GALCOR Documentation

Angelino Lefevers

May 1, 2017

Contents

1 Introduction

GALCOR is an assistance program in which performs various calculations and functions for the user. GALCOR functionality can be accessed by either calling a variety of functions from a program script, or by using the GALCOR GUI.

2 gGENERATOR.c

2.1 Function Prototypes

- ▷ int *__gGENERATOR_DCSignalGenerator0__(int *size*, int *amplitude*, int *delay*);
- ▷ double *__gGENERATOR_DCSignalGenerator1__(int *size*, double *amplitude*, int *delay*);
- ▷ int *__gGENERATOR_DCSignalInserter0__(int **signalSource*, int *index*, int *sourceSize*, int *amplitude*, int *signalSize*);
- ▷ double *__gGENERATOR_DCSignalInserter1__(int **signalSource*, int *index*, int *sourceSize*, double *amplitude*, int *signalSize*);
- ▷ int *__gGENERATOR_ImpulseSignalGenerator0__(int *size*, int *amplitude*, int *delay*);
- ▷ double *__gGENERATOR_ImpulseSignalGenerator1__(int *size*, double *amplitude*, int *delay*);
- ▷ int *__gGENERATOR_RandomIntegerSignalGenerator__(int *size*, int *minRange*, int *maxRange*);

2.2 Functions

2.2.1 gGENERATOR_DCSignalGenerator0

- Prototype:
  - int *__gGENERATOR_DCSignalGenerator0__(int ***size***, int ***amplitude***, int ***delay***);
- Description:
  - Consists of integer values
  - Generates a constant DC signal of length *size*
  - Returns a pointer to the loaction of the generated DC signal
- Arguments:
  - *int* ***size*** := Number of samples in the signal to be generated.
  - *int* ***amplitude*** := The amplitude of the DC signal to be generated.
  - *int* ***delay*** := Number of samples the generated DC signal will be delayed from TIME 0.
- Intermediate Variables:
  - *int** **dcSignalSource** := Pointer to first element of the generated signal.
  - *int** **tSignal** := Temporary offset pointer used to iterate through and initialize the signal.
  - *int* **i, j** := Counters to be used in looping.
- Return:
  - *int** **dcSignalSource**
- Tested:
  - **YES** - LIMITED

2.2.2 gGENERATOR_DCSignalGenerator1

- Prototype:
  - double *__gGENERATOR_DCSignalGenerator1__(int ***size***, double ***amplitude***, int ***delay***);
- Description:
  - Consists of double floating-point values
  - Generates a constant DC signal of length *size*
  - Returns a pointer to the loaction of the generated DC signal
- Arguments:
  - *int* ***size*** := Number of samples in the signal to be generated.
  - *double* ***amplitude*** := The amplitude of the DC signal to be generated.
  - *int* ***delay*** := Number of samples the generated DC signal will be delayed from TIME 0.
- Intermediate Variables:
  - *double** **dcSignalSource** := Pointer to first element of the generated signal.
  - *double** **tSignal** := Temporary offset pointer used to iterate through and initialize the signal.
  - *int* **i, j** := Counters to be used in looping.
- Return:
  - *double** **dcSignalSource**
- Tested:
  - **YES** - LIMITED

2.2.3 gGENERATOR_DCSignalInserter0

- Prototype:
  - int *__gGENERATOR_DCSignalGenerator0__(int *__signalSource__, int __index__, int __sourceSize__, int __amplitude__, int __sourceSize__);
- Description:
  - Consists of integers
  - Takes an array of integer elements of size *sourceSize* and pointed to by *signalSource*.
  - Inserts a DC signal into the index value *index* from the signal source.
  - The DC signal inserted is *signalSize* elements long and has an amplitude of *amplitude*.
- Arguments:
  - *int** __signalSource__ := Pointer to the first element of the inputted signal.
  - *int* __index__ :=Index value of where the generated DC signal is inserted.
  - *int* __sourceSize__ := The number of elements in the inputted signal.
- Intermediate Variables:
  - *int** **dcSignalSource** := Pointer to first element of the generated signal.
  - *int** **tSignal** := Temporary offset pointer used to iterate through and initialize the signal.
  - *int* **i, j** := Counters to be used in looping.
- Return:
  - *int** **dcSignalSource**
- Tested:
  - **YES** - LIMITED

