FRANCHISE_VALUATION_FUNC

This function is used to calculate the value of Apple's iPhone franchise. It returns an array with the resulting values of the calculation.

Parameters and Variables

For a full list of parameters and variables, as well as their descriptions, see exhibit 1.

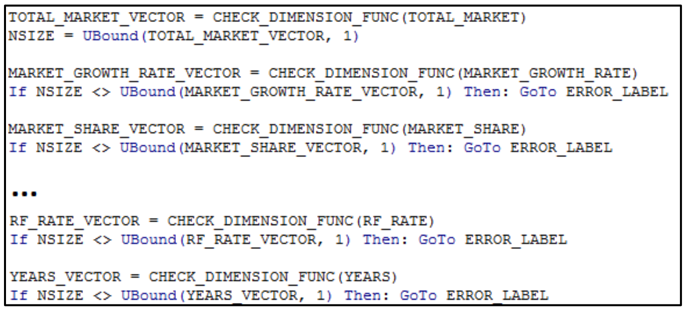
TEMP_MATRIX holds all of the data from each valuation per year that is calculated. In our recursive calculation, this matrix holds the base case of the calculation, as well as the ending total after the recursive calculation is complete. The data from this variable is then returned by the function at the end. The size of the matrix is defined by **NROWS** and **NCOLUMNS**, **NROWS** being the number of years to be calculated, and **NCOLUMNS** being the number of output columns.

TEMP_VECTOR temporarily holds the data from the results of recursive calculations for each valuation per year. This data is then transferred to **TEMP_MATRIX** once the recursive calculation is complete.

HEADINGS_STRING holds a list of headings to be used to display the headings for the results of the calculation when they are printed out on Excel.

NSIZE stores the number of calculations that are required from the valuation. This number represents each year of the valuation, and also represents the number of output rows that will be returned within the array when the calculation is complete.

Ensuring All Parameters Passed Are Arrays



```
TOTAL_MARKET_VECTOR = CHECK_DIMENSION_FUNC(TOTAL_MARKET)
NSIZE = UBound(TOTAL_MARKET_VECTOR, 1)

MARKET_GROWTH_RATE_VECTOR = CHECK_DIMENSION_FUNC(MARKET_GROWTH_RATE)
If NSIZE <> UBound(MARKET_GROWTH_RATE_VECTOR, 1) Then: GoTo ERROR_LABEL

MARKET_SHARE_VECTOR = CHECK_DIMENSION_FUNC(MARKET_SHARE)
If NSIZE <> UBound(MARKET_SHARE_VECTOR, 1) Then: GoTo ERROR_LABEL

...
RF_RATE_VECTOR = CHECK_DIMENSION_FUNC(RF_RATE)
If NSIZE <> UBound(RF_RATE_VECTOR, 1) Then: GoTo ERROR_LABEL

YEARS_VECTOR = CHECK_DIMENSION_FUNC(YEARS)
If NSIZE <> UBound(YEARS_VECTOR, 1) Then: GoTo ERROR_LABEL
```

This section checks that each range passed to the function as a parameter is an array. If the range provided as one of the parameters only has one value, it will not be stored as an array, and would result in errors in our program. The **CHECK_DIMENSION_FUNC** checks to see if the range is an array, and if not, it converts the single value to array form. To do so, the function redims the size of the array that will store the value to 1x1, and stores the value in the (1,1) location of the array. In addition, **CHECK_DIMENSION_FUNC** also ensures that the inputs are passed as vertical arrays. If not, then the function invokes the **MATRIX_TRANSPOSE_FUNC**.

This function changes an array of dimension A x B to B x A. Our code is built to traverse vertical arrays not horizontal arrays for the parameters. The code [If UBound(DATA_VECTOR, 1) = 1] within **CHECK_DIMENSION_FUNC** checks to see if there is only one row, meaning that the data is stored in a single row rather than a single column. If this is the case, the **MATRIX_TRANSPOSE_FUNC** moves all of the values from the first row to the first column, converting it to a vertical array.

