MechCommander Gold Campaign Builder's Guide

Part 1 – The Basics

Revision 1.1

By Cmunsta

Revision History

Rev 1.1

- Fixed the hyperlinks to the tools to reflect their new home.
- Fixed a few more typos

Rev 1.0

- Reworded info about the LastScenario value to try to make it a little clearer.
- Added note about changing the music for a campaign mission.
- Expanded information on the removing extraneous mechs and vehicles from the store.
- Expanded the information about testing the campaign.
- Fixed Final Words section to reflect the new situation with the guide series.
- Duplicated all tables into their own appendix at the end of the document.
- Fixed More Typos

Rev 0.3

- Removed extraneous semi-colon in step 6 and in file listings for step 6.
- Added extra note that the start0.pkk file to be changed in step 4 is the one previously overwritten with the .SOL file.
- Changed the file0.txt and file1.txt filenames to something more appropriate.
- Added, fixed and/or reworded sections due to the discovery of the effect of the Setting entry in the Planet section of the master definition file in the PKK file.
- Created sidebars for tips and other information.
- Enhanced the section describing the command shell, and moved it into a sidebar.
- Moved the section describing the variable types in a FIT file to a much earlier point in the document, and placed it with a general FIT file description.
- Placed the whole FIT file description into a sidebar.
- Reworked the entire packaging section (step 8).
- Modified the proposed filename prefixes for FSTs to reflect the changes in step 8.
- Added information about editing TGA image files.
- Fixed improper description of where to place the edited main menu button images.
- Added description of possible font to use for re-creating the main menu button typeface.
- Added description of how the warrior profile filename is constructed.
- Added assigned vehicles section to step 6 and file appropriate listings to appendices.
- Added link to the InstallCDP campaign installation tool.
- Added notes on how to add, remove, and change the components, warriors, and mechs in the store.
- Added a table of component IDs.
- Fixed various typos.

Rev 0.2 (not released)

- Fixed various typos.
- Changed the main text layout to use full justification for a cleaner look.
- Added a few clarifications.
- Changed “version” to “revision” as this is more appropriate.
- Added revision history page.

Rev 0.1

- First release of the document.

Introduction

This document is meant as a tutorial on creating user campaigns for MechCommander Gold. It is subtitled The Basics as there are many advanced topics I could go into but would be inappropriate if starting out. This tutorial should, however, contain enough information for someone to create a working campaign.

Some of the information presented here was collected from various sources. Many of the websites no longer had the relevant pages available, or were simply not around anymore (thank you "wayback machine"). Further information presented here is from my own research into the subject.

Before Starting

Currently, the only known method for creating a working user made campaign is to override either the original or expansion campaigns. This means that some of the files originally installed by MechCommander Gold will need to be backed up.

Though backing up the entire game folder will certainly do the trick, only a few files actually need to be backed up. These files can be found in the `{MCX}\data\savegame` folder. Where `{MCX}` represents the folder you installed MechCommander Gold in.

Open Windows Explorer or whichever program you use to navigate around your files, and go to the `{MCX}\data\savegame` folder and copy the files `START0.PKK` and `XSTART0.PKK` to a safe location.

To restore the normal campaigns after playing a user made campaign or in case of some trouble, do the following:

To restore the original campaign, copy `START0.PKK` from the safe location back to the `{MCX}\data\savegame` folder and remove, rename or delete the `MechCmdr1.FIT` file from the `{MCX}\data\missions` folder if present.

To restore the expansion campaign, copy `XSTART0.PKK` from the safe location back to the `{MCX}\data\savegame` folder and remove, rename or delete the `xMechCmdr1.FIT` file from the `{MCX}\data\missions` folder if present.

You may also need to remove, rename or delete the user campaign's .FST file. More information on FSTs will be explained in Step 8.

Tools of The Trade

Obviously, to create a campaign we are going to need to create maps. So install the MechCommander Editor from the game CD or ISO if it isn't already installed.

A good text editor will be needed to edit many of the files. Though Notepad or WordPad can certainly do the job, I recommend downloading Notepad++ (a free program available on the Internet).

As many of the files that will need to be edited are compressed and packaged into other files, some tools to extract and repackage them will be needed. I've created a few simple command line tools to do this and will be the tools described in this document. Simply download them and unzip the tools somewhere easily accessible. These tools were made quickly so don't expect too much from them. You can get the tools from my web site here:

http://www.freewebs.com/therealcmunsta/downloads.htm

For the programmers out there, the source code to my tools can be downloaded from the above link as well.

Also, a text document describing the data file formats can also be downloaded from the same location.

These are not needed for general campaign editing; I'm just providing these links so others can have working examples with which to make better tools.

I also suggest getting the Mission Editor Extender. Though not needed for this tutorial, it allows adding extra objectives and other fun things. It does come with a mission linker for campaigns, but its linker assumes a few things that are not desired so I won't be using it to link campaigns. Download the Mission Editor Extender from the author's website.

http://www.colossus.demon.co.uk/mechcommander/mechcommander.htm

The source code to the Mission Editor Extender is also available from the link listed above

For a simple tool to install CDP campaign packages, you can try my simple tool InstallCDP also available on my site (the source code for it is also available).

Campaigns by Example

To help explain how to make a campaign, I'll be creating a simple one in step-by-step fashion.

The first step should be rather obvious. Maps, maps, maps. Of course the ones I'm going to create are going to be incredibly simple as the contents of each map are not important at this point.

The order I'll be doing things in isn't necessarily the best way of making a full campaign, but it should help demonstrate every step that is needed to be done.

This tutorial will be making use of simple tools I made and manually editing files. The reason for this is that not many tools currently exist to help automate the process, and of those that do, have made certain assumptions that may not hold true any longer.

Further, though it seems to complicate things at first, learning how to do things manually will grant a greater understanding of how a campaign is created and used by the game.

In the appendices of this document are the full listings of the files that have been modified. This way, if a section is not completely clear as to what needs to be changed, refer to the listings in the appendices to review how the entire file should be.

Step 1: Making The Maps

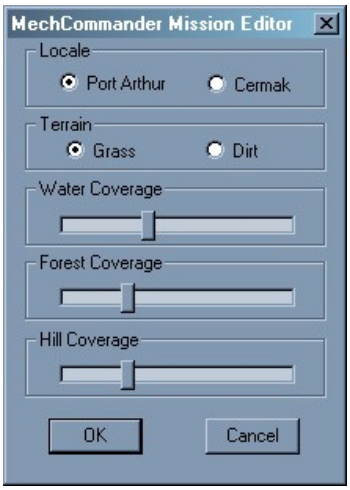
For this example I'll be creating a three-mission campaign.

All maps, no matter how simple or complex, must contain three things: A drop zone, an objective, and at least one enemy. As this isn't a tutorial on how to use the editor that comes with MechCommander Gold, I'll assume some minimal knowledge on using the editor and will not go into how to place drop zones etc. The best way to learn how to use the editor is to simply open it up and use the built-in help.

Start the editor and select the "Solo Mission Editor". The solo maps will be converted for use in the campaign in the next step. For now, we just want to get some maps.

Select "New Map" in the next dialog.

There have been some rumors going around that a campaign has to start with a Port Arthur map if overriding the original campaign, and use a Cermak map if overriding an expansion campaign. This is not quite true. We will see in step 4 that we can in fact start either campaign with either type of map, but until then if you are making a Cermak map to start while following this guide, you will want to override the expansion campaign. For this tutorial, I'll start with a Port Arthur map.



Once in the map editor itself, place a building down somewhere on the map, and also place a drop zone. I always try to ensure that there is enough space around the drop zone for the game to put in other Mechs.

Now make the objective of this mission to capture the single building.

Only one thing remains to be done to make this a valid map. An enemy unit needs to be placed somewhere on the map. As this is only an example map, I'm only going to place a single Swiftwind. These don't shoot back so it is easy to test things.

Add a suitable player briefing and save the map. Don't use the "Save map only" option, as this will remove any objectives etc. Also don't have the editor write a distribution package (.DPK). That step will be covered later, and will be done using my tools instead of the editor.

I chose to call this map something sophisticated like MAP1. Almost any name will do but it is easier to remember which map is which if given appropriate names. The name will not show up in game (unless playing the game in a window), so it doesn't really matter what it is called here. Notice that a maximum of eight characters can be used for the map name.

Tips for Naming Your Maps

Be aware that we will be overriding one of the official campaigns. Also, the player may have another user campaign installed as well. Therefore, though the game doesn't care what the name of the map actually is, if the name happens to be identical to a campaign you are **not** overriding, then if the player tries to play that other campaign, your maps may show up in that campaign, or worse, those maps may show up in your campaign.

There are a few easy solutions for this. One way is to always start your map names with a few characters specific to you (your initials, nickname, etc). Another would be to use a short version of the campaign's name (a campaign called "tutorial" could have all its maps start with "tut").

Yet another method would be to name them the same as the campaign you are overriding (the original campaign names its maps "`mis`*OOMM*", and the expansion campaign names its maps "`mcx`*OOMM*" where *OO* is the operation number and *MM* is the mission number). I don't suggest this last method however for a few reasons. First, I'll be describing a method that will allow your campaign to be installed as either an original or expansion campaign. Second, the official campaigns don't always use the same name for all the parts of a map, so even though you may have overridden part of the map, you may not have overridden all the files comprising that map. Therefore, if you wish to use this method, you should only use the generic packaging method I describe in step 8.

For the second map, I decided to make it a "Destroy all enemies" mission. So create a new map, put in a drop zone, a couple of easy enemy mechs, and save the map. Note that the default objective for a map is always "Destroy all enemies" so no need to go add that objective in. Give the map another suitable player briefing and save the map as MAP2.

For the third map, I decided to make it a “reach the extraction zone” type of mission, so again, start a new map, place a drop zone and an extraction zone marker on the map, make the objective to reach the extraction zone marker, and also put an enemy somewhere on the map (I chose another Switfwind). Give the mission an appropriate player briefing and save it as MAP3.

Close the editor and try out each map as a solo mission to make sure they work as intended. Once satisfied that each map works, exit the game. It’s time to turn these maps into a campaign.

Step 2: Linking the Maps

Before explaining how to link maps together, I should very briefly describe what is going on behind the scenes when a player hits either the "Original Campaign" or "Expansion Campaign" buttons in game.

When a player presses the "Original Campaign" button in the game's main menu, the game goes off and looks for a file called `MechCmdr1.FIT`. If the player pressed the "Expansion Button" then the game looks for the file `xMechCmdr1.FIT`. These are simply text files that describe, among other things, which map to use for the campaign. Therefore, it should be obvious that these are the files of interest. Since this example campaign is going to be using the "Original Campaign" button to launch itself, the `MechCmdr1.FIT` file is what must be created.

Luckily, the map editor has already done most of the work of creating this file for us. All solo missions also have a campaign file made for them by the editor, but it is named differently and only has one map, but otherwise is exactly what is needed.

So the first step is to find this file and copy or rename it to `MechCmdr1.FIT`. In the `{MCX}\data\missions` folder will be three files called `campaign`*XXX*`.fit` (where *XXX* is one of the map names we created in step 1). Copy one of these files and call this copy `MechCmdr1.FIT`. I chose to copy `campaignmap1.fit` as this is the first map for this example campaign, but any of them will do. Keep the `MechCmdr1.FIT` file in the same folder as the solo campaign file.

Open `MechCmdr1.FIT` in a text editor. Look for the heading named `[Scenarios]` around line 15 or so. This is where the needed changes will start. Don't change the entries before this header, as they are needed by the game when it starts up and may cause problems.

About .FIT Files

Almost all of the files we'll be editing are .FIT files, so getting to know how they work is important.

All .FIT files start with `FITini` and end with `FITend`. Everything between these two words are the contents of the .FIT file. Anything after `FITend` will be ignored by the game.

The contents of .FIT files work much like a Windows .INI file. That is, they have a section heading enclosed in square brackets and a list of data entries under that section heading.

Those one and two letter codes at the start of each entry tell the game what type of data to expect for a given entry.

About .FIT Files

For example, the "st" in front of the movie0 entry in the [Movies] section means that movie0 will have a text string after it. The "ul" means that the numScenarios entry (in the [Scenarios] section) will contain a positive long integer (32 bits wide from the game's stand point).

Here is a table of valid codes:

Code	Long Name	Description	Value Range
b	boolean	A true or false value	Only two values. True (t) and False (f)
c	char	A character or small integer	-128 to 127
uc	unsigned char	A character or small positive integer	0 to 255
l	long	A 32 bit integer value	-2147483648 to 2147483647
ul	unsigned long	A positive 32-bit integer value	0 to 4294967295
f	float	A real or floating point number	-3.4028e38 to 3.4028e38
st	string	A series of characters delineated by double quotes	

The first change should be obvious at this point. As the example campaign has three maps, we need to change the number of "scenarios" to 3, so change the line:

```
ul numScenarios = 1
```

to

```
ul numScenarios = 3
```

This entry actually represents the total number of scenario maps listed in this file. It is important to realize that more maps can be listed than actually used, but this number must reflect the number listed.

The next line in the file also needs to be changed. It currently looks like this:

```
ul LastScenario = 0
```

This line tells the game which scenario number is last in the list to be played. We need to change this to the correct scenario number. This entry also should reflect the last scenario in the list that is to be played even if not all maps are used. Notice however, that

the scenario list starts counting at 0 and not 1. This makes the number we need to use one less than the total count. The actual number to use for this entry can be gotten once the scenario list is created but as this is just an example, the value to use is easy to get. Change the line to this:

```
ul LastScenario = 2
```

The actual list of scenario maps comes next. Currently the next line doesn't look like much of a list, as there is only one map entry here. But recall that this came from a solo mission so this is to be expected. Change and/or add lines so that the scenario list looks like this:

```
st Scenario0 = "map1"
st Scenario1 = "map2"
st Scenario2 = "map3"
```

Note how this list is made. The names are all called ScenarioX where X is a number that increases by one for each map, and also starts at 0. Now notice how the last scenario listed is Scenario2. The "2" is the number given on the LastScenario line. These two entries must be equal. Of course, the names given for each scenario is the names of the maps I made in step 1. If I had saved the maps as tutorial1, tutorial2 and tutorial3, then those would be the names I would need to use instead of map1, map2 and map3.

Next up to be changed is the list that describes the operation and mission numbers for the campaign. This list shows what scenario map is associated with which operation/ mission. I think this also affects the save game files that are created. Since this example campaign is overriding the original campaign, the save files the game creates will be called `PortArthurBefore_Op`*X*`Mis`*Y*`.sav` etc. Note that the *X* and *Y* entries will reflect the operation and mission numbers provided in the scenario list.

I decided to make the first map its own operation, and the next two to be operation two, missions one and two respectively. Edit the operations list to look like this:

```
l Scenario0Operation = 1
l Scenario0Mission = 1
l Scenario1Operation = 2
l Scenario1Mission = 1
l Scenario2Operation = 2
l Scenario2Mission = 2
```

Note how the names reflect which scenario it refers to from the scenario list previously created. This is how the game will associate each scenario map with the correct operation and mission numbers.

Now all that's left to change is the order in which the game will play each map. Huh? Wasn't that done already with the scenario list? Actually, no.

The scenario list is just that; A list of maps that this campaign may use. It doesn't even have to be in any particular order. Nor is the operation list we just finished making need to be in any particular order. It just needs to say what operation and mission number a given map represents. Nothing more.

The order in which the game actually plays each map is defined in a series of entries called game segments. But a game segment is more than just a map. It is also used to run the game's main menu for example.

In fact, note that there are two game segments already defined in the file even though this came from a solo mission. The first game segment, called `[GameSegment0]`, is actually used by the game to run the main menu. So this first game segment should not be changed. The second game segment should look like this:

```
[GameSegment1]
ul GameState = 3
ul NextGameState = 16
ul NextGameSegment = 2
ul SmackerMovieId = -1
ul ScenarioId = 0
```

This entry is also correct and doesn't actually need to be changed. There are three entries in this segment that should be of note. First is the `GameState` entry. This tells the game what to do. Valid game states are:

START_GAME	0
MAIN_SCREEN	1
START_CAMPAIGN	2
LOGISTICS_SCREEN	3
BRIEFING_SCREEN	4
GENERAL_ORDERS_SCREEN	5
SCENARIO_RESULTS_SCREEN	6
IN_SCENARIO	7
IN_SMACKER_MOVIE	8
NEXT_SEGMENT	9
START_SMACKER_MOVIE	10
START_SCENARIO	11
EXIT_GAME	16

So the `GameState` entry says to go to the logistics screen. Which is what we want.

Ignore the `NextGameState` entry, it is correct but doesn't do exactly what it suggests. Even though the value seems to indicate that the game should exit, this value is correct.

The next entry to take note of is the `NextGameSegment` entry. This entry does do what it suggests, which is when the current game segment is done, which game segment should come next. This is how the mission maps can be ordered. It is best to simply keep game segments in order to avoid confusion, though any order could be established other than `GameSegment0` which, as explained, is used by the game to run the main menus. The number to put here is the game segment number. Note that this already points to "2" even though there is no `GameSegment2` listed in the file. This is how the game knows it is the last game segment. The next segment simply doesn't exist.

The `SmackerMovieId` entry should be set to –1 as we have no videos to play.

The last entry to note is the `ScenarioId`. This entry associates a game segment with one of the mission maps listed earlier. Since the first map to play in the example campaign is map1, and that mission map is listed as Scenario0 above, the number to put here is 0.

So as stated, `GameSegment1` is correct so no changes needed here. But there are to be three maps in this campaign, so copy the whole `GameSegment1` entry, and paste it in twice to make two new game segments.

Change the name of the first new segment to `GameSegment2` and the other to `GameSegment3`, and change the `GameState` entry of both new segments to be 11. The first map should always have a `GameState` of 3; the remaining maps should have a `GameState` of 11. I'm not sure why, but that is how the official campaigns are made, so lets not change this.

The `NextGameSegment` entry for `GameSegment2` should be changed to 3 so that the maps are linked, and the one for `GameSegment3` should be changed to 4. As before, though `GameSegment4` doesn't exist, this will tell the game that this is the end of the campaign.

The `ScenarioId` entry should be changed to 1 for `GameSegment2` and changed to 2 for `GameSegment3`.

