Matlab Timers (v. 1.4)

Dr. C. S. Tritt
January 25, 2012

Motivation

- There are a number of reasons to use Matlab's timing and timers features. These include:
  - Controlling user interfaces.
  - Collecting data from patents and/or devices.
  - Controlling devices.
  - Simulating situations.
  - Evaluating program performance.

2

The *pause* function

- **`pause`**, by itself, causes M-files to stop and wait for you to press any key before continuing.
- **`pause(n)`** pauses execution for n seconds before continuing, where *n* can be any nonnegative real number.
- The pause resolution is about 0.01 seconds on most systems.
- *Ctrl-c* escapes the pause.

3

tic and toc

Note variations in results on successive runs.

- The functions, *tic* and *toc*, operate like a stopwatch, with *tic* starting a timer and *toc* stopping it and returning the elapsed time. Here's an example (ticTocTest.m):

```
for n = 1:100
    A = rand(n,n);
    b = rand(n,1);
    tic % Start timer.
    x = A\b;
    t(n) = toc; % Stop it a get interval.
end
plot(t)
```

4

Controlling GUI's

- The functions *uiwait* and *uiresume* are designed for use in GUI programs. They block and resume MATLAB program execution, respectively.
- Calling **`uiwait(h,timeout)`**, where *h* is a figure handle and *timeout* is in seconds, blocks execution until *uiresume* is called, figure *h* is deleted, or *timeout* seconds elapse.

5

Other *uiwait* Calls

- Calling **`uiwait(h)`** blocks execution until *uiresume* is called or the figure *h* is deleted.
- Simply calling **`uiwait`** blocks execution until *uiresume* is called or the current figure is deleted.
- This syntax produces the same result as **`uiwait(gcf)`**.
- See WaitTest.m demo program.

6

Another u*iwait* Example

- The following three lines added to the exit callback of a GUIDE generated GUI program would change the displayed text, block program execution and then delete the program after 5 seconds:

```
set(handles.theText, 'String', 'Goodbye...');
uiwait(gcbf, 5);
delete(gcbf);
```

7

Matlab Timers

- Matlab timers are sophisticated programming objects that can be used in a variety of ways to control the behavior of executing programs.
- Timers can generate calls to specified functions when they start or stop. Their period and start delay can be controlled. They can fire once (single shot mode) or repeatedly.
- Timers can make things happen.

8

Basic Timer Steps

- Create a timer.
- Modify its configuration (settings).
- Start it.
- Let it run to do whatever it was intended to do.
- Stop it (or let it stop on its own after its programmed run).
- Delete it.

9

Simple Example

- Here's 2 lines of code that creates and starts a timer and the resulting output:

```
>> t = timer('ExecutionMode', 'FixedRate', ...
'Period', 3, 'TasksToExecute', 4, ...
'TimerFcn', 'disp(''Hello World!'')');
>> start(t)
Hello World!
Hello World!
Hello World!
Hello World!
>> delete(t) % Delete timer from memory.
>> clear t % Remove timer from workspace.
```

10

What's an Object?

- Matlab objects are accessed like structures but have states, properties and can do things (like call functions).
- Some timer properties can not be set (they are read only). The *AveragePeriod* is such a property.
- Rerun the previous command, but don't delete the timer. Then enter:

```
get(t, 'AveragePeriod')
```

11

What State are We In?

- Repeat the previous example:

```
t = timer('ExecutionMode', 'FixedRate', ...
'Period', 3, 'TasksToExecute', 4, ...
'TimerFcn', 'disp(''Hello World!'')'); and
  start.t
```

- Here are all the properties of our timer, *t* (obtained using `get(t)`):

```
AveragePeriod: 3.0050                StartDelay: 0
BusyMode: 'drop'                     StartFcn: ''
ErrorFcn: ''                         StopFcn: ''
ExecutionMode: 'fixedRate'           Tag: ''
InstantPeriod: 3                     TasksExecuted: 4
Name: 'timer-4'                      TasksToExecute: 4
ObjectVisibility: 'on'               TimerFcn: 'disp('Hello World!')'
Period: 3                            Type: 'timer'
Running: 'off'                       UserData: []
```

12

Timer Call Back Functions

- In the previous example, the timer function was simply a Matlab command.
- It is more common for the timer function to be a Matlab function having particular parameters.
- These parameters are:
  - An object representing the timer that called the function.
  - An Event

