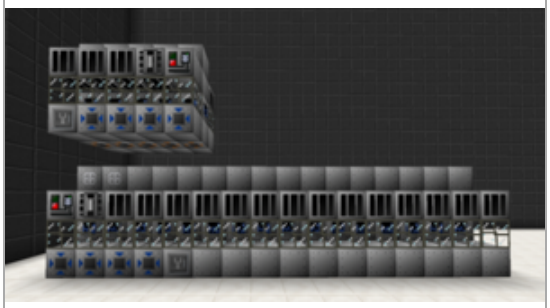
GT New Horizons

# Advanced Assembly Line

The **Advanced Assembly Line** (AAL) is an IV tier multiblock and an upgraded version of the Assembly Line that allows for simultaneous processing (item pipelining) and higher overclocking. It mimics a real assembly line by consuming ingredients one-by-one instead of all at once which allows the AAL to offer parallelism up to the number of input ingredients.

Although craftable at the same time as the Assembly Line, it is *not* recommended to upgrade right away for multiple reasons. 1) The AAL is much more complicated than the Assembly Line and is prone to getting stuck if not setup correctly. A solid understanding of the basics is strongly recommended before building this multiblock. 2) Parallelism is not free and will cost a tremendous amount of power which may be difficult to support without laser power transfer. 3) Automation is significantly more expensive as it requires a lot more AE2 components.

**Advanced Assembly Line**



| | |
|---|---|
| **Mod** | GigaGramFab |
| **Type** | Tile Entity |
| **Relevant Quest** | Advanced Assembly Line |
| **Tier** | IV |
| **Size** | 5-16x4x3 |
| **Pollution** | None |

## Construction

All construction requirements of the AAL are identical to the Assembly Line which means upgrading is as simple as replacing the controller block. Using a Multiblock Structure Hologram Projector will show the player the minimum length AAL, but holding multiple in a single stack can show/build different lengths up to the maximum of sixteen.

### Buses and Hatches

Unlike the Assembly Line, the AAL accepts TecTech Multi-Amp Energy Hatches and Laser Target Hatches to handle the increased power requirements of parallel processing. However, there is no tier skipping; UV energy hatch(es) with any number of amps is not sufficient for a UHV recipe and the AAL will stop due to a crash. This is unlike the regular Assembly Line which can run higher tier recipes with multiple energy hatches.

The input buses can be any tier, but typically Advanced Stocking Input Buses are used because they can auto-pull from buffer inventories. The output bus should replace a solid steel machine casing on layer 1 rather than replacing an input bus on the last slice.

Only four input hatches are required because no AAL recipe uses more than four fluids. There can be gaps between the input hatches if necessary.

# Usage

The AAL will start processing once the input bus and input hatch contents align with any stored Data Stick. The first slice will consume the ingredient in the first input bus and all input hatches in (recipe time / number of inputs) seconds. Once complete, the second slice will start processing in the same amount of time and consume the ingredient in the second input bus. This will continue until the last ingredient in the recipe. If the next slice cannot find the materials in its input bus or is still busy doing something else, the just-finished slice will remain in a STUCK state which will hang the AAL. If this happens, the controller's front face will have its status light turned orange. WAILA HUD will display which slices are stuck.

### Power

The energy cost of this machine is the number of slices active multiplied by the original recipe EU/t. STUCK slices do not consume power. The AAL will use the worst energy supplying hatch's input voltage for calculating the tier of the recipe and overclocks. With higher amp energy hatches, it can overclock beyond the named voltage tiers, but will consume even more power than a usual imperfect overclock. Every such laser overclock will add 0.3 to the power multiplier. For example, one laser overclock will have 50% recipe time and use 430% power but two laser overclocks will have 25% recipe time and use 1978% power (4.3 * 4.6) contrary to the usual 1600%. It is not possible to overclock faster than one tick. The AAL first tries to parallelize, then normal imperfect overclock, then laser overclock.

