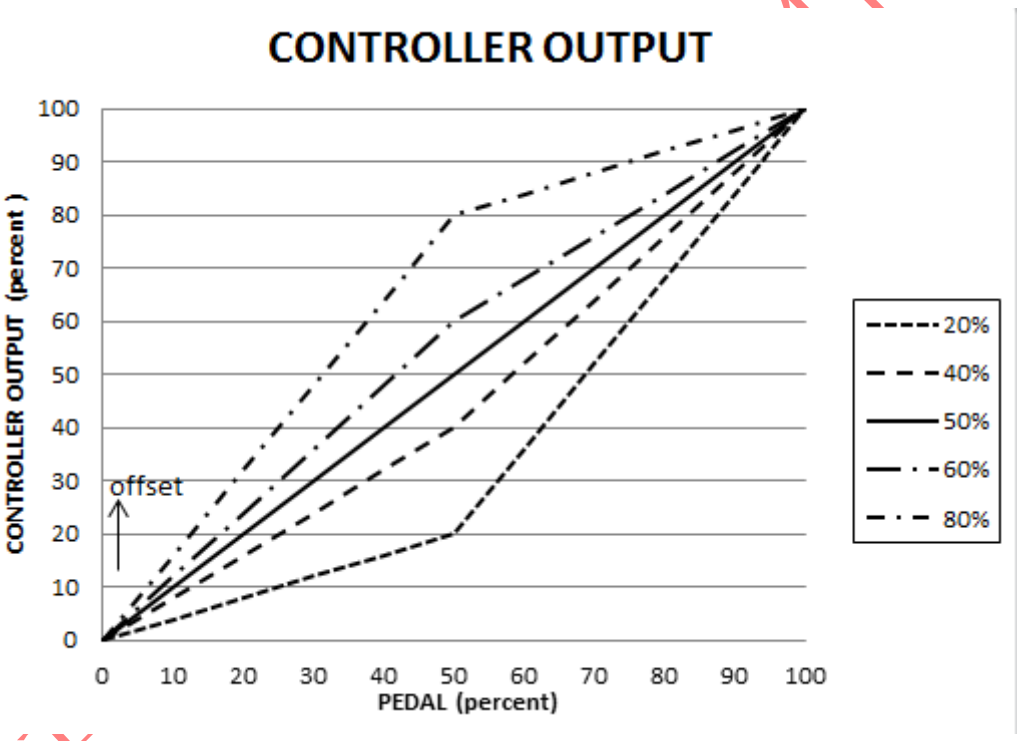


图 2.3　踏板百分数与控制输出百分数对应关系

相关参数说明：

```
word MidVolPercent;      // the middle voltage percent
word MidVol_RefPercent;// the ref percent for middle voltage percent
word ratio1;             // 1/(MidVolPercent-0%)
word ratio2;             // 1/(100%-MidVolPercent)
word ratio_K1;           // k1 = (MidVol_RefPercent-0%)/(MidVolPercent-0%)
word ratio_b1;          // offset  起点垂直方向偏置
word ratio_K2;           //k2= 100%-MidVol_RefPercent)/(100%-MidVolPercent)
word ratio_b2;            // b = 100% - K2
```

代码运算使用 Q15 格式。

2.5 throttleMap.c

此函数只要是用于踏板的故障检测和将踏板电压值转换为踏板输出百分数。踏板类型有多种，本代码中有两种类型。可根据需要做修改。踏板故障检测主要是判断踏板的最大和最小电压值，在这个范围内表示踏板正常，否则不正常。同时加入 IO 判断，当踩下踏板时会有 IO 口的输入。因此如果检测到踏板输入有正常的电压值但是未检测到 IO 口的输入，表示踏板异常；如果检测到 IO 口的输入但是为检测到踏板的电压值表示踏板异常。踏板输出百分数如图 2.4 所示为一种设置情况。 0.5V 对应 0% ； 4.5V 对应 100%。