After all this, the three game segments for the campaign should look like this:

```
[GameSegment1]
ul GameState = 3
ul NextGameState = 16
ul NextGameSegment = 2
ul SmackerMovieId = -1
ul ScenarioId = 0
```

```
[GameSegment2]
ul GameState = 11
ul NextGameState = 16
ul NextGameSegment = 3
ul SmackerMovieId = -1
ul ScenarioId = 1

[GameSegment3]
ul GameState = 11
ul NextGameState = 16
ul NextGameSegment = 4
ul SmackerMovieId = -1
ul ScenarioId = 2
```

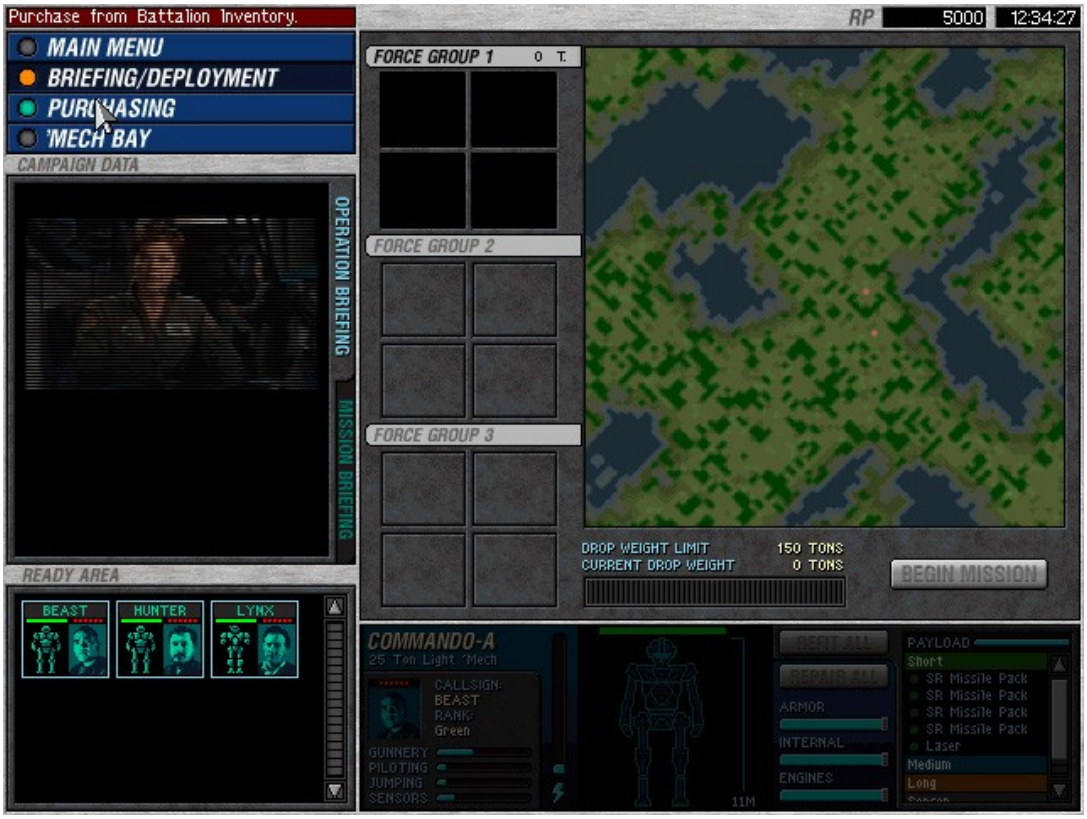
Save the file and exit the text editor. It is time to test this out. Start MechCommander Gold and start an “Original Campaign”.

The first thing of note is that the original briefing video shows up. Also, the original campaign’s money limit of 5000 RPs is present, as well as having Beast, Hunter and Lynx already in mechs.

Going into the Mech Bay reveals that Hawk is also available just like in the original campaign.

The purchasing screen also reveals the current component inventory and what is available to purchase is also identical to that of the original campaign.

Here is a picture of the result I got:



I set the 150-ton limit for the first mission myself when I made the first map in the editor. So that too has come through.

The campaign should be completely playable at this point. Try it out.

Something to note while playing through the campaign is that none of the objectives have cash rewards.

Clearly, some work still needs to be done.

Step 3: Objective Rewards and Drop Zones

Now that a basic campaign is functional, it is time to add in a few features to the missions. Specifically, cash rewards for each objective.

Yes, I know that the Editor Extender can do this, and it certainly does help in the task, and you are welcome to use it. But as I stated at the beginning of this document, I'm doing things manually because it shows what is actually going on instead of having it hidden behind the scenes. Knowing what is being changed will help understand what the Editor Extender is doing in the first place.

Further, the Editor Extender cannot change the rewards for missions and/or campaigns created by someone else as the files it changes are for the editor prior to saving the mission, not the game files. What I'm showing here will apply to any campaign or mission. Well it will when I show how to get to those files in a later step. For now, this step concentrates on what should be changed.

The objectives of a mission are held in a definition file for each mission. The editor has already created the files for each mission in the campaign; all that needs to be done is to edit them. The editor saves the definition in a .FIT file named after the map. In the case of the example campaign, this is `map1.fit`, `map2.fit` and `map3.fit`.

Lets start with the first map. Open `map1.fit` in a text editor (you can find the file in the `{MCX}\data\missions` folder).

Take a quick look through the file, there is a fair number of things we can change in here. I won't describe everything as that will take much to long, but there is some interesting things to play with in this file.

First, lets change the number of artillery strikes available. Lets have two camera drones, and three small artillery strikes. They aren't really needed, but this is a tutorial after all. Again, I know this can be done directly in the editor, but I did it on purpose. For the same reason I'm not using the Editor Extender.

Change the appropriate lines under the `[Artillery]` heading to this:

```
1 NumSmallStrikes =  3
1 NumCameraStrikes =  2
```

While changing the artillery strikes is interesting, it is not the reason for editing this file. Scroll down and find the heading `[Objectives]` (around line 143 if following this tutorial closely). Here is where the objectives for the mission are defined. The map only has one objective of course, and this is reflected in the `NumObjectives` entry.

Following this however is the heading `[Objective0]`. This is the definition of the first objective. It starts counting off at 0 instead of 1, which is why the first objective is shown as 0. If there were a second objective, this would appear as `Objective1`, the third as `Objective2` etc, all the way to `Objective7`, which would be the last possible objective. The game imposes a maximum of eight objectives per mission, even though the editor only allows for four.

The first entry under the `Objective0` heading is `Name`. This is the name that will appear in the "briefing" panel in game and at the end of the mission when the game ticks off the completed objectives and shows the reward given for each. This is also the name that was set in the editor's objectives dialog. Note that this is not the same as the objective list shown in the player briefing (the player briefing text will be dealt with in a later step).

The next entry is `Type`. Note that the valid values for this entry are already shown in a comment at the end of this line. The invisible type means that it does not show up in the briefing panel until it is time to do the mission. This is how to hide an objective in game until another one is complete.

The `TimeLeft` entry should be obvious. I won't be covering timed objectives in this tutorial, so leave it alone for now.

The `Status` entry is only really there for saved games. It would be rather pointless to start a mission with an already completed or failed objective.

And finally, we get to the `Points` entry. This is the entry to change in order to hand out cash rewards for an objective. Lets change this entry from 0 to 10000. The line will then looks like this:

```
l Points = 10000
```

That handles the objective rewards, now what about the tonnage limit and the number of mech slots in a drop zone.

Lets change the max tonnage for the mission to 150. I had already done this in the mission editor, but lets do it here anyway. Scroll back up the file to find the `[Campaign]` heading. The entry that interests us at the moment is the third entry called `MaxTonnage`. Change this value to 150.

```
l MaxTonnage = 150
```

The next entry in this section dictates how many drop zones (1-3) are present. As I only defined one drop zone in the map editor, this should already be set to 1.

Now lets changed the drop zone.

Scroll down a little until you see the heading `[DropZone0]`. This is the definition for the first drop zone. Since a map can have up to three of these, there could also be a `DropZone1` and `DropZone2`. But I only placed one on my map so there is only `DropZone0`.

The first entry in the drop zone section is `NumSlots`. Not exactly rocket science as to what this controls. Lets limit this drop zone to 3 mechs.

```
l NumSlots = 3
```

Next up are the two entries `PositionX` and `PositionY`. This is the map coordinate location where the drop zone is. It is better to leave this to the editor as setting numbers here could place the drop zone in an invalid location.

Next are four sets of three entries. It should be fairly obvious how these work. Each entry in a set ends in a number from 0-3. This makes sense as these came from a drop zone that had four entries. So the last set (`OffsetX3`, `OffsetY3` and `Rotation3`) need not be present anymore as the number of slots has been reduced to 3. Leaving them in shouldn't hurt anything, but lets keep things nice and tidy. Remove that last set.

As to the other sets, they work like this. The `OffsetX` and `OffsetY` entries tell the game how far from the drop zone position a mech should be placed. By changing these values, any desired formation can be created. Be warned that putting the mechs too close to each other can cause havoc. The default 100 seems to be a good number as that is what the editor puts in.

The `Rotation` entry dictates how to rotate the mech when placing it at the drop zone, in effect changing the direction the mech is facing. 0 = Facing South, 90 = Facing East, 180 = Facing North, –90 = Facing West. Rotations of 45, 135, -45 and –135 are also valid.

Lets make the first mech face north, the second face southwest and the third southeast. Change the relevant entries as follows:

```
f Rotation0 = 180

f Rotation1 = 45

f Rotation2 = -45
```

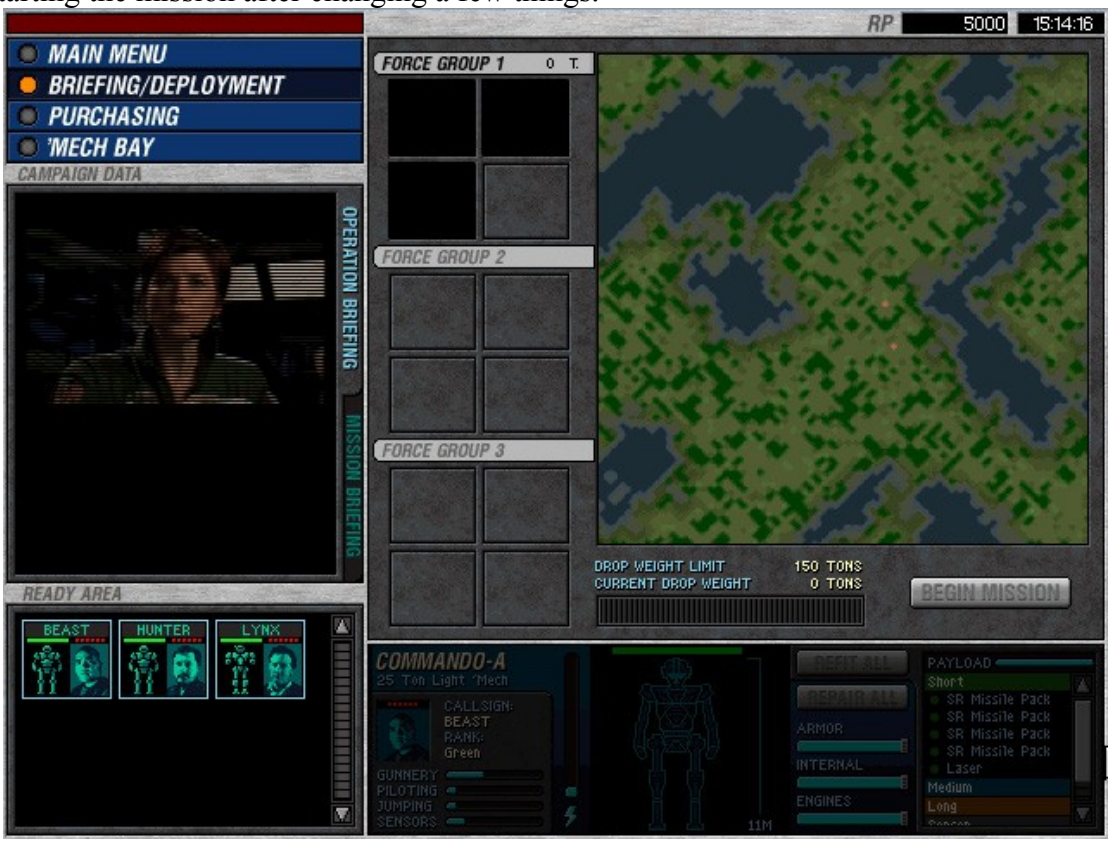
This about covers everything that needed to be changed for now. Save this file and open `map2.fit` and change it as desired. Do the same again to `map3.fit`.

Once done, save the files, exit the text editor, and try out the campaign.

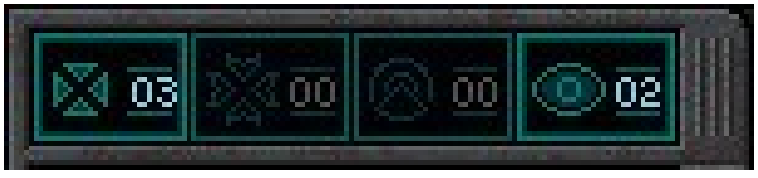
Note how the drop zone has only 3 slots and the drop weight limit is 150 tons.

The briefing video is still playing and the campaign still starts with Beast, Hunter and Lynx ready to deploy. Hawk is still in the Mech Bay, and the mission starts with only 5000 RPs.

Starting the mission after changing a few things.



During play, note that the first mission now has 3 small strikes and 2 camera drones.



Playing through the first mission, the objective rewards are now handed out.

After playing the first mission:

Something else to notice during play. Even though the mission does indeed have three small strikes and two camera drones, the mission's briefing text at the logistics screen still says that there are no strikes, cameras, or anything else. This is actually because the briefing text is held elsewhere and is not generated at mission start. I'll be going over that detail later.

Well, things are starting to look up. Still a fair bit of work ahead though.

Step 4: Using PKK Files

At the start of step 2, I briefly explained what was going on behind the scenes when one of the campaign buttons was pressed. I purposely skipped mentioning that it also looks for a second file. Now it is time to explain about this file.

After opening the `MechCmdr1.FIT` file (or `xMechCmdr1.FIT` file in the case of the expansion), the game also looks for and loads a file called `Start0.PKK` (in the case of the expansion, it looks for `XStart0.PKK`).

This PKK file is effectively a saved game containing the state the campaign should have when the campaign starts.

Now it should become obvious why I asked that these PKK files be backed-up. These files are going to have to change in order to get the proper setup desired in the campaign.

Also note that the solo mission maps that were created in step 1 also come with their own starting files. The map editor creates one of these files for each solo mission map. Though in this case, the file ends up with a .SOL extension instead of .PKK. But apart from the file extension, they are the same thing.

So to start things off, simply copy one of the .SOL files created by the editor over the `Start0.PKK` file (you did back up the file first right?). I replaced `Start0.PKK` by copying the `map1.sol` file over it.

Once this is done, try out the campaign again.

The first thing to note is that there is no briefing video that appears, nor do Beast, Hunter and Lynx show up ready to be deployed.

The starting money is now 100000 RPs, the Mech Bay is empty, and we have no components, mechs, or MechWarriors pre-assigned.

The purchasing screen has all IS mechs available, and in unlimited quantities. All MechWarriors are available, as well as unlimited components and vehicles.

Well it is an improvement. But not quite what is desired.

After copying the .SOL file over the Start0.PKK file:

Since editing the contents of the PKK file is clearly what needs to be done in order to give the campaign the proper starting conditions, learning how to do this will be critical.

PKK files are not text files, but rather a collection of text files, compressed and packaged into a single file. This is where my tools come into play. Create a folder elsewhere on the hard drive for PKK editing purposes, and copy the `Start0.PKK` file into it (use the version of Start0.PKK that we just overwrote with the .SOL file, not the backed-up copy).

Also copy the tools `solextract.exe` and `makesol.exe` into this same folder.

Open a command shell and navigate to your work folder.

Using The Command Shell

Because the tools that I use in this tutorial all use the command shell to execute, here is a crash course in using the Windows Command Shell.

To start a command shell, Press the Windows Start button and select Run. Type `CMD` in the edit box and then hit OK.

Note how the start of the command line tells you what drive and which folder you are currently in. This is the current folder.

To create a folder, use the "`md`" command. For example:

```
md WorkFolder
```

Will create the folder "`WorkFolder`" in the current directory.

To change the folder you are currently in, use the "`cd`" command. Example:

```
cd WorkFolder
```

This will change to the folder "`WorkFolder`" that is in the current folder.

You can also go down multiple folders by separating them with a backslash (\).

```
cd WorkFolder\Data
```

This will take you to the "`Data`" folder inside the "`WorkFolder`" folder.

The "`cd`" command also has a couple of special forms. One that you will want to know about is this:

```
cd ..
```

When you give the "`cd`" command two periods instead of a folder name, the "`cd`" command will go to the current folder's parent folder.

Another special form of the "`cd`" command is:

```
cd \
```

The single backslash (\) will make the current folder be the root of the current drive you are on.

You can also combine the above with the "cd" command. For instance:

```
cd \WorkFolder\Data
```

Which would take you to the "`Data`" folder inside the "`WorkFolder`" folder that is itself, in the root of the current drive.

If you wanted to back up one folder and enter the "OtherFolder" folder, you could use this:

```
cd ..\OtherFolder
```

Though the "`cd`" command can switch folders, it cannot switch to another drive.

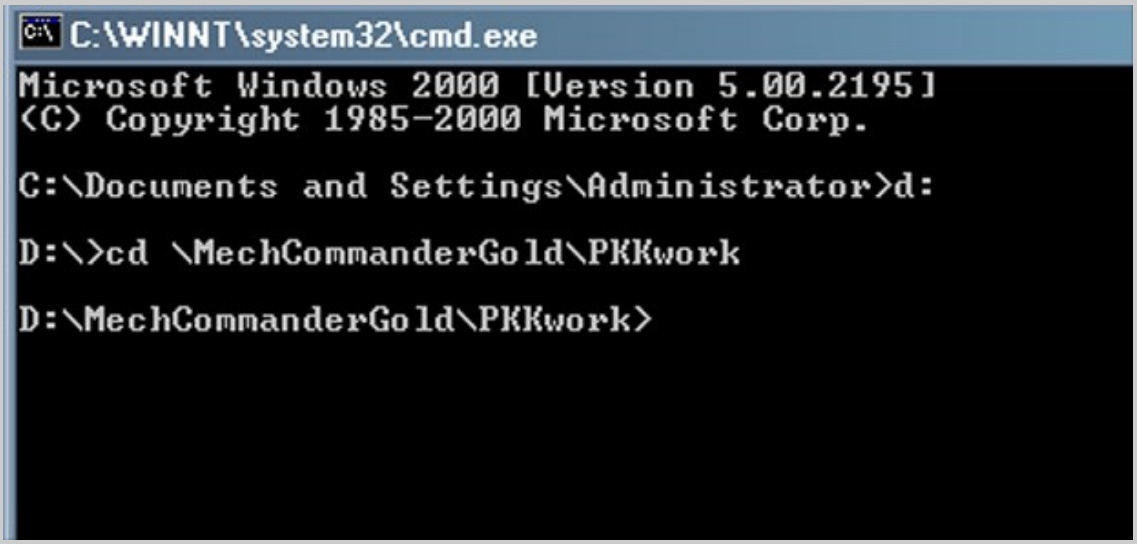
Using The Command Shell

To do this, you simply have to type the name of the drive followed by a colon. Example:

```
d:
```

This will change to D-Drive (assuming you actually have one).

Given the above, after I made my work folder, this is what I typed to get to it.



As can be seen, I first switch to my D-Drive, then switched to the "`PKKwork`" folder that is in the "`MechCommanderGold`" folder off the root of my D-Drive.

The prompt has changed showing that the current folder is now "`D:\MechCommanderGold\PKKwork`"

There is one more thing to know. The command shell assumes that parameters to commands (what you type after the command name) are separated by spaces. This can be problematic if the folder you are trying to change to has spaces in its name. The trick here is to enclose the entire parameter in double quotes like this:

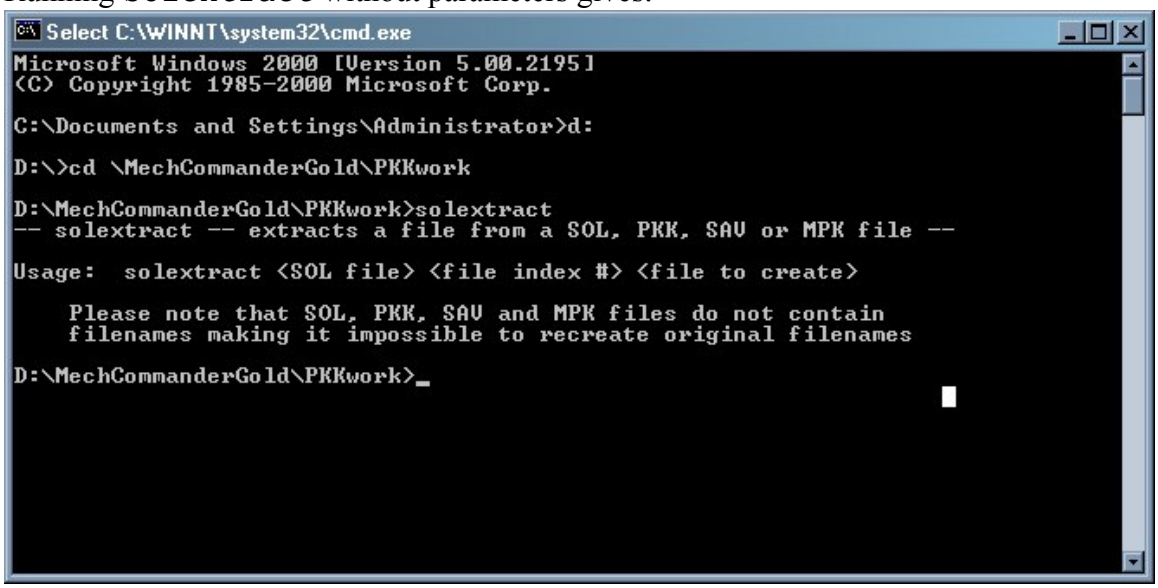
```
cd "Folder with spaces in name"
```

This will properly switch to the folder "`Folder with spaces in name`".