# Automation (Universal AAL/UAAL)

AAL can be automated in two ways: dedicated AAL and universal AAL.

Dedicated AAL are dedicated to process one recipe. This is more common for highly common ingredients such as circuits. There is practically no automation challenge needed here as you just stuff the things into each hatches in sequence. Ordinary assembly line automation is more than enough to handle this kind of use case.

UAAL aims to process any possible assembly line recipes. However, this is not as easy to achieve. Two of the known automation means are discussed in detail here. It is also easy to implement with an AE2 auto-craft system. A prerequisite is the ME Fluid Processing Pattern Terminal which is the only terminal big enough to pattern all the ingredients in AAL crafts.

### Method 1: Applied Energistics

The core mechanic behind this approach is that ME Chests with set priorities can determine the order in which items are inserted and hold several recipes worth of a single item. Inputs are blocked using an Advanced Blocking Card to avoid recipes from being mixed.

To actually run recipes in parallel on the same AAL, *all patterns must be multiplied*. Instead of a single motor, for example, have the pattern craft sixteen motors (hence the need for buffer chests). This is because the Advanced Blocking Card will prevent any additional ingredients from being inserted while there are still some in the buffer chests.

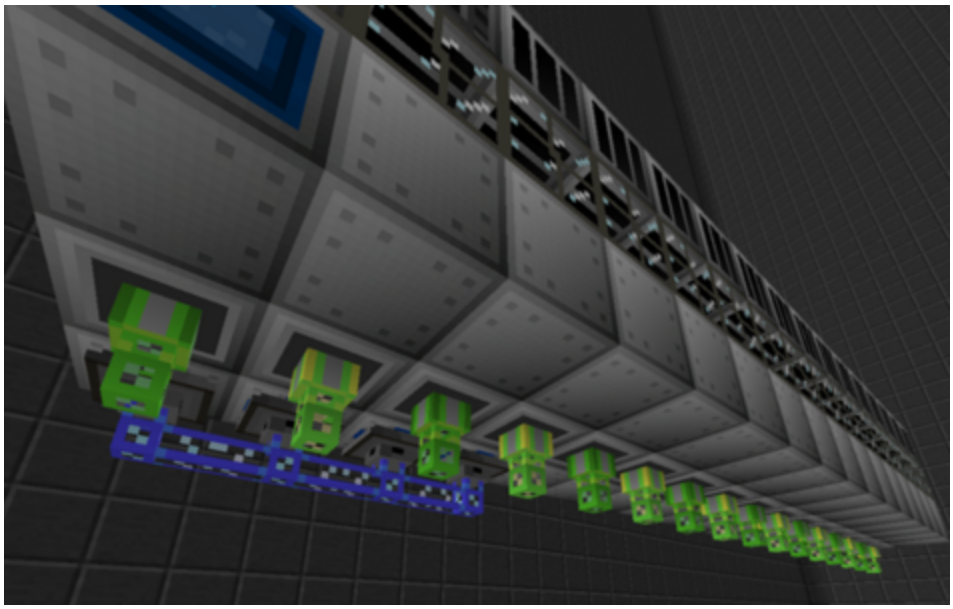
Duplicate items that are full stacks also need to be renamed in an Industrial Material Press. This is because the buffer chests can hold more than a single stack of items, unlike the ULV input buses used in regular Assembly Line automation.