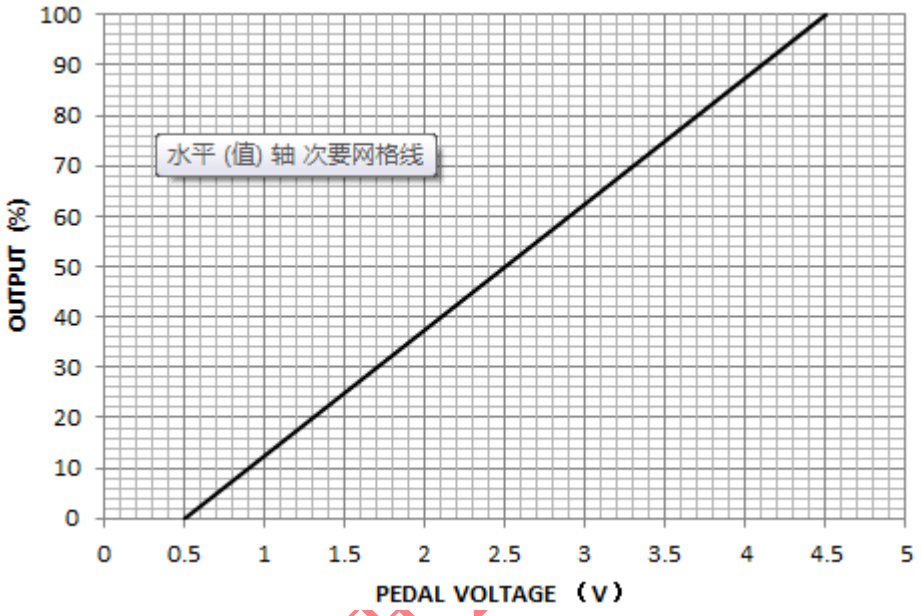


图 2.4 踏板电压对应输出百分数

相关的函数参数说明

```
word    VerifyRange;// Range of for check throttle
word S1_HighVoltage;// the voltage of S1 for 100% throttle ,IQ15
word S1_LowVoltage;// the voltage of S1 for 0% throttle,IQ15
word S2_HighVoltage;//    the voltage of S2 for 100% throttle ,IQ15
word S2_LowVoltage;//    the voltage of S2 for 10% throttle,IQ15
dword DeltaS1HiLowVoltage;  // throttle signal1,
                            //1/(high voltage - low voltage),IQ15
dword DeltaS2HiLowVoltage;  // throttle signal2,
                            //1/(high voltage - low voltage),IQ15
word PercentS1;       // throttle signal(S1) percent ,IQ15
word PercentS2;       // throttle    signal(S2) percent ,IQ15

word MaxVoltage;      // the Max voltage of Throttle Signal 1 for Protection ,over it throttle failed
word MinVoltage;    // the Min voltage of Throttle Signal 1 for Protection ,Under it throttle failed
```

```
word CheckEnable:1; // 0:disable throttle check ,1:enable throttle
word CheckType:3;    // Type of check tow throttle signal
word ErrorInf:4;      // Error information of throttle,
                      //0:normal;1: out range;2:S1 failed ;3:S2 failed
word OldErrorInf:4;        // Last error information
word DeltaS1;              // throttle signal(S1) Protect Voltage Delta ,IQ15
word DeltaS2;              // throttle signal(S2) Protect Voltage Delta ,IQ15
```