To extract a single file from a PKK (or .SOL or .SAV file for that matter) requires the use of the `solextract` tool. The syntax is quite simple and if unsure, simply type `solextract` without parameters and the tool will display a usage reminder.

PKK files do not contain the names of the original files that were used to make them. This means that filenames will need to be provided. This also means that an index to the file to extract needs to be provided. As the number of files in the PKK is not currently known, we need a way to find this out.

Running `solextract` without parameters gives:

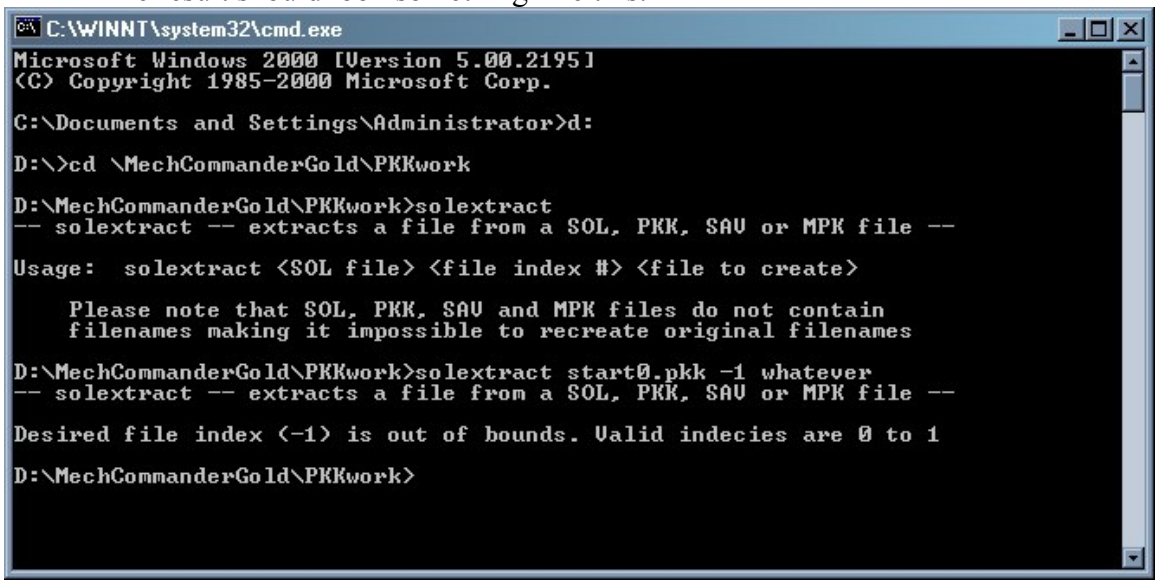


By asking `solextract` to extract a non-existing file index, it will give an error message that displays the valid file indices. So type the command

```
solextract start0.pkk -1 whatever
```

Note that the index of –1 is always invalid. The filename given at the end of the command doesn't actually matter right now as we are purposely creating an error to discover the valid indices in this PKK file.

The result should look something like this:



We can see that there are only two files embedded in this PKK. The first being at index 0, the other at index 1.

Now that the valid indices are revealed, just enter the command properly to extract a single file. For example, to extract the first file, use:

```
solextract start0.pkk 0 file0.txt
```

Do this again for the second file (index 1) and call it `file1.txt`.

Tip

If we want to extract all the files or a consecutive series of files, it would be tedious to do this one file at a time. But we can use the power of the command shell to help us here. Try this command:

```
for /L %n in (0, 1, 1) do solextract start0.pkk %n file%n.txt
```

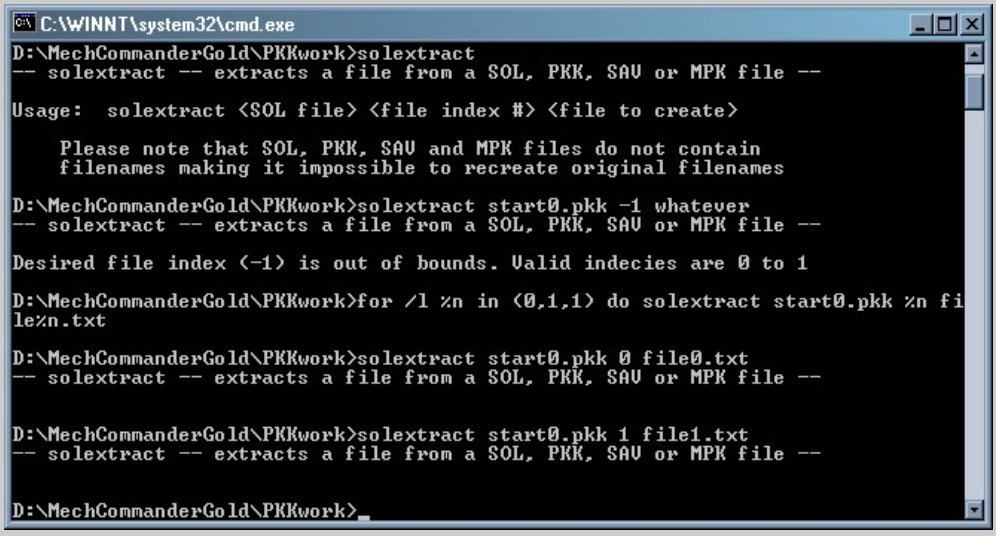
The above command will extract all files from the PKK and call each `file`*X*`.txt` where *X* is the index number in the PKK. The above command works like this:

The "`for /L`" part tells the command shell we want to count something. The "`%n`" part is going to represent the number that is being counted. The "`in (0, 1, 1)`" part tells the command how we want to count and works like this:

`in (`*start #*, *increment by*, *last #*`)`

In other words, if the indices were from 0 to 10, we could extract those by using "`in (0, 1, 10)`" etc. The "`do`" part says what to actually do each iteration count. The last part looks an awful lot like the previous command. In fact it is the same, as the actual iteration number will replace the `%n` before it runs the command.

This is what running that command looks like:



Tip

Note how each iteration runs the "do" part of the command as if we typed each in individually. Of course, with only two entries in this PKK file, this wasn't strictly required, but I wanted to show the possibility.

Lets examine the two files extracted.

The first file, called file0.txt is the master starting condition definition file. Lets rename it to masterdef.txt. This will be the first file to be changed when we get to modifying the starting equipment. This file will also need to be changed when giving players starting MechWarriors and assigning them to pre-allocated mechs. In other words, this is going to be a very important file so lets just skim through this file and get to know it a little first.

The first section, called [General] is defining a few things. First up is the master purchase file. It turns out this entry isn't actually used for the purchase list at all, but for a few other things. We won't be changing this entry in this tutorial.

The next entry seems rather odd. I am not really sure what it is used for, but we can specify a string here, though obviously it doesn't seem to really matter as the campaign is running without a hitch already.

The next entry, MissionNumber, seems to be indicating what mission this file is for. As this is the campaign's starting file, leave this at 0.

The same can be said for LastScenarioTime and LastLogisticsTime. All three of these entries are likely used for save games. Remember that a PKK file is really nothing more than a save game that is loaded prior to running the campaign.

The next section, [Planet], is defining the planet (i.e. Port Arthur or Cermak) that the mission is for, or so it would seem. Remember back while creating the maps, I mentioned about starting an expansion campaign if the first map was a Cermak map or original campaign if the starting map was a Port Arthur map. I also promised to say how to get around this. This section is what allows us to do this. The Setting entry in this section needs to be set to 0 if the user campaign is overriding an original campaign, and set to 1 if overriding the expansion campaign. Failing to do this can confuse the game as it will attempt to use files from the other campaign. As the campaign I'm building in this tutorial is overriding the original campaign, make sure the Setting entry is set to 0.

The next entry is rather interesting. The Solo entry is set to true. Since these files were copied from the solo mission's .SOL file, it makes sense that this entry indicates that this is a solo mission, but the campaign has run successfully without changing this entry. It turns out the game only looks for this entry when loading a solo mission. The

game completely ignores this entry when running a campaign. We can safely remove this entire line now, though obviously it doesn't hurt to leave it in.

In the next section we find… Aha! Resource Points! This entry is defining how much money to give out at the start of the campaign. Lets set this to half a million just for laughs.

The remaining entries I won't comment on just yet, but they are related to allocating Components, MechWarriors, Mechs, and assigning MechWarriors to Mechs.

At the very end of the file, right after `FITend`, you may see a couple of garbage characters. These can be deleted, though they will have no effect if left in.

Save the file.

Lets now take a look at `file1.txt`. First, rename it to `initstore.txt`, then open it in a text editor.

This file is the initial purchase file (hence why we renamed it). It lists all the Mechs, MechWarriors, Vehicles and Components that will be available in the store when the campaign is started. It is fairly self-explanatory as the names make some sense, the rest can be figured out. The only thing that needs explanation is that a value of –1 means unlimited, while any other number indicates that only that number of the specific item is available for purchase. The section of the file dealing with the MechWarriors can be a little confusing, but, as this is dealt with in the next step, you can mostly ignore it.

For now, all I'm going to change here is to make only two Commando-As available for purchase. Under the `Mech0` heading, change the `TypeAAvailable` to 2, and set the Type Js and Type Ws to 0.

Time to repackage the files. For this use the `makesol` tool. Using this tool will require a list of files to include in the final packed file. So first, using a text editor, create a new text file. The name isn't overly important, put don't go calling it the same as one of the other files.

As there are only two files to package up, we just need to make two lines in the file list. However, it is not enough to simply put the names of the files to add. The tool is also capable of compressing the files before packaging them up. Since this can be done on a per-file basis, the tool expects a special command character at the start of the line telling it what to do with the file, then the name of the file separated from the command with a space.

In short, the file list we need to create will look like this:

```
c masterdef.txt
c initstore.txt
```

Fairly simple no? The 'c' tells `makesol` that it should compress the file. If the file shouldn't be compressed, then put a 'u' as the first character instead. This should be used for very small files (32 bytes or less), as attempting to compress files that small can crash the tool (I did warn that I made these tools rather quickly and were somewhat fragile).

Save the file list. I called mine `filelist.txt` (yeah I know, real original).

Now to create the new `Start0.PKK` file, go back to the command shell window and enter this command:

```
makesol filelist.txt start0.pkk
```

When it's done, copy the newly created `Start0.PKK` from the work folder back into the `{MCX}\data\savegame` folder and overwrite the old `Start0.PKK` with this new one.

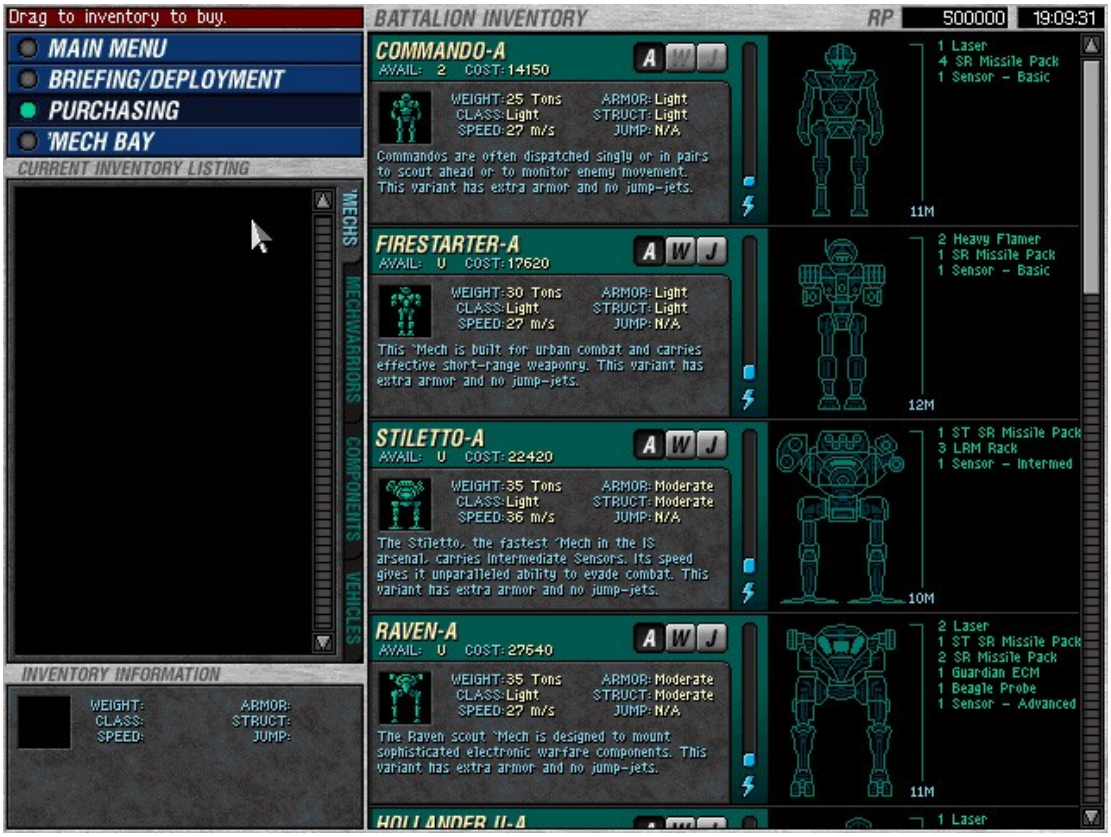
Now run the game again to see the changes.

Note how the amount of money the campaign gives is now 500000 RPs. Exactly what was asked for.

Also, going to the purchasing screen, we see that there are only two Commando-As and no Commando-Ws or Commando-Js. This is exactly what was specified.

The campaign is coming along, and this step has given important lessons in editing the needed files in the PKK.

This is how things look after the first edit of the PKK file



Step 5: Purchase Files

In step 4, we learned how to deal with PKK files. While doing so, we caught a glimpse of what is going to be involved in making the files needed for setting up the store.

Unfortunately, it turns out that this is one area that shows the game was rushed. There seems to be structures in place to make things relatively easy to make the purchase files, but they don't appear to be used. Instead, what is needed is to override yet more files because the filenames are hard coded in the game. It is a shame, but that is what must be dealt with.

The news isn't too grim though, as I'll be explaining how to package everything up in the last step, which will make the final campaign package as clean as possible.

But before we reach that stage, lets examine how the game presents the store to the player on a per-mission basis.

As was shown in step 4, the game starts the campaign by loading `Start0.PKK` (or `xStart0.PKK` in the case of the expansion). Which has in it the master definition file (that was the `masterdef.txt` file that was extracted from the PKK file in step 4) that references another file called `solopurchase.fit`. This file appears to have entries for individual purchase files for each mission map (we'll see later that this is not quite true). The PKK also contained another file (`initstore.txt` that was extracted in step 4) that was the initial state of the purchase store.

The initial state of the store does indeed come from the PKK as was proven in step 4 earlier. However, the game doesn't use any of the purchase file information from the master purchase file (`solopurchase.fit`), but does use other information in there, so the file is still needed. Instead, the game looks for a file called `purchase`*XX*`.fit` where *XX* is the mission number starting from 0 (if using the expansion campaign, the purchase files are `xpurchase`*XX*`.fit`). So for this example campaign, the actual purchase files the game will attempt to load will be `purchase00.fit`, `purchase01.fit` and `purchase02.fit`. It should also be noted that though the game will indeed look for `purchase00.fit`, it has no effect since the initial store, as mentioned, is stored in that `initstore.txt` from the PKK.

So to setup the store, that purchase file of the PKK needs to be setup properly, and then for each mission we also need to create a `purchase`*XX*`.fit` file. I know this sounds a little oddball, but this is how it must be done.

There is one more thing to know before beginning this journey. The purchase files for each mission are *added* to the current store (with the exception of the first mission, as this is the initial state of the store and so already contains the "addition" so to speak). This means that if the store has 2 Commando-Ws in the previous mission, and the

purchase file for the next mission indicates 2 Commando-Ws, then the net result is that the store will have 4 Commando-Ws. This also means there is no way to remove mechs from a store once they are added in (except by the player buying them up of course).

Well time to get our hands dirty.

First, go back to that working folder created in step 4. Since the file needed to be changed is already extracted from the PKK, that's the one to edit.

Open `initstore.txt` in a text editor. This is the initial store. I've decided to only make a few things available in the store. But as this is a rather long file, I'll simply mention what I'm going to change, and you can look at the final version of the file in the appendices.

First, set the `TypeAAvailable`, `TypeJAvailable` and `TypeWAvailable` entries for all the mechs to 0. This will remove all mechs. Then go back and find the entry for the Bushwacker and make 2 Type-As and 2 Type-Ws available (In the next step, we will see how to properly identify a mech from its profile number and vice-versa, but for now, simply use the comments in the file to find the appropriate mechs).

I don't actually want any vehicles in this campaign, so for all the vehicle headings, set the `NumAvailable` to 0 as well.

For components, set the `NumAvailable` to 0 for all except for the following: 8 Pulse Lasers, 2 Heavy Flamers, 4 Large Pulse Lasers, and 4 PPCs (not the ER-PPCs).

Now for the list of available MechWarriors. These entries act a little differently than the previous ones. Note that there are only two entries under each Warrior heading. Namely `Profile` and `Status`. The profile entry is the filename to a MechWarrior profile file. To find out who an entry actually represents, we use the filename in the `Profile` entry and use the following table.

The profile entry is really the filename of a warrior profile file (more on this later). But the filename is constructed like this:

`PMW`*`nnnnn`*`.FIT`

Where *`nnnnn`* is a 5 digit ID representing the warrior in question. This ID number will equal one of the numbers in the following table. Just read off the name to find out who it is (you can ignore the `.FIT` part for now).

The names marked with "(Ex)" at the end are the updated MechWarrior profiles used in the expansion.

51	Beast	76	Blaze
52	Hawk	77	Thunder
53	Siren	78	Paingod
54	Rooster	79	Firestorm
55	Isis	80	Vixen
56	Gunman	86	Siren (Ex)
57	Mystique	87	Rooster (Ex)
58	Skater	88	Isis (Ex)
59	Spice	89	Gunman (Ex)
60	Baron	90	Skater (Ex)
61	Fiend	91	Countess (Ex)
62	Ronin	92	Scarab (Ex)
63	Gator	93	Mantis (Ex)
64	Rebel	94	Dragon (Ex)
65	Hunter	95	Blaze (Ex)
66	Lynx	96	Thunder (Ex)
67	Hitman	97	Goblin (Ex)
68	Countess	102	Spice (Ex)
69	Scarab	103	Baron (Ex)
70	Outlaw	104	Fiend (Ex)
71	Falcon	105	Ronin (Ex)
72	Burnout	106	Gator (Ex)
73	Goblin	107	Rebel (Ex)
74	Mantis	108	Hitman (Ex)
75	Dragon	109	Burnout (Ex)

The `Status` entry tells the game how to treat this warrior. Valid values are in the following table.