**STEP 1**

Place fluid storage buses on all the input hatches with DESCENDING PRIORITY from the first slice

Use advanced stocking input buses on each slice and enable AUTO-PULL on all of them

Alternate cable colors (such as green and lime in the figure) since each slice is its own AE2 subnetwork

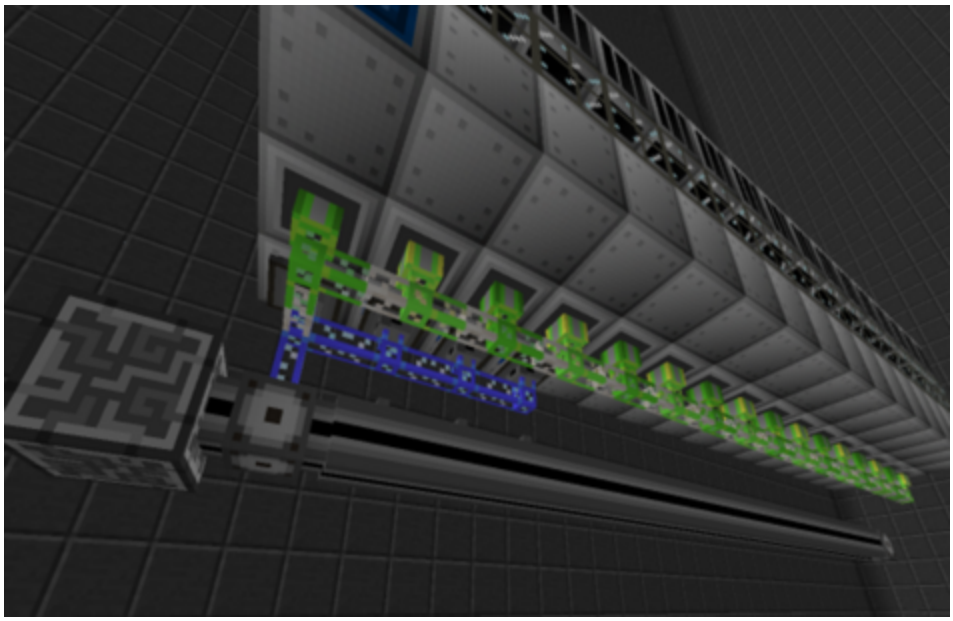


**STEP 2**

Place quartz fiber cables between each subnetwork for sharing power to all devices

Notice there is a quartz fiber cable between the blue and green cable on the first slice

Place an ME controller and dense fluix cables as shown (cable anchors are optional)

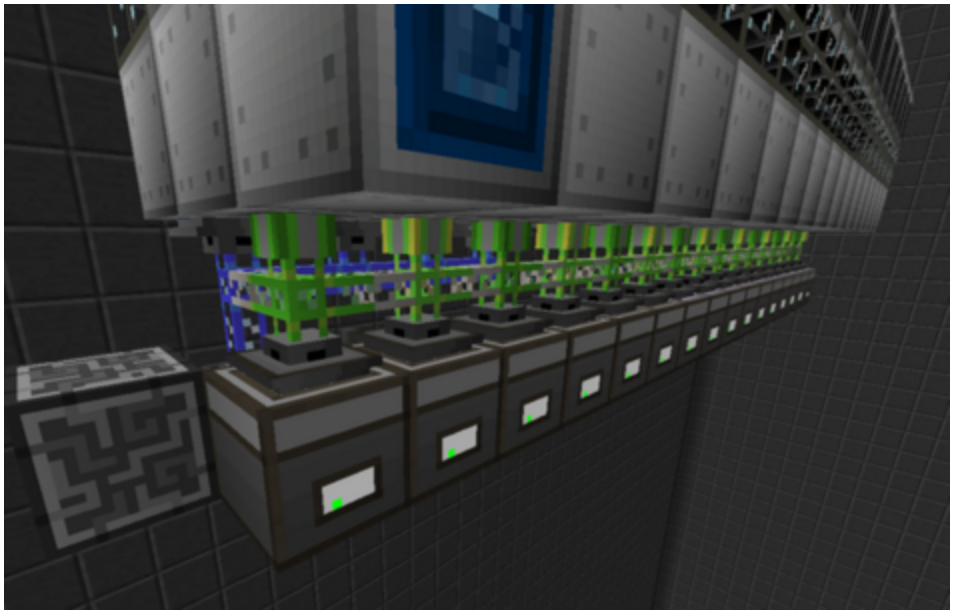


**STEP 3**

Place a storage bus on the bottom of each green cable to read the contents of the ME chest directly underneath