代码运算使用 Q15 格式。

2.6 TMThrottleResponse.c

此函数主要力矩模式下，用于计算力矩模式下的油门百分数对应的力矩输出。如下图 2.5 所示

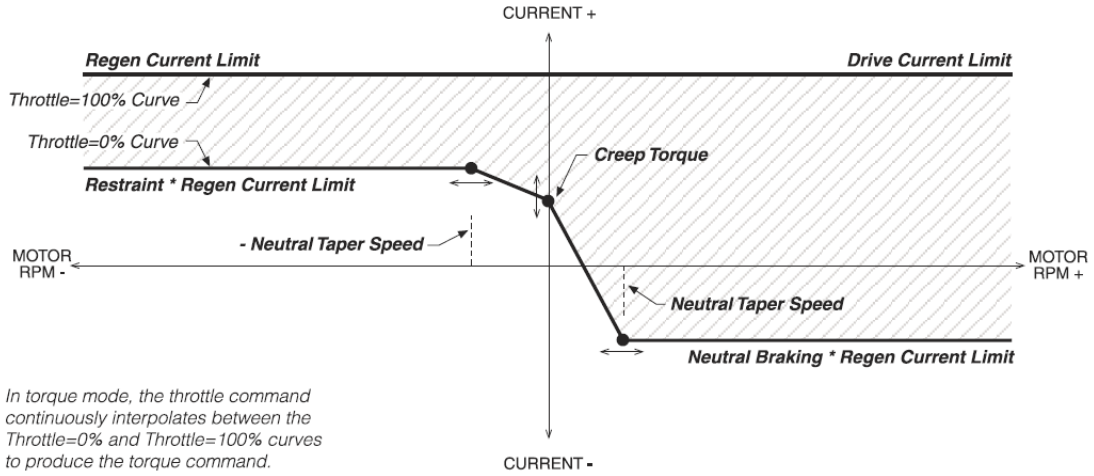


图 2.5 油门百分数对应的力矩输出

相关的函数参数说明

```
word     DriveLimit;              //驱动最大力矩  pu
word     RegenLimit;              //制动最大力矩  pu
word     CreepTorque;             //启动力矩  pu
int   ReversalSoftenSpeed;        //驱动和制动最大力矩不一致时，使用
int   NeutralTaperSpeed;          //空档制动力矩变化速度     pu
word     NeutralBrakeTorque;      //空档制动力矩百分数
word     RestraintTorque_F;       // 正向时制动最大力矩百分数
word     RestraintTorque_B;       //反向时制动最大力矩百分数
// input
int   SpeedInput;                 // 速度输入          pu
word     DirectionInput;          // 方向判断标志
word     ThrottleInput;           //油门信号百分数输入
// output
int    TorqueOut;                 // 力矩输出    pu
```

代码运算使用 Q15 格式。

2.7 TMBrakeResponse.c

此函数主要用于力矩模式，用于计算刹车踏板对应的力矩输出。如下图 2.6 所示。在电机速度在 Brake Taper Speed 设置值以上以恒定力矩制动，当在设置值以下，随着速度下降，力矩也随之下降，直到速度为零力矩为零。

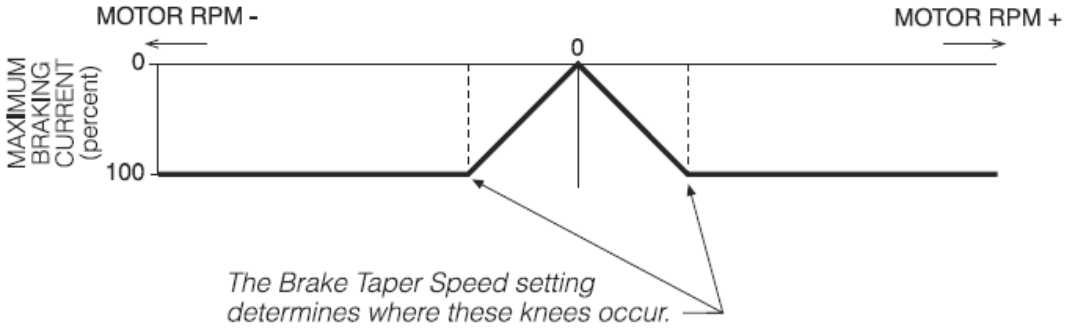


图 2.6　踏板百分数对应的刹车力矩

相关函数参数说明

```
word    BrakeLimit; //刹车力矩 pu
int   BrakeTaperSpeed;//刹车力矩减小点速度
// input
int   SpeedInput;//速度输入
word    BrakeInput;//刹车输入
// output
int   TorqueOut;//力矩输出
```