Horizontal Array:

w	x	y	z

Vertical Array:

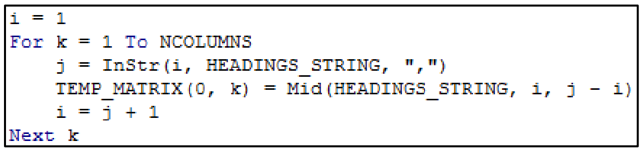
w	
x	
y	
z	

Then, the program checks to make sure that there are enough items in each array for the number of calculations that are required (stored in **NSIZE**). For example, if three calculations are required, there need to be three items stored in each array to be used in inputs for each of the three calculations.

This process is done to each parameter to ensure that they are formatted correctly for our calculation.

Headings

The string **HEADINGS_STRING** holds a list of headings to be used to display the headings for the results of the calculation when they are printed out on Excel. Each heading in the string is separated by a comma.



```
i = 1
For k = 1 To NCOLUMNS
    j = InStr(i, HEADINGS_STRING, ",")
    TEMP_MATRIX(0, k) = Mid(HEADINGS_STRING, i, j - i)
    i = j + 1
Next k
```

The above loop adds the separate headings from the string to the first row of the **TEMP_MATRIX**, so that when the matrix is printed out on Excel, the first row displays the headings.

To do this, the loop first finds the location of the comma character in the string, and stores this location in **j**. The **i** counter stores the location of the position of the first character of the string to be isolated. Since the position of the first character of the string being isolated, **i** is initialized at 1.

Then, the heading is isolated from the **HEADINGS_STRING** using the **Mid** function, which takes the whole string, the position of the first character of the string to be isolated, and the position last character of the string to be isolated as parameters. Using these parameters, the function returns the isolated string and the result is stored in **TEMP_MATRIX** in the first row, and in the corresponding column, which are populated in order using k as counter, which increases by one each time the loop runs.

Finally, the starting position of the next string is the position of the current comma character plus one. The loop repeats this for each header, and stores it in **TEMP_MATRIX**.

Using Recursion to Calculate iPhone Franchise Value

Since our function uses recursion to calculate the value of the iPhone franchise, it needs a base case where the calculation actually occurs, as well as a section that breaks up the problem into pieces until a base case is reached. The following two sections describe the two sections.

