AcroTEX.Net

The cntdwn Package

Creating short and long countdowns

D. P. Story

Prepared: September 1, 2010

www.acrotex.net
Version 1.0

Table of Contents

1. Introduction

The cntdwn package provides two types of countdowns, short and long.

- A short countdown, accessed through the `shortcount` option, is a countdown (or count-up) that is for a relatively short time period (less than a day). Such a countdown is designed for a talk (or presentation).
- A long countdown, accessed through the `longcount` option, is a countdown to a distant event, perhaps many days or even years in the future.
- As a bonus of long countdown, a clocks can be defined for local or time zones.

Each type of countdown (clocks excepted) has several events: (1) the main event, which is the target of the countdown; (2) pre-events, events that occur *before* the occurrence of the main event; and (3) post-events, events that occur *after* the occurrence of the main event. Each event occurs at a definable instant in time, and may have a (JavaScript) action associated with the event.

2. Requirements and Sample files

The cntdwn requires the eforms package, part of AeB (the AcroTEX eDucation Bundle), and, of course, the hyperref package.

The PDFs need to be viewed in Adobe Reader (or Acrobat), not some other nonconforming PDF viewer that does not support document level JavaScript.

Basic examples are provided in the `examples` folder; advanced examples can be found, in time, on my AeB Blog site, http://www.math.uakron.edu/~dpstory/aebblog.html.

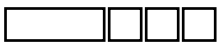
3. The short countdown

To input the code for the short count, type either

`\usepackage{cntdwn}` or `\usepackage[shortcount]{cntdwn}`

in the preamble.

We begin with an example of the default behavior of a short countdown:



To start the countdown, press the Start button (the second form field from the left). The left-most field is a text field, the others are buttons. The original idea behind this behavior was that this seemed a nice way to do a count for a talk: (1) at a preselected time (45 seconds in this example) the first button would turn green indicating to the speaker that time is running out; (2) a little later, the second notification signal appears, a yellow button (at 30 seconds); at a third notification time, the third button starts blinking red (at 15 seconds), this tells the speaker to wrap it up; (4) finally, at the end of the defined length of the timer, the third button stops blinking indicating that the speaker's time is up.

The buttons also play roles as controls over the counter, the three buttons are **Start**, **Pause**, and **Stop**. The user can press the **Pause** button to pause the count (without taking away from his/her time), then restart it by pressing the **Start** button.

The code for the above countdown consisted to two sets of lines. In the preamble, we find,

```
\setShortCntDwn{Timer1}{%
    length=1*\minutes,
    notify1=45*\seconds,
    notify2=30*\seconds,
    notify3=15*\seconds
}
```

This defines a timer name `Timer1`, with key-value pairs give above. The length of the countdown is 1 minutes, following by the defining of three notification times. The mean of these key-values is clear in light of the above example.

The second line of code is the laying down of the timers themselves.

```
\cntdwnDisplay{Timer1}{.5in}{11bp}
\cntdwnStartT{Timer1}{11bp}{11bp}
\cntdwnPauseT{Timer1}{11bp}{11bp}
\cntdwnStopT{Timer1}{11bp}{11bp}
```

The first argument of each is the name of the timer to use, `Timer1`, in this case. All of these form fields are optional; if none is included in the document, the timer ticks away silently.[1]

3.1. Using `\setShortCntDwn` in the preamble

For each short countdown timer, the `\setShortCntDwn` must be used to define the properties of the timer.

```
\setShortCntDwn{<t_name>}{<key-values>}
```

Command Description: The command defines the properties of the countdown. Internally, the command defines macros `\seconds`, `\minutes`, and `\hours`. For key-value pairs that take a time as its value, use these commands to define the time value, for example, 20 minutes should be denoted `20*\minutes` (use `*` for multiplication); for 1 minute and 30 seconds, you can type either `1*\minutes+30*\seconds` or, alternatively, `1.5*\minutes`; and so on.