Each ME chest should be limited to a single item type (block container cell works well) and have DESCENDING PRIORITY from the first slice

The ME chests serve as large buffers for each individual slice

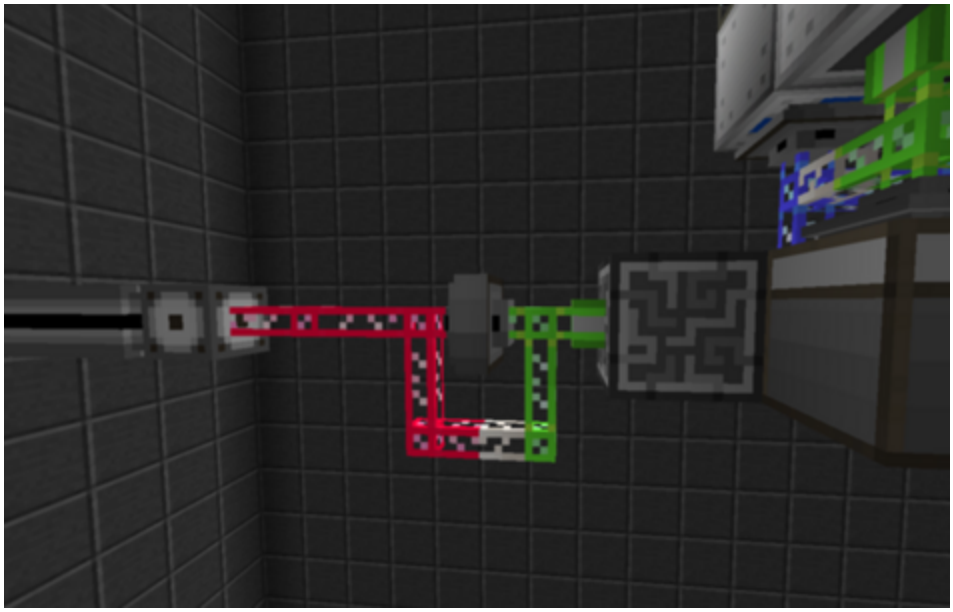


**STEP 4**

On the subnetwork (green), place an ME Dual Interface with an ADVANCED BLOCKING CARD inside

On the main network (red), place an output P2P tunnel – ME Dual Interface with BLOCKING MODE enabled

Power the subnetwork by connecting a quartz fiber cable to the main network or adding a neutronium energy cell (better for performance in the late game)