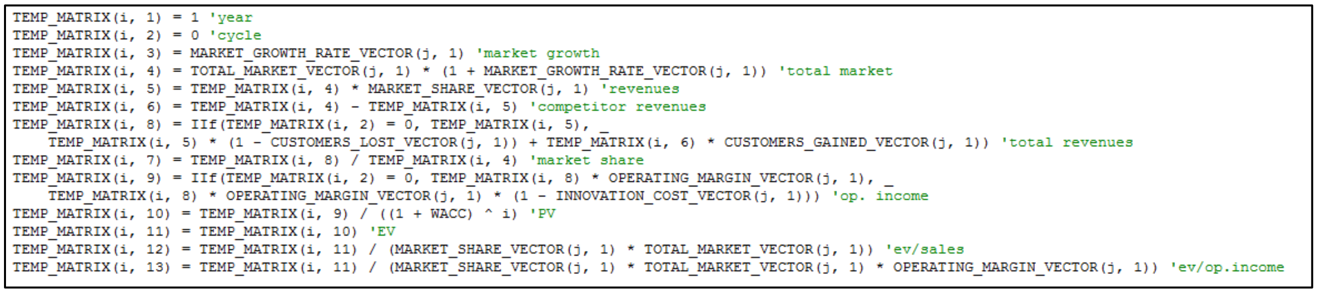
Calculating the Base Case

```
If UBound(YEARS_VECTOR, 1) = 1 Then

i = 1: j = 1

    TEMP_MATRIX(i, 1) = 1 'year
    TEMP_MATRIX(i, 2) = 0 'cycle
    TEMP_MATRIX(i, 3) = MARKET_GROWTH_RATE_VECTOR(j, 1) 'market growth
    TEMP_MATRIX(i, 4) = TOTAL_MARKET_VECTOR(j, 1) * (1 + MARKET_GROWTH_RATE_VECTOR(j, 1)) 'total market
    TEMP_MATRIX(i, 5) = TEMP_MATRIX(i, 4) * MARKET_SHARE_VECTOR(j, 1) 'revenues
    TEMP_MATRIX(i, 6) = TEMP_MATRIX(i, 4) - TEMP_MATRIX(i, 5) 'competitor revenues
    TEMP_MATRIX(i, 8) = IIf(TEMP_MATRIX(i, 2) = 0, TEMP_MATRIX(i, 5), _
        TEMP_MATRIX(i, 5) * (1 - CUSTOMERS_LOST_VECTOR(j, 1)) + TEMP_MATRIX(i, 6) * CUSTOMERS_GAINED_VECTOR(j, 1)) 'total revenues
    TEMP_MATRIX(i, 7) = TEMP_MATRIX(i, 8) / TEMP_MATRIX(i, 4) 'market share
    TEMP_MATRIX(i, 9) = IIf(TEMP_MATRIX(i, 2) = 0, TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1), _
        TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1) * (1 - INNOVATION_COST_VECTOR(j, 1))) 'op. income
    TEMP_MATRIX(i, 10) = TEMP_MATRIX(i, 9) / ((1 + WACC) ^ i) 'PV
    TEMP_MATRIX(i, 11) = TEMP_MATRIX(i, 10) 'EV
    TEMP_MATRIX(i, 12) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1)) 'ev/sales
    TEMP_MATRIX(i, 13) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1) * OPERATING_MARGIN_VECTOR(j, 1)) 'ev/op.income

    For i = 2 To YEARS_VECTOR(1, 1)

    GoSub CALC_LINE

    Next i

Else
...
```

This section of the code is evoked when the base case is currently being calculated by the function. In our function, the base case is when there is only one input from each array that was passed as a parameter. As mentioned above, our function is able to take multiple sets of inputs in order to do multiple calculations. The calculations are calculated separately, and then aggregated using recursion.

If currently each array only contains one variable each, then the base case is being calculated. The if statement checks the **YEARS_VECTOR**, but any of the parameters could be checked since they all need to have the same amount of variables currently being stored.

To calculate the base case, the function first sets the counters **i** and **j** to 1, with **j** always holding the position in **TEMP_MATRIX** of the first row (1), and **i** holding the current row that is being and stored in **TEMP_MATRIX**. Each row represents the calculation for one year's worth of cash flows, from year one to the final year of the valuation.

