Michael Blodgett
MBA 614
12/09/10

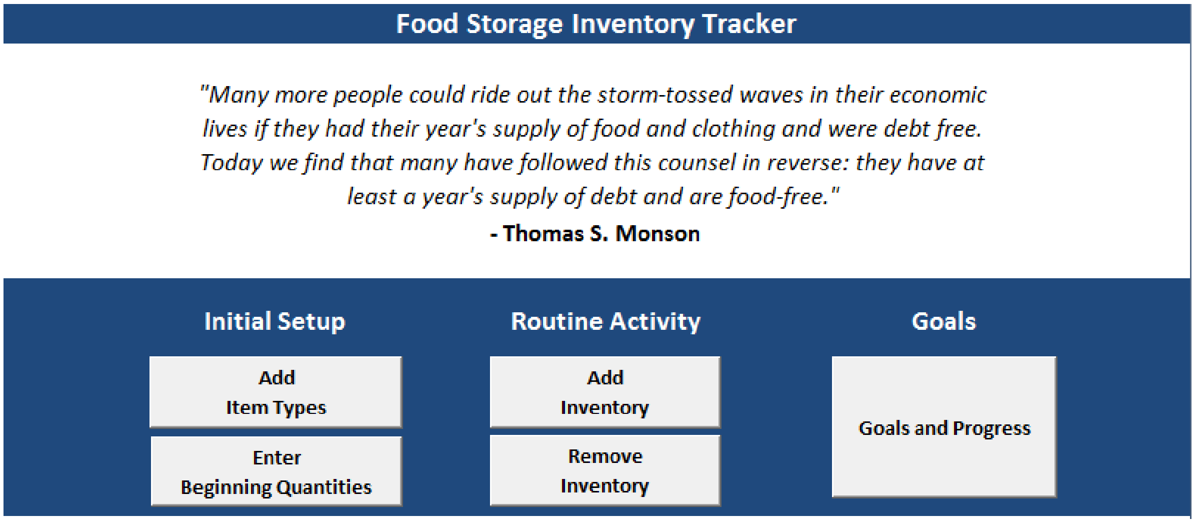
Executive Summary

Leaders in my church leaders often speak about the importance of keeping a food storage to prepare for times of need such as natural disaster or economic downturn. My wife and I have made a weak effort to follow this counsel at times, but we would like to do better. My problem is that I do not have a systematic way to keep track of what we have or what we are likely to need.

To help myself to better focus my food storage efforts, I have created an Excel workbook that allows me to enter food storage goals and track current storage levels. I can then see how I am doing relative to my current goals at any given time.

Implementation documentation

To have my workbook function the way that I wanted, I thought of it like a relational database with several different tables. I set up several user forms to collect the information to populate these tables. Below is a screen shot of the first worksheet in the workbook. The worksheet contains buttons that open the user forms or move to another place in the workbook.

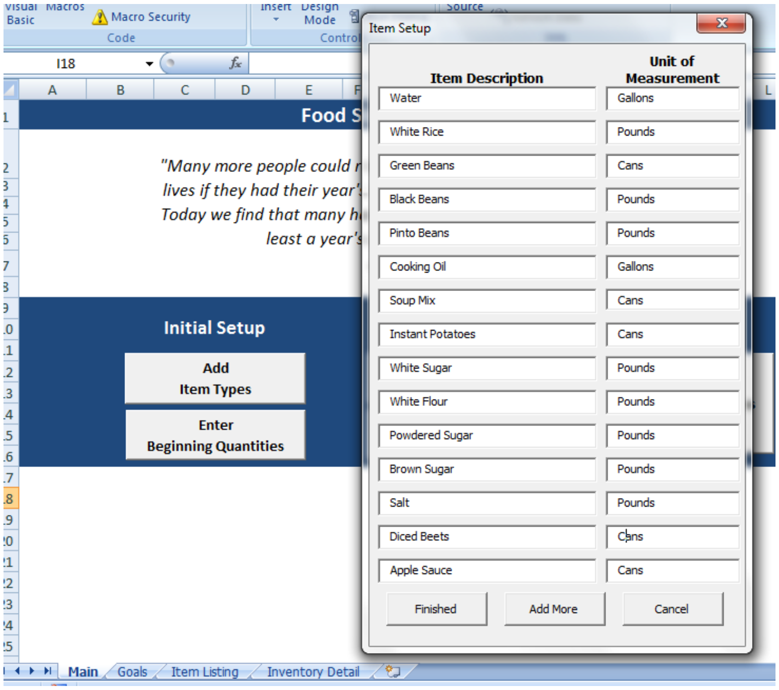


Food Storage Inventory Tracker

"Many more people could ride out the storm-tossed waves in their economic lives if they had their year's supply of food and clothing and were debt free. Today we find that many have followed this counsel in reverse: they have at least a year's supply of debt and are food-free."