Parameter Description: The first parameter `<t_name>` is a name you assign the timer. The name must be unique among all timer names defined in the document. The second parameter consists of key-value pairs, described below.

[1] Why would anyone do this, you might ask?

Key-Value Pairs: The second parameter takes several key-value pairs.

- `length`: The length of the countdown, the default is 20 minutes. (`20*\minutes`)
- `stopwatch`: A Boolean, which if `true`, the counter counts up, like a stopwatch; he default is `false`, in this case the counter counts down. If stopwatch is not in the list of parameters, the counter counts down.
- `onfinish`: A choice key that determines the behavior of the counter when it reaches the main time event. Possible values are `stop` (the default) and `continue`. If `onfinish=continue`, the clock continues to count even after the main time has been attained.
- `endmsg`: When the timer reaches the main time event, the default behavior of the timer is to write a message to a text field created by the `\cntdwnEndTarget` command. The default message is "This ends the Presentation, any questions?" This message can be changed for this timer using the `endmsg` key. To globally change the message, redefine the command `\cnddwnDefaultEndMsg`. The redefinition of this command must occur before the expansion of any `\setShortCntDwn` command.

The next three keys take time as a value. Use the special macros `\hours`, `\minutes`, and `\seconds`, as explained above.

- `notify1`: The first notification time; the time before/after the main time event. The time is a prior time if the counter is counting down; and is a post time if the counter is counting up (`stopwatch=true`). The default is `5*\minutes`.
- `notify2`: The second notification time; the time before/after the main time event. The time is a prior time if the counter is counting down; and is a post time if the counter is counting up (`stopwatch=true`). The default is `3*\minutes`.
- `notify3`: The third notification time; the time before/after the main time event. The time is a prior time if the counter is counting down; and is a post time if the counter is counting up (`stopwatch=true`). The default is `1*\minutes`.

If `stopwatch=false`, the default, the counter is counting down to the main time event (at time 0 seconds); in this case `notify1 > notify2 > notify3`. If `stopwatch=false`, the counter is counting up to the main time event (at time `length`); in this case `notify1 < notify2 < notify3`. If these restrictions are not met, the timer and notification may not be as expected.

The next four keys concern the actions that are taken at the four notification times (`notify1`, `notify2`, `notify3`, and the main time event). These actions are in the form of JavaScript functions, which may be re-defined by the document author to obtain custom behaviors.

- `event1`: A JavaScript function (which the document author can create) to handle the first notification event. The default event turns the **Start** button green and causes a beep to sound.

- event2: A JavaScript function (which the document author can create) to handle the second notification event. The default event turns the **Pause** button yellow and causes a beep to sound.
- event3: A JavaScript function (which the document author can create) to handle the third notification event. The default event causes the **Stop** button to blink red and causes a beep to sound.
- endEvent: When main time is reached (time 0 or length), this function turns the Stop button to solid red (non-blinking), and writes a message to the text field created by the command \cntdwnEndTarget.
- startcolor: The color used by the default event1 function to color the **Start** button. The color is a JavaScript color; the default is color.green.
- pausecolor: The color used by the default event2 function to color the **Pause** button. The color is a JavaScript color; the default is color.yellow.
- stopcolor: The color used by the default event3 and endEvent functions to color the **Stop** button. The color is a JavaScript color; the default is color.red.
- autorun: A Boolean that determines whether the count begins when the page containing the counter is opened. The default is false.
- refreshrate: The refresh rate of the counter, the default is 1000 (milliseconds).

3.2. Commands that go in the body

The main field for the counter is \cntdwnDisplay, and it has three supporting fields \cntdwnStartT, \cntdwnPauseT, and \cntdwnStopT.

```
\cntdwnDisplay[<eforms_options>]{<t_name>}{<width>}{<height>}
\cntdwnStartT[<eforms_options>]{t_name}{<width>}{<height>}
\cntdwnPauseT[<eforms_options>]{t_name}{<width>}{<height>}
\cntdwnStopT[<eforms_options>]{t_name}{<width>}{<height>}
```

Parameter Description: The first optional parameter is used to change the appearance of the fields (these fields use the eforms package). The second parameter is the name of a timer (<t_name>) that has already been defined by \setShortCntDwn. The last two parameters sets the width and height of the form fields.