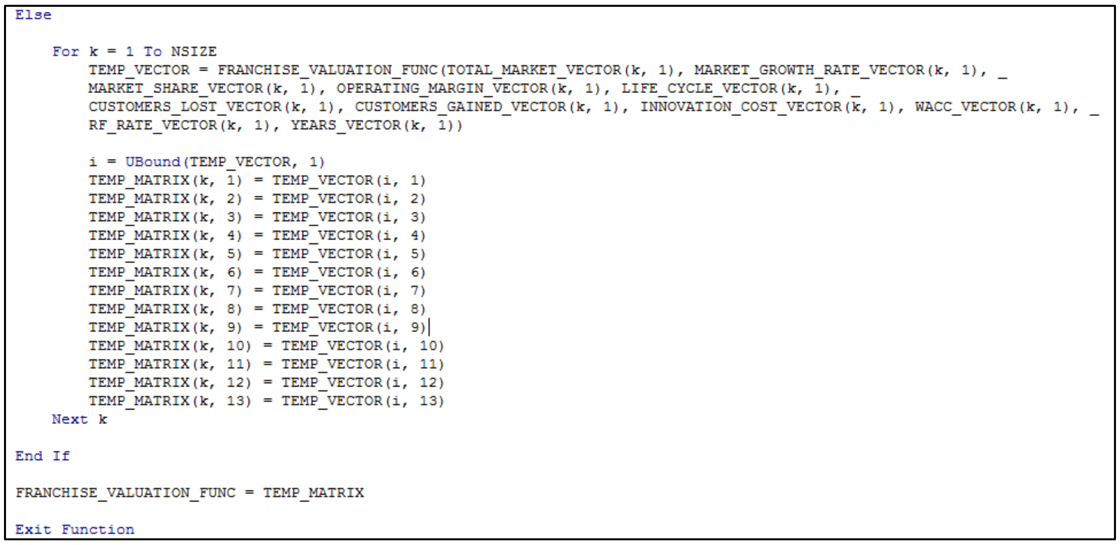


```
TEMP_MATRIX(i, 1) = 1 'year
TEMP_MATRIX(i, 2) = 0 'cycle
TEMP_MATRIX(i, 3) = MARKET_GROWTH_RATE_VECTOR(j, 1) 'market growth
TEMP_MATRIX(i, 4) = TOTAL_MARKET_VECTOR(j, 1) * (1 + MARKET_GROWTH_RATE_VECTOR(j, 1)) 'total market
TEMP_MATRIX(i, 5) = TEMP_MATRIX(i, 4) * MARKET_SHARE_VECTOR(j, 1) 'revenues
TEMP_MATRIX(i, 6) = TEMP_MATRIX(i, 4) - TEMP_MATRIX(i, 5) 'competitor revenues
TEMP_MATRIX(i, 8) = IIf(TEMP_MATRIX(i, 2) = 0, TEMP_MATRIX(i, 5), _
    TEMP_MATRIX(i, 5) * (1 - CUSTOMERS_LOST_VECTOR(j, 1)) + TEMP_MATRIX(i, 6) * CUSTOMERS_GAINED_VECTOR(j, 1)) 'total revenues
TEMP_MATRIX(i, 7) = TEMP_MATRIX(i, 8) / TEMP_MATRIX(i, 4) 'market share
TEMP_MATRIX(i, 9) = IIf(TEMP_MATRIX(i, 2) = 0, TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1), _
    TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1) * (1 - INNOVATION_COST_VECTOR(j, 1))) 'op. income
TEMP_MATRIX(i, 10) = TEMP_MATRIX(i, 9) / ((1 + WACC) ^ i) 'PV
TEMP_MATRIX(i, 11) = TEMP_MATRIX(i, 10) 'EV
TEMP_MATRIX(i, 12) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1)) 'ev/sales
TEMP_MATRIX(i, 13) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1) * OPERATING_MARGIN_VECTOR(j, 1)) 'ev/op.income
```

In order to use a loop to calculate the rest of the years, the first year must first be calculated and stored in the second row of the **TEMP_MATRIX** (code above), with the first row being headings. The results of this first calculation are then used in the loop to calculate the other years. Since a valuation needs at least one year for a calculation, and since this calculation only needs to be done once if there are more years, this section of the code is left outside of the loop. See exhibit 2 for calculations of each column in year 1.

Next, in order to calculate the cash flow from the rest of the years of the valuation, the above loop is used. It uses the sub **CALC_LINE** to perform the calculation for each year and store it in the row of **TEMP_MATRIX** that is designated by the **i** counter. By using GoSub, the program returns back to the loop after **CALC_LINE** is done its calculation. Also, by using GoSub, **CALC_LINE** has access to all of the variables of **FRANCHISE_VALUATION_FUNC** without needing any parameters. Then, the loop increments **i**.