- Thomas S. Monson

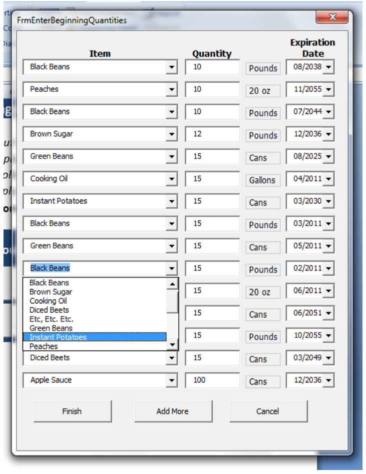
The first table that needs to be populated is a list of inventory items. I set up a user form to allow myself to enter the types of items that are (or should be) in my food storage. Below is a screenshot of what the form looks like. It is accessed with the “Add Item Types” button.



Item Description	Unit of Measurement
Water	Gallons
White Rice	Pounds
Green Beans	Cans
Black Beans	Pounds
Pinto Beans	Pounds
Cooking Oil	Gallons
Soup Mix	Cans
Instant Potatoes	Cans
White Sugar	Pounds
White Flour	Pounds
Powdered Sugar	Pounds
Brown Sugar	Pounds
Salt	Pounds
Diced Beets	Cans
Apple Sauce	Cans

The "Add More" button saves the items to the "Item Listing" tab and clears the form so that more items can be entered. The data in the item listing tab is sorted alphabetically, and blank rows in the data are deleted each time new items are added. The "Finished" button adds the items and closes the form. After items are in the item listing, the user is then able to enter goals or current food storage levels.

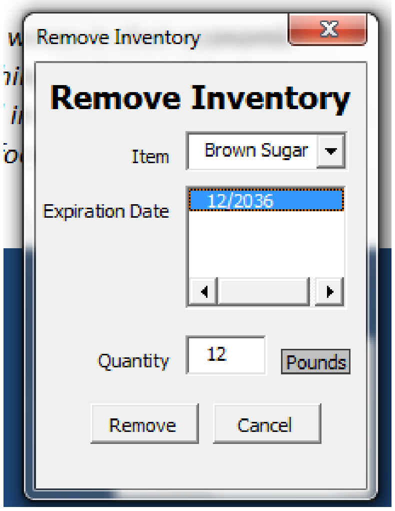
I will first look at the process of adding inventory amounts. There are two different forms that allow the user to do this. The first is with the "Enter Beginning Quantities" button. This produces a form that allows the user to enter multiple amounts on the same form.



Item	Quantity		Expiration Date
Black Beans	10	Pounds	08/2038
Peaches	10	20 oz	11/2055
Black Beans	10	Pounds	07/2044
Brown Sugar	12	Pounds	12/2036
Green Beans	15	Cans	08/2025
Cooking Oil	15	Gallons	04/2011
Instant Potatoes	15	Cans	03/2030
Black Beans	15	Pounds	03/2011
Green Beans	15	Cans	05/2011
Black Beans	15	Pounds	02/2011
	15	20 oz	06/2011
	15	Cans	06/2051
	15	Pounds	10/2055
Diced Beets	15	Cans	03/2049
Apple Sauce	100	Cans	12/2036

There are 15 comboxes on the form that are populated from the inventory listing. When a change is made to the combobox, a label displaying the unit of measurement (pounds, gallons, etc.) is automatically updated next to the quantity. There is a separate combo box on the right side of the form that allows the user to enter an expiration date. I chose to set up another table with possible expiration dates. The date comboboxes are populated from this list. The "Add More" and Finished buttons function much like those on the previous form, but this time the amounts are updated to the "Inventory Detail" worksheet. Like the inventory listing, the inventory detail is sorted by item and date each time new items are added.

There are also buttons and forms for routine additions and removals after the initial setup. These forms allow the user to add or remove item s one at a time.The "Add Inventory" form has the same functionality as the "Enter Beginning Quantities" form.

