1. General
   1. I wrote a custom disassembler to produce each of the prg and chr asm files. The disassembly was performed with reference to a code & data log (CDL) from Mesen 0.9.7
2. Building
   1. To build, you need ca65 and ld65 (not provided). Put the \_cryasm.bat, crystalis.cfg, and all the asm files into the directory with ca65 and ld65, then run the \_cryasm.bat file
   2. \_crystalis\_master.asm .includes all of the other asm files and contains the iNES header
   3. There are a few asm files for defining constants and memory locations
3. PRG Banks
   1. PRG banks 0x0 – 0x6 have no known code (just data and unknowns)
   2. PRG bank 0x6 contains the sprite (including monsters) data array, and also contains a pointer table with pointers to each sprite in the array (size of sprites appears to be variable based on prior analysis from romhacking.net)
   3. PRG bank 0x8 contains some GUI and shop processing code including price/inventory array for the shops (likely the inn as well) and program logic for the inn (presumably other shops as well)
   4. PRG bank 0xD contains sword charging code, level up code and combat code (damage monsters and the player in combat). It also has code to handle the refresh spell. Also has sword attack (and level 3 mp cost), armor/shield defense power and gold drop amount arrays. Don’t know what else
   5. PRG bank 0xF probably has the main game code. It seems to always be loaded at 0xC000 and never switched out for a different bank. Don’t have many specifics known yet, but it does have code to update the player stats after selecting items and equipment in the menu. It also has the main game loop and some code to change scenes (like entering/leaving a building or a shop, moving from town to/from overworld, etc.)
4. CHR Banks
   1. All the CHR banks have verified code in them

**IDENTIFIED SUBROUTINES**

|  |  |  |
| --- | --- | --- |
| Bank | Offset | Name |
| PRG 0x8 | 0x04DA | GUI\_EVENT\_INN\_PREPARE\_DIALOGUE\_CHECK\_LOOP |
| PRG 0x8 | 0x02B7 | UpdateEquipmentSelection |
| PRG 0xD | 0x20B0 | NestedSubroutine\_RestoreHPFromRefresh |
| PRG 0xF | 0x0008 | UpdatePlayerEquipmentStats |
| PRG 0xF | 0x0169 | UnknownSub\_0xC169 |
| PRG 0xF | 0x2F55 | ApplyEnvironmentalEffectsToPlayer |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**INVENTORY MANAGEMENT**

So far, a few things are known about inventory management:

* UpdateEquipmentSelection (PRG 0x8 offset 0x2B7) is called when the player selects/deselects something in their inventory.
* Based on analysis of the UpdateEquipmentSelection function, there is an inventory numbering system that begins on the weapon/armor screen in the top left corner (starting with the slot for the Sword of Wind and ending with the slot for the Ball of Thunder / Storm Bracelet). The slot for the Sword of Wind is index 0 and the index goes up by one from left to right (so the index for the Sword of Thunder slot is 3) and when the right most slot is reached, dropping down to the next row at the left again (so that the slot for the Ball of Wind / Tornado Bracelet is index 0xC, and the slot for the Ball of Thunder / Storm Bracelet is 0xF). The index picks up at 0x10 on the items page of the inventory, again starting at the top left and going to the right and then down once the end of a row is reached. The final inventory index is 0x2F (the slot for the Flight spell).
* Also based on analysis of the UpdateEquipmentSelection function, there seems to be a master item index that is different from the categorical indices corresponding to swords, armor, shields, consumable items, etc. This is also evident from the shop management code (not yet documented in this file but noted from the crystalis bytes program on romhacking.net)