# *Factorio Ratio Calculator*

*The example represents an attempt to comprehend the problem of calculating ratios for my factory in the context of gears. These values are all wrong because they don't properly account for crafts vs. units, but this was just for practice so who cares.*

## Example Of Problem: Calculating Gear Ratios/Units Per Second Required To Satisfy Factory's Current Needs

<u>What do I want automated?</u>

- Fast inserter per second
- Long inserter per second
- Normal inserter per second
- 2 x Assembler per base crafting time
- 2 x Electric mining drill per base crafting time
- 2 x ammo per base crafting time
- Hand supply of gears
    - Means producing slightly extra to fit inserter grab rate
- Hand supply of chips

<u>Units being used:</u>

- Gears: 1 gps per assembler
- Inserter can grab 0.85 gps
- gps : gears per second
    - == gpc/spc
- spc : seconds per craft
    - == gpc/gps
- gpc : gears per craft
    - == gps*spc

- uc : units crafted
    - == gr/gpc
- gr : gears required
    - == uc*gpc
- ups : units per second
- ar : assemblers required
    - == ups*spc

<u>Gear ratios required for needed recipes:</u>

```
/*
more of a placeholder to show that gpc is coming from
a gear requirement instead of a secondary craft
*/
```

**(0)** Gear @ 1 spc

**(1)** Inserter @ 1 spc
- ➢ **0** req @ 1 upc
- ■ 1 gpc
- ■ gps = 1 * ups

**(2)** Fast Inserter @ 1 spc
- ➢ **1** req @ 1 upc
- ■ 1 gpc
- ■ gps = 1 * ups

**(3)** Long Inserter @ 1 spc
- ➢ **0** req @ 1 upc
- ➢ **1** req @ 1 upc
- ■ 2 gpc
- ■ gps = 2 * ups

**(4)** Assembler @ 1 spc
- ➢ **0** req @ 5 upc
- ■ 5 gpc
- ■ gps = 5 * ups

**(5)** Electric Mining Drill @ 4 spc
- ➢ **0** req @ 5 upc
- ■ 5 gpc
- ■ gps = 1.25 * ups

<u>Updated factory specs (output  of calculator will look something like this):</u>

**1** @ 1 ups + (**2** req. 1 ups, **3** req. 1 ups)
- 3 ar
- 3 gps

**2** @ 1 ups
- 1 ar
- ➢ **1** req @ 1 ups
    - 1 gps
- 0 gps (gps is already included in child recipe)

**3** @ 1 ups
- 1 ar
- ➢ **1** req @ 1 ups
    - 1 gps
- 1 gps (half of gps is already included in child recipe)

**4** @ 0.5 ups
- 0.5 ar
- 2.5 gps

**5** @ 0.5 ups
- 2 ar
- 0.625 gps

---

## Ratios Calculator Design Outline

### Problem Statement
I want to know, given any combination of outputs (inputs in the context of the program, simply units per second of a series of items), what items I will need to produce in order to match these outputs as well as the # of these items per second, and the number of assemblers required to produce these items.

### Inputs/Outputs Conceptualized
Input Info:
- Units per second of a given item I want to produce

Output Info:
- Units per second of ingredients required to make said item
- Crafter counts required to make said ingredients

### All Starting Info Req. For Calculations
- Crafter speed multipliers (assembler, furnace, etc.)
    - a short decimal value
- Crafting time for each recipe
    - a short decimal value
- Ingredients for each recipe (link to a recipe)
    - an integer

### Values Manipulated By The Program

These values are changed upon input, are used in calculations by the program, and will functionally serve as the program’s outputs. Therefore, it’d be good to have some system that would prevent these values from becoming corrupted / make it easy to fix them if they do.