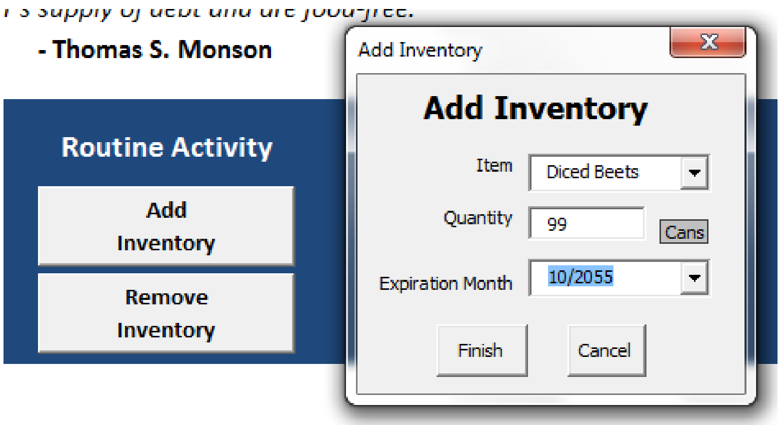

Remove Inventory

Remove Inventory

Item: Brown Sugar

Expiration Date: 12/2036

Quantity: 12 Pounds


- Thomas S. Monson

Add Inventory

Add Inventory

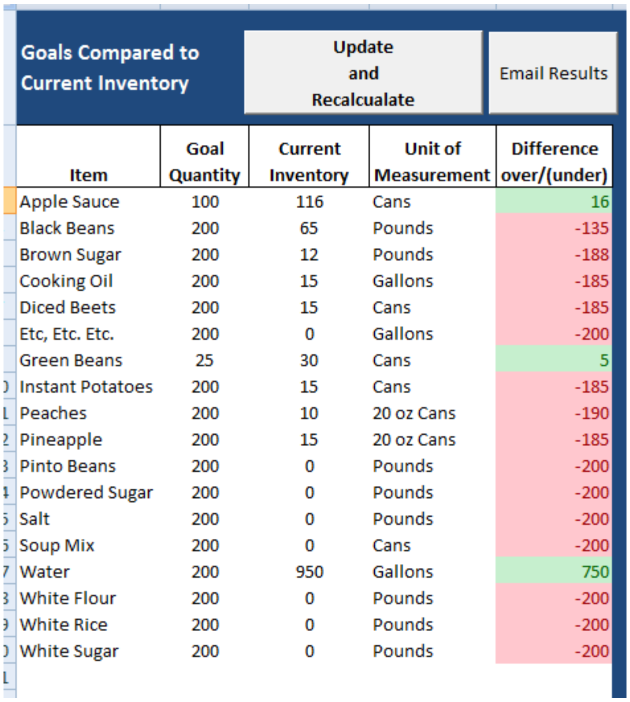
Item: Diced Beets

Quantity: 99 Cans

Expiration Month: 10/2055

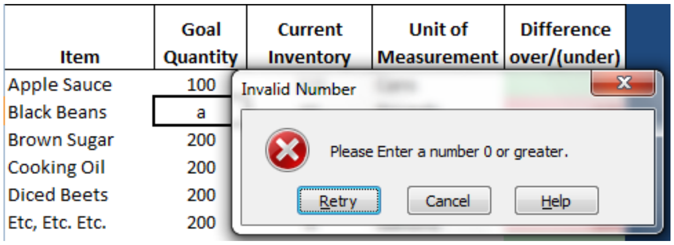
The "Remove Inventory" form allows the user to select any items that currently exist in the Inventory Detail list. When the user clicks the expiration date, the "Quantity" text box is populated with the current inventory level for the selected item and expiration date. I added this as a way to decrease the likelihood that I would accidentally remove more items than I currently had in storage. In addition, if the user tries to select an item with no current inventory in the "Remove Inventory" form, they will get an error saying that no inventory is listed for that item. I tried to add data validation functions in the program wherever I could, but that is an area that should still be improved.

The "Goals and Progress" button on the home tab, takes the user to the goals worksheet. This worksheet will initially be blank. When the user clicks the "Update and Recalulate" button, the first column is populated with the inventory listing. The current inventory summarizes the information for each item in the inventory detail worksheet. There is no user form for the goal quantities. I felt it would be better to input the amounts directly into the worksheet. If any goals or inventory amounts are modified, the user can again click the "Update and Recalculate" button to update the Inventory amounts and the difference column. The difference is simply the current inventory minus the goal for that item. The difference is highlighted green if current levels are above the specified goals. Otherwise, they are red. To the right is the error that is produced if any value other than a nonnegative number is entered.