代码运算使用 Q15 格式。

2.8 SMThrottleResponse.c

此函数用于速度模式，用于计算速度模式下的油门对应的速度加速时间。如下图 2.7 所示。

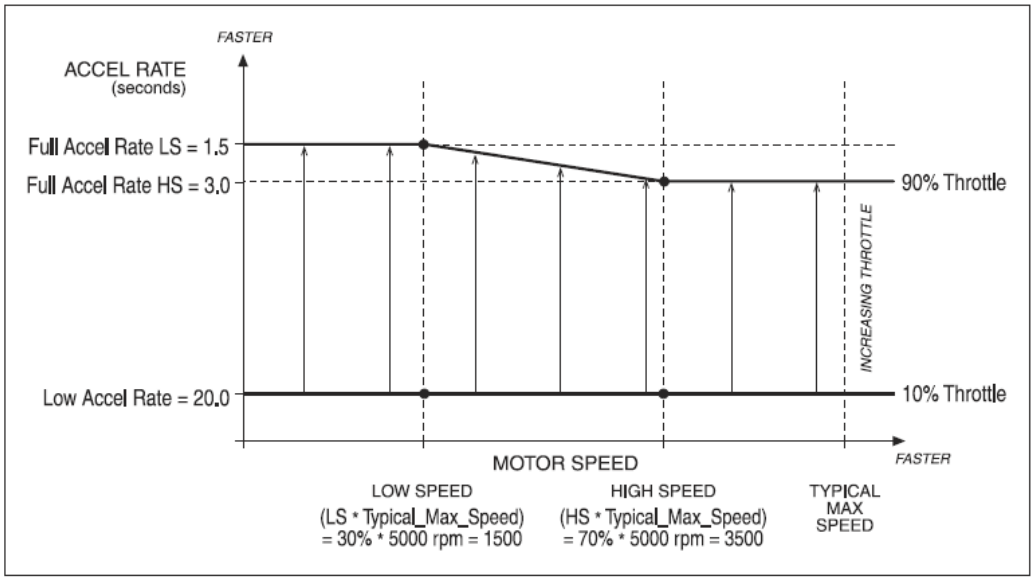


图 2.7 速度模式下，油门百分数对应的加速时间

函数相关参数说明

```
word    HSFullThrottleStep;    //高速满油门对于的加速时间
word    LSFullThrottleStep;    //低速满油门对于的加速时间
word    LowThrottleStep;       //小油门对应的加速时间
int     LowSpeed;              //低速度设定值
int     HighSpeed;             //高速度设定值
// input
int     SpeedInput;            //  速度输入值
word    DirectionInput;        //方向标志位
word    ThrottleInput;         //油门踏板的百分数输入
// output
int   StepOut;                 //  当前输出的时间步长
```