0	Available for hire
1	Hired
2	Dead
3	Dismissed (sold back)
4	Unseen

Now obviously, we don't want to be killing off or dismissing warriors right from the start, so the only values that are of use to us are 0 and 4. The value 1 may come in handy later, but ignore it for now, as this will introduce unneeded complications.

If a given MechWarrior should be available for hire, set their status to 0. If they are to be hidden from the hiring list, but need to be available later, set the status to 4.

For this campaign, I decided to have Beast (51), Gator (63), Hunter (65) and Hitman (67) available for hire at the beginning, all the others will be set to Unseen status.

Save the file.

Now for the per-mission purchase files. As mentioned, each mission map has its own purchase file, but they are additions to the base store allocations. So the easiest thing to do is to simply take the file just edited, save it as `purchase00.fit` to create the first mission's purchase file (remember that the file must exist, but the real data is in the `initstore.txt` just edited. Then make the changes and additions we want for the next mission, save it as `purchase01.fit`, etc. It is a good idea to reset all the components, mechs and vehicle counts to 0 before doing the changes.

For this campaign, I'm going to add two ER-PPCs and an Atlas-W for mission 2, but no additional hardware at all for mission 3. I will add Blaze (76) as available for hire for mission 3 however.

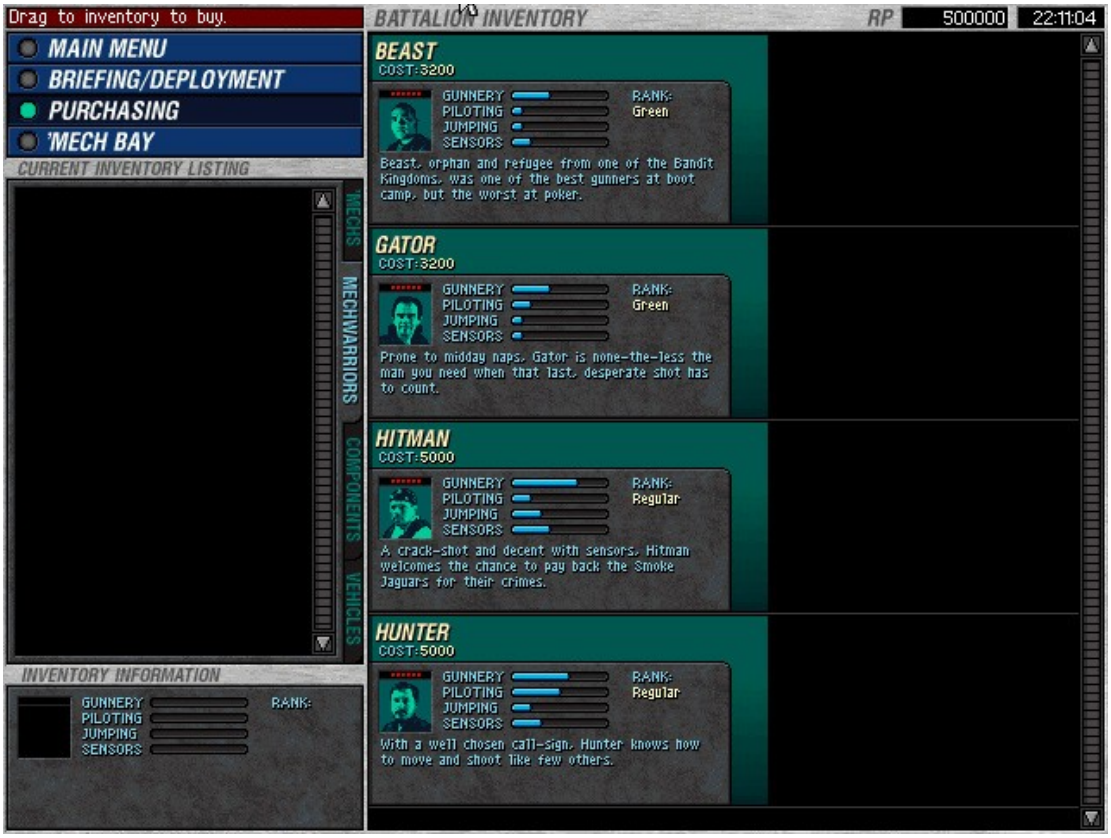
See if you can work out the details yourself. Compare your results with the listings in the appendices.

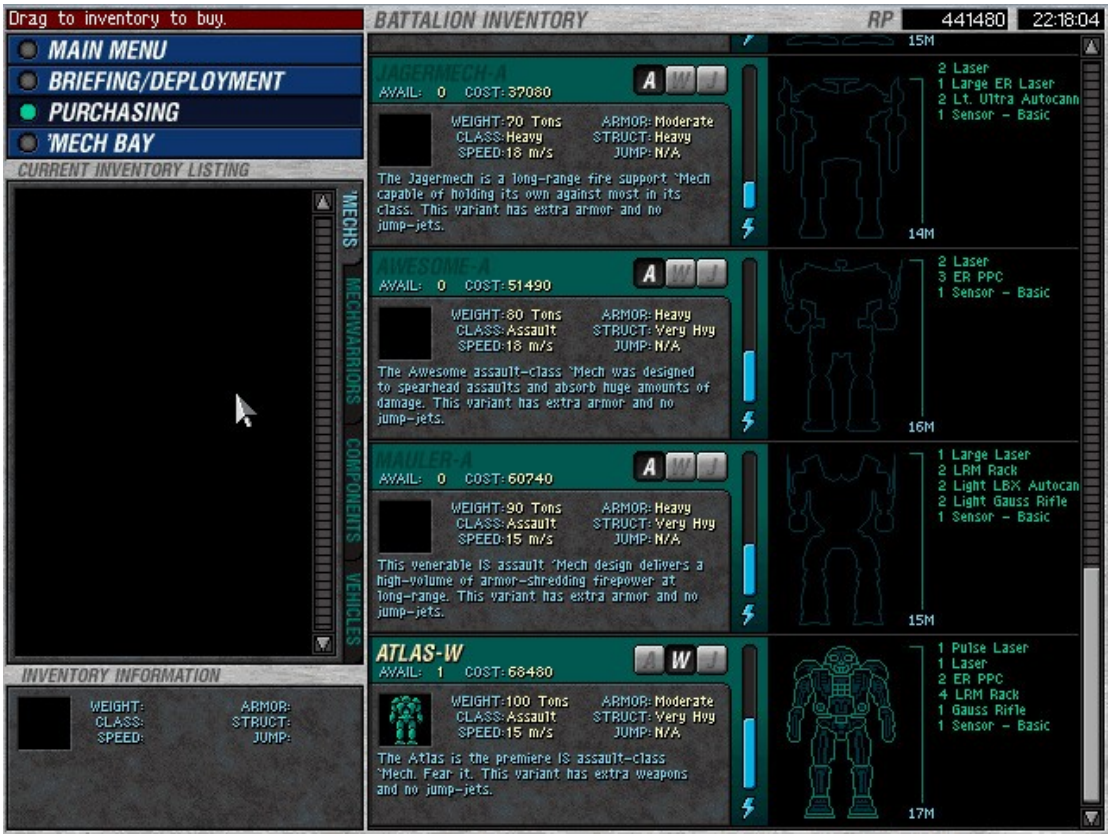
When all three purchase files (`purchase00.fit`, `purchase01.fit` and `purchase02.fit`) are done, copy them into the `{MCX}\data\missions` folder. Also remake the `Start0.PKK` file and copy it over to the `{MCX}\data\savegame` folder.

OK, lets try out this new version.

Coming in, it looks like the store is all set as was wanted. The Warriors selected are available, as are the mechs and equipment.

After playing the first mission, an Atlas-W is now available, as are the ER-PPCs. The rest is as it was before. Exactly what was wanted.


Drag to inventory to buy.
BATTALION INVENTORY
RP 500000 22:11:04
MAIN MENU
BRIEFING/DEPLOYMENT
PURCHASING
'MECH BAY
CURRENT INVENTORY LISTING
MECHS
MECHWARRIORS
COMPONENTS
VEHICLES
INVENTORY INFORMATION
GUNNERY
PILOTING
JUMPING
SENSORS
RANK:
BEAST
COST:5200
GUNNERY
PILOTING
JUMPING
SENSORS
RANK:
Green
Beast, orphan and refugee from one of the Bandit Kingdoms, was one of the best gunners at boot camp, but the worst at poker.
GATOR
COST:5200
GUNNERY
PILOTING
JUMPING
SENSORS
RANK:
Green
Prone to midday naps, Gator is none-the-less the man you need when that last, desperate shot has to count.
HITMAN
COST:5000
GUNNERY
PILOTING
JUMPING
SENSORS
RANK:
Regular
A crack-shot and decent with sensors, Hitman welcomes the chance to pay back the Smoke Jaguars for their crimes.
HUNTER
COST:5000
GUNNERY
PILOTING
JUMPING
SENSORS
RANK:
Regular
With a well chosen call-sign, Hunter knows how to move and shoot like few others.


Drag to inventory to buy.
BATTALION INVENTORY
RP 441480 22:18:04
MAIN MENU
BRIEFING/DEPLOYMENT
PURCHASING
'MECH BAY
CURRENT INVENTORY LISTING
MECHS
MECHWARRIORS
COMPONENTS
VEHICLES
INVENTORY INFORMATION
WEIGHT:
CLASS:
SPEED:
ARMOR:
STRUCT:
JUMP:
15M
JAGERMECH-A
AVAIL: 0 COST: 37080
A W J
WEIGHT: 70 Tons
CLASS: Heavy
SPEED: 18 m/s
ARMOR: Moderate
STRUCT: Heavy
JUMP: N/A
The Jagermech is a long-range fire support 'Mech capable of holding its own against most in its class. This variant has extra armor and no jump-jets.
2 Laser
1 Large ER Laser
2 Lt. Ultra Autocann
1 Sensor - Basic
14M
AWESOME-A
AVAIL: 0 COST: 51490
A W J
WEIGHT: 80 Tons
CLASS: Assault
SPEED: 18 m/s
ARMOR: Heavy
STRUCT: Very Hvy
JUMP: N/A
The Awesome assault-class 'Mech was designed to spearhead assaults and absorb huge amounts of damage. This variant has extra armor and no jump-jets.
2 Laser
3 ER PPC
1 Sensor - Basic
16M
MAULER-A
AVAIL: 0 COST: 60740
A W J
WEIGHT: 90 Tons
CLASS: Assault
SPEED: 15 m/s
ARMOR: Heavy
STRUCT: Very Hvy
JUMP: N/A
This venerable IS assault 'Mech design delivers a high-volume of armor-shredding firepower at long-range. This variant has extra armor and no jump-jets.
1 Large Laser
2 LRM Rack
2 Light LBX Autocan
2 Light Gauss Rifle
1 Sensor - Basic
15M
ATLAS-W
AVAIL: 1 COST: 68480
A W J
WEIGHT: 100 Tons
CLASS: Assault
SPEED: 15 m/s
ARMOR: Moderate
STRUCT: Very Hvy
JUMP: N/A
The Atlas is the premiere IS assault-class 'Mech. Fear it. This variant has extra weapons and no jump-jets.
1 Pulse Laser
1 Laser
2 ER PPC
4 LRM Rack
1 Gauss Rifle
1 Sensor - Basic
17M

And after mission 2, Blaze is now available for hire (I had hired Hitman and Hunter, for testing the campaign, which is why they aren't in the "for hire" list).

It was a fair bit of monkey work, but I think the effort was worth it.

Now that covers a lot, but what about adding components other that the ones currently in that initial store.

We can do this simply by adding or changing the component IDs in the component lists of the purchase files. To find out what IDs can be used, look through this table:

ID	Component Name
13	Sensor (Basic) (IS)
14	Sensor (Intermediate) (IS)
15	Sensor (Advanced) (IS)
16	Sensor - Basic (Clan)
17	Sensor - Advanced (Clan)
37	Beagle Probe (IS)

ID	Component Name
38	Guardian ECM (IS)
42	ECM Suite (Clan)
43	Active Probe (Clan)
98	Rail Gun
99	Light Gauss (IS)
100	Light Autocannon (IS)
101	Autocannon (IS)
102	Heavy Autocannon (IS)
103	Light Ultra Autocannon (IS)
104	Gauss Rifle (IS)
107	Light LBX AC (IS)
108	LBX AC (IS)
109	Hvy LBX AC (IS)
110	Ultra AC/5 (Clan)
111	Ultra AC/10 (Clan)
112	Ultra AC/20 (Clan)
113	Gauss Rifle (Clan)
116	Clan Lt LBX (Clan)
117	Clan LBX (Clan)
118	Clan Hvy LBX (Clan)
120	Long-Range Missile Rack (IS)
123	SRM Pack (IS)
125	Streak SRM Pack (IS)
130	LRM/5 (Clan)
133	SRM/2 (Clan)
135	Streak SRM/2 (Clan)
139	X-Pulse Laser (IS)
140	Large Laser (IS)
141	Large ER Laser (IS)
142	Large Pulse Laser (IS)
143	Laser (IS)
144	Pulse Laser (IS)

ID	Component Name
145	Particle Projector Cannon (IS)
146	ER Particle Projector Cannon (IS)
147	Heavy Flamer (IS)
150	Large ER laser (Clan)
151	Large PS laser (Clan)
152	ER laser (Clan)
153	PS laser (Clan)
154	ER PPC (Clan)
155	Flamer (Clan)
160	Long Tom Cannon(IS)

For warriors, just replace the profile entries with the ones you want. We'll learn more about warrior profiles in the next step.

As for mechs, note how each mech has three profiles. We can do the same for the mechs as we can do with warriors, change the profile names in the entries. Though I haven't tested this, it would seem possible to assign different mechs to each of the A,W and J categories since they each take a full profile name (though it would likely be very confusing to do so). We will learn more about mech profile names in the next section as well.

Well, no rest for the weary. Even though the effort so far has given us a nicely working campaign, there is still more yet to do and learn.

Step 6: Starting Equipment, Mechs, and Warriors

In the previous section, we setup the store both at the start of the campaign, and for every mission thereafter. But that's the store. What about starting the player off with some base equipment in the Mech Bay.

It turns out that this was probably the hardest thing to figure out. So hold onto your hats folks because, to paraphrase an episode of American Dad, things are going to get a little weird and a whole lot crazy.

Back in step 4, we started playing around with the PKK file. I mentioned that the first file (`masterdef.txt`) was used for starting equipment and such. This is where we start playing with those sections.

So once again, go back to the work folder for the PKK file and open `masterdef.txt` in a text editor.

To start, I'm going to give 4 ER Lasers to the player's starting inventory. Scroll down to the bottom of the file and find the `[Components]` heading. As I'm only adding one thing to the inventory, change the number of components to 1.

Now we need to tell the game which component we are adding. Just like in the purchase file we need to add the entry immediately after the Components heading. To make things a little easier, copy the appropriate entry from one of the purchase files we created in the previous step. I added the following:

```
[Componant0]
// ER laser
uc ComponantID = 152
l NumAvailable = 4
```

Note the spelling of the heading here.

OK, that should do the trick for the ER Lasers. Now to add an Atlas-W. Find the `[Mechs]` heading in the file and change the number of mechs to 1.

Now we need to add an entry called `[Mech0]` to tell the game what mech to add (if we wanted to add 2 mechs, we would also need a `[Mech1]` entry of course).

The entries under this `[Mech0]` heading are `Profile` and `NumAvailable`. The profile entry tells the game what type of mech to add, and the number available says how many. So yes, we could give multiples of the same mech to the player with a single entry. But we only want one. Now all that is needed is to know what filename the profile of an Atlas-W is.

Every mech that can be used in the game has a profile file associated with it. The filename is constructed in a special way, like so:

`PM`*`nnntvv`*`.FIT`

The *`nnn`* part is a three-digit code identifying the mech chassis, the *`t`* part identifies which type of mech it is (A, W or J) and the *`vv`* part says which special variant to use.

So to construct the mech profile filename to use, we look up what is wanted in the following tables (note that for the special variant, normally this will be 0, but a few variants are defined by the game, likely used for special mechs in the official campaigns).

The Mech ID Code Table:

	IS Mechs		**Clan Mechs**
100	Commando	200	Uller
101	Firestarter	201	Cougar
102	Raven	203	Hunchback IIc
103	Hollander II	204	Vulture
104	Hunchback	205	Loki
105	Centurion	206	Thor
106	Catapult	207	Mad Cat
107	JagerMech	208	Masakari
108	Awesome	209	Shadow Cat
109	Atlas	210	Nova Cat
110	Fire Falcon	211	Turkina
111	Bushwacker		
112	Mauler		

The Mech Types Table:

	Mech Type
1	Armor (A)
2	**invalid**
3	Jump Jet (J)
4	Weapons (W)

The Defined Mech Variants Table:

Mech & Type	Variants
Awesome-A	1
Atlas-A	50
Uller-A	50, 60, 70
Cougar-J	50
Vulture-A	1
Vulture-J	50
Thor-W	1
Mad Cat-J	50, 60
Masakari-A	50
Masakari-J	50
Shadow Cat-A	50
Shadow Cat-W	50
Turkina-J	50
Turkina-W	50

Since I wish to add an Atlas-W, I first look what three digit code is given for an Atlas and see that it is code 109. From the mech type, I see the type code is 4, and I'll use the default variant (usually wise, though in this case there are no other variants), so the variant will be 0.

Given this, the profile's complete filename becomes:

```
PM109400.FIT
```

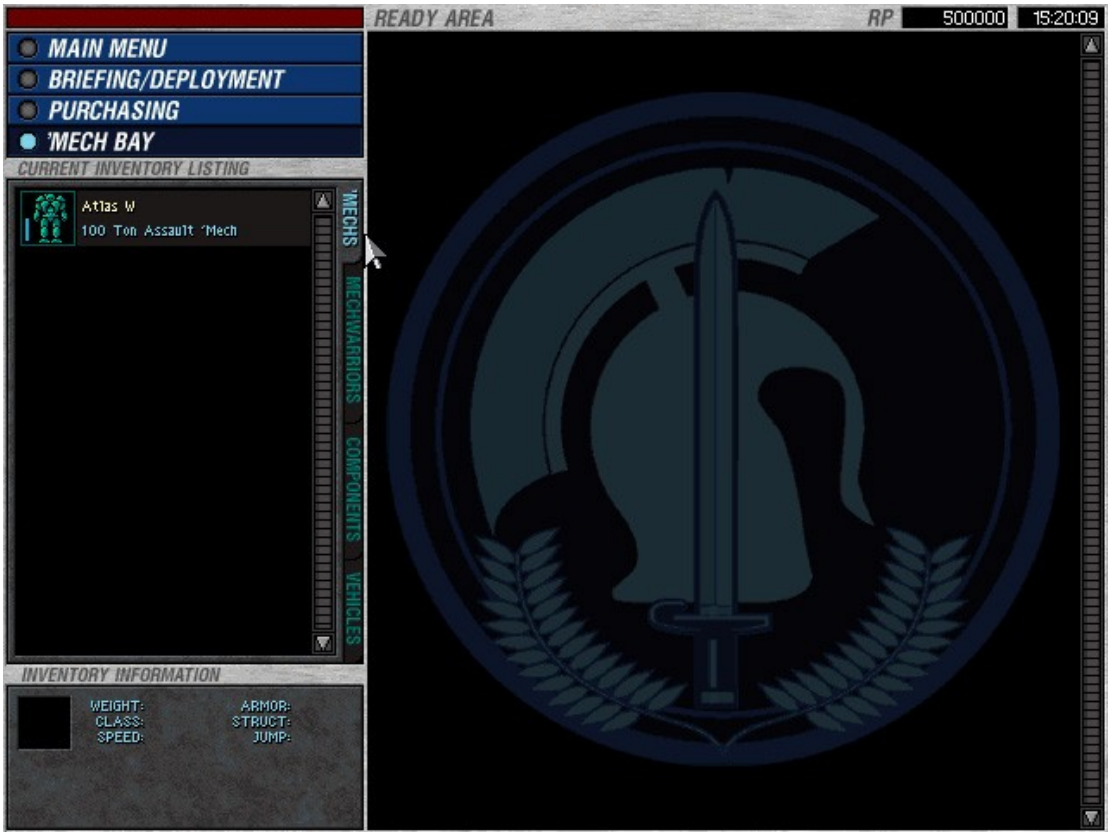
For the profile entry in `masterdef.txt`, we leave off the .FIT part of the filename (it is assumed to be that by the game) to finally make the entry this:

```
st Profile = "PM109400"
```

And as I only want one of these, I set the number available to 1, which makes the completed `[Mech0]` entry look like this:

```
[Mech0]
st Profile = "PM109400"
l NumAvailable = 1
```

Now that the mech and ER lasers are added, save the file and rebuild the `Start0.PKK`. Copy it to the savegame folder and try it out.