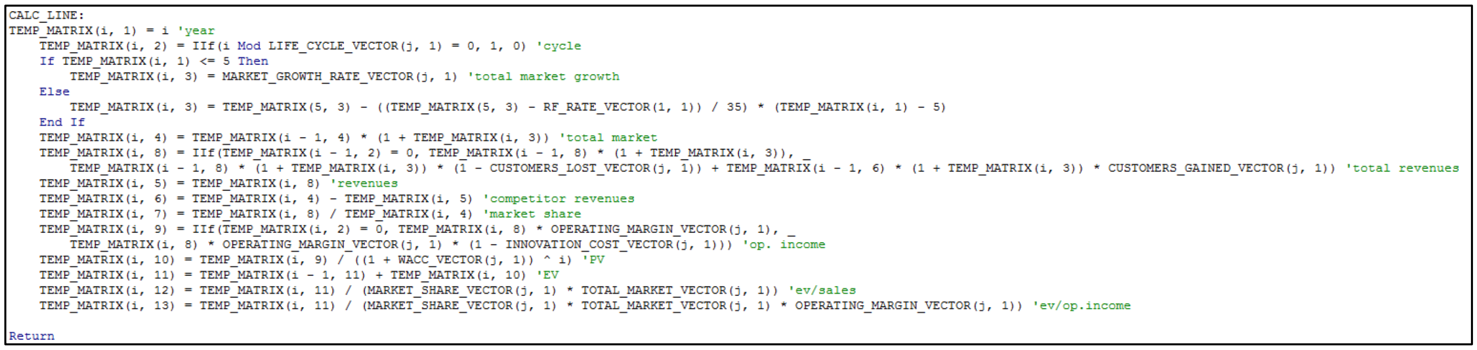
Recursive Section



```
Else

    For k = 1 To NSIZE
        TEMP_VECTOR = FRANCHISE_VALUATION_FUNC(TOTAL_MARKET_VECTOR(k, 1), MARKET_GROWTH_RATE_VECTOR(k, 1), _
        MARKET_SHARE_VECTOR(k, 1), OPERATING_MARGIN_VECTOR(k, 1), LIFE_CYCLE_VECTOR(k, 1), _
        CUSTOMERS_LOST_VECTOR(k, 1), CUSTOMERS_GAINED_VECTOR(k, 1), INNOVATION_COST_VECTOR(k, 1), WACC_VECTOR(k, 1), _
        RF_RATE_VECTOR(k, 1), YEARS_VECTOR(k, 1))

        i = UBound(TEMP_VECTOR, 1)
        TEMP_MATRIX(k, 1) = TEMP_VECTOR(i, 1)
        TEMP_MATRIX(k, 2) = TEMP_VECTOR(i, 2)
        TEMP_MATRIX(k, 3) = TEMP_VECTOR(i, 3)
        TEMP_MATRIX(k, 4) = TEMP_VECTOR(i, 4)
        TEMP_MATRIX(k, 5) = TEMP_VECTOR(i, 5)
        TEMP_MATRIX(k, 6) = TEMP_VECTOR(i, 6)
        TEMP_MATRIX(k, 7) = TEMP_VECTOR(i, 7)
        TEMP_MATRIX(k, 8) = TEMP_VECTOR(i, 8)
        TEMP_MATRIX(k, 9) = TEMP_VECTOR(i, 9)
        TEMP_MATRIX(k, 10) = TEMP_VECTOR(i, 10)
        TEMP_MATRIX(k, 11) = TEMP_VECTOR(i, 11)
        TEMP_MATRIX(k, 12) = TEMP_VECTOR(i, 12)
        TEMP_MATRIX(k, 13) = TEMP_VECTOR(i, 13)
    Next k

End If

FRANCHISE_VALUATION_FUNC = TEMP_MATRIX

Exit Function
```

If there is multiple calculations to be performed, in other words, if there are multiple sets of inputs for the franchise value calculation, this section of code calculates each set of inputs individually, thus passing only one input in each array, resulting in the base case. The base case is calculated and the result is stored in the **TEMP_VECTOR**. Then, the results from the **TEMP_VECTOR** are transferred to the **TEMP_MATRIX**, which stores the valuation, and its values are then returned by the array function.