代码运算使用 Q15 格式。

2.9 SMBrakeResponse.c

此函数用于速度模式下，用于计算速度模式下的刹车速度减速时间。如图 2.8 所示。

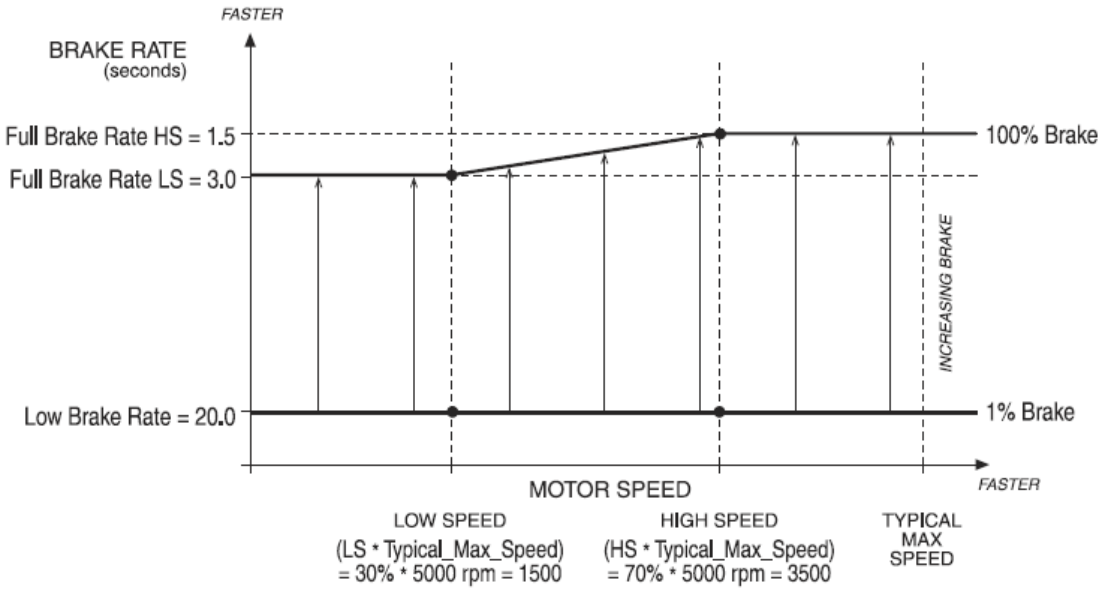


图 2.8 速度模式下，刹车百分数对应的加速时间

函数相关参数说明

```
word  HSFullBrakeStep;     //高速满油门下，刹车减速时间
word  LSFullBrakeStep;     //低速满油门下，刹车减速时间
word  LowBrakeStep;        //小油门下，减速时间
int     LowSpeed;          //低速度设定值
int     HighSpeed;         //高速度设定值
// input
int       SpeedInput;      //  速度输入
word BrakeInput;           //  刹车百分数输入
// output
int    StepOut;            //当前计算的的减速时间
```

代码运算使用 Q15 格式。

2.10 checkio.c

此函数主要用于滤波 IO 口输入信号，判断当前 IO 口状态的变化。设定计数次数后，当检测到 IO 口的状态维持到计数次数到之后不发生变化，说明当前 IO 口的状态是此状态。

相关函数参数说明

word Check_IO(word IO,int* pdata,word CountNum)

