

# **VBA Final Project**

**Scott De Burgh**

**I SYS 520**

## **Executive Summary**

Los Angeles Records Service is a southern California-based company aimed at providing support services to the legal community. They mainly provide subpoena preparation, medical record copying, and x-ray reproduction services.

Recently, Los Angeles Records Service outsourced their payroll function to a company called Paychex. Information to do payroll, such as hours, is currently collected using a paper system. Employees fill out the times that they come in and out of the office on paper time cards. These time cards are then submitted every week to the payroll manager, who then calculates all of the hours worked for each employee by hand. This process takes about one to two hours to complete usually. The payroll manager wanted to keep paper time cards, but wanted a way to easily enter the hours worked by employees because she has to provide this information to Paychex bi-weekly.

To shorten the time it takes to do this task, I have created a timesheet in Microsoft Excel as a userform where the payroll manager can now easily enter the times that employees clock in and out of the office. This form then automatically calculates the hours and provides data in a simple spreadsheet that can be sent to Paychex in order to process payroll. In addition to the timesheet form, I have also provided an employee form for the payroll manager to be able to add or remove employees as turnover occurs.

## **Workbook Sheets**

The workbook contains three different sheets. One is the Timesheet sheet, which is what will eventually be sent to Paychex. This sheet will receive the employee name and hours worked from the Employee Hours userform. There is also an Employees sheet which is what another userform (Employee userform) will utilize. This sheet contains the names of employees that have

been added. Lastly, there is a Pay Type sheet where the four different types of pay (Regular, Sick, Vacation, Holiday) are stored.

## Employee Hours UserForm

The screenshot shows a UserForm titled "Information on Hours Worked". At the top, there is an "Employee:" dropdown menu and a "Pay Period Ending:" text box. To the right of these are buttons: "Edit Employees", "Clear Info", "Next Employee", "Build Report", and "Cancel".

The main area is divided into two sections: "Week 1" and "Week 2". Each section has a table with columns for "Time In", "Time Out", and "Type". The rows represent the days of the week: Mon, Tue, Wed, Thu, and Fri. Each cell in the "Time In" and "Time Out" columns contains a text box for input. The "Type" column contains a dropdown menu.

Below the weekly input tables, there is a checkbox labeled "Auto Advance" and a "Calculate" button. To the right of the "Calculate" button, there are summary rows for "Regular", "Vacation", "Sick", "Holiday", and "Total" hours for both Week 1 and Week 2. Each of these rows has a corresponding text box for the total hours.

When the user first opens the workbook (which has been named frm\_EmployeePayroll), the Employee Hours userform automatically pops up. This was done by using the Workbook\_Open subprocedure. The tab order of entry boxes on this userform is done logically so the user can skip to text boxes that he or she is interested in putting data in. The first box in the order is a combo box that allows the user to choose an employee from the employee list. There is then a text box where the user enters the date of the end of the pay period.

Following the pay period box are the text boxes where the user inputs the times for the past two weeks that the employees have clocked in and clocked out. Once the user has put in the times, he or she then chooses the type of pay it is (Regular, Vacation, Sick, Holiday). If the user chooses Vacation, Sick, or Holiday, the hours will automatically populate to compute eight hours. This feature will be useful because it can save the time of having to input the information about hours. There is also a feature on this form that automatically turns the Pay Type combo boxes to Regular if the user just tabs over them. This will be useful because the hours worked will

predominantly be regular, thus letting the user just hit the tab button rather than having to reach for the mouse and pull up the drop-down box.

Another useful feature on this form is the Auto Advance feature. If this checkbox is checked, when the user inputs the numbers for hours worked it will automatically advance to the next box (i.e. if a user inputs 1020, they do not have to tab over. They will automatically be moved to the next box). The user then inputs data for the rest of the days in Week 1 and the days in Week 2. Once the user has input all necessary data, they can click the calculate button in order to get the totals for the information that has been entered. These total boxes are towards the bottom of the form.

Other command buttons on the form are Clear Info, Next Employee, Build Report, and Cancel. They are at the top of the userform to the right of the Edit Employee button. The Clear Info Button clears information on the userform if the user realizes that they did something wrong and simply want to start over. The Next Employee button saves data that has been entered into the userform and places the information on the Timesheet sheet. Once the last employee has been entered, the user can then click the Build Report button. This saves all the information on the Timesheet sheet as a sheet in a new workbook with the file name “TimesheetMM\_DD\_YY.xls” (MM\_DD\_YY corresponds to the Pay Period Ending date that the user has inserted).

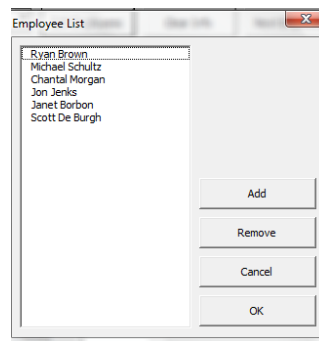
The following is a short summary of some code used in the Employee Hours userform:

- An if statement is used to automatically populate hours worked when Vacation, Sick or Holiday is chosen.
- An if statement that is used to automatically put “Regular” in the pay type box if the user exits the box without choosing a pay type.
- A select case statement is used to calculate the hours worked
- An if statement is used for the checkbox auto advance feature.

- Code used to save the form will have to be changed once it is installed on the user's computer. The code, shown below, currently reflects how it would be saved in my computer.

```
Dim fn As String
Sheets("Timesheet").Select
Sheets("Timesheet").Copy
ChDir "C:\Users\Sierra\Desktop"
fn = "C:\Users\Sierra\Desktop\" & "Timesheet" & Replace(txt_PayPeriod.Value, "/", ".")
Debug.Print fn
ActiveWorkbook.SaveAs Filename:=fn
ActiveWindow.Close
End Sub
```

## Employee UserForm



Next to the employee combo box is an Edit Employee button. When this button is clicked, it opens up a new form, called the Employees form, which allows the user to add or remove employees. This will be a useful feature for the client as there is a good amount of employee turnover. The names of employees in this form correspond to a list on the Employees sheet tab, so if the user ever just wants to look at the names of the employees on the spreadsheet, they can. The Add button brings up a message box that asks for the name of the employee, which is added in once the user clicks ok in the message box. The Remove button removes an employee name if a name is highlighted in the listbox. If a name is not highlighted and the Remove button is pushed, a box pops up that says the user must choose a name to be removed. The OK button closes the form and saves change and the Cancel button simply closes the form. The following is a short summary of code used in this form:

- An if statement on the Remove button that checks to see if an employee name is highlighted to remove.
- A For/Next statement that ensure new employee edit information is saved.

### **Learning and Conceptual Difficulties Encountered**

Since I am very inexperienced at VBA and the concepts are difficult for me to understand, this entire project for me was a hard learning experience. Most of the elements I have learned in this class I understood, except for userforms. Since this was a project for someone that I knew, I wanted to be able to create this userform for them even though my expertise at creating a good userform was lacking. I was able to include all the elements I set out to do but had to come see you (Dr. Allen) multiple times to help me get through problems. The code that was the most difficult for me was calculating hours, which were the five text boxes at the bottom of each week. I did not know how to use the Select Case code at all, but now I know how to use it on a future project. This project also taught me how important naming conventions are. If I had not had a clear naming convention that was common across certain text boxes, it would have made writing my code a lot more difficult. Now I know on any future project the first thing I want to do is have good naming conventions in place before I start writing lots of code.