3 gECON.c

3.1 Function Prototypes

- ▷ double *__gECON_SellingPricePerUnit__(double *constant_a*, double *constant_b*, double *demand*);
- ▷ double *__gECON_Demand__(double *constant_a*, double *constant_b*, double *pricePerUnit*);
- ▷ double *__gECON_TotalRevenue0__(double *price*, double *demand*);
- ▷ double *__gECON_TotalRevenue1__(double *constant_a*, double *constant_b*, double *demand*);
- ▷ double *__gECON_TotalCost__(double *fixedCosts*, double *variableCostsPerUnit*, double *demand*);
- ▷ double *__gECON_Profit0__(double *totalRevenue*, double *totalCosts*);
- ▷ double *__gECON_Profit1__(double *constant_a*, double *constant_b*, double *demand*, double *fixedCosts*, double *variableCostsPerUnit*);

3.2 Functions

3.2.1 gECON_SellingPricePerUnit

- Prototype:
  - double **gECON_SellingPricePerUnit**(double ***constant_a***, double ***constant_b***, double ***demand***);
- Description:
  - Returns the selling price per unit given constants *constant_a* and *constant_b*, and the demand *demand*.
- Equation: $p = a - bD$
  - p := Price per Unit
  - D := "Demand" - Total number of units sold.
  - a, b := constants
  - Conditions:
    - $0 \leq D \leq \frac{a}{b}$
    - $a > 0$
    - $b > 0$
- Arguments:
  - *double* ***constant_a*** := Constant
  - *double* ***constant_b*** := Constant
  - *double* ***demand*** := The total number of units sold.
- Intermediate Variables: **NONE**
- Return:
  - *double* **pricePerUnit**
- Tested:
  - **YES** - LIMITED

3.2.2 gECON_Demand

- Prototype:
  - double **gECON_Demand**(double ***constant_a***, double ***constant_b***, double ***pricePerUnit***);
- Description:
  - Returns the selling price per unit given constants *constant_a* and *constant_b*, and the price per unit *pricePerUnit*.
- Equation: $D = \frac{a-p}{b}$
  - D := "Demand" - Total number of units sold.
  - p := Price per Unit
  - a, b := constants
  - Conditions:
    - $b \neq 0$
- Arguments:
  - *double* ***constant_a*** := Constant
  - *double* ***constant_b*** := Constant
  - *double* ***pricePerUnit*** := The price per unit which was manufactured.
- Intermediate Variables: **NONE**
- Return:
  - *double* **demand**
- Tested:
  - **YES** - LIMITED

3.2.3 gECON_TotalRevenue0

- Prototype:
  - double **gECON_TotalRevenue0**(double ***pricePerUnit***, double ***demand***);
- Description:
  - Returns the total revenue given the *pricePerUnit* and the *demand* of the product.
- Equation: $TR = pD$
  - TR := Total revenue earned from a production.
  - p := Price per Unit
  - D := Demand, or the total number of units sold after production.
- Arguments:
  - *double* ***pricePerUnit*** := Price per Unit
  - *double* ***demand*** := Total number of units sold.
- Intermediate Variables: **NONE**
- Return:
  - *double* **totalRevenue**
- Tested:
  - **YES** - LIMITED

3.2.4 gECON_TotalRevenue1

- Prototype:
  - double **gECON_TotalRevenue1**(double ***constant_a***, double ***constant_b***, double ***demand***);
- Description:
  - Returns the total revenue given the constants *constant_a* and *constant_b*, and the *demand* of the product.
- Equation: $TR = (a - bD)D$
  - TR := Total revenue earned from a production.
  - D := Demand, or the total number of units sold after production.
  - a, b := Constants
  - Conditions:
    - $0 \leq D \leq \frac{a}{b}$
    - $a > 0$
    - $b > 0$
- Arguments:
  - *double* ***constant_a*** := Constant
  - *double* ***constant_b*** := Constant
  - *double* ***demand*** := Total number of units sold after production.
- Intermediate Variables: **NONE**
- Return:
  - *double* **totalRevenue**
- Tested:
  - **YES** - LIMITED