13

Another Example

- Configure the timer to call a function (*myTimerCB* in this case):

```
set(t, 'TimerFcn', @myTimerCB)
```

- This function simply echoes its parameters:

```
function myTimerCB(obj, event)
   fprintf('In timer function...\n');
   fprintf('Object (timer) name: %s\n', obj.Name);
   fprintf('Event type: %s\n', event.Type);
   fprintf('Event data: ');
   disp(event.Data);
end
```

14

Fragment of Output

```
>> start(t)
In timer function...
Object (timer) name: timer-7
Event type: TimerFcn
Event data:     time: [2008 1 27 22 3 25.2770]

In timer function...
Object (timer) name: timer-7
Event type: TimerFcn
Event data:     time: [2008 1 27 22 3 28.2880]
```

Etc.

Notes: *Event* is structure with fields *type* (which contains a string indicating the calling conditions) and *data* (which is a structure containing a *time* field which contains a *time* vector). See datestr and datevec regarding the format of the date information.

15

Passing Arguments

- It is possible to specify arguments to be passed to timer call back functions by using a cell array for the function specification.
- The next slides shows a function designed to accept such a parameter, how to set up the call backs and the resulting output.

16

New Function & Commands

- New Function …

Note there was an error here in v. 1.1 of show.

```
function myTimerCB2(obj, event, myString)
   fprintf('In timer function...\n');
   fprintf('Event type: %s\n', event.Type);
   fprintf('My string: %s\n\n', myString);
end
```

- Command window setup …

```
set(t, 'TimerFcn', {@myTimerCB2, 'Regular Call'})
set(t, 'StartFcn', {@myTimerCB2, 'Start Call'})
set(t, 'StopFcn', {@myTimerCB2, 'Stop Call'})
start(t)
```

17

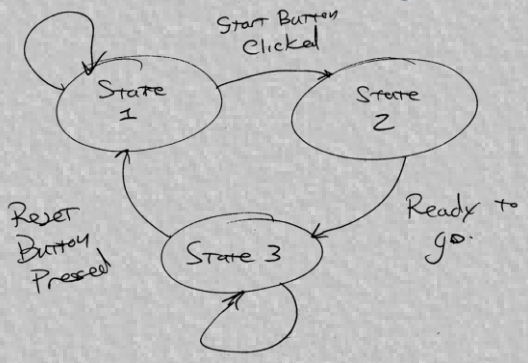
Resulting Output

```
In timer function...
Event type: StartFcn
My string: Start Call

In timer function...
Event type: TimerFcn
My string: Regular Call

… Snip (2 TimerFcn calls) …

In timer function...
Event type: TimerFcn
My string: Regular Call

In timer function...
Event type: StopFcn
My string: Stop Call
```

Admittedly, the event type contains the same information as the string in this example, but then, this is just an example.

18

Review of State Diagrams



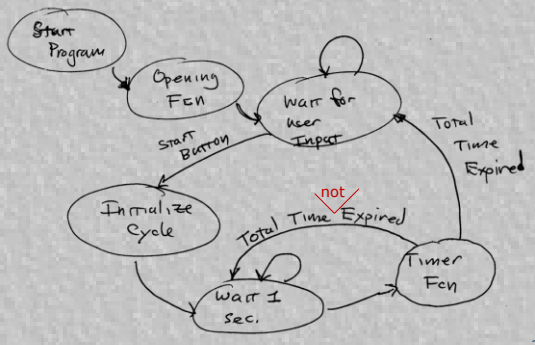
19

Using Timers with GUI's

- Egg Timer Example.
- This program demonstrates how to coordinate timers and GUI components. The user enters an interval in an edit box and clicks the start button. The starting time is displayed in a large static text box and proceeds to count down to zero. At which point the process can be repeated or the program exited.
- See attached documentation for more information.

20

Egg Timer State Diagram



21

A "Real" Example

- Monte Carlo hospital patient census simulation.
- Patients arrive at random intervals (governed by a parameter), patients stay for random intervals, during their stay some fraction of patients need infusion pumps for some fraction of their stay. The total number of patients and infusion pumps in use at any given time is displayed and saved to a file.
- The time scale of 1 second = 6 hours is applied to the entire simulation.

22