- Req. crafter counts for each item
    - a short decimal value

- Req. ups per item
    - a short decimal value

**General Program Steps**

1. accept an input value
2. store this input value
    a. I'm assuming this will just be done by default with the way sheets is setup
3. access the recipe the input value is referencing
    1. recipe will need to be looked up
    2. ingredients will need to be individually processed
4. use the recipe's information to update/populate a visual output representing the factory's required output to meet the input demand

## How to calculate child URPS and CR from parent URPS (these rather simple calculations are at the core of the program)

Calculate URPS before CR

- to get parent crafts required per second, divide parent units required per second by parent units produced per craft
- *get child units required per second by multiplying child units required per parent craft by # of parent crafts, calculated above
- get child crafts required per second by dividing child units produced per second by child units produced per craft
- find time per child craft by dividing crafting time by crafting divisor of crafter
- *get crafters required by multiplying child crafts required per second by time per child craft

---

## Ratio Calculator Example Program Trace

**Page One**

- Program Trace Explanation
    - Input is introduced as 3 inserters per second
    - Various objects are introduced to "program memory" (this will be done before input is introduced in the actual program by filling in cells. I could potentially grab .csv data from the game files themselves in order to fill in all the recipes.)
        - A recipe-type object is introduced, which represents all the information required to create an item (an item-type object will have to be introduced in some way; I've only needed recipes so far because recipes only reference other recipes. Items will likely store their own ups & cr).

- A crafter type object is introduced, which represents all the information required to craft an item.

Program Trace : Inserter Prod. (1)

*Input : Inserter @ 3 ups

Recipe object (constant)

Name : Inserter
- crafting time : 0.5s
- Ingredients :
  - Green Chip @ 1 upc
  - Gear @ 1 upc
  - Iron Plate @ 1 upc

Crafter : Assembler

Crafter Object (constant)

Name : Assembler
- Crafting divisor : 0.5

Recipe object (constant)*

Name : Iron plate
- Crafting time : 3.2s
- Ingredients :
  - Iron Ore @ 1 upc

Crafter : Stone Furnace

Crafter object (constant)

Name : Stone furnace
- Crafting Divisor : 1

**Page Two**

- Meta Introductions
    - A legend was introduced describing certain notation
    - The format of recipes was tweaked to include units of the recipe produced per craft (UPPC) as well as loosely group related data points
- Program Trace Explanation
    - A resource-type object is introduced, and will be used to represent a base material such as ore or oil (basically anything harvested).
    - More recipes are introduced to “program memory”.

Legend:
- * represents top-level ingredient
- UPPC - units produced per craft
- URPC - units required per craft

Resource Object (constant)

Name: Iron ore

Recipe Object (constant) *

Name: Iron Gear Wheel
- Crafting time: 0.5s
- Crafter: assembler
- UPPC: 1 units
- Ingredients:
    - Iron plate @ 2 URPC

Recipe object (constant) *

Name: Electronic circuit
- Crafting time: 0.5s
- crafter: assembler
- UPPC: 1 units
- Ingredients:
    - copper cable @ 3 URPC

oops - Iron Plate @ 1 URPC

Recipe object (constant)

Name: copper cable
- Crafting time: 0.5s
- crafter: assembler
- UPPC: 2 units
- Ingredients:
    - copper plate @ 1 URPC

---

**Program Considerations**

Items will likely need a link to their respective recipe

- How will this link be represented/created?
    - The item and its recipe could just have the same name/ID, but have different types; the link would be implied in this case.
    - The item and the recipe could be combined into one object with constant and manipulated values.
    - I think this will ultimately come down to how sheets handles stuff. I doubt I'll be doing any explicit object oriented programming.

When and how will all the objects of various types be initialized?

- Recipes will be hard-coded by the user in sheets, and will be a sort of constant in the context of the program. They will need to reference the crafter they use from the start, and they will be linked to an item at some point (still don't know exactly how this will work). The crafter link will be outgoing and the item link will probably be incoming.
- Crafters will be similar to recipes
- Items will be created as needed (aka when the URPS and CR need to be tracked for a particular recipe). However, they'll be a singleton because only one count of URPS and CR needs to exist per item.
- Resources will be akin to items because they will track URPS and CR; however, they won't be linked to any recipe.