Command Description: All four of these fields are optional (no JavaScript exceptions are thrown if they do not exist). \cntdwnDisplay show the countdown of the timer; \cntdwnStartT is the start button, \cntdwnPauseT is the pause button, and \cntdwnStopT is the stop button. The capital letter T indicates that these buttons are the Target of the default event functions. Below, we describe non-target start, pause, and stop buttons.

When the timer reaches it main time event (time 0 for a countdown clock, and time length for a count-up clock), the default endEvent writes message to the multiline text field created by \cntdwnEndTarget.

```
\cntdwnEndTarget[<eforms_options>]{t_name}{<width>}{<height>}
```

Parameter Description: The four parameters are the same as described above for the `\cntdwnDisplay` field, for example. This field is optional, if it does not exist, no exception is thrown.[2]

The `cntdwn` package also provides three form field buttons for starting, pausing, and stopping the target count. The parameters are the same as those of the 'T' counterparts.

```
\cntdwnStart[<eforms_options>]{t_name}{<width>}{<height>}
\cntdwnPause[<eforms_options>]{t_name}{<width>}{<height>}
\cntdwnStop[<eforms_options>]{t_name}{<width>}{<height>}
```

4. The long countdown

A long countdown is one where the main countdown event is in the distant future. Since a short countdown is designed for less than one day, a long countdown is for times greater than a day.

As an example, let us countdown to New Year's Day, which is 1 second after midnight.

This countdown is in *local time*. No matter where you are in the world, in whatever time zone, the countdown reflect the time until your New Year's Day.

The time until my friend Jürgen celebrates New Year's Day is shown below:

Note that there is a difference in the count between the two counters, probably in the hour position; this is due difference between his timezone offset, and yours. This counter gives the time until my friend celebrates the New Year; it should read the same throughout the world, for he celebrates at a unique time in the world.

As with the short countdown, to obtain such clocks we must set the clocks parameters in the preamble (using the `\setLongCntDwn` command), and place the countdown clock anywhere we like in the body of the document (using `\lcntdwnDisplay`).

In the preamble, we have

```
\setLongCntDwn{NewYearsLocal}{%
    date=2011/01/01,
    time=00:01:00,
}
\setLongCntDwn{NewYearsCEST}{%
    date=2011/01/01,
    time=00:01:00,
    tzoffset=+0100
}
```

[2] Actually, an exception is thrown, but it is "caught" so no harm is done.

The first one sets the parameters of the local clock. The `date` and `time` are specified in the obvious way. We set the counter to run and pause automatically. The parameters for the Central European New Year clock are the same, with one exception; I've included a value for the `tzoffset` key (time zone offset), this is `+0100` (in Germany during the Winter).

In the body of the document, we place the countdown clocks,

```
\lcntdwnDisplay{NewYearsLocal}{3.5in}{11bp}
...
\lcntdwnDisplay{NewYearsCET}{3.5in}{11bp}
```

As with the short count, we reference the counter parameters through the name of the counter, as defined using `\setLongCntDwn`.

4.1. Using `\setLongCntDwn` in the preamble

For each long countdown timer, the `\setLongCntDwn` must be used to define the properties of the timer.

```
\setLongCntDwn{<t_name>}{<key-values>}
```

Command Description: The command defines the properties of the countdown. Internally, the command defines macros `\seconds`, `\minutes`, `\hours`, `\weeks`, and `\years`. For key-value pairs that take a time as its value, use these commands to define the time value, for example, 20 minutes should be denoted `20*\minutes` (use `*` for multiplication); for 2 weeks and 3 days, you can type either `2*\weeks+3*\days` or `13*\days`; and so on.