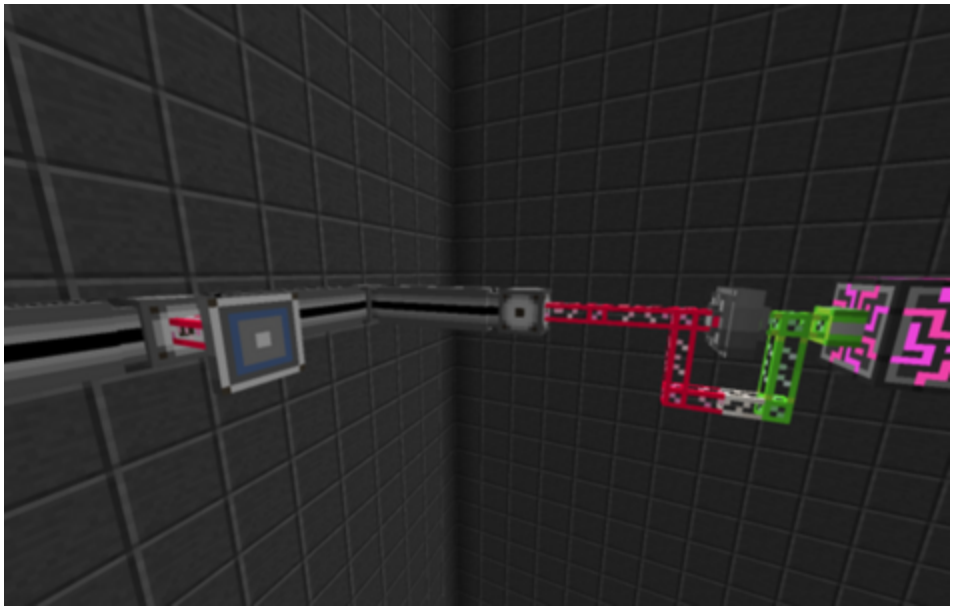
**STEP 5**

Somewhere on the main network (red) should be an input P2P tunnel – ME dual interface with BLOCKING MODE enabled which should hold all the patterns

Parallelizing this approach is as simple as linking more output P2P tunnels to this one input P2P tunnel

More P2P tunnels can be added if more than 36 recipes are needed (4 total is recommended)

TIP: Many P2P tunnels can be linked at the same time with the basic memory card and ring of loki

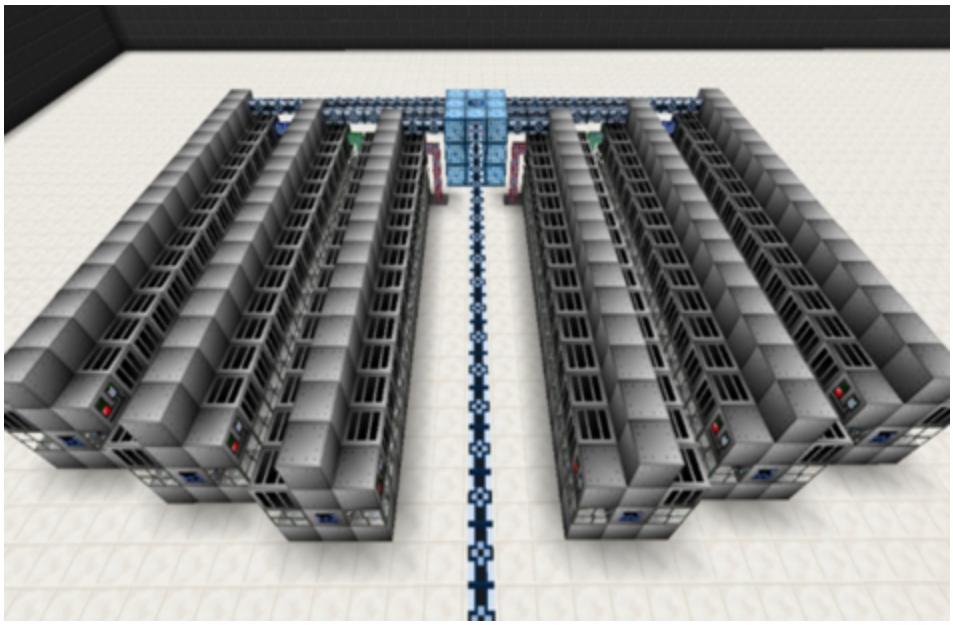


**BONUS**

There are a lot of AE2 cables, laser vacuum pipes, and optical fiber cables surrounding these machines

A setup such as this can help compact everything, save on materials, and scale up very easily

Downward facing data banks are daisy-chained underneath with the optical fiber cables coming up through the ground