As can be seen, the Atlas-W is now in the Mech Bay right from the start, The ER Lasers are present as well.

Well that didn't seem so bad, now did it? But this is where it starts to get complicated.

Lets first start by adding a warrior so the player doesn't actually have to go hire anyone. Lynx would be a nice addition, lets add him.

Back we go to edit `masterdef.txt` again. Once again, open it in a text editor. This time go to the `[Warriors]` section and change the number of warriors to 1.

Immediately after the `[Warriors]` section, we need to start adding the warriors themselves. As you may guess, we need to add a `[Warrior0]` section. But there is only one entry in this section, which will be the "packet number" for this warrior. The entire section will look like this:

```
[Warrior0]
ul PacketNum = 0
```

The packet number assigned will increase for each warrior. But what exactly is a packet?

Packets are what the game calls the files that are in the PKK file (it is likely that this is where the "P" comes from in PKK). So we are telling the game that this warrior will be the next packet file in the PKK. But notice that packet numbers start counting *after* the first file in the PKK, which is the file we are currently editing.

This means that we will have to re-order the files in our PKK. But first, lets find out what it is that we'll be adding to the PKK.

What the game expects as a packet file for a warrior definition is a warrior profile. We can make one of these by editing one of the default profiles stored with the game. The trick is getting to it.

Most of the files the game needs are packaged up in the `mission.fst` file in the game's root folder (where the game executable is). We can extract files from the FST with my tool `fstextract`. So first, copy `fstextract.exe` into the game's root folder. Now all we need is to figure out is what file to extract from `mission.fst`.

We've already seen in the previous step how to identify warriors from their IDs when we were setting up the purchase file. We now do the reverse process. Since I want Lynx to be hired at the start, I look through the warrior ID table from step 5 and see that Lynx has code 66. And as explained in the previous step, this means that the warrior profile is `PMW00066`. Add a `.FIT` extension to that and we have the filename we need.

The tool `fstextract` expects the first parameter to be the name of the FST file to extract from, then the name of the file to be extracted, then finally, the name of the file we want to create (it doesn't have to have the same name).

I issued this command:

```
fstextract mission.fst pmw00066.fit Lynx.txt
```

This command creates the file `Lynx.txt` that will be a copy of the contents of `PMW00066.FIT` from the `mission.fst` file.

Now copy or move `Lynx.txt` into the PKKwork folder. And open it in a text editor. This profile is close to what we need for the PKK, but not quite. We still need to change a few things.

First, under the `[General]` section, add the following two lines:

```
b Assigned=FALSE
b NotMineYet=FALSE
```

I'm not quite sure what "NotMineYet" really does, but the "Assigned" entry is used when assigning a mech to a warrior at startup. As Lynx is only meant to be hired but not assigned, leave this set to false.

Now find the [Skills] section, copy the entire thing, and paste it in twice more (giving three skills sections in all). The first copy should be renamed to [OriginalSkills] and the other should be renamed to [LatestSkills]. This give us three sections like so:

```
[Skills]
c Piloting =                    50
c Jumping =                     45
c Sensors =                     50
c Gunnery =                     55

[OriginalSkills]
c Piloting =                    50
c Jumping =                     45
c Sensors =                     50
c Gunnery =                     55

[LatestSkills]
c Piloting =                    50
c Jumping =                     45
c Sensors =                     50
c Gunnery =                     55
```

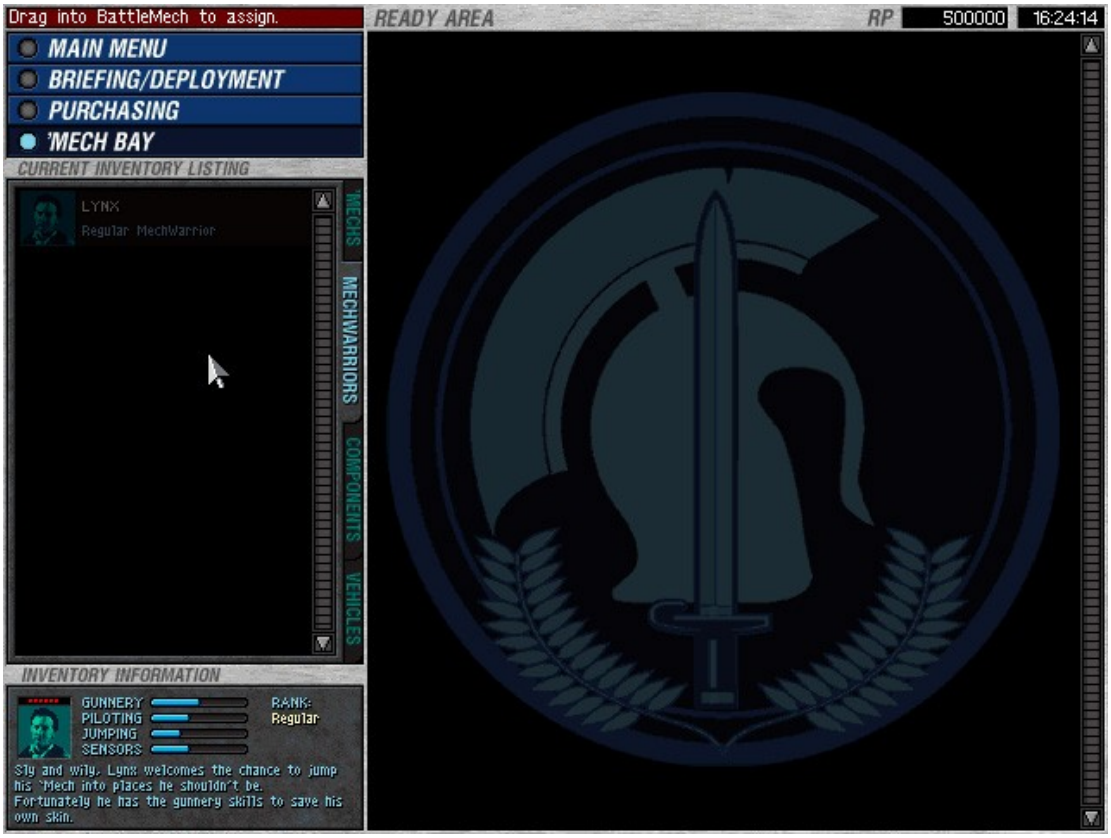
Finally, remove the entire [Affinities] section. It is not needed.

Save the file.

The last step is to get the Lynx.txt file into the PKK. So open the filelist.txt file and add Lynx.txt between the lines for masterdef.txt and initstore.txt (remember, I said that this was the definition for packet 0 which was to come immediately after masterdef.txt). The new filelist.txt should look like this:

```
c masterdef.txt
c Lynx.txt
c initstore.txt
```

Save this file, and now rebuild Start0.PKK and copy it to the savegame folder. Start the game and check the results.



It is a little hard to see as I simply went straight to the MechWarrior tab in the Mech Bay, but yes, Lynx is now there and usable.

So that handles handing out warriors into the player's inventory, but what about having a warrior starting already assigned to a mech like they did in the official campaigns.

As you may expect, it is handled somewhat similarly to how warriors are added, but there are a couple of added twists that took me a little while to figure out.

To truly demonstrate how this works, and hence show what those "gotchas" are, I'm going to assign two warriors to two different mechs. Specifically, I'll put Mystique in a Cougar-A and Burnout in a Catapult-W. Both will be available at the start, ready to deploy.

Once more, we must first go edit that `masterdef.txt` file we are getting to know so well.

The first thing to change will be the number of assigned warriors. Find the `[AssWarriors]` section and change the "`NumAssWarriors`" to 2.

Now we add warrior entries *as if we were continuing the list from the warriors section*. Also, we need to *continue incrementing* the packet number values. This is very important. Since we are adding two more warriors, the entries will look like this:

```
[Warrior1]
ul PacketNum = 1

[Warrior2]
ul PacketNum = 2
```

Now, as we did with Lynx, we need to go create warrior profiles for Mystique and Burnout for inclusion in the PKK. This is done exactly as was done for Lynx. Including the editing of the files with one exception. As these warriors will be assigned mechs right from the start, we need to set their "`Assigned`" flag to true instead of false.

I won't list all the changes here, review the listings in the appendices to see the completed changes. As I did with Lynx, I called the profiles `Mystique.txt` and `Burnout.txt`. So these files need to be added to `filelist.txt` immediately after the `Lynx.txt` line like so:

```
c masterdef.txt
c Lynx.txt
c Mystique.txt
c Burnout.txt
c initstore.txt
```

But we aren't done yet. We need to tell the game what mechs to assign them to. First, as we did for the assigned warriors, we need to list the assigned mechs in `masterdef.txt`. So open that file again, look for the `[AssMechs]` section and change the assigned mechs to 2.

Now, as was the case with the assigned warriors, we now list the mechs as if we were listing them following the un-assigned mech list. Further, we also need to continue increasing the packet number count since these refer to files following the `masterdef.txt` in the PKK. So our new mech entries look like this (I added the comments):

```
[Mech1]
ul PacketNum=3  // Cougar-A

[Mech2]
ul PacketNum=4  // Catapult-W
```

Now we need to go get the mech profiles. I've already mentioned how to create the name, so extracting them from the `mission.fst` should not be difficult. For the Cougar-A, the mech profile to extract will be `PM201100.FIT` (I saved it as `Cougar-A.txt`) and the Catapult-W's profile will be `PM106400.FIT` (I saved it as `Catapult-W.txt`).

As with the Warrior profiles, the mech profiles also need a little editing for use. For each of them, load them in the text editor.

In the `[General]` section, delete the two lines called "`Morale`" and "`MoraleModifier`", as they are not needed. Still in the `[General]` section, add the following four lines:

```
b Assigned = TRUE
b NotMineYet = FALSE
l Pilot = 0
b Required = FALSE
```

Now look for the section called `[HeatSinks]` and completely remove it. It is not needed.

Now there is one more thing to do here. The Pilot entry we just added needs to point to the correct warrior pilot. The number to use here is defined by the order we listed the assigned warriors in `masterdef.txt`. However, the list is ***backwards***. This means that the last warrior listed is pilot 0, the second to last is pilot 1 etc. As I've listed Mystique first, then Burnout. Burnout will be pilot 0, and Mystique will be pilot 1. So in the profile for the Cougar-A, set the `Pilot` entry to 1, and in the profile of the Catapult-W, set the `Pilot` to 0.

Now we go back to the file list and add the mechs to `filelist.txt` so that they are in the PKK as well. The file list now looks like this:

```
c masterdef.txt
c Lynx.txt
c Mystique.txt
c Burnout.txt
c Cougar-A.txt
c Catapult-W.txt
c initstore.txt
```

Save everything, rebuild `Start0.PKK` and copy it to the savegame folder, and try this thing out.

Burnout and Mystique are ready to deploy right from the start, and they are in the correct mechs too.

Phew. Only one thing remains. Adding vehicles to the starting inventory.

Just like when we added mechs, to add an un-assigned vehicle we change the number under the `[Vehicles]` heading, then start adding entries after. Lets add a Bulldog to the player's inventory.

Yup, go back and load the `masterdef.txt` file again. Find the `[Vehicles]` heading and set the number of vehicles to 1, then add a vehicle entry like so:

```
[Vehicle0]
st Profile = "PV20800"
l NumAvailable = 1
```

By now you are probably wondering how I got the profile name. Vehicle profile names, like the mech profiles, are constructed in a specific way. The pattern for vehicles is:

PV*nnnvv*.FIT

Where *nnn* is the vehicle's ID number from the table below. And like the mechs, the *vv* part is the variant. Normally this will be 0, but the third column shows the variant defined for a given ID (if any) and the name of the variant as it isn't always the same as the default one.

So the Bulldog is ID 208, and there are no variants, so it is 0. This gives a filename of `PV20800.FIT`. Strip off the ".FIT" and we have the name of the profile entry above.

Table of Vehicle IDs and variants

	Vehicle Name	**Var.**	**Variant Name**
200	Switfwind		
201	Rommel		
202	MobileHQ		
203	Fuel Truck		
204	Elem. Carrier		
205	Harasser		
206	APC	50	APC
208	Bulldog		
209	SRM Carrier		
210	LRM Carrier		
211	Mobile Artillery	50	Mobile Artillery
212	Artillery Loader		
213	Shrek		
214	Saracen		
215	Towed Turret		
216	AeroSpace Spotter		
217	Striker		
218	Pegasus		
219	J. Edgar		
220	VonLuckner		
221	Condor		
222	Ordnance Truck		
224	Salvage Rig		
227	Light Transport	01	Armored Car
228	Mine Sweeper		
229	Mine Layer		
304	Semi Tractor		
305	Flatbed	01	Armored Flatbed
306	Farm Truck	01	Armed Farm Truck
307	Car		
308	Sports Car		
309	Sports Car		
310	Car		
311	Hover Limo		

	Vehicle Name	Var.	Variant Name
312	Limo		
313	Sports Car		
314	Car		
315	Car		
316	Sports Car		
317	Car		
318	Car		
319	Savannah Master		
320	Sports Car		
321	Salvage Truck		
322	Refit Truck		
323	Refit Truck		
324	Ambulance		
333	Centipede		
334	Regulator		
335	Manticore		
336	Pilum		
337	Alacorn		
338	Ammo Truck		

(Note that I haven't tried all these vehicles, these are just the vehicles I found in the `mission.fst` file)

That should be it for getting a vehicle into the inventory, so lets try adding one already assigned. Once more, open `masterdef.txt` and find the `[AssVehicles]` section and change the number of assigned vehicles to 1. Now add another vehicle entry, again as if we were continuing the vehicle section like this:

```
[Vehicle1]
ul PacketNum = 5
```

Note that we are still continuing the packet numbers.

Now let's add a Swiftwind as the assigned vehicle. As we did with assigned warriors and mechs, we need the vehicle profile. So as I've just explained above, we figure out the file to extract from `mission.fst` and discover that it is `PV20000.FIT`. Extract it and open it up in the text editor (I called mine the obvious `Swiftwind.txt`).

Again, as with the mech and warrior profiles, we must edit this file for our use. It turns out these profiles require a little more editing than the others.

First, we must find and remove the following entries (all in the `[General]` section): `ID`, `VehicleTypeID`, `ResourcePoints`, `BattleRating`, `Morale`, and `MoraleModifier`.

Now add the following four entries:

```
b Assigned=TRUE
b NotMineYet=FALSE
b Deployed=FALSE
b Required=FALSE
```

Note the new entry "`Deployed`". I think this is only used for save games, but I'm not sure.

Well, save the file and open the file list again. Add the `Swiftwind.txt` right before the `initstore.txt` entry so that the file list is like this:

```
c masterdef.txt
c Lynx.txt
c Mystique.txt
c Burnout.txt
c Cougar-A.txt
c Catapult-W.txt
c Swiftwind.txt
c initstore.txt
```

Save and rebuild the PKK (getting used to that yet?), copy it to the `{MCX}\data\savegame` folder and give this puppy a whirl.

When you enter, there is now a Swiftwind just waiting to be deployed. And in the mech bay under vehicles, we find the Bulldog.


MAIN MENU
BRIEFING/DEPLOYMENT
PURCHASING
'MECH BAY
CAMPAIGN DATA
OPERATION BRIEFING
MISSION BRIEFING
*****CLASSIFIED*****
Tutorial Map 1
1. Capture the enemy headquarters
FIRE SUPPORT:
ARTILLERY: 0 Small 0 Large
SENSOR PROBE: 0
CAMERA DRONE: 0
Simple example map for the MechCommander Gold Campaign Creation tutorial.
TACTICAL WARNING
*no warnings
TACTICAL NOTICE
*no notices
<<<<END DISPATCH
READY AREA
MYSTIQUE
BURNOUT
Swiftwind
RP
500000
19:08:16
FORCE GROUP 1
0 T.
FORCE GROUP 2
FORCE GROUP 3
DROP WEIGHT LIMIT
150 TONS
CURRENT DROP WEIGHT
0 TONS
BEGIN MISSION
COUGAR-A
35 Ton Light 'Mech
CALLSIGN:
MYSTIQUE
RANK:
Green
GUNNERY
PILOTING
JUMPING
SENSORS
REFIT ALL
REPAIR ALL
ARMOR
INTERNAL
ENGINES
11M
PAYLOAD
Short
C Heavy Flamer
Medium
C Large PS Laser
Long
C LR Missile Rack
C LR Missile Rack
C LR Missile Rack


READY AREA
RP
500000
19:06:30
MAIN MENU
BRIEFING/DEPLOYMENT
PURCHASING
'MECH BAY
CURRENT INVENTORY LISTING
MECHS
MECHWARRIORS
COMPONENTS
VEHICLES
Bulldog
60 Ton Heavy Vehicle
INVENTORY INFORMATION
WEIGHT:65 Tons
CLASS:Heavy
SPEED:18 m/s
ARMOR:Moderate
STRUCT:Heavy
JUMP:N/A
The Catapult is a second-line, fire support 'Mech with a solid layer of armor. This variant has extra weapons and no jump-jets.
COUGAR-A
35 Ton Light 'Mech
CALLSIGN:
MYSTIQUE
RANK:
Green
GUNNERY
PILOTING
JUMPING
SENSORS
REFIT ALL
REPAIR ALL
ARMOR
INTERNAL
ENGINES
11M
PAYLOAD
Short
C Heavy Flamer
Medium
C Large PS Laser
Long
C LR Missile Rack
C LR Missile Rack
C LR Missile Rack
CATAPULT-W
65 Ton Heavy 'Mech
CALLSIGN:
BURNOUT
RANK:
Regular
GUNNERY
PILOTING
JUMPING
SENSORS
REFIT ALL
REPAIR ALL
ARMOR
INTERNAL
ENGINES
15M
PAYLOAD
Short
Laser
Medium
Large PS Laser
Long
LRM Rack
LRM Rack
LRM Rack
SWIFTWIND
5 Ton Light Vehicle
MAX SPEED:
21 m/s
VEHICLE TYPE:
Light
VEHICLE PAYLOAD
1 Vehicle Sensor

Step 7: Final Touches

The campaign is effectively finished, but there are a few extra touches that could be done. Here I will only describe some of the things that can be changed as these are cosmetic and therefore depend more on the designer's choices than anything else. But can help give extra little polish.

Briefing Text

Back in step 3, after adding some small strikes and camera drones, the briefing text was no longer accurate. Also, it would be nice to be able to remove unwanted text etc. Why always have the tactical warnings and notices if there are none for a mission.

Each map has a text file associated with it that was created by the editor. This file actually contains the briefing text itself. Simply go to the `{MCX}\data\missions` folder and open the text file with your map's name (`map1.txt`, `map2.txt` or `map3.txt` for the example campaign) to edit the appropriate map's briefing text.