IO 表示当前的 IO 口状态值，,int* pdata 用于计数的指针参数，CountNum 需要计数的次数设定值。

2.11 TimeBasedRamp.C

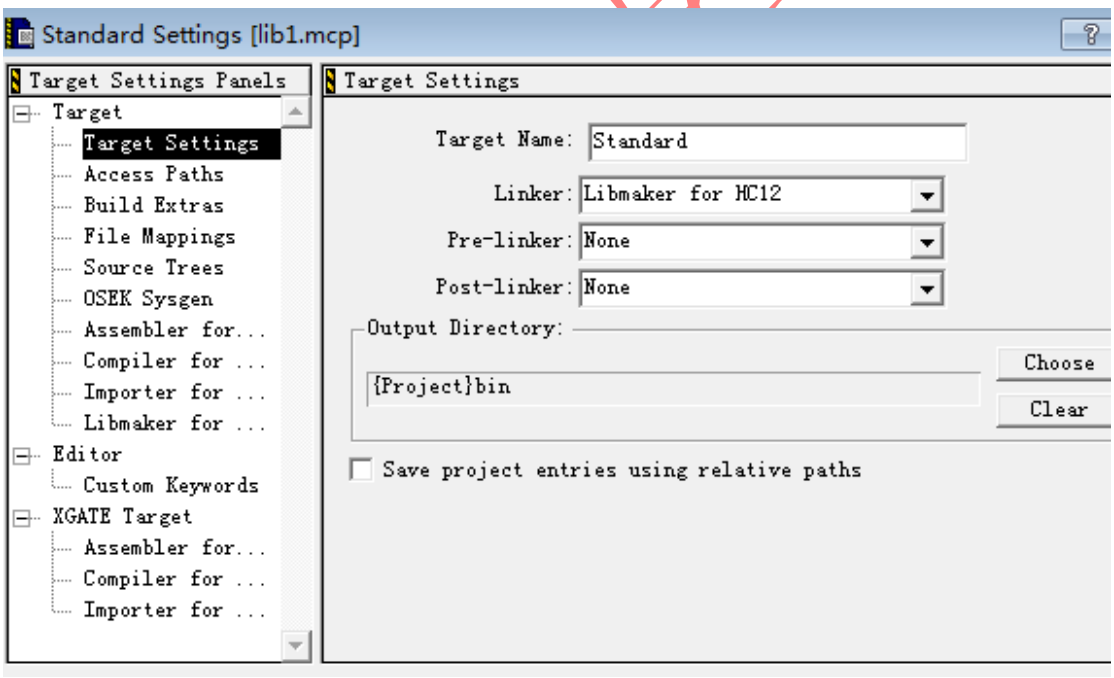
此函数是一个斜坡函数，主要对目标值输出做斜坡处理。

如力矩的加速时间，决定了斜坡函数的步长。速度加速时间也决定了斜坡的步长。相关函数参数说明

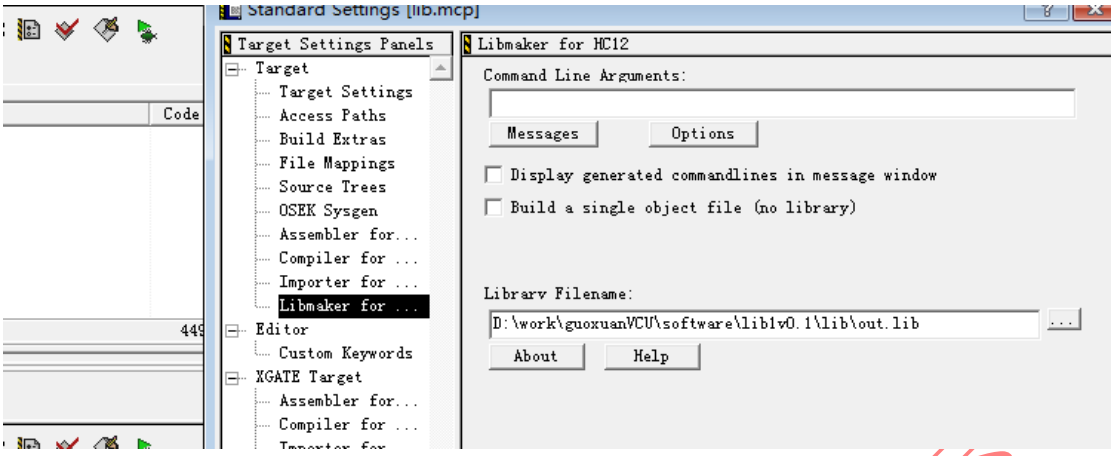
```
int TargetValue;        // Input: Target input (pu)
int RampLowLimit;       // Parameter: Minimum limit (pu)
int RampHighLimit;      // Parameter: Maximum limit (pu)
int SetpointValue;      // Output: Target output (pu)
int RampUpStep;         // Parameter: Ramp Up Step
int RampDownStep;       // Parameter: Ramp Down Step
word EqualFlag;         // Output: Flag output
```

2.12 lib 库说明

关于库的生成，在工程中打开如下图所示窗口 ，将 Linker 设置为 Libmaker for HC12 即可。



关于库生产的存放位置，如下图中路径所示。



此图中的路径说明了库文件产生的路径，会生成两个文件如下

将两个文件复制到主工程文件夹下的 lib 文件夹里，然后添加到主工程中调用。

另一种比较简单的方法就是直接修改路径，将路径改为你需要放在库文件的路径。

在使用库文件的同时，需要将库文件中的头文件加入到主工程中，同时建立一个头文件，名为 lib.h 。将需要的头文件加入，下次调用直接用此头文件即可。

```
#ifndef _LIB_H
#define _LIB_H

#include "filter_lp.h"
#include "limit.h"
#include "los.h"
#include "refmap.h"
#include "throttle_map.h"
#include "TMThrottleResponse.h"
#include "TMBrakeResponse.h"
#include "SMThrottleResponse.h"
#include "SMBrakeResponse.h"
#include "checkio.h"
#include "TimeBasedRamp.h"
#endif
```