CALC_LINE Function



```
CALC_LINE:
TEMP_MATRIX(i, 1) = i 'year
    TEMP_MATRIX(i, 2) = IIf(i Mod LIFE_CYCLE_VECTOR(j, 1) = 0, 1, 0) 'cycle
    If TEMP_MATRIX(i, 1) <= 5 Then
        TEMP_MATRIX(i, 3) = MARKET_GROWTH_RATE_VECTOR(j, 1) 'total market growth
    Else
        TEMP_MATRIX(i, 3) = TEMP_MATRIX(5, 3) - ((TEMP_MATRIX(5, 3) - RF_RATE_VECTOR(1, 1)) / 35) * (TEMP_MATRIX(i, 1) - 5)
    End If
    TEMP_MATRIX(i, 4) = TEMP_MATRIX(i - 1, 4) * (1 + TEMP_MATRIX(i, 3)) 'total market
    TEMP_MATRIX(i, 8) = IIf(TEMP_MATRIX(i - 1, 2) = 0, TEMP_MATRIX(i - 1, 8) * (1 + TEMP_MATRIX(i, 3)), _
        TEMP_MATRIX(i - 1, 8) * (1 + TEMP_MATRIX(i, 3)) * (1 - CUSTOMERS_LOST_VECTOR(j, 1)) + TEMP_MATRIX(i - 1, 6) * (1 + TEMP_MATRIX(i, 3)) * CUSTOMERS_GAINED_VECTOR(j, 1)) 'total revenues
    TEMP_MATRIX(i, 5) = TEMP_MATRIX(i, 8) 'revenues
    TEMP_MATRIX(i, 6) = TEMP_MATRIX(i, 4) - TEMP_MATRIX(i, 5) 'competitor revenues
    TEMP_MATRIX(i, 7) = TEMP_MATRIX(i, 8) / TEMP_MATRIX(i, 4) 'market share
    TEMP_MATRIX(i, 9) = IIf(TEMP_MATRIX(i, 2) = 0, TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1), _
        TEMP_MATRIX(i, 8) * OPERATING_MARGIN_VECTOR(j, 1) * (1 - INNOVATION_COST_VECTOR(j, 1))) 'op. income
    TEMP_MATRIX(i, 10) = TEMP_MATRIX(i, 9) / ((1 + WACC_VECTOR(j, 1)) ^ i) 'PV
    TEMP_MATRIX(i, 11) = TEMP_MATRIX(i - 1, 11) + TEMP_MATRIX(i, 10) 'EV
    TEMP_MATRIX(i, 12) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1)) 'ev/sales
    TEMP_MATRIX(i, 13) = TEMP_MATRIX(i, 11) / (MARKET_SHARE_VECTOR(j, 1) * TOTAL_MARKET_VECTOR(j, 1) * OPERATING_MARGIN_VECTOR(j, 1)) 'ev/op.income

Return
```

In this sub, the program calculates the value of the cash flows from year 2 up to the last year designated by the user in the parameter YEARS. See exhibit 3 for the calculation of each column for these years.

EXHIBIT 1 – PARAMETERS AND VARIABLES OF THE FUNCTION

Parameter	**Description**	**Stored in Vector**
TOTAL_MARKET	Market size of Smartphones	TOTAL_MARKET_VECTOR
MARKET_GROWTH_RATE	Growth of the market (%)	MARKET_GROWTH_RATE_VECTOR
MARKET_SHARE	iPhone's marketshare	MARKET_SHARE_VECTOR
OPERATING_MARGIN	Apple's Operating Margin	OPERATING_MARGIN_VECTOR
LIFE_CYCLE	Life cycle of the iPhone	LIFE_CYCLE_VECTOR
CUSTOMERS_LOST	Customers lost to competitors	CUSTOMERS_LOST_VECTOR
CUSTOMERS_GAINED	Customers gained from competitors	CUSTOMERS_GAINED_VECTOR
INNOVATION_COST	% of after-tax income used for R&D	INNOVATION_COST_VECTOR
WACC	Cost of capital for Apple	WACC_VECTOR
RF_RATE	Risk Free Rate	RF_RATE_VECTOR
YEARS	Duration of the franchise in years	YEARS_VECTOR