The briefing text also uses some special commands to help format and color the text.

The special commands table:

Code	Description	Example Use
`%c`	Center text line	
`%n`	New line	
`%%`	Percent ('%') character	
`%fc#`	Select font color (1-6)	`%fc2`
`%t##`	Tab over ## pixels	`%t12  %t03`
`%fs`	Select small font	
`%fm`	Select medium font	
`%fl`	Select large font	

The usable colors table:

1	Red
2	Yellow
3	Green (bluish green really)
4	Cyan
5	Dark Grey
6	White

I found while testing that a single percent sign that is not followed by a recognized command character also seems to print correctly. Though using the double percent is probably safer.

The large font does not work for the grey and white colors for some reason. Instead it uses the medium size font, though the space character will be enlarged. Also, not all characters seem to be defined in the large font (e.g. the '|' character will come out as a space.)

Store Cleanup

Remove extraneous mechs and components from the store if none of those mechs ever become available during the campaign. This helps to keep things tidy. Do the same for components and vehicles. As for warriors, they are already hidden to begin with, so no real problem there.

Just be aware that any mech, vehicle, or component that you wish to allow the player to purchase must exist in the master purchase file (the initial store file) at the start of the mission or it will never show up in there. So even if you will only be allowing the player to purchase an Atlas in the last mission, you will need to have an Atlas entry in the other purchase files and the initial store file as well. The same goes for components etc.

Balance and Test the Campaign

Play the campaign a few times, and on different difficulty settings. Make sure that the campaign as a whole can be completed as intended. Ideally, the campaign should be neither too difficult, nor too easy.

Of course, this is somewhat subjective. The perceived difficulty will be influenced by the skill of the person playing the campaign. This is why you should have people of varying skill levels try out the campaign as well.

You should also keep in mind the story of the campaign itself. Removing too many enemies from a level may break the believability of a mission if you are supposed to be getting swamped by enemies in the story. For such a situation, it may be better to simply dumb down the pilots and maybe give them weaker mechs and/or vehicles.

Conversely, if a mission is supposed to have few enemies, don't go adding tons of enemies to make the mission harder. You can make the mechs they have harder instead, and maybe give them air support. Get creative.

Try different approaches to the missions themselves. Do you really need to always allow the player to bring 12 heavy mechs into a mission? Think of how the official campaigns are put together. Remember the mission where you have to bring an APC (supposedly with Firestorm in it) to a repair bay and commandeer a mech? All you were allowed to bring into that mission was a lousy 15 tons. That mission wasn't even all that hard, but it was both interesting and fun to play.

Change the Starting Music

You don't have to be stuck with that same old music track the editor slaps in you know. Open your mission's .FIT file and look for the heading `[Music]`. The entry `scenarioTuneNum` can be changed from 0 to 23. Granted some of these are less appropriate since some are the victory music etc. But you don't have to be stuck with the same old tune all the time.

Step 8: Packaging it All Up

This is it, the final step. The campaign is working as intended, but it is spread out across various files throughout the data folder.

For solo missions, the map editor does a fine job of creating distribution packages (.DPK files). But for a user made campaign with multiple maps, the editor isn't adequate. So instead, we will package this all up using my tools.

I will explain two methods for packaging the campaign. The first will be a generic method, but will need to be installed manually by any player wanting to use it. However, I will discuss a few method to consider to get around this limitation. The second, though a little more complex to setup, will use a method that will allow the player to use my campaign package installer to install the campaign as either an original campaign or an expansion campaign.

Before we delve into package making however, lets examine what the game is going to expect. This will help us understand what is needed in order to create a proper package.

When the game starts the original campaign, as explained earlier in this document, the game first looks for the file `MechCmdr1.fit`. Currently we have the version of this file for our campaign in the `{MCX}\data\missions` folder, but where was the original file that the game used before we made our user campaign?

The answer is that it resides in the `mission.fst` file with many of the other files the game uses (we've already used this file when we extracted the mech and warrior profiles in step 6). The game always looks for its files first in the `{MCX}\data` folder (and sub-folders) and if not found, scans through the list of FST files that it knows about.

At start-up, the game scans the game's root folder looking for any file with the `.FST` extension and opens it as one of the FST files it can use. The game always opens all FST files that it finds since this is done at game startup. So from our perspective, we can view the FSTs as a kind of virtual disk drive the game uses (this is just a simplification, but a useful one).

This means that we can put our files in one or more FSTs and the game will find the files, but we must make sure that the game opens our FSTs before it opens the game's original ones or the game will use the original version first. We can ensure this by noting that the game will open the FSTs in alphabetical order.

Programmer Note

The game actually uses the Microsoft specific C functions `_findfirst()` and `_findnext()` on "`*.fst`" when scanning the game's root folder. This has the side effect of opening the FST files in alphabetical order.

The only file that cannot be placed in an FST that the game uses is the PKK file as this is, as discussed earlier, really just a savegame MechCommander Gold uses when starting the campaign. If we did put it in an FST, though the game would indeed find the file, it would be unable to perform all the operations it needs to do on the PKK file.

So what we need to do in order to create a package is prepare one or more FST files with all the data needed for our campaign, give them filenames that will ensure that it is opened prior to the game's standard FSTs, and place the needed PKK file in the `{MCX}\data\savegame` folder (overwriting the one that is there).

Since I'll be describing two methods for packaging our campaign, and I would like to keep the methods from stomping on each other as much as possible, I am going to propose the following naming conventions for FST files.

Filenames Starting With	**Proposed Use**
`_` and `0_`	Reserved for special data overrides that must override everything
`1_`	Special user supplied patch data (original campaign) (CDP packages)
`2_`	Special user supplied patch data (expansion campaign) (CDP packages)
`3_`	CDP user campaign data files overriding the original campaign. Does not include purchase info or the MechCmdr1.fit files.
`4_`	CDP user campaign purchase files and MechCmdr1.fit file for overriding the original campaign. Also contains start menu button and images.
`5_`	CDP user campaign data files overriding the expansion campaign. Does not include purchase info or the xMechCmdr1.fit files.
`6_`	CDP user campaign purchasing and xMechCmdr1.fit file for overriding the expansion campaign. Also contains the start menu button and images.
`7_`	CDP user supplied non-critical patch data that can be overridden by above (original campaign)
`8_`	CDP user supplied non-critical patch data that can be overridden by above (expansion campaign)
`9_`	Reserved for special non-critical data patches that can be overridden by above.
`A_` and `AA_` to `AM_`	User campaigns using the generic packaging method.
`AN_` to `AR_`	Reserved for special non-critical data patches that can be overridden by above.

So let's get started.

Preparation:

Regardless of the packaging method we intend to use, we need to prepare in the same way.

Create a work folder somewhere in order to do the FST work. Inside this work folder, create a folder called `data`. Inside the `data` folder create a `missions` folder and another called `terrain`. Finally inside the `missions` folder, create a `warriors` folder. You should end up with a folder structure that looks like this (this is the same folder layout as the game uses):

Copy the tool `makefst.exe` into the work folder.

Now from the `{MCX}\data\missions` folder, select all files that have your map names in them (in my case, this is all files that have `map1`, `map2`, and `map3` in the names) and move (not copy) them to the `data\missions` folder in your work folder. Also move the `MechCmdr1.fit` (or `xMechCmdr1.fit`) and the purchase files we made (for this example campaign, they are `purchase00.fit`, `purchase01.fit` and `purchase02.fit`). This is quite a few files as these include a bunch of .ABI and .ABL files among others.

Move all the files with your map names in them from the `{MCX}\data\missions\warriors` folder and move them to the `data\missions\warriors` folder in your work folder.

Now move all the files with your map names from the `{MCX}\data\terrain` folder and put them in the `data\terrain` folder in your work folder.

Also copy or move the `start0.pkk` (or xstart0.pkk if using the expansion) file from `{MCX}\data\savegame` to the root of your work folder. (You may want to also copy the original `start0.pkk` back, but do still keep the backup copy around).

List of files for the tutorial campaign

Here is the list of files that I copied for this tutorial campaign.

data\missions\campaignmap1.fit
data\missions\campaignmap2.fit
data\missions\campaignmap3.fit
data\missions\map1.ABL
data\missions\map1.FIT
data\missions\map1.txt
data\missions\map1_INIT.ABI
data\missions\map1_LOP.ABI
data\missions\map1_MP.ABI
data\missions\map1_STR.ABI
data\missions\map1_VAR.ABI
data\missions\map2.ABL
data\missions\map2.FIT
data\missions\map2.txt
data\missions\map2_INIT.ABI
data\missions\map2_LOP.ABI
data\missions\map2_MP.ABI
data\missions\map2_STR.ABI
data\missions\map2_VAR.ABI
data\missions\map3.ABL
data\missions\map3.FIT
data\missions\map3.txt
data\missions\map3_INIT.ABI
data\missions\map3_LOP.ABI
data\missions\map3_MP.ABI
data\missions\map3_STR.ABI
data\missions\map3_VAR.ABI
data\missions\MechCmdr1.fit
data\missions\purchase00.fit
data\missions\purchase01.fit
data\missions\purchase02.fit
data\missions\warriors\map1Params.fit
data\missions\warriors\map2Params.fit
data\missions\warriors\map3Params.fit
data\terrain\map1.bdg
data\terrain\map1.dat
data\terrain\map1.elv
data\terrain\map1.fit
data\terrain\map1.gmm
data\terrain\map1.log.tga
data\terrain\map1.obj
data\terrain\map1.pre

List of files for the tutorial campaign
data\terrain\map1.tga data\terrain\map2.bdg data\terrain\map2.dat data\terrain\map2.elv data\terrain\map2.fit data\terrain\map2.gmm data\terrain\map2.log.tga data\terrain\map2.obj data\terrain\map2.pre data\terrain\map2.tga data\terrain\map3.bdg data\terrain\map3.dat data\terrain\map3.elv data\terrain\map3.fit data\terrain\map3.gmm data\terrain\map3.log.tga data\terrain\map3.obj data\terrain\map3.pre data\terrain\map3.tga

Generic Packaging Method:

Now as we've seen, to create a generic package that can be used by the game, we are going to need to make an FST file. This can be made fairly easily by using the `makefst` tool.

The makefst tool works like the makesol tool we've already used. That is, it requires a list of files to be included in the FST. Unlike the PKK files however, FST files contain not only the names of the files in them, but the path they would be in if in a sub-folder (this is why we are re-creating the folder structure of the game). As there are quite a few files to include here, it would be inconvenient to have to type each in individually, so we are going to use the command shell to help us here as well.

We can get this by issuing the following command from the command shell in our work folder:

```
dir /a-d/s/b >fstlist.txt
```

This will create a complete file list that we can trim down for our uses. Edit this file list (`fstlist.txt` if entered the command above exactly) in a text editor.

The "dir" Command Explained

The dir command displays the list of files in the current folder. It has many options, but these are the ones used in the command above.

/a List files with a given attribute.

This option has sub-options with it. The d sub-option refers to directories (that is, folders). The option can look for a specific attribute, or exclude that attribute by placing a dash (-) in front of the sub-option. Therefore, the /a-d option in the command above says: Do not show folders when listing files (it still shows the path to a given file, which we want, but it will not list individual folder names themselves).

/s Recurse through sub-folders.

This option makes the dir command not only look in the current folder, but also in all sub-folders.

/b Brief listing.

This option makes the dir command use the brief form of it's display. Normally the dir command lists each folder separately, will show file sizes and other information, but as we only want a list of files with path information, we use the brief version.

The greater-than symbol (>) in the command above is an output redirection command. It is not an option of the dir command, but a feature of the command shell. Output redirection is used to send the output of a command not to the screen, but to a file. In this case to the file fstlist.txt.

There are a few more files listed here than we want in here as this list really is simply a directory listing. Remove any entries that don't refer to the data folder. The only files that should be in this list are the ones we copied from the {MCX}\data\missions folder, the {MCX}\data\missions\warriors folder and the {MCX}\data\terrain folder (refer to the list of files in the grey box above for what the list should be like for the tutorial campaign).

Also remove the references to the campaign*XXX*.fit files as these are only needed for the solo versions of the maps we made back in step 1 (the campaign uses MechCmdr1.fit instead which we based off of one of them anyway). This left me with 58 files listed.

Now we need to trim the path info as it is much too long and will be incompatible with MechCommander Gold. The game expects to have a relative path starting from the game's root, even in the FSTs. So delete the start of each line until the line starts with the word "data". For example, the first file in my list is:

```
D:\MechCommanderGold\CDPWork\data\missions\map1.ABL
```

So I changed it to:

```
data\missions\map1.ABL
```

Do this for the entire list of files.

Tip
This is where an editor such as Notepad++ really comes in handy. If you hold down Alt while dragging a selection box with the mouse, you can select by columns and so easily remove the start of the lines.

As with the `makesol` tool, `makefst` also expects the first character on a line to be the compression command. So on each line add "`c` " to the start of the line.

Example:

```
data\missions\map1.ABL
```

becomes:

```
c data\missions\map1.ABL
```

Once the above is done, we can build the FST itself. However, we must decide on a name. Recall that we want the game to open our FST before it opens its standard ones. As I've explained, we can do this by using the table proposed standard filename starting characters. From that table we find that we can use "`A_`". This will ensure that the FST will be opened before the others, yet not interfere with the other packaging method I'll describe.

We build the FST similarly to how we built the PKK files. The tool to use is `makefst` and is used in the same way we used the `makesol` tool. I decided to call my campaign "tutorial", and as suggested, I'll put "`A_`" as the first characters of the filename, this is the command I use in the command shell to build my FST.

```
makefst fstlist.txt a_tutorial.fst
```

In the root folder of the work directory are the two files you need to distribute to others if they are to play your campaign. The FST just made and the PKK file.

To install the campaign, copy the FST into the game's root folder, and the PKK into the game's `{MCX}\data\savegame` folder.

To remove the campaign, we reverse the process. First delete or move the FST from the game's root, then restore the original PKK file (`start0.pkk` or `xstart0.pkk`) and the campaign will be removed.

So that is how the generic package is made, installed and removed. But it is still a little cumbersome to use. So lets discuss methods to make things a little easier for the user.

One simple distribution method would be to create a ZIP file with the correct files and folders in it. That is, the FST would be in the root, and the PKK would be in the `data\savegame` folder. The advantage here is that there is simply one file to distribute, and we can usually assume that users will have WinZIP or some compatible program. The disadvantage is that the PKK file still needs to be backed-up by the player.

Another idea would be to use an install maker. There are a few free ones available on the internet. The advantages with this method is that it uses an interface that is familiar (it may or may not use a single file however, so I won't mark this down as an advantage). This method also suffers from the same disadvantage as ZIP files. The user will still need to first backup their PKK files.

Along the same lines as an installer, we could also use a patch making program. Again, there are a few free ones available on the internet. Unlike an installer, a patching program will add, remove or modify files as needed. The advantages and disadvantages are similar to ZIP files as well. Single file for distribution, but user needs to back up the PKK files.

One workaround for the backup problem is to also provide a means of restoring the original PKK. This could be achieved by providing a copy of the original PKK in another installer, ZIP or patch file. For ZIP files, we could place this PKK in a special sub-folder with, say, a batch file that would allow the user to “un-install” the campaign by removing the FST and restoring the PKK from this backup copy. Installer's typically also provide an un-install method, but this might not restore the original PKK (likely depends on the installer used). For patch programs, we could also provide an “un-installer” patch that would reverse the process.

Clearly, there are many choices, and probably many more that I haven't mentioned or even thought of.

The choice of method to use should be dictated by the following factors: Ease of installation, number of files that must be distributed (lower is better here), and ease of un-installation. As a rule, try to view things from the user's standpoint. If things are too complex, they simply won't bother.

With that said, time to start discussing the other packaging method.

CDP Packaging Method:

There are a few problems with the generic packaging method described above. First, you end up with more than one files (unless using one of the distribution tricks). Second, if using one of the discussed methods of distributing the campaign as a single file, there are an infinite ways to do so. Meaning that there won't be a single method for the player to learn in order to install a user made campaign. Third, there is no easy way to allow the user the choice of installing the campaign as an expansion campaign or original campaign (at least not without effectively replicating what will be discussed here). And finally, restoring the original campaigns can also be problematic.

In an attempt to alleviate these problems, I created the Campaign Distribution Package (CDP) and the tool `installCDP`. The tool itself is not the nice front-end I had envisioned it to be, I merely created a simple command line tool after all, but it could serve as either the backbone for a front-end GUI, or have an independent front-end use some of the code as an example on using and installing CDPs.

So to understand some of the differences between CDPs and the generic method described above, lets first examine what goes into a CDP and see how this can solve problems outlined above.

First, a CDP is really nothing more than an FST with no compression done on any of the files (the compression is not needed as most of the files going into it will be compressed anyway) which allows for a very simple and compact extraction program.

The files that the CDP will contain are three FSTs, two PKK files, and a text file. The text file always has the same name which is `campaign_info.txt`. This file merely contains a description of the campaign as whole. This could be used by a front-end program to display the campaign description to the user so they know what it is about. The two PKK files in the CDP are versions of `Start0.PKK` and `xStart0.PKK`. A front-end (and therefore `installCDP`) would install the correct PKK file depending on which campaign the user wishes to override.

This brings us to the three FST files. The first of these is the campaign's main data. The name of this FST is the same as the CDP (except for the extension of course). This FST, when being installed by either a front-end or installCDP will have "`3_`" added to the front of the name if it is to be installed overriding the original campaign, or with "`5_`" added to the front if overriding the expansion campaign. The contents of this FST is identical to that created in the generic method, with the exception that it does not contain the `MechCmdr1.FIT` (or `xMechCmdr1.FIT`) file, nor any of the purchase files.

The other two FSTs are named after the CDP as well, but one has "`o_`" as a prefix to the name and the other has "`x_`" as it's prefix. These FST with the "`o_`" contains the `MechCmdr1.fit` and the purchase*XX*.fit files. While the FST starting with "`x_`" contains the same files but named `xMechCmdr1.fit` and the `xpurchase`*XX*`.fit` files.

When installing the CDP as an original campaign, the front-end (and also installCDP) would extract the FST with "o_", replace the first letter with "4" and install it. If the CDP was being installed as an expansion campaign, the front-end would use the FST with "x_" instead, and replace the first letter of the name with "6". In this way, the campaign can be installed as overriding either the original or expansion campaigns since all required data would be present in the CDP file.

As an example, lets consider installing a CDP of this tutorial campaign called "tutorial". A front-end would open tutorial.cdp and (optionally) look for the campaign_info.txt file so it can display a description for the user. If the user wants to install it, he can then choose to install it as an original or expansion campaign. If he chose original, then the front-end would copy out the tutorial.fst file to the game's root folder and rename it to 3_tutorial.fst, then it would extract the o_tutorial.fst and call it 4_tutorial.fst, finally it would extract Start0.pkk and put it in the {MCX}\data\savegame folder. If the user chose to install it as an expansion campaign, then the front-end would again extract tutorial.fst but this time call it 5_tutorial.fst, it would then extract x_tutorial.fst and call it 6_tutorial.fst and finally extract xStart0.pkk to the {MCX}\data\savegame folder.

Removing the campaign from the "original campaign" slot would simply be a matter of removing the 3_*.fst and 4_*.fst files from the game's root folder, and replacing start0.pkk with a copy of the original. In the case of removing an expansion user campaign, then it would remove the 5_*.fst and 6_*.fst files and replace xstart0.pkk from a copy of the original. This, installCDP can also do.

So what we need to do is create three FST files, two PKK files and a description text file. This is easier than you may think.

The description file, as noted, is merely a text file, so create a text file called campaign_info.txt containing the description of your campaign.

