```
/*
  This is Time.h, the .h file for the Time library
  This library is implements low level time and date functions

  ------------------------------------------------------
  This code is found online. It was not written by team 26
  ------------------------------------------------------

  July 3 2011 - fixed elapsedSecsThisWeek macro (thanks Vincent Valdy for this)
              - fixed  daysToTime_t macro (thanks maniacbug)
*/

#ifndef _Time_h
#ifdef __cplusplus
#define _Time_h

#include <inttypes.h>
#ifndef __AVR__
#include <sys/types.h> // for __time_t_defined, but avr libc lacks sys/types.h
#endif


#if !defined(__time_t_defined) // avoid conflict with newlib or other posix libc
typedef unsigned long time_t;
#endif


// This ugly hack allows us to define C++ overloaded functions, when included
// from within an extern "C", as newlib's sys/stat.h does.  Actually it is
// intended to include "time.h" from the C library (on ARM, but AVR does not
// have that file at all).  On Mac and Windows, the compiler will find this
// "Time.h" instead of the C library "time.h", so we may cause other weird
// and unpredictable effects by conflicting with the C library header "time.h",
// but at least this hack lets us define C++ functions as intended.  Hopefully
// nothing too terrible will result from overriding the C library header?!
extern "C++" {
typedef enum {timeNotSet, timeNeedsSync, timeSet
}  timeStatus_t ;

typedef enum {
    dowInvalid, dowSunday, dowMonday, dowTuesday, dowWednesday, dowThursday, dowFriday,
    dowSaturday
} timeDayOfWeek_t;

typedef enum {
    tmSecond, tmMinute, tmHour, tmWday, tmDay,tmMonth, tmYear, tmNbrFields
} tmByteFields;

typedef struct  {
  uint8_t Second;
  uint8_t Minute;
  uint8_t Hour;
  uint8_t Wday;   // day of week, sunday is day 1
  uint8_t Day;
  uint8_t Month;
  uint8_t Year;   // offset from 1970;
}   tmElements_t, TimeElements, *tmElementsPtr_t;

//convenience macros to convert to and from tm years
#define  tmYearToCalendar(Y) ((Y) + 1970)  // full four digit year
#define  CalendarYrToTm(Y)   ((Y) - 1970)
#define  tmYearToY2k(Y)      ((Y) - 30)    // offset is from 2000
#define  y2kYearToTm(Y)      ((Y) + 30)

typedef time_t(*getExternalTime)();
//typedef void  (*setExternalTime)(const time_t); // not used in this version


/*=====================================================================*/
```

```
69  /* Useful Constants */
70  #define SECS_PER_MIN  ((time_t)(60UL))
71  #define SECS_PER_HOUR ((time_t)(3600UL))
72  #define SECS_PER_DAY  ((time_t)(SECS_PER_HOUR * 24UL))
73  #define DAYS_PER_WEEK ((time_t)(7UL))
74  #define SECS_PER_WEEK ((time_t)(SECS_PER_DAY * DAYS_PER_WEEK))
75  #define SECS_PER_YEAR ((time_t)(SECS_PER_DAY * 365UL)) // TODO: ought to handle leap
    years
76  #define SECS_YR_2000  ((time_t)(946684800UL)) // the time at the start of y2k
77
78  /* Useful Macros for getting elapsed time */
79  #define numberOfSeconds(_time_) ((_time_) % SECS_PER_MIN)
80  #define numberOfMinutes(_time_) (((_time_) / SECS_PER_MIN) % SECS_PER_MIN)
81  #define numberOfHours(_time_) (((_time_) % SECS_PER_DAY) / SECS_PER_HOUR)
82  #define dayOfWeek(_time_) ((((_time_) / SECS_PER_DAY + 4)  % DAYS_PER_WEEK)+1) // 1 =
    Sunday
83  #define elapsedDays(_time_) ((_time_) / SECS_PER_DAY)  // this is number of days since
    Jan 1 1970
84  #define elapsedSecsToday(_time_) ((_time_) % SECS_PER_DAY)   // the number of seconds
    since last midnight
85  // The following macros are used in calculating alarms and assume the clock is set to a
    date later than Jan 1 1971