# Method 2: Ender Chest + Robot Arm

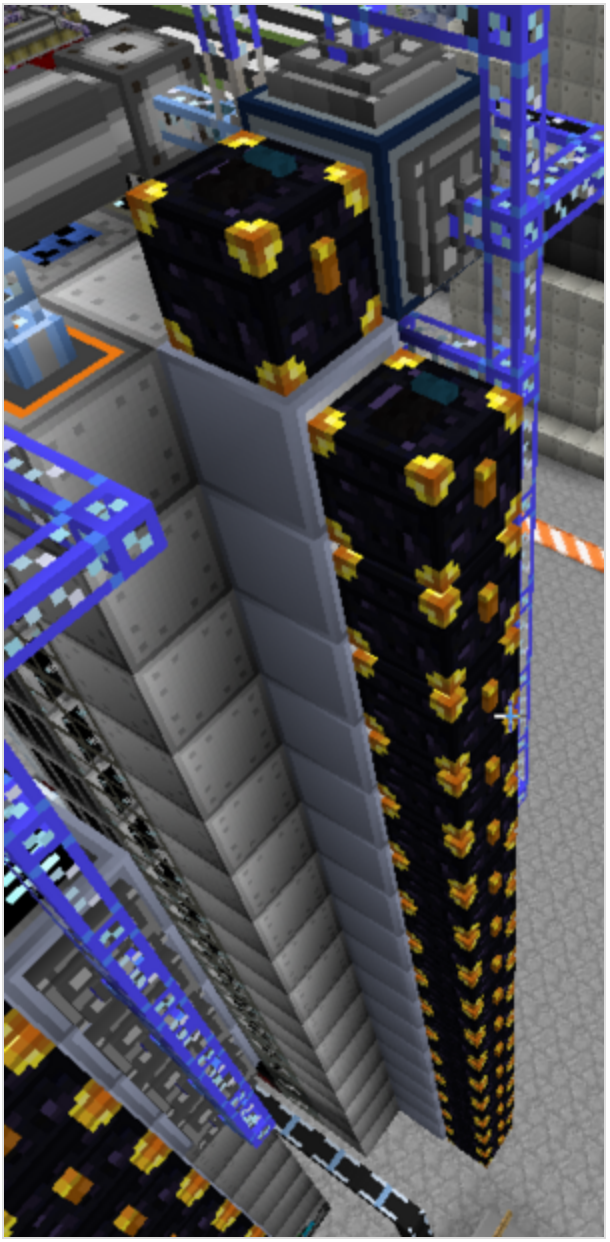
ChickenBone's Ender chest mod (not to be confused by the vanilla ender chest) offers 16*16*16 interdimensional shared inventory globally or privately. This method take advantage of the fact that ME Interface is able to push each ingredients in its original encoding order and can block when there are item/fluid still inside the target buffer.

This system cannot use multiple AAL as executors, so it's most suited for early to mid game setup, and most probably cannot meet your end game production needs. Most notably this system does not require item renaming of any kind.

The system runs on multiple level of buffers:

1. ME Dual Interface block as system input. This interface should be configured to be on blocking mode, and insert stacks to first empty slot. A part form Dual Interface will not work as we need to exploit the fluid output order.
2. Inbound buffer. This uses two types of buffer: ender chests and super tanks. Items will be injected into the ender chest in encoding order. Fluids will go to up to 4 super tank linked by 4 pair of fluid p2ps. The sending end of fluid p2ps should be on different side of Dual Interface. P2P to first input hatch should be on bottom side, second on top and so on. The order should be Down, Up, North, South, West, East.
3. Outbound buffer. This level of buffer takes item from inbound buffer using Robot Arms and a preconfigured slot number. The first buffer will take from slot 1, second buffer from slot 2, and so on. This level is why Ender Chests are required. Each outbound buffer takes item from one ender chest linked to the original inbound buffer. A fluid variant of this tier is not required, but can be helpful for some late game crafts.
4. Input bus/Hatch. For items, we need ULV Input bus in AAL structure to take items from the Outbound buffer with a conveyor belt cover. For fluid, a large enough input hatch with a fast pump cover is sufficient.

Fluid Handling



Item handling

---

Retrieved from "https://wiki.gtnewhorizons.com/wiki/Advanced_Assembly_Line?oldid=11911"