Parameter Description: The first parameter `<t_name>` is a name you assign the timer. The name must be unique among all timer names defined in the document. The second parameter consists of key-value pairs, described below.

Key-Value Pairs: The second parameter takes several key-value pairs.

- `date`: The date of the event. The value of the `date` key has the form `YYYY/MM/DD`. If `date` key is not specified then the default date of `1970/01/01` is used and a warning message is written to the log. Valid variations on the `date` key are `YYYY` (in which case the default month and day values are used, `YYYY/01/01`, and `YYYY/MM` (again, the default day is used `YYYY/MM/01`). Year must be specified with four numbers, and the month and day with two numbers. (The first month is 01.)
- `time`: The time of the event on the specified date. The format for the value of time is `HH:mm:SS` (hours, minutes, seconds). All time components are specified with two digits, if specified at all. The default value of `00` is taken for any missing component. If no `time` is specified then `00:00:00` is used; if `HH` only is specified, then the value for time is `HH:00:00`; if `HH:mm` is specified, then the value of time is `00:mm:00`.

- `tzoffset`: The time zone offset of the time of the event. If `tzoffset` is not specified, then time is interpreted as local time. The format for `tzoffset` is `Z|OHHmm`, where `Z` means that local time is equal to UT (Universal Time). For the `OHHmm` pattern, the `O` is + (plus) or – (minus); a + (plus) means that local time is later than UT, and a – (minus) means that local time is earlier than UT. For example CST (Central Standard Time) is –0600 while CET (Central European Time) is +0100. See http://www.timeanddate.com for time zone information. `HH` is the number of hours offset from UT and `mm` is the number of minutes (some time zones are measured in hours and minutes, for example, Australian Central Standard Time is +0930).
- `refreshrate`: The refresh rate of the counter, the default is 1000 (milliseconds).
- `autorun`: A Boolean that determines whether the count begins when the page containing the counter is opened. The default is `true`.
- `autopause`: A Boolean that determines whether the count is paused when the page containing the counter is closed. The default is `true`.
- `autorunenabled`: A Boolean that enables the `autorun` feature. The default value of this key is `true`. The purpose of this key-value is to turn off `autorun` dynamically (through JavaScript). The timer object that keeps all timer information has a key named `bAutorunEnabled`. If the name of the timer is `MyTimer`, and you execute `_oMyTimer.bAutorunEnabled=false`, the timer, if already paused, *will not start* (automatically) when the page containing the timer is opened again. For an example of usage, see the file `armistice_day.pdf`, titled "The cntdwn Package: Handling Notification Events for the Long Countdown Timer, Remembrance Day," available on the AeB Blog site.
- `notify1`, `notify2`, `notify3`: Leading up to the main time event are three *pre-events* that occur at times `notify1 > notify2 > notify3` *before* the main event. These are the first, second, and third notification event times. The times are relative to the main event, so if `notify1=1*\weeks`, then the first notification event occurs 1 week *before* the main event.
- `notify5`, `notify6`, `notify7`: Following the main time event are three *post-events* that occur at times `notify5 < notify6 > notify7` *after* the main event. These are the fifth, sixth, and seventh notification event times.[3] The times are relative to the main event, so if `notify5=5*\hours`, then the fifth notification event (the first after the main event) occurs 5 hours *after* the main event.
- `eventhandler`: When the timer reaches any of the seven notification times, a event handler function is launched, the default function is `_NoOpt`. The `_NoOpt` does nothing. The document author ca define his/her own event handler using this key.

  An event handler should take three parameters `doc`, `cTimer`, and `nEvent`. Use the `insDLJS` environment to define your custom handler. A very simple example is

[3] You may ask about the fourth notification event, that event is the main time event, the event that occurs when the countdown reaches 0.

```
\begin{insDLJS}[myEventHandler]{dps}{My Event Handlers}
function myEventHandler (doc,cTimer,nEvent) {
    console.show();
    console.println("Event number " + nEvent + " just occurred);
}
\end{insDLJS}
```

and type `eventhandler=myEventHandle` as part of the key-values of `\setLongCntDwn`.