86  // Always set the correct time before settting alarms
87  #define previousMidnight(_time_) (((_time_) / SECS_PER_DAY) * SECS_PER_DAY)  // time at
    the start of the given day
88  #define nextMidnight(_time_) (previousMidnight(_time_)  + SECS_PER_DAY)   // time at
    the end of the given day
89  #define elapsedSecsThisWeek(_time_) (elapsedSecsToday(_time_) +  ((dayOfWeek(_time_)-1)
    * SECS_PER_DAY))   // note that week starts on day 1
90  #define previousSunday(_time_) ((_time_) - elapsedSecsThisWeek(_time_))      // time at
    the start of the week for the given time
91  #define nextSunday(_time_) (previousSunday(_time_)+SECS_PER_WEEK)          // time at
    the end of the week for the given time
92
93
94  /* Useful Macros for converting elapsed time to a time_t */
95  #define minutesToTime_t ((M)) ( (M) * SECS_PER_MIN)
96  #define hoursToTime_t   ((H)) ( (H) * SECS_PER_HOUR)
97  #define daysToTime_t    ((D)) ( (D) * SECS_PER_DAY) // fixed on Jul 22 2011
98  #define weeksToTime_t   ((W)) ( (W) * SECS_PER_WEEK)
99
100 /*============================================================================*/
101 /*  time and date functions   */
102 int     hour();             // the hour now
103 int     hour(time_t t);     // the hour for the given time
104 int     hourFormat12();     // the hour now in 12 hour format
105 int     hourFormat12(time_t t); // the hour for the given time in 12 hour format
106 uint8_t isAM();             // returns true if time now is AM
107 uint8_t isAM(time_t t);     // returns true the given time is AM
108 uint8_t isPM();             // returns true if time now is PM
109 uint8_t isPM(time_t t);     // returns true the given time is PM
110 int     minute();           // the minute now
111 int     minute(time_t t);   // the minute for the given time
112 int     second();           // the second now
113 int     second(time_t t);   // the second for the given time
114 int     day();              // the day now
115 int     day(time_t t);      // the day for the given time
116 int     weekday();          // the weekday now (Sunday is day 1)
117 int     weekday(time_t t);  // the weekday for the given time
118 int     month();            // the month now  (Jan is month 1)
119 int     month(time_t t);    // the month for the given time
120 int     year();             // the full four digit year: (2009, 2010 etc)
121 int     year(time_t t);     // the year for the given time
122
123 time_t now();               // return the current time as seconds since Jan 1 1970
124 void    setTime(time_t t);
125 void    setTime(int hr,int min,int sec,int day, int month, int yr);
126 void    adjustTime(long adjustment);
127
```

```
128  /* date strings */
129  #define dt_MAX_STRING_LEN 9 // length of longest date string (excluding terminating null)
130  char* monthStr(uint8_t month);
131  char* dayStr(uint8_t day);
132  char* monthShortStr(uint8_t month);
133  char* dayShortStr(uint8_t day);
134
135  /* time sync functions  */
136  timeStatus_t timeStatus(); // indicates if time has been set and recently synchronized
137  void    setSyncProvider( getExternalTime getTimeFunction); // identify the external
     time provider
138  void    setSyncInterval(time_t interval); // set the number of seconds between re-sync
139
140  /* low level functions to convert to and from system time                     */
141  void breakTime(time_t time, tmElements_t &tm);  // break time_t into elements
142  time_t makeTime(const tmElements_t &tm);  // convert time elements into time_t
143
144  } // extern "C++"
145  #endif // __cplusplus
146  #endif /* _Time_h */
147
```