Variable	**Description**
TEMP_MATRIX	Holds the total results of the valuation, each year calculated is stored in each row
TEMP_VECTOR As Variant	Holds temporary results from the recursive calculation, then transferred to TEMP_MATRIX
HEADINGS_STRING	Holds the name of each column, used when data is displayed in Excel
NCOLUMNS	Number of columns of the output array
NROWS	Number of variables in each array that is passed by the parameters
NSIZE	Number of variables in the year array

EXHIBIT 2 – Year 1 Calculations

Item	Description	Calculation	Notes
TEMP_MATRIX(i, 1)	Year of Valuation	=1	
TEMP_MATRIX(i, 2)	Cycle	=0 (first year)	
TEMP_MATRIX(i, 3)	Market Growth	=Market growth provided in MARKET_GROWTH_RATE_VECTOR	
TEMP_MATRIX(i, 4)	Total Market	=Total Market * growth rate	
TEMP_MATRIX(i, 5)	Revenues	=Total Market * Market Share	
TEMP_MATRIX(i, 6)	Competitor Revenues	=Total Market Size - Apple Revenues	
TEMP_MATRIX(i, 8)	Total Revenues	if cycle is 0, then it's simply total market size. if cycle <> 0: = Revenues * (1 - customers lost) + Competitor Revenues * customers gained	Accounts for Customers lost and Gained
TEMP_MATRIX(i, 7)	Market Share	Total Revenues / Total Market	
TEMP_MATRIX(i, 9)	Operating Income	If cycle is 0, then it's simply Revenues * Margin if cycle <> 0: = Revenues * (margin) * (1- % spend on innovation)	
TEMP_MATRIX(i, 10)	Present Value	Present value formula	
TEMP_MATRIX(i, 11)	Enterprise Value (EV)	Same as Present value	
TEMP_MATRIX(i, 12)	EV / Sales	EV / Market Share * Total Market	
TEMP_MATRIX(i, 13)	EV / Operating Income	EV / Market Share * Total Market * Margin	

EXHIBIT 3 – CALC_LINE Function Calculations

Item	**Description**	**Calculation**
TEMP_MATRIX(i, 1)	Year of Valuation	=i (i cycles through all the years of the valuation)
TEMP_MATRIX(i, 2)	Cycle	=1 if cycle, 0 if not
TEMP_MATRIX(i, 3)	Market Growth	=if first 5 years, use same growth as provided in parameter. After 5 years, growth is reduced evenly until year 40 to reach 2%
TEMP_MATRIX(i, 4)	Total Market	=Previous Year Total Market * growth rate
TEMP_MATRIX(i, 8)	Total Revenues	if cycle is 0, then it's simply previous total market size*growth. if cycle <> 0: = Previous yr Revenues * (1 - customers lost%) + Competitor Revenues * customers gained%
TEMP_MATRIX(i, 5)	Revenues	=Previous Year Total Market * Market Share
TEMP_MATRIX(i, 6)	Competitor Revenues	=Total Market Size - Apple Revenues
TEMP_MATRIX(i, 7)	Market Share	Total Revenues / Total Market
TEMP_MATRIX(i, 9)	Operating Income	If cycle is 0, then it's simply Revenues * Margin if cycle <> 0: = Revenues * (margin) * (1- % spend on innovation)
TEMP_MATRIX(i, 10)	Present Value	Present value formula
TEMP_MATRIX(i, 11)	Enterprise Value (EV)	Same as Present value
TEMP_MATRIX(i, 12)	EV / Sales	EV / Market Share * Total Market
TEMP_MATRIX(i, 13)	EV / Operating Income	EV / Market Share * Total Market * Margin