- `endtimecolor`: When main time event is reached (0 seconds), the color of the display is changed to the JavaScript color determined by this key. The default is `color.red`.
- `displayfunc`: This JavaScript function displays the count; the default display function is `_defaultLDisplayFunc`, it takes parameters

  `function _defaultLDisplayFunc(f,nYears,nDays,nHours,nMinutes,nSeconds)`

  where `f` is the field object of the display field, the meaning of the other parameters is obvious.

  The default display function uses the strings year, years, day, days, hours, hour, minutes, minute, seconds, second. For localization of these strings, the document author may redefine the commands

```
\newcommand{\cntdwnYear}{year}
\newcommand{\cntdwnYears}{years}
\newcommand{\cntdwnDay}{day}
\newcommand{\cntdwnDays}{day}
\newcommand{\cntdwnHour}{hour}
\newcommand{\cntdwnHours}{hours}
\newcommand{\cntdwnMinute}{minute}
\newcommand{\cntdwnMinutes}{minutes}
\newcommand{\cntdwnSecond}{second}
\newcommand{\cntdwnSeconds}{seconds}
```

- `onfinish`: A choice key that determines the behavior of the counter when it reaches the main time event. Possible values are `stop` and `continue` (the default). If `onfinish=stop`, the clock stops the count when main event time is attained (when the timer reaches 0 seconds).
- `endmsg`: When the timer reaches the main time event and `onfinish=stop`, the default behavior is to write a message to a text field created by the `\lcntdwnDisplay` command. The default message is "The time has expired." This message can be changed for this timer using the `endmsg` key. To globally change the message, redefine the command `\lcnddwnDefaultEndMsg`. The redefinition of this command must occur before the expansion of any `\setLongCntDwn` command.

4.2. Commands that go in the body

In the body of the document there are several commands used with the long countdown timer.

```
\lcntdwnDisplay[<eform_options>]{<t_name>}{<width>}{<height>}
```

Command Description: The command creates a text field that holds the current countdown.

Parameter Description: The first optional parameter is used to change the appearance of the field (these field use the `eforms` package). The second parameter is the name of a timer (`<t_name>`) that has already been defined by `\setLongCntDwn`. The last two parameters sets the width and height of the form fields.

```
\lcntdwnToggle[<eform_options>]{<t_name>}{<width>}{<height>}
```

Command Description: The command creates a push button field that is used to toggle the count on and off (start and pause). Normally, this button is not needed, but there may be situations that it may be useful.

Parameter Description: The first optional parameter is used to change the appearance of the field (these field use the `eforms` package). The second parameter is the name of a timer (`<t_name>`) that has already been defined by `\setLongCntDwn`. The last two parameters sets the width and height of the form fields.

For example, the button [ ] will toggle the New Years Day counter.

The verbatim listing is as follows:

```
For example, the button \lcntdwnToggle{NewYearsLocal}{11bp}{11bp}\kern1bp
\lcntdwnDisplay{NewYearsLocal}{2.5in}{11bp} will toggle the New Years
Day counter.
```

4.3. The clock timer

I was just getting to wrap up this package, when I decided that it wouldn't be too much trouble to define clock timers. I include the code in as part of the `longcount` option because the code for the clock is derived from the `longcount` count. Multiple clocks can be activated at once, both clocks in the local time zone, as well as other time zones, as shown below.

	Time	**Date**
Local Time:		
CEST:		

Note: You can specify the time zone you want your clock to function in, there is, however, no way to automatically adjust time zone when there is a change from standard

to/from summer time.[4]

For these two clocks, we need to execute the command \setClockTimer in the preamble.

```
\setClockTimer{LocalClock}{}
\setClockTimer{CESTClock}{tzoffset=+0200}
```

The first parameter is the name of the clock and the second takes key-value pairs. For the LocalClock, we take the defaults, for the CESTClock, we set the time zone offset from UTC to CEST (Central European Summer Time).

