Micromega Corporation

# Application Note 37

# Working with Dates and Times

## Introduction

Date and time information can be represented in a variety of ways. For example, it can be stored as the number of seconds that have elapsed since a know time, or as separate fields for seconds, minutes, hours, days, etc. This application note describes an implementation of the Unix Time format for working with date and time values. It also shows how the `TIMESET`, `TIMELONG` and `TICKLONG` instructions can be used for keeping track of elapsed time and time delays.

## Unix Time

Unix Time is a method of storing time as a 32-bit value specifying the number of seconds that have elapsed since midnight UTC of January 1, 1970. It has been widely used on Unix operating systems and many other systems, and is usually part of the standard libraries provided with C compilers. There are a number of free web-based utilities available for converting between Unix Time and readable date/time strings. Unix time can be easily implemented on the uM-FPU V3 chip since there's comprehensive support for 32-bit integer instructions. The examples provided with this application note handle Unix Time values from January 1, 1970 00:00:00 to January 19, 2038 03:14:07.

## *unixTime.fpu* Functions

A set of uM-FPU V3.1 user-defined functions have been implemented to convert between individual date and time fields and 32-bit Unix Time values. Functions to convert date and time fields to text strings are also included. The functions in *unixTime.fpu* use the following registers:

| Register Name | Description | Range of 32-bit Integer Values Min | Max | |
|---|---|---|---|---|
| tm_sec | seconds | 0 | 59 | |
| tm_min | minutes | 0 | 59 | |
| tm_hour | hours | 0 | 23 | |
| tm_mday | day of month | 1 | 31 | |
| tm_mon | months since January | 0 | 11 | (note: not 1 to 12) |
| tm_year | years since 1900 | 70 | 138 | |
| tm_wday | days since Sunday | 0 | 6 | |
| tm_yday | days since January 1 | 0 | 365 | |
| tm_unix | 32-bit Unix Time value | 0 | 2147483647 | |

### dateToUnix

This function converts individual date and time fields to a 32-bit Unix Time value.

*Input*:

| | | |
|---|---|---|
| register tm_sec | 32-bit integer | seconds |
| register tm_min | 32-bit integer | minutes |
| register tm_hour | 32-bit integer | hour |
| register tm_mday | 32-bit integer | day of month |
| register tm_mon | 32-bit integer | months since January |

| | | |
|---|---|---|
| register tm_year | 32-bit integer | years since 1970 |

*Output*:

| | | |
|---|---|---|
| register tm_unix | 32-bit integer | Unix Time |

### unixToDate

This function converts a 32-bit Unix Time value to individual date and time fields.

*Input*:

| | | |
|---|---|---|
| register tm_unix | 32-bit integer | Unix Time |

*Output*:

| | | |
|---|---|---|
| register tm_sec | 32-bit integer | seconds |
| register tm_min | 32-bit integer | minutes |
| register tm_hour | 32-bit integer | hour |
| register tm_mday | 32-bit integer | day of month |
| register tm_mon | 32-bit integer | months since January |
| register tm_year | 32-bit integer | years since 1970 |
| register tm_wday | 32-bit integer | days since Sunday |
| register tm_yday | 32-bit integer | days since January 1 |
| register tm_unix | 32-bit integer | Unix Time |

### getDateTimeStamp

This function converts individual date and time fields to a text string that is stored in the FPU string buffer. The format of the string is YYYY-MM-DD HH:MM:SS.

*Input*:

| | | |
|---|---|---|
| register tm_sec | 32-bit integer | seconds |
| register tm_min | 32-bit integer | minutes |
| register tm_hour | 32-bit integer | hour |
| register tm_mday | 32-bit integer | day of month |
| register tm_mon | 32-bit integer | months since January |
| register tm_year | 32-bit integer | years since 1970 |

*Output*:

| | | |
|---|---|---|
| string buffer | string | *YYYY-MM-DD HH:MM:SS*<br>e.g. `2007-07-19 09:16:20` |

### getDateString

This function converts individual date fields to a text string that is stored in the FPU string buffer. The format of the string is Www, Mmm DD/YY.

*Input*:

| | | |
|---|---|---|
| register tm_sec | 32-bit integer | seconds |
| register tm_min | 32-bit integer | minutes |
| register tm_hour | 32-bit integer | hour |
| register tm_mday | 32-bit integer | day of month |
| register tm_mon | 32-bit integer | months since January |
| register tm_year | 32-bit integer | years since 1970 |

*Output*:

| | | |
|---|---|---|
| string buffer | string | *www, mmm DD/YY*<br>e.g. `Thu, Jul 19, 2007` |

### insertDigits

Converts an integer value to a two-digit (leading zero) string and stores it at the current string selection point.