3.2.5 gECON_TotalCost0

- Prototype:
  - double **gECON_TotalCost0**(double ***fixedCosts***, double ***variableCosts***);
- Description:
  - Returns the total costs of production given the *fixedCosts* the *variableCosts* of the production.
- Equation: $C_T = C_F + C_V$
  - C_T := Total costs in production.
  - C_F := Total fixed (initial) costs of production.
  - C_V := Total variable costs of production.
- Arguments:
  - *double* ***fixedCosts*** := Fixed costs of the production.
  - *double* ***variableCosts*** := Variable costs of the production
- Intermediate Variables: **NONE**
- Return:
  - *double* **totalCosts**
- Tested:
  - **YES** - LIMITED

3.2.6 gECON_TotalCost1

- Prototype:
  - double **gECON_TotalCost1**(double ***fixedCosts***, double ***variableCostsPerUnit***, double ***demand***);
- Description:
  - Returns the total costs of production given the *fixedCosts*, the *variableCostsPerUnit*, and the *demand* of the production.
- Equation: $C_T = C_F + (c_v * D)$
  - C_T := Total costs in production.
  - C_F := Total fixed (initial) costs of production.
  - c_v := Variable costs of production per unit.
  - D := Total number of units sold after production.
- Arguments:
  - *double* ***fixedCosts*** := Fixed costs of the production.
  - *double* ***variableCostsPerUnit*** := Variable costs per unit.
  - *double* ***demand*** := Total number of units sold after production.
- Intermediate Variables: **NONE**
- Return:
  - *double* **totalCosts**
- Tested:
  - **YES** - LIMITED

3.2.7 gECON_Profit

- Prototype:
  - double **gECON_Profit**(double ***totalRevenue***, double ***totalCosts***);
- Description:
  - Returns the total costs of production given the *fixedCosts*, the *variableCostsPerUnit*, and the *demand* of the production.
- Equation: $P = TR - C_T$
  - P := Total profit made from production.
  - TR := Total revenue earned from production.
  - C_T := Total costs in production.
- Arguments:
  - *double* ***fixedCosts*** := Fixed costs of the production.
  - *double* ***variableCostsPerUnit*** := Variable costs per unit.
  - *double* ***demand*** := Total number of units sold after production.
- Intermediate Variables: **NONE**
- Return:
  - *double* **profit**
- Tested:
  - **YES** - LIMITED

3.2.8 gECON_RevenueMaximizedDemand

- Prototype:
  - double **gECON_RevenueMaximizedDemand**(double ***constant_a***, double ***constant_b***, double ***demand***);
- Description:
  - Returns the number of units sold which would maximize the revenue earned from the production.
- Equation: $\hat{D} = \frac{a}{2b}$
  - $\hat{D}$:= Total units sold to maximize the total revenue earned from production.
  - a, b := Constants
  - Conditions:
    - $\frac{dTR}{dD} = a - 2bD = 0$
- Arguments:
  - *double* ***constant_a*** := Constant
  - *double* ***constant_b*** := Constant
  - *double* ***demand*** := Total number of units sold after production.
- Intermediate Variables: **NONE**
- Return:
  - *double* **revenueMaximizedDemand**
- Tested:
  - **YES** - LIMITED

3.2.9 gECON_BreakEvenPoints

- Prototype:
  - double **gECON_BreakEvenPoints**(double ***constant_a***, double ***constant_b***, double ***variableCostsPerUnit***, double ***fixedCosts***);
- Description:
  - Returns the number of units which need to be sold after production for the total revenue earned from the production will equal to the total costs of the production.
- Equation: $BEP = \frac{-(a-c_v) \pm \sqrt{(a-c_v)^2 - 4(b)(C_F)}}{2(-b)} = \frac{i0 \pm i1}{i2}$
  - BEP := Number of units which need to be sold for the total revenue to equal total costs.
  - C_F := The fixed or inital costs of the production
  - c_v := The costs per unit produced.
  - a, b := Constants
- Arguments:
  - *double* ***constant_a*** := Constant
  - *double* ***constant_b*** := Constant
  - *double* ***variableCostsPerUnit*** := The costs per each unit produced.
  - *double* ***fixedCosts*** := The fixed of initial costs of the production.
- Structure and Type Definition:

```
struct breakEvenPoints {
    double low;
    double high;
};

typedef struct breakEvenPoints BEP;
```

- Intermediate Variables:
  - *double* **i0** := Intermediate variable 0.
  - *double* **i1** := Intermediate variable 1.
  - *double* **i2** := Intermediate variable 2.
  - *BEP* **out** := breakEvenPoints structure whose elements represent the low and high break even points.
- Return:
  - *double* **out**
- Tested:
  - **YES** - LIMITED