The code for the clocks themselves that appear above follows:

```
\begin{tabular}{rcc}
&\textbf{Time}&\textbf{Date}\\[3bp]
Local Time:&
\cntdwnclocktime{LocalClock}{1in}{11bp}&
\cntdwnclockdate{LocalClock}{1in}{11bp}\\[3bp]
CEST:&
\cntdwnclocktime{CESTClock}{1in}{11bp}&
\cntdwnclockdate{CESTClock}{1in}{11bp}%
\end{tabular}
```

The current time is displayed by \cntdwnclocktime and the date is displayed by \cntdwnclockdate.

• Using \setClockTimer in the preamble

For each clock , the \setClockTimer must be used to define the properties of the clock.

```
\setClockTimer{<t_name>}{<key-values>}
```

Command Description: The command defines the properties of the clock.

Parameter Description: The first parameter <t_name> is a name you assign the clock. The name must be unique among all timer names defined in the document. The second parameter consists of key-value pairs, described below.

Key-Value Pairs: The second parameter takes several key-value pairs.

- tzoffset: The time zone offset of the time of the event. If tzoffset is not specified, then time is interpreted as local time. The format for tzoffset is Z|OHHmm, where Z means that local time is equal to UT (Universal Time). For the OHHmm pattern, the O is + (plus) or - (minus); a + (plus) means that local time is later than UT, and a - (minus) means that local time is earlier than UT. For example

[4]The statement has limited truth to it. If the dates are known when a given time zone changes UT offsets (between standard and summer/daylight savings time), you can write some JavaScript to make this adjustment dynamically. The problem is the dates/times may change from year to year. The ultimate solution is have access to a time/date database/server.

CST (Central Standard Time) is -0600 while CET (Central European Time) is +0100. See http://www.timeanddate.com for time zone information. HH is the number of hours offset from UT and mm is the number of minutes (some time zones are measured in hours and minutes, for example, Australian Central Standard Time is +0930).

- refreshrate: The refresh rate of the counter, the default is 1000 (milliseconds).
- autorun: A Boolean that determines whether the count begins when the page containing the counter is opened. The default is true.
- autopause: A Boolean that determines whether the count is paused when the page containing the counter is closed. The default is true.
- currtimefunc: When a long count is active, cntdwn provides the current time and date. The document author can design a custom display through this key. The default value of this key is _defaultTimeDateFunc, the definition of which is

```
function _defaultTimeDateFunc(oTime,cTimer) {
    try{ this.getField(cTimer+".clock.time").value
        =util.printd("H:MM:ss",oTime); } catch(e) {};
    try { this.getField(cTimer+".clock.date").value=
        util.printd("mm/dd/yyyy", oTime); } catch(e) {};
```

where oTime is the Date object containing current time/date.

• Commands that go in the body

In the body of the document there are several commands used with the long countdown timer.

```
\cntdwnclocktime[<eform_options>]{<t_name>}{<width>}{<height>}
\cntdwnclockdate[<eform_options>]{<t_name>}{<width>}{<height>}
```

Command Description: Each command creates a read-only text field, the first displays the time, and second displays the date. countdown.

Parameter Description: The first optional parameter is used to change the appearance of the field (these field use the eforms package). The second parameter is the name of a timer (<t_name>) that has already been defined by \setClockTimer. The last two parameters sets the width and height of the form fields.

```
\clockToggle[<eform_options>]{<t_name>}{<width>}{<height>}
```

Command Description: The command creates a push button field that is used to toggle the clock on and off (start and pause). Normally, this button is not needed, but there may be situations that it may be useful.

Parameter Description: The first optional parameter is used to change the appearance of the field (these field use the eforms package). The second parameter is the name of a timer (<t_name>) that has already been defined by \setClockTimer. The last two parameters sets the width and height of the form fields.

Note: It is easy to create a single button to toggle all clocks, or a selection of clocks, but this button is not provided with this package.

That's all for now, I simply must get back to my retirement. ☺