Goals Compared to Current Inventory

Item	Goal Quantity	Current Inventory	Unit of Measurement	Difference over/(under)
Apple Sauce	100	116	Cans	16
Black Beans	200	65	Pounds	-135
Brown Sugar	200	12	Pounds	-188
Cooking Oil	200	15	Gallons	-185
Diced Beets	200	15	Cans	-185
Etc, Etc. Etc.	200	0	Gallons	-200
Green Beans	25	30	Cans	5
Instant Potatoes	200	15	Cans	-185
Peaches	200	10	20 oz Cans	-190
Pineapple	200	15	20 oz Cans	-185
Pinto Beans	200	0	Pounds	-200
Powdered Sugar	200	0	Pounds	-200
Salt	200	0	Pounds	-200
Soup Mix	200	0	Cans	-200
Water	200	950	Gallons	750
White Flour	200	0	Pounds	-200
White Rice	200	0	Pounds	-200
White Sugar	200	0	Pounds	-200

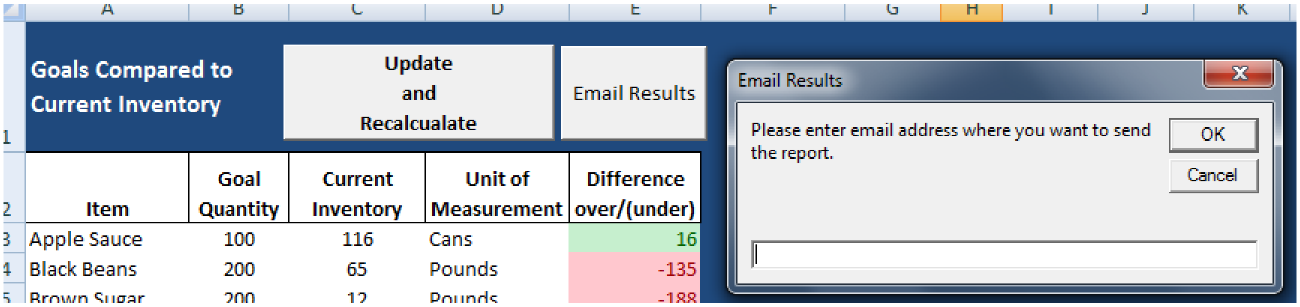


Item	Goal Quantity	Current Inventory	Unit of Measurement	Difference over/(under)
Apple Sauce	100			
Black Beans	a			
Brown Sugar	200			
Cooking Oil	200			
Diced Beets	200			
Etc, Etc. Etc.	200			

Invalid Number

Please Enter a number 0 or greater.

At the top of the goals tab, is a button that reads “Email Results”. When the user clicks this button, an input box ask them for an email address. I created a gmail account with address vbafoodstroage@gmail.com through which the programs sends the emails. The email includes the values from range beginning in cell A2 through the last value in column E. I modified the formatting so that the report in the body of the email is still in the proper rows and columns, but that is another area that I should improve.



Goals Compared to Current Inventory

	Item	Goal Quantity	Current Inventory	Unit of Measurement	Difference over/(under)
2					
3	Apple Sauce	100	116	Cans	16
4	Black Beans	200	65	Pounds	-135
5	Brown Sugar	200	12	Pounds	-188

Email Results

Please enter email address where you want to send the report.

Learning and Conceptual Difficulties

I had zero programming experience prior to taking this class, so I learned a great deal going through this project. I struggled with the logic behind a lot my code initially, but I was eventually able to think through most of it. While I was eventually able to do most of the things I wanted to do, I probably did them in ways that was not very efficient. I relied very heavily on loops to do perform a lot of my tasks. I feel like my code would have been much more efficient if I had a better understanding of arrays and how to use them.

One example of writing inefficient code had to do with one of my user forms. I had 15 comboboxes that I needed to perform essentially the same task. I needed each of them to trigger some event when they were changed. I copied the “ComboBox_Change” subprocedure 15 times in my code, but I imagine there was a better way to do this. I probably should have asked for help on that. I also struggled with how to include information from my workbook in the body of the emails that the program sent. I eventually used some formulas to get the email data to stay in a format is somewhat easy to follow. I researched the issue online and didn’t find any good solutions, so I probably should have asked for help on that part of the program.

Overall, I feel like the project allowed me to become much more comfortable using VBA. I was able to practice and review the areas we have focused on in our assignments and class examples as well venturing into several other new areas. Between this project and the user form homework assignment, I feel comfortable using user forms. I also discovered other areas that I’d to learn more about in the future. Perhaps the biggest benefit I gained from the project is an increased confidence going forward. I learned of the vast resources that are available to help me in the future. This project forced me to look beyond the textbook to online tutorials and examples. I will be more comfortable attempting to write programs in the future knowing that with my basic knowledge of VBA and these resources, I can do some powerful things.