三、第二部分工程函数说明

3.1 函数说明

3.1.1 atd.c

此函数主要是配置 ADC 采样口，本产品中使用 ADC0,1,2,3 四路，配置为 12 位模数转换，右对齐模式，模块时钟频率为 4MHz。

3.1.2 can.c

此函数主要用于配置 CAN 通信模块，本产品使用两路 CAN ，CAN0 波特率为 500，CAN1 波特率为 250。由于本产品使用的 CAN 通信协议是基于 J1939 协议修改的，因此使用标识符为 29 位扩展帧。

3.1.3 pit.c

此函数主要用于配置中期中断定时器，定时器 0 定时周期为 1ms ,定时器 1 定时周期为 0.125ms。定时器作为采样和相关逻辑处理的时基。

3.1.4 sci.c

此函数主要配置串口通信模块，设置 SCI0 为正常模式，八位数据位，无奇偶校验，波特率为 12500 。同时使能接收中断。485 相关的底层协议解析处理也在此函数中完成。应用数据解析在 hmi.c 中完成。但此文件调用其中的处理函数。

3.1.5 vehicle.c

此函数主要定义整车参数的默认初始值。作为备用数据的储存。

3.1.6 eeprom.c

此函数主要是对 eeprom 的处理，包括初始化，功能设置和相关的数据读写。

3.1.7 crc16.c

文件中包含两个用于数据校验的函数 CRC 校验计算函数和 LRC 计算校验函数。

3.1.8 hmi.c

主要是对 485 协议的数据应用层进行解析，对数据进行读写操作，同时将相关的数据写入 eeprom 中。此文件中还包含一个实时采样上传函数，此函数可实现采样数据的翻页功能。

3.1.9 VCUctr.c

此文件中包含多个函数，主要的整车控制功能都在其中实现。下面对各个函数做相关介绍。

1.void Vcuctr_Init(void)

此函数主要用于整车的参数的初始化设置。将 eeprom 中读到的数据填入相应的设置值中。

2.void CAN_Process(void)

此函数运行在周期定时器 1 的中断函数中。对 CAN 进行接收判断，然后将 CAN 接收到的数据根据 ID 进行数据的解码。

3.void CAN_SendTimeReg(void)

CAN 通信发送时间处理函数。此函数运行在周期定时器 0 的中断函数中。对 CAN 的发送数据做计时用。相应的时间到，标志位置 1。

4.void CAN_MsgUpdate(void)

函数运行在 1ms 任务中。更新相关需要发送的 CAN 数据的内容信息。

5.void CAN_Transfer(void)

CAN 数据发送函数，函数运行在 1ms 任务中，根据 CAN 发送时间处理函数的计时，发送 CAN 的信息数据。

6.void System_Check(void)

对错误检测与处理函数。功能是根据 can 通信的解码内容 判断各个外围系统状态，有故障进行处理，保存故障前的状态和各个量值情况， 并进入 FLAUT_STATUE (相关给定值为零，如果故障消失，恢复之前状态)，记录故障发生次数于 EEPROM 中。

7.void SignalProcess(void)

对油门刹车等相关信号检测与处理函数 ，运行在 4ms 任务。对油门踏板和刹车踏板的输入进行处理，调用相关的函数计算出控制的百分数。同时此函数调用 void System_Check(void) 函数。

8.void VEHICLE_refreshState(void)

整车状态刷新函数。此函数主要对整车的状态进行处理，当 VCU 内部 eeprom 读取正常后，进入整车准备状态。

在准备状态中，判断外部 BMS 自检是否通过 ，然后对充电信号进行采集，判断是否进入充电状态，只要在此时才能进入充电状态。在没有充电信号后，判断钥匙开关位置，并进行相关的继电器动作。当处于 SATRT 位置一定时间，将进入启动状态。启动状态下，判断电机控制器和电池管理系统没有故障的情况下，使能电机控制器进入工作状态，当电机控制器准备就绪后，判断是否有高踏板保护，如果有等待保护消去，如果没有进去驱动状态。驱动状态下，先判断当前的电机控制模式是力矩模式还是速度模式。不同的模式下调用不用的处理函数。根据当前的档位信号，进行 DNR 判断。在前进档，调用前进处理函数，后退档调用后退处理函数，中档调用中档处理函数。一旦检测到刹车信号，进行刹车处理。在充电状态下，如果充电机没有故障，闭合充电接触器，通知 BMS 允许充电。然后进入充电进行状态。充电进行状态下，主要判断是否断开充电连接，如果断开连接，关闭充电接触器，进入整车准备状态。当有错误发生时，进入错误状态。在错误状态下，主要进行相关数据的清零和错误的恢复。