The first FST to make is the main data. We use the exact same procedure we used for the generic package, but we do not include the MechCmdr1.fit or any of the purchase*XX*.fit files as these will be used for making the other FSTs. Create the tutorial.fst as in the generic package. Now the next FST file is the one that will contain the MechCmdr1.fit file and all the purchase*XX*.fit files. Make a file list for these files like so:

```
c data\missions\MechCmdr1.fit
c data\missions\purchase00.fit
c data\missions\purchase01.fit
c data\missions\purchase02.fit
```

Then build the FST calling it `o_tutorial.fst`.

Now we need to make the third FST. To do this, we simply copy the `MechCmdr1.fit` file to `xMechCmdr1.fit` and copy the `purchase`*XX*`.fit` files to `xpurchase`*XX*`.fit`. Make an appropriate file list and build it as `x_tutorial.fst`.

Now we need to create the xStart0.PKK file. I've already described the method to do this, but let me re-iterate it here. Go to your FST work folder, find the masterdef.txt file and change the Setting entry in the [Planet] section to 1. build a version of this PKK file as xStart0.pkk and copy it the working folder for packaging.

All the required files are now prepared, so all that is left is to put them in a CDP. Since a CDP is nothing more than an FST with uncompressed files, we can again use makefst to create it.

In this case, my file list for the CDP was this:

```
u tutorial.fst
u o_tutorial.fst
u x_tutorial.fst
u start0.pkk
u xstart0.pkk
u campaign_info.txt
```

Use this list to build `tutorial.cdp` and we are done. We have a single file that can be distributed and installed using `installCDP` or a front-end, and it can be either an expansion campaign or an original campaign at the user's choice. Further, the campaign can be removed by `installCDP` as well making it much easier for the user to deal with our campaign.

Extra Details for our Campaign:

There is one last thing we can do that would be a nice touch that I haven't mentioned yet. How to change the look of the button in the main menu so the user can immediately see what campaign is loaded in which slot. This little trick will work regardless of the method used to package things up.

In your package work folder, under the data folder, crate a folder called art. In this folder, we will put the edited button images.

The .TGA images for the buttons can be extracted from the `art.fst` file. The filenames are:

Expansion Campaign button:

```
mm_xbh00.tga    // Normal button
mm_xbn00.tga    // Grayed-out (disabled) button
mm_xbg00.tga    // Pressed button
```

Original Campaign button:

```
mm_xbh01.tga    // Normal button
mm_xbn01.tga    // Grayed-out (disabled) button
mm_xbg01.tga    // Pressed button
```

Add the images to the `data\art` folder and then add the appropriate images to the desired FST file list(s). For instance, you would add the line

```
c data\art\mm_xbh01.tga
```

to add the normal button for the original campaign.

Of course, also add the other images as appropriate. The correct FST to use will depend on the packaging method you decided to use.

For the generic package method:

Edit the images for the button of the campaign you are overriding and include it in your campaign FST.

For the CDP package method:

Edit both versions of the buttons. Put the expansion button images in the `x_*.fst` and the original campaign button images in the `o_*.fst`.

The button images all have palettes, so you should preserve this color palette when editing the button images.

Now when your campaign is installed, the name of your campaign will appear on the correct button in the main menu.

Something to note if you do edit the .TGA images. When I tested this, the image came out inverted in game after I saved the edited version. If this happens, merely flip the image vertically and re-save.

Though I do not know which font the game developers used for the text on the buttons, I have found a font that closely matches it (though not quite correct). Try the Franklin Gothic Medium Condensed font with italics and a 21 point font size. This comes very close to matching the original button font. Of course, there is nothing that says you must stick to that font.

Final Words

Well, that about wraps it up for this tutorial. I hope the information in here is clear enough, I do have a tendency to ramble at times.

Some of you may have noticed that this tutorial is called “Part 1”. It is indeed. Part 2 of the guide is about the ABL scripting language. It is effectively a reference guide for the ABL language and the built-in functions available. Part 3 is about more advanced features for campaigns, most of which require some form of scripting.

As of this writing, I’m planning a fourth installment for creating custom content for MechCommander Gold campaigns, and also a fifth installment on global edits (changes that affect the whole of the game) and hacks both to the game executable and from within using ABL scripts.

The rest of this document is the appendices. They contain complete file listing for the files edited in the other steps, as well as a collection of all the tables seen throughout the guide.

-Cmunsta

Appendix 1: Step 2 File Listings

Listing of MechCmdr1.fit:

```
FITini

[HeapInfo]
ul heapSize = 8192

[Movies]
ul numMovies = 5

st movie0 = "logos1"
st movie1 = "opening"
st movie2 = "credits"
st movie3 = "closing"
st movie4 = "credits"

[Scenarios]
ul numScenarios = 3
ul LastScenario = 2

st Scenario0 = "map1"   // name of the scenario file
st Scenario1 = "map2"
st Scenario2 = "map3"


[OpInfo]
l Scenario0Operation = 1
l Scenario0Mission = 1
l Scenario1Operation = 2
l Scenario1Mission = 1
l Scenario2Operation = 2
l Scenario2Mission = 2

[GameSegment0]
ul GameState = 0
ul NextGameState = 10
ul NextGameSegment = 1
ul SmackerMovieId = 0
ul ScenarioId = 0

[GameSegment1]
ul GameState = 3
ul NextGameState = 16
ul NextGameSegment = 2
ul SmackerMovieId = -1
ul ScenarioId = 0

[GameSegment2]
ul GameState = 11
ul NextGameState = 16
ul NextGameSegment = 3
ul SmackerMovieId = -1
ul ScenarioId = 1
```

```
[GameSegment3]
ul GameState = 11
ul NextGameState = 16
ul NextGameSegment = 4
ul SmackerMovieId = -1
ul ScenarioId = 2

FITend
```

Appendix 2: Step 3 File Listings

Listing of map1.fit

```
FITini

// ---------------------------------------------------------------
//
//      Mission Name: map1  Creation Date: 8/19/2008
//
// ---------------------------------------------------------------


[Output]

[GameScale]                       //For 90 Pixel mechs
f WorldUnitsPerMeter = 5.01
f MetersPerWorldUnit = 0.2
ul Duration = 60
f CycleLength = 0.125

[Artillery]
l NumLargeStrikes =  0
l NumSmallStrikes =  3
l NumSensorStrikes =  0
l NumCameraStrikes =  2

[ElementSystem]
ul ElementHeapSize = 358399
ul MaxElements = 2500
ul MaxGroups = 1000

[CraterSystem]
l NumCraters = 100
ul CraterShapeSize = 20479
st CraterFile = "feet"

[ContactManager]
l MaxContacts = 1600

[PotentialContactManager]
l MaxPotentialContacts = 500

[Smoke Manager]
l NumSmokeTypes = 2
l MaxSmokesPerType = 100
ul SmokeSphereHeapSize = 50000

[Smoke0]
l SmokeType = 11

[Smoke1]
l SmokeType = 12
```

```
[CameraSystem]
ul CameraHeapSize = 8191
st CameraFileName = "cameras"

[ObjectSystem]
ul ObjectHeapSize = 1535999
ul ObjectTypeHeapSize = 716799
ul NumObjects = 5000
st ObjectFileName = "object2"

[SpriteSystem]
ul SpriteHeapSize = 524287
st SpriteFileName = "sprites"

ul SpriteManagerHeapSize = 2097151
ul SpriteDataHeapSize = 65535
st ShapeFileName = "shapes"

[SpriteManager]
ul LegHeapSize = 1572863
ul TorsoHeapSize = 2097151
ul RightArmHeapSize = 786431
ul LeftArmHeapSize = 786431
ul TotalMechs = 19
ul Use90Pixel = 1

[TerrainSystem]
st TerrainFileName = "map1"
st TacMapGifName = "map1"

[PaletteSystem]
st PaletteSystem = "Palette"

[Music]
uc scenarioTuneNum = 0

[CollisionSystem]
ul XGridSize =24
ul YGridSize =24
ul GridRadius = 200
ul MaxObjects = 300
ul MaxCollisions = 100
ul MaxPending = 40
ul CollisionHeapSize = 16383
f WarningDist = 250.0                          //This is in world Units!!!
f AlertTime = 2.5                              //This is in seconds
ul NumAlerts = 20

[SensorContactShape]
st shapeName = "blip"

[ABLibraries]
st Library0 = "orders"
st Library1 = "miscfunc"

[Script]
st ScenarioScript = "map1"
```

```
[StatusWindow]
ul PosX = 200
ul PosY = 200
ul Width = 300
ul Height = 100

[Campaign]
st BriefingFile = "map1.txt"
st MapFile = "map1"
l MaxTonnage = 150
l NumDropZones = 1

[DropZone0]
l NumSlots = 3

f PositionX = 1471
f PositionY = 65

f OffsetX0 = 0
f OffsetY0 = 0
f Rotation0 = 180

f OffsetX1 = 0
f OffsetY1 = -100
f Rotation1 = 45

f OffsetX2 = 100
f OffsetY2 = 0
f Rotation2 = -45

[Objectives]
ul NumObjectives = 1       //There can be a maximum of eight objectives

[Objective0]
st Name = "Capture Buildings"
//Types are 0=primary, 1=secondary, 2=other, -1=invisible
ul Type = 0

// Time to accomplish objective.  -1.0 means forever
f TimeLeft = -1

// Status are 0=incomplete, 1=success, 2=failed
ul Status = 0
l Points = 10000

[Warriors]
ul NumWarriors = 1
uc CaptureChance = 0
st BrainParameterFile = "map1Params"
[Warrior1]
st Profile = "PMW00025"
st Brain = "DredAttack01"

[Parts]
ul NumParts = 1
b AlliedTeam = False
```

```
[Part1]
ul ObjectNumber         = 426
ul ControlType          = 2           // player = 1, ai = 2, net = 3
b PlayerPart            = False
ul ControlDataType      = 2           // vehicle control data = 1
c MyIcon                = 0           //
c TeamID                = 1    // 0 = IS, 1 = CLAN, 2 = ALLIED
c CommanderID           = 1    // 0 = IS, 1 = CLAN, 2 = ALLIED
st ObjectProfile        = "PV20000"   // Swiftwind
ul Pilot                = 1
f PositionX             =  447        // Starting position in world.
f PositionY             =  577
f PositionZ             = -1.0        // Elevation automatically set
                                      // by terrain.
f Rotation              =  45         // Rotation of Object in Degrees
ul Gesture              = 2           // Initial Sprite Gesture --
                                      // Determines velocity in Mechs
                                      // -- non-mechs ignore
f Velocity              = 0.0         // Initial Velocity - Start
                                      // velocity for non-mech objects
                                      // -- mechs ignore
l Active                =  1
l Exists                =  1

[Teams]
b AlliedTeam = False

[Commander1Group:0]
l[12] Mates             = 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

FITend
```

Appendix 3: Step 4 File Listings

Listing of filelist.txt:

```
c masterdef.txt
c initstore.txt
```

Listing of masterdef.txt:

```
FITini

[General]
st PurchaseFile="solopurchase"
st GameStateName=(null)
l MissionNumber=00
f LastScenarioTime=0.000000
f LastLogisticsTime=0.000000

[Planet]
l Setting=0

[ResourcePoints]
ul numPoints=500000

[Warriors]
ul NumWarriors=0

[AssWarriors]
ul NumAssWarriors=0

[Mechs]
ul NumMechs=0

[AssMechs]
ul NumAssMechs=0

[Vehicles]
ul NumVehicles=0

[AssVehicles]
ul NumAssVehicles=0

[Components]
ul NumComponents=0

FITend
```

Listing of initstore.txt

```
FITini

[Header]

l NumGifts = 0
l NumMechs = 13
l NumVehicles = 23
l NumComponants = 28
l NumWarriors = 30

[Mech0]
//Commando (IS)
l TypeAAvailable = 2
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM100100"
st TypeWFile = "PM100400"
st TypeJFile = "PM100300"

[Mech1]
//Firestarter (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM101100"
st TypeWFile = "PM101400"
st TypeJFile = "PM101300"

[Mech2]
//Stiletto (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM110100"
st TypeWFile = "PM110400"
st TypeJFile = "PM110300"

[Mech3]
//Raven (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM102100"
st TypeWFile = "PM102400"
st TypeJFile = "PM102300"

[Mech4]
//Hollander (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM103100"
st TypeWFile = "PM103400"
st TypeJFile = "PM103300"
```

```
[Mech5]
//Hunchback (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM104100"
st TypeWFile = "PM104400"
st TypeJFile = "PM104300"

[Mech6]
//Bushwacker (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM111100"
st TypeWFile = "PM111400"
st TypeJFile = "PM111300"

[Mech7]
//Centurion (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM105100"
st TypeWFile = "PM105400"
st TypeJFile = "PM105300"

[Mech8]
//Catapult (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM106100"
st TypeWFile = "PM106400"
st TypeJFile = "PM106300"

[Mech9]
//Jagermech (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM107100"
st TypeWFile = "PM107400"
st TypeJFile = "PM107300"

[Mech10]
//Awesome (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM108100"
st TypeWFile = "PM108400"
st TypeJFile = "PM108300"
```

```
[Mech11]
//Mauler (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM112100"
st TypeWFile = "PM112400"
st TypeJFile = "PM112300"

[Mech12]
//Atlas (IS)
l TypeAAvailable = -1
l TypeJAvailable = -1
l TypeWAvailable = -1
st TypeAFile = "PM109100"
st TypeWFile = "PM109400"
st TypeJFile = "PM109300"

//VEHICLES

[Vehicle0]
// Savannah Master
l NumAvailable = -1
st Filename = "pv31900"

[Vehicle1]
// J. Edgar Light Hover Tank
l NumAvailable = -1
st Filename = "pv21900"

[Vehicle2]
// Pegasus Light Scout Tank*
l NumAvailable = -1
st Filename = "pv21800"

[Vehicle3]
// Saracen Hover Tank
l NumAvailable = -1
st Filename = "pv21400"

[Vehicle4]
// Striker Light Tank
l NumAvailable = -1
st Filename = "pv21700"

[Vehicle5]
// Harasser Light Missile Tank
l NumAvailable = -1
st Filename = "pv20500"

[Vehicle6]
// Condor Heavy Hover Tank
l NumAvailable = -1
st Filename = "pv22100"
```

```
[Vehicle7]
// SRM Carrier
l NumAvailable = -1
st Filename = "pv20900"

[Vehicle8]
// Minelayer*
l NumAvailable = -1
st Filename = "pv22900"

[Vehicle9]
// Minesweeper*
l NumAvailable = -1
st Filename = "pv22800"

[Vehicle10]
// Bulldog Medium Tank
l NumAvailable = -1
st Filename = "pv20800"

[Vehicle11]
// Rommel Medium Tank
l NumAvailable = -1
st Filename = "pv20100"

[Vehicle12]
// Von Luckner Heavy Tank
l NumAvailable = -1
st Filename = "pv22000"

[Vehicle13]
// LRM Carrier
l NumAvailable = -1
st Filename = "pv21000"

[Vehicle14]
// Schrek PPC Carrier
l NumAvailable = -1
st Filename = "pv21300"

[Vehicle15]
// Swiftwind Scout Car*
l NumAvailable = -1
st Filename = "pv20000"

[Vehicle16]
// Refit Truck*
l NumAvailable = -1
st Filename = "pv32300"

[Vehicle17]
// Centipede
l NumAvailable = -1
st Filename = "pv33300"
```

```
[Vehicle18]
// Regulator
l NumAvailable = -1
st Filename = "pv33400"

[Vehicle19]
// Manticore
l NumAvailable = -1
st Filename = "pv33500"

[Vehicle20]
// Pilum*
l NumAvailable = -1
st Filename = "pv33600"

[Vehicle21]
// Alacorn*
l NumAvailable = -1
st Filename = "pv33700"

[Vehicle22]
// Ammo Truck*
l NumAvailable = -1
st Filename = "pv33800"

//COMPONENTS
//short range

[Componant0]
//Pulse Laser (IS)
uc ComponantID = 144
l NumAvailable =-1

[Componant1]
//Short-Range Missile Pack (IS)
uc ComponantID = 123
l NumAvailable =-1

[Componant2]
//Laser (IS)
uc ComponantID = 143
l NumAvailable =-1

[Componant3]
//Streak Short-Range Missile Pack(IS)
uc ComponantID = 125
l NumAvailable =-1

[Componant4]
//Heavy Flamer (IS)
uc ComponantID = 147
l NumAvailable =-1
```

```
[Componant5]
// Heavy Autocannon (IS)
uc ComponantID = 102
l NumAvailable =-1

//medium range

[Componant6]
//  Autocannon (IS)
uc ComponantID = 101
l NumAvailable =-1

[Componant7]
// Large Laser (IS)
uc ComponantID = 140
l NumAvailable =-1

[Componant8]
// Large Pulse Laser (IS)
uc ComponantID = 142
l NumAvailable =-1

[Componant9]
// Particle Projector Cannon (IS)
uc ComponantID = 145
l NumAvailable =-1

//long range

[Componant10]
//Light Autocannon (IS)
uc ComponantID = 100
l NumAvailable =-1

[Componant11]
// Long-Range Missile Rack (IS)
uc ComponantID = 120
l NumAvailable =-1

[Componant12]
// Light Ultra Autocannon (IS)
uc ComponantID = 103
l NumAvailable =-1

[Componant13]
// Large ER Laser (IS)
uc ComponantID = 141
l NumAvailable =-1

[Componant14]
// ER Particle Projector Cannon (IS)
uc ComponantID = 146
l NumAvailable =-1
```

```
[Componant15]
// Gauss Rifle (IS)
uc ComponantID = 104
l NumAvailable =-1

// Sensor

[Componant16]
// Sensor (Basic) (IS)
uc ComponantID = 13
l NumAvailable =-1

[Componant17]
// Sensor (Intermediate) (IS)
uc ComponantID = 14
l NumAvailable =-1

[Componant18]
// Sensor (Advanced) (IS)
uc ComponantID = 15
l NumAvailable =-1

[Componant19]
// Beagle Probe (IS)
uc ComponantID = 37
l NumAvailable =-1

[Componant20]
// Guardian ECM (IS)
uc ComponantID = 38
l NumAvailable =-1

[Componant21]
// Light LBX AC(IS)
uc ComponantID = 107
l NumAvailable =-1

[Componant22]
// LBX AC(IS)
uc ComponantID = 108
l NumAvailable =-1

[Componant23]
// Hvy LBX AC(IS)
uc ComponantID = 109
l NumAvailable =-1

[Componant24]
// Light Gauss(IS)
uc ComponantID = 99
l NumAvailable =-1

[Componant25]
// Long Tom Cannon(IS)
uc ComponantID = 160
l NumAvailable =-1
```

```
[Componant26]
// X Pulse(IS)
uc ComponantID = 139
l NumAvailable =-1

[Componant27]
// Rail Gun
uc ComponantID = 98
l NumAvailable = -1


//pilots

[Warrior0]
st Profile = "pmw00051"
l Status =0

[Warrior1]
st Profile = "pmw00052"
l Status =0

[Warrior2]
st Profile = "pmw00053"
l Status =0

[Warrior3]
st Profile = "pmw00054"
l Status =0

[Warrior4]
st Profile = "pmw00055"
l Status =0

[Warrior5]
st Profile = "pmw00056"
l Status =0

[Warrior6]
st Profile = "pmw00057"
l Status =0

[Warrior7]
st Profile = "pmw00058"
l Status =0

[Warrior8]
st Profile = "pmw00059"
l Status =0

[Warrior9]
st Profile = "pmw00060"
l Status =0

[Warrior10]
st Profile = "pmw00061"
l Status =0
```

```
[Warrior11]
st Profile = "pmw00062"
l Status =0

[Warrior12]
st Profile = "pmw00063"
l Status =0

[Warrior13]
st Profile = "pmw00064"
l Status =0

[Warrior14]
st Profile = "pmw00065"
l Status =0

[Warrior15]
st Profile = "pmw00066"
l Status =0

[Warrior16]
st Profile = "pmw00067"
l Status =0

[Warrior17]
st Profile = "pmw00068"
l Status =0

[Warrior18]
st Profile = "pmw00069"
l Status =0

[Warrior19]
st Profile = "pmw00070"
l Status =0

[Warrior20]
st Profile = "pmw00071"
l Status =0

[Warrior21]
st Profile = "pmw00072"
l Status =0

[Warrior22]
st Profile = "pmw00073"
l Status =0

[Warrior23]
st Profile = "pmw00074"
l Status =0

[Warrior24]
st Profile = "pmw00075"
l Status =0
```

```
[Warrior25]
st Profile = "pmw00076"
l Status =0

[Warrior26]
st Profile = "pmw00077"
l Status =0

[Warrior27]
st Profile = "pmw00078"
l Status =0

[Warrior28]
st Profile = "pmw00079"
l Status =0

[Warrior29]
st Profile = "pmw00080"
l Status =0

FITEnd
```

