**宏定义：**

#define TIME\_INTERVAL 2 //时间间隔

#define DEBUG\_PRINT\_TIMME\_ON 1 //调试输出采集数据

#if DEBUG\_PRINT\_TIMME\_ON

#define DEBUG\_PRINT\_TIMME(fmt,args...) printf (fmt ,##args)

#else

#define DEBUG\_PRINT\_TIMME(fmt,args...)

#endif

**变量定义：**

INT32U StartTime[100] = {0}; //数组，起始时间，基准值，记录订单下发时，硬件系统中的挡墙时钟节拍。

InternetTime current\_internet\_time[100]; //结构体数组，该数组中每个元素是一个类型为InternetTime的变量，用于同一批次不同订单下发的时间。

**结构体定义：**

typedef struct

{

u32 hour;

u32 min;

u32 sec;

u32 msec;

}InternetTime;

**函数定义：**

/\*\*

\* @brief 增加时间

\* @param time-增加的毫秒，current\_internet\_time-当前时间，结构体类型InternetTime

\* @return 执行结果

\*/

void AddTimes(INT32U time, InternetTime current\_internet\_time)

{

int rhour = 0, rmin = 0, rsec = 0, rmsec = 0;//时分秒 毫秒

OS\_CPU\_SR cpu\_sr = 0u;

// 此函数有BUG，当原本是900ms ，新增2600ms 结果的ms理应为500，而不是1500

// 经检查发现，ms未完全转化成s，待修复

if (time > 999)//大于999毫秒

{

rsec = current\_internet\_time.sec + time / 1000;//赋值给秒

rmsec = current\_internet\_time.msec + time % 1000;//赋值给毫秒

if (rsec > 59)//大于59秒

{

rmin = current\_internet\_time.min + rsec / 60;//赋值给分

rsec = rsec % 60;//赋值给秒

if (rmin > 59)//大于59分

{

rhour = current\_internet\_time.hour + rmin / 60;//赋值给时

rmin = rmin % 60;//赋值给分

}

else

{

rhour = current\_internet\_time.hour;//赋值给时

}

}

else

{

rmin = current\_internet\_time.min;//赋值给分

rhour = current\_internet\_time.hour;//赋值给时

}

}

else//小于等于999毫秒 直接赋值

{

rmsec = current\_internet\_time.msec + time;//赋值给毫秒

rsec = current\_internet\_time.sec;//赋值给秒

rmin = current\_internet\_time.min;//赋值给分

rhour = current\_internet\_time.hour;//赋值给时

}

DEBUG\_PRINT\_TIMME("时间：%d:%d:%d:%d\r\n",rhour, rmin, rsec, rmsec);

}

**使用方式：**

置DEBUG\_PRINT\_TIMME\_ON为1即可查看输出的采集信息，其中：

StartTime：当前时钟节拍，作为基准值

序号：当前批次内的订单的序号

时间：依次为时、分、秒、毫秒

输出如下：

StartTime:494784，序号1

订单下发时间15:58:4:475，序号为1，

订单编号：344

订单长度：480

订单进入缓冲区，容量为9732，序号为1，时间：15:58:5:35

分配内存大小：1K，序号为1，时间：15:58:5:63

进入打印队列开始，序号为1，时间：15:58:5:99

打印成功结束，序号为1，时间：15:58:5:221

释放内存块，序号为1，时间：15:58:5:221