*Input*:

| | | |
|---|---|---|
| register 1 | 32-bit integer | 0 to 99 |

*Output*:

| | | |
|---|---|---|
| string selection | string | *nn*<br>e.g. `19` |

**insertWeekDay**

Converts the tm_wday value to a three character abbreviation for the weekday and stores it at the current string selection point.

*Input*:

| | | |
|---|---|---|
| register tm_wday | 32-bit integer | days since Sunday |

*Output*:

| | | |
|---|---|---|
| string selection | string | *www*<br>e.g. `Thu` |

**insertMonth**

Converts the tm_mon value to a three character abbreviation for the month and stores it at the current string selection point.

*Input*:

| | | |
|---|---|---|
| register tm_mon | 32-bit integer | months since January |

*Output*:

| | | |
|---|---|---|
| string selection | string | *mmm*<br>e.g. `Jul` |

## Using the TIMESET, TIMELONG and TICKLONG instructions

The uM-FPU V3 chip has two 32-bit elapsed time counters. One counter tracks elapsed time in seconds, and the other tracks elapsed time in milliseconds. By default, the time counters are disabled, but they can be enabled using the `TIMESET` instruction. The `TIMESET` instruction sets the initial value of the seconds counter, clears the milliseconds counter, and enables the counters. The `TIMELONG` instruction is used to read the seconds counter, and the `TICKLONG` instruction is used to read the milliseconds counter. See the *uM-FPU V3.1 Instruction Set* for details.

The millisecond counter is used for measuring small time intervals. It can run for 4294967.295 seconds (~49.7 days) before it rolls over to zero. The seconds counter can be used for longer time periods, or to keep track of date and time. For example, the seconds counter could be loaded with the Unix Time value. However, it should be noted, that the elapsed time counters run from the internal clock. This internal clock frequency can vary up to 0.1% from the specified frequency, which is not significant for most operations, but is not accurate enough for keeping track of time over an extended period. For applications that require precise time over an extended period, the counter could be periodically reset (e.g. with a time signal from a GPS receiver, or other time source), or an external realtime clock chip could be used to keep track of the time, while the FPU is used to work with time values.

## Elapsed Time Calculations

Elapsed time calculations can be done simply by subtracting two time values. For example, the following code calculates the elapsed time in milliseconds for a section of code. Note: elapsed time in seconds can be calculated by replacing the `TICKLONG` instruction with `TIMELONG`.

```
    SELECTA, 1              ; get start time
    TICKLONG
    LSET0
( code sequence being timed)
    SELECTA, 2              ; get end time
    TICKLONG
```

```
    LSUB, 1                 ; calculate elapsed time
```

## Time Delays

Time delays can be implemented in uM-FPU user-defined functions. The following routine assumes the timer is already running, and register 1 specifies the number of milliseconds to delay. Note: a time delay in seconds can be implemented by replacing the `TICKLONG` instruction with `TIMELONG`.

```
    SELECTA, 1              ; get current time
    TICKLONG
    LADD0                   ; add the delay amount

_wait:
    TICKLONG                ; wait until current time >= delay time
    LUCMP0
    BRA, GT, _wait
```

The example shown above assumes that the counters will not rollover to zero during the delay loop. Fot the milliseconds timer this will occur approximately every 49.7 days. If this is a concern, it can be avoided by periodically resetting the timers, or always resetting the timer before a time delay.

## Additional Files

There are additional files located on the Micromega website that accompany this application note. They include:

| | |
|---|---|
| *unixTime.fpu* | contains uM-FPU V3.1 user-defined functions |
| *unixTime.bs2* | Basic Stamp demo application |
| *unixTime.bas* | PICAXE demo application |

Before running the demo application, the user-defined functions in *unixTime.fpu* must be programmed into the uM-FPU V3.1 chip. This can be done using the uM-FPU V3 IDE software.

## Further Information

See the Micromega website (http://www.micromegacorp.com) for additional information regarding the uM-FPU V3.1 floating point coprocessor, including:

*uM-FPU V3.1 Datasheet*
*uM-FPU V3.1 Instruction Set*
*Using the uM-FPU V3 Integrated Development Environment (IDE)*