In this tutorial we will create a simple clock program.

Step 1: Open a new **Single View Application** and call it **Clock** or something that you like. Use Auto Reference Counting (**ARC**). Don't select the storyboard option. Don't really need it for this application.

Step 2: Add a Label to the ViewController.xib file. Label it **clockDisplay**. The following will be added.

```
@property (weak, nonatomic) IBOutlet UILabel *clockDisplay;
```

Step 3: Edit **ViewController.h** to add an instance of **NSTimer**. The finished code looks like this. Notice the opening and closing brackets that were added.

```
@interface ViewController : UIViewController
{
        NSTimer *myTicker;
}
```

Note: The myTicker variable is going to be responsible for updating the clockLabel.

Add the following new method declarations to ViewController.h. These get added between @interface and @end. They define two methods we will write ourselves to control the functionality of our program.

```
/* New Methods */
- (void) runTimer;
- (void) showActivity;
```

Step 4: Open **ViewController.m** and add the @synthesize for the label property. This goes between @implementation and @end.

```
@synthesize clockDisplay;
```

Step 5: Add the method implementations that we just declared in the .h file.

```
- (void)runTimer {
    // This starts the timer which fires the showActivity
    // method every 0.5 seconds
```

```
    myTicker = [NSTimer scheduledTimerWithTimeInterval: 0.5 target:
self selector: @selector(showActivity) userInfo: nil repeats: YES];
}

// This method is run every 0.5 seconds by the timer created
// in the function runTimer
- (void)showActivity {

    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    NSDate *date = [NSDate date];

    // This will produce a time that looks like "12:15:00 PM".
    [formatter setTimeStyle:NSDateFormatterMediumStyle];

    // This sets the label with the updated time.
    [clockDisplay setText:[formatter stringFromDate:date]];
}
```

Step 6: The runTimer method only has 1 line of code split into multiple lines. All it does is call the showActivity method every 0.5 seconds.

The showActivity method formats the clockDiplay label so it looks like 12:15:00 PM and sets it to the current time. As mentioned above, this method is called every 0.5 seconds.

We want to call runTimer after the view loads. This is a common method that, when Xcode generated ViewController.m, they included a method called viewDidLoad. This method is called immediately after the View items are loaded. Find the viewDidLoad method and now we will edit it.

Add [self runTimer]; to the end of the method. It should now look like this:

```
- (void)viewDidLoad
{
    [self runTimer];
    [super viewDidLoad];
}
```

Step 7: Compile and run to see the clock working.

Step 8: Add 2 additional buttons to **ViewController.xib** and wire their IBAction (Touch Down events) as **startClock** and **stopClock**.

```
- (IBAction)startButton:(UIButton *)sender;
- (IBAction)stopButton:(UIButton *)sender;
```

Add a label and wire the property as **CounterTime**.

```
@property (weak, nonatomic) IBOutlet UILabel *counterTime;
```

Step 9: Add: **int counter** as a variable in the **ViewController.h** file. The variable declarations now look like this:

```
@interface ViewController : UIViewController
{
    NSTimer *myTicker;
    int counter;
}
```

Step 10: Switch to ViewController.m and add the **counterTime** to the existing @synthesize line. The finished line now looks like:

```
@synthesize clockDisplay, counterTime;
```

Step 11: Add the following code to **ViewController.m** to create the start and stop counter button behaviors.

```
- (IBAction)startButton:(UIButton *)sender {
    myTicker = [NSTimer scheduledTimerWithTimeInterval:1
target:self selector:@selector(showTimerActivity) userInfo:nil
repeats:YES];
}

- (IBAction)stopButton:(UIButton *)sender {
    [myTicker invalidate];
    myTicker = nil;
}
```

Step 12: Add the following additional method, showTimerActivity, which is run by the other methods we just added.

```
-(void)showTimerActivity;
{
    counter += 1;
    int hours = counter / 3600;
    int minutes = counter / 60 - hours * 60;
    int seconds = counter - hours * 3600 - minutes * 60;
    counterTime.text = [NSString
stringWithFormat:@"%02d:%02d:%02d", hours, minutes, seconds];
}
```

Step 13: Run your finished app. Test the stop and start functionality. You now have a working stop clock.