9.void Get_VehicleInput(word* pData)

整车信号输入处理函数。对数字量和模拟量的采集。运行于周期定时器 1 中断函数中。

10.void SetTorqueCommand(void)

力矩模式下对输出力矩设定值的处理函数，包括力矩的斜坡，减载和力矩限制处理。最终通过 CAN 通信发送给电机控制器，运行于 4ms 的任务中。

11.void SetSpeedCommand(void)

速度模式下对速度设定值的处理。主要是对速度的斜坡处理。最终通过 CAN 通信发送给电机控制器，运行于 4ms 的任务中。

12.void TMForwardThrottleHandle(void)

力矩模式前进方向油门处理函数。在前进档时，根据油门的输入百分数和当前的电机速度调用相关库函数计算力矩输出值。同时根据设定值，判断加减速时间。

13.void TMBackwardThrottleHandle(void)

力矩模式后退方向油门处理函数。在后退档时，根据油门的输入百分数和当前的电机速度调用相关库函数计算力矩输出值。同时根据设定值，判断加减速时间。

14.void TMNeutralThrottleHandle(void)

力矩模式中档处理函数。在中档时，根据转速和中档制动力矩的设定值，计算出最后的力矩输出值和相应的斜坡步长。

15.void TorquemodeBrakeHandle(void)

力矩模式刹车处理函数。根据刹车踏板百分数和当前的转速计算出制动力矩的输出值，同时计算相应的斜坡步长。

16.void SpeedmodeBrakeHandle(void)

速度模式刹车处理函数。根据当前转速和刹车踏板百分数计算速度斜坡的步

长，决定减速的时间。

17.void SMForwardThrottleHandle(void)

速度模式前进方向油门处理函数。根据油门踏板的输入百分数和当前的转速计算出前进档下速度的斜坡步长。

18.void SMForwardThrottleHandle(void)

速度模式后退方向油门处理函数。根据油门踏板的输入百分数和当前的转速计算出后退档下速度的斜坡步长。

19.void SMNeutralThrottleHandle(void)

速度模式中档处理函数。根据当前的转速计算出中档下速度的斜坡步长。

20.void SystemShutDown(void)

系统给定目标值清零函数。在故障和非法指令中断时，调用此函数对相应的目标设定值清零。

3.2 相关中断说明

本代码使用了多个中断。SCi 的接收中断，主要用于接收数据，存于定义的数组缓存中。

周期定时器 0 中断，时间为 1ms，主要用于主循环的任务时基和 CAN 通信的发送数据计时处理。

中期定时器 1 中断，时间为 0.125ms，主要用于对 ADC 数据的采集和整车的输入信号处理。485 接受与发送数据函数和 CAN 通信接收并处理数据函数也处于此中断中。

指令错误中断主要清零数据和复位芯片。

3.3 主循环任务说明

主循环任务有三个，1ms 任务，4ms 任务和 5 分钟任务。

1ms 任务中函数有：

```
CAN_MsgUpdate();          //更新 CAN 通信的数据内容
CAN_Transfer();           //发送 CAN 通信
Rs485_Process(&rs485); //对 485 通信接收内容进行解码和发送内容编码
```

4ms 任务中函数有：

```
SignalProcess();          //整车信号处理
VEHICLE_refreshState(); //整车状态刷新
//根据电机工作模式，决定不同命令
if(Vehicle.VehiclePara.MotorWorkMode == TORQUE_MODE)
{
   SetTorqueCommand();
}
else
{
   SetSpeedCommand();
}
```

5min 任务中主要记录电池电量和里程等数据。