Appendix 4: Step 5 File Listings

Listing of initstore.txt and purchase00.fit (they are exactly the same, so use the same listing for both)

```
FITini

[Header]

l NumGifts = 0
l NumMechs = 13
l NumVehicles = 23
l NumComponants = 28
l NumWarriors = 30

[Mech0]
//Commando (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM100100"
st TypeWFile = "PM100400"
st TypeJFile = "PM100300"

[Mech1]
//Firestarter (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM101100"
st TypeWFile = "PM101400"
st TypeJFile = "PM101300"

[Mech2]
//Stiletto (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM110100"
st TypeWFile = "PM110400"
st TypeJFile = "PM110300"

[Mech3]
//Raven (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM102100"
st TypeWFile = "PM102400"
st TypeJFile = "PM102300"
```

```
[Mech4]
//Hollander (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM103100"
st TypeWFile = "PM103400"
st TypeJFile = "PM103300"

[Mech5]
//Hunchback (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM104100"
st TypeWFile = "PM104400"
st TypeJFile = "PM104300"

[Mech6]
//Bushwacker (IS)
l TypeAAvailable = 2
l TypeJAvailable = 0
l TypeWAvailable = 2
st TypeAFile = "PM111100"
st TypeWFile = "PM111400"
st TypeJFile = "PM111300"

[Mech7]
//Centurion (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM105100"
st TypeWFile = "PM105400"
st TypeJFile = "PM105300"

[Mech8]
//Catapult (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM106100"
st TypeWFile = "PM106400"
st TypeJFile = "PM106300"

[Mech9]
//Jagermech (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM107100"
st TypeWFile = "PM107400"
st TypeJFile = "PM107300"
```

```
[Mech10]
//Awesome (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM108100"
st TypeWFile = "PM108400"
st TypeJFile = "PM108300"

[Mech11]
//Mauler (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM112100"
st TypeWFile = "PM112400"
st TypeJFile = "PM112300"

[Mech12]
//Atlas (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM109100"
st TypeWFile = "PM109400"
st TypeJFile = "PM109300"

//VEHICLES

[Vehicle0]
// Savannah Master
l NumAvailable = 0
st Filename = "pv31900"

[Vehicle1]
// J. Edgar Light Hover Tank
l NumAvailable = 0
st Filename = "pv21900"

[Vehicle2]
// Pegasus Light Scout Tank*
l NumAvailable = 0
st Filename = "pv21800"

[Vehicle3]
// Saracen Hover Tank
l NumAvailable = 0
st Filename = "pv21400"

[Vehicle4]
// Striker Light Tank
l NumAvailable = 0
st Filename = "pv21700"
```

```
[Vehicle5]
// Harasser Light Missile Tank
l NumAvailable = 0
st Filename = "pv20500"

[Vehicle6]
// Condor Heavy Hover Tank
l NumAvailable = 0
st Filename = "pv22100"

[Vehicle7]
// SRM Carrier
l NumAvailable = 0
st Filename = "pv20900"

[Vehicle8]
// Minelayer*
l NumAvailable = 0
st Filename = "pv22900"

[Vehicle9]
// Minesweeper*
l NumAvailable = 0
st Filename = "pv22800"

[Vehicle10]
// Bulldog Medium Tank
l NumAvailable = 0
st Filename = "pv20800"

[Vehicle11]
// Rommel Medium Tank
l NumAvailable = 0
st Filename = "pv20100"

[Vehicle12]
// Von Luckner Heavy Tank
l NumAvailable = 0
st Filename = "pv22000"

[Vehicle13]
// LRM Carrier
l NumAvailable = 0
st Filename = "pv21000"

[Vehicle14]
// Schrek PPC Carrier
l NumAvailable = 0
st Filename = "pv21300"

[Vehicle15]
// Swiftwind Scout Car*
l NumAvailable = 0
st Filename = "pv20000"
```

```
[Vehicle16]
// Refit Truck*
l NumAvailable = 0
st Filename = "pv32300"

[Vehicle17]
// Centipede
l NumAvailable = 0
st Filename = "pv33300"

[Vehicle18]
// Regulator
l NumAvailable = 0
st Filename = "pv33400"

[Vehicle19]
// Manticore
l NumAvailable = 0
st Filename = "pv33500"

[Vehicle20]
// Pilum*
l NumAvailable = 0
st Filename = "pv33600"

[Vehicle21]
// Alacorn*
l NumAvailable = 0
st Filename = "pv33700"

[Vehicle22]
// Ammo Truck*
l NumAvailable = 0
st Filename = "pv33800"

//COMPONENTS
//short range

[Componant0]
//Pulse Laser (IS)
uc ComponantID = 144
l NumAvailable = 8

[Componant1]
//Short-Range Missile Pack (IS)
uc ComponantID = 123
l NumAvailable =0

[Componant2]
//Laser (IS)
uc ComponantID = 143
l NumAvailable =0

[Componant3]
//Streak Short-Range Missile Pack(IS)
uc ComponantID = 125
l NumAvailable =0
```

```
[Componant4]
//Heavy Flamer (IS)
uc ComponantID = 147
l NumAvailable =2

[Componant5]
// Heavy Autocannon (IS)
uc ComponantID = 102
l NumAvailable =0

//medium range

[Componant6]
//  Autocannon (IS)
uc ComponantID = 101
l NumAvailable =0

[Componant7]
// Large Laser (IS)
uc ComponantID = 140
l NumAvailable =0

[Componant8]
// Large Pulse Laser (IS)
uc ComponantID = 142
l NumAvailable =4

[Componant9]
// Particle Projector Cannon (IS)
uc ComponantID = 145
l NumAvailable =4

//long range

[Componant10]
//Light Autocannon (IS)
uc ComponantID = 100
l NumAvailable =0

[Componant11]
// Long-Range Missile Rack (IS)
uc ComponantID = 120
l NumAvailable =0

[Componant12]
// Light Ultra Autocannon (IS)
uc ComponantID = 103
l NumAvailable =0

[Componant13]
// Large ER Laser (IS)
uc ComponantID = 141
l NumAvailable =0
```

```
[Componant14]
// ER Particle Projector Cannon (IS)
uc ComponantID = 146
l NumAvailable =0

[Componant15]
// Gauss Rifle (IS)
uc ComponantID = 104
l NumAvailable =0

// Sensor

[Componant16]
// Sensor (Basic) (IS)
uc ComponantID = 13
l NumAvailable =0

[Componant17]
// Sensor (Intermediate) (IS)
uc ComponantID = 14
l NumAvailable =0

[Componant18]
// Sensor (Advanced) (IS)
uc ComponantID = 15
l NumAvailable =0

[Componant19]
// Beagle Probe (IS)
uc ComponantID = 37
l NumAvailable =0

[Componant20]
// Guardian ECM (IS)
uc ComponantID = 38
l NumAvailable =0

[Componant21]
// Light LBX AC(IS)
uc ComponantID = 107
l NumAvailable =0

[Componant22]
// LBX AC(IS)
uc ComponantID = 108
l NumAvailable =0

[Componant23]
// Hvy LBX AC(IS)
uc ComponantID = 109
l NumAvailable =0

[Componant24]
// Light Gauss(IS)
uc ComponantID = 99
l NumAvailable =0
```

```
[Componant25]
// Long Tom Cannon(IS)
uc ComponantID = 160
l NumAvailable =0

[Componant26]
// X Pulse(IS)
uc ComponantID = 139
l NumAvailable =0

[Componant27]
// Rail Gun
uc ComponantID = 98
l NumAvailable = 0


//pilots

[Warrior0]
//Beast
st Profile = "pmw00051"
l Status =0

[Warrior1]
st Profile = "pmw00052"
l Status =4

[Warrior2]
st Profile = "pmw00053"
l Status =4

[Warrior3]
st Profile = "pmw00054"
l Status =4

[Warrior4]
st Profile = "pmw00055"
l Status =4

[Warrior5]
st Profile = "pmw00056"
l Status =4

[Warrior6]
st Profile = "pmw00057"
l Status =4

[Warrior7]
st Profile = "pmw00058"
l Status =4

[Warrior8]
st Profile = "pmw00059"
l Status =4
```

```
[Warrior9]
st Profile = "pmw00060"
l Status =4

[Warrior10]
st Profile = "pmw00061"
l Status =4

[Warrior11]
st Profile = "pmw00062"
l Status =4

[Warrior12]
st Profile = "pmw00063"
l Status =0

[Warrior13]
st Profile = "pmw00064"
l Status =4

[Warrior14]
st Profile = "pmw00065"
l Status =0

[Warrior15]
st Profile = "pmw00066"
l Status =4

[Warrior16]
st Profile = "pmw00067"
l Status =0

[Warrior17]
st Profile = "pmw00068"
l Status =4

[Warrior18]
st Profile = "pmw00069"
l Status =4

[Warrior19]
st Profile = "pmw00070"
l Status =4

[Warrior20]
st Profile = "pmw00071"
l Status =4

[Warrior21]
st Profile = "pmw00072"
l Status =4

[Warrior22]
st Profile = "pmw00073"
l Status =4
```

```
[Warrior23]
st Profile = "pmw00074"
l Status =4

[Warrior24]
st Profile = "pmw00075"
l Status =4

[Warrior25]
// Blaze
st Profile = "pmw00076"
l Status = 4

[Warrior26]
st Profile = "pmw00077"
l Status =4

[Warrior27]
st Profile = "pmw00078"
l Status =4

[Warrior28]
st Profile = "pmw00079"
l Status =4

[Warrior29]
st Profile = "pmw00080"
l Status =4

FITEnd
```

Listing of purchase01.fit

```
FITini

[Header]

l NumGifts = 0
l NumMechs = 13
l NumVehicles = 23
l NumComponants = 28
l NumWarriors = 30

[Mech0]
//Commando (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM100100"
st TypeWFile = "PM100400"
st TypeJFile = "PM100300"
```

```
[Mech1]
//Firestarter (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM101100"
st TypeWFile = "PM101400"
st TypeJFile = "PM101300"

[Mech2]
//Stiletto (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM110100"
st TypeWFile = "PM110400"
st TypeJFile = "PM110300"

[Mech3]
//Raven (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM102100"
st TypeWFile = "PM102400"
st TypeJFile = "PM102300"

[Mech4]
//Hollander (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM103100"
st TypeWFile = "PM103400"
st TypeJFile = "PM103300"

[Mech5]
//Hunchback (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM104100"
st TypeWFile = "PM104400"
st TypeJFile = "PM104300"

[Mech6]
//Bushwacker (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM111100"
st TypeWFile = "PM111400"
st TypeJFile = "PM111300"
```

```
[Mech7]
//Centurion (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM105100"
st TypeWFile = "PM105400"
st TypeJFile = "PM105300"

[Mech8]
//Catapult (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM106100"
st TypeWFile = "PM106400"
st TypeJFile = "PM106300"

[Mech9]
//Jagermech (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM107100"
st TypeWFile = "PM107400"
st TypeJFile = "PM107300"

[Mech10]
//Awesome (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM108100"
st TypeWFile = "PM108400"
st TypeJFile = "PM108300"

[Mech11]
//Mauler (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM112100"
st TypeWFile = "PM112400"
st TypeJFile = "PM112300"

[Mech12]
//Atlas (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 1
st TypeAFile = "PM109100"
st TypeWFile = "PM109400"
st TypeJFile = "PM109300"

//VEHICLES
```

```
[Vehicle0]
// Savannah Master
l NumAvailable = 0
st Filename = "pv31900"

[Vehicle1]
// J. Edgar Light Hover Tank
l NumAvailable = 0
st Filename = "pv21900"

[Vehicle2]
// Pegasus Light Scout Tank*
l NumAvailable = 0
st Filename = "pv21800"

[Vehicle3]
// Saracen Hover Tank
l NumAvailable = 0
st Filename = "pv21400"

[Vehicle4]
// Striker Light Tank
l NumAvailable = 0
st Filename = "pv21700"

[Vehicle5]
// Harasser Light Missile Tank
l NumAvailable = 0
st Filename = "pv20500"

[Vehicle6]
// Condor Heavy Hover Tank
l NumAvailable = 0
st Filename = "pv22100"

[Vehicle7]
// SRM Carrier
l NumAvailable = 0
st Filename = "pv20900"

[Vehicle8]
// Minelayer*
l NumAvailable = 0
st Filename = "pv22900"

[Vehicle9]
// Minesweeper*
l NumAvailable = 0
st Filename = "pv22800"

[Vehicle10]
// Bulldog Medium Tank
l NumAvailable = 0
st Filename = "pv20800"
```

```
[Vehicle11]
// Rommel Medium Tank
l NumAvailable = 0
st Filename = "pv20100"

[Vehicle12]
// Von Luckner Heavy Tank
l NumAvailable = 0
st Filename = "pv22000"

[Vehicle13]
// LRM Carrier
l NumAvailable = 0
st Filename = "pv21000"

[Vehicle14]
// Schrek PPC Carrier
l NumAvailable = 0
st Filename = "pv21300"

[Vehicle15]
// Swiftwind Scout Car*
l NumAvailable = 0
st Filename = "pv20000"

[Vehicle16]
// Refit Truck*
l NumAvailable = 0
st Filename = "pv32300"

[Vehicle17]
// Centipede
l NumAvailable = 0
st Filename = "pv33300"

[Vehicle18]
// Regulator
l NumAvailable = 0
st Filename = "pv33400"

[Vehicle19]
// Manticore
l NumAvailable = 0
st Filename = "pv33500"

[Vehicle20]
// Pilum*
l NumAvailable = 0
st Filename = "pv33600"

[Vehicle21]
// Alacorn*
l NumAvailable = 0
st Filename = "pv33700"
```

```
[Vehicle22]
// Ammo Truck*
l NumAvailable = 0
st Filename = "pv33800"

//COMPONENTS
//short range

[Componant0]
//Pulse Laser (IS)
uc ComponantID = 144
l NumAvailable = 0

[Componant1]
//Short-Range Missile Pack (IS)
uc ComponantID = 123
l NumAvailable =0

[Componant2]
//Laser (IS)
uc ComponantID = 143
l NumAvailable =0

[Componant3]
//Streak Short-Range Missile Pack(IS)
uc ComponantID = 125
l NumAvailable =0

[Componant4]
//Heavy Flamer (IS)
uc ComponantID = 147
l NumAvailable =0

[Componant5]
// Heavy Autocannon (IS)
uc ComponantID = 102
l NumAvailable =0

//medium range

[Componant6]
//  Autocannon (IS)
uc ComponantID = 101
l NumAvailable =0

[Componant7]
// Large Laser (IS)
uc ComponantID = 140
l NumAvailable =0

[Componant8]
// Large Pulse Laser (IS)
uc ComponantID = 142
l NumAvailable =0
```

```
[Componant9]
// Particle Projector Cannon (IS)
uc ComponantID = 145
l NumAvailable =0

//long range

[Componant10]
//Light Autocannon (IS)
uc ComponantID = 100
l NumAvailable =0

[Componant11]
// Long-Range Missile Rack (IS)
uc ComponantID = 120
l NumAvailable =0

[Componant12]
// Light Ultra Autocannon (IS)
uc ComponantID = 103
l NumAvailable =0

[Componant13]
// Large ER Laser (IS)
uc ComponantID = 141
l NumAvailable =0

[Componant14]
// ER Particle Projector Cannon (IS)
uc ComponantID = 146
l NumAvailable =2

[Componant15]
// Gauss Rifle (IS)
uc ComponantID = 104
l NumAvailable =0

// Sensor

[Componant16]
// Sensor (Basic) (IS)
uc ComponantID = 13
l NumAvailable =0

[Componant17]
// Sensor (Intermediate) (IS)
uc ComponantID = 14
l NumAvailable =0

[Componant18]
// Sensor (Advanced) (IS)
uc ComponantID = 15
l NumAvailable =0
```

```
[Componant19]
// Beagle Probe (IS)
uc ComponantID = 37
l NumAvailable =0

[Componant20]
// Guardian ECM (IS)
uc ComponantID = 38
l NumAvailable =0

[Componant21]
// Light LBX AC(IS)
uc ComponantID = 107
l NumAvailable =0

[Componant22]
// LBX AC(IS)
uc ComponantID = 108
l NumAvailable =0

[Componant23]
// Hvy LBX AC(IS)
uc ComponantID = 109
l NumAvailable =0

[Componant24]
// Light Gauss(IS)
uc ComponantID = 99
l NumAvailable =0

[Componant25]
// Long Tom Cannon(IS)
uc ComponantID = 160
l NumAvailable =0

[Componant26]
// X Pulse(IS)
uc ComponantID = 139
l NumAvailable =0

[Componant27]
// Rail Gun
uc ComponantID = 98
l NumAvailable = 0


//pilots

[Warrior0]
//Beast
st Profile = "pmw00051"
l Status =0

[Warrior1]
st Profile = "pmw00052"
l Status =4
```

```
[Warrior2]
st Profile = "pmw00053"
l Status =4

[Warrior3]
st Profile = "pmw00054"
l Status =4

[Warrior4]
st Profile = "pmw00055"
l Status =4

[Warrior5]
st Profile = "pmw00056"
l Status =4

[Warrior6]
st Profile = "pmw00057"
l Status =4

[Warrior7]
st Profile = "pmw00058"
l Status =4

[Warrior8]
st Profile = "pmw00059"
l Status =4

[Warrior9]
st Profile = "pmw00060"
l Status =4

[Warrior10]
st Profile = "pmw00061"
l Status =4

[Warrior11]
st Profile = "pmw00062"
l Status =4

[Warrior12]
st Profile = "pmw00063"
l Status =0

[Warrior13]
st Profile = "pmw00064"
l Status =4

[Warrior14]
st Profile = "pmw00065"
l Status =0

[Warrior15]
st Profile = "pmw00066"
l Status =4
```

```
[Warrior16]
st Profile = "pmw00067"
l Status =0

[Warrior17]
st Profile = "pmw00068"
l Status =4

[Warrior18]
st Profile = "pmw00069"
l Status =4

[Warrior19]
st Profile = "pmw00070"
l Status =4

[Warrior20]
st Profile = "pmw00071"
l Status =4

[Warrior21]
st Profile = "pmw00072"
l Status =4

[Warrior22]
st Profile = "pmw00073"
l Status =4

[Warrior23]
st Profile = "pmw00074"
l Status =4

[Warrior24]
st Profile = "pmw00075"
l Status =4

[Warrior25]
// Blaze
st Profile = "pmw00076"
l Status = 4

[Warrior26]
st Profile = "pmw00077"
l Status =4

[Warrior27]
st Profile = "pmw00078"
l Status =4

[Warrior28]
st Profile = "pmw00079"
l Status =4
```

```
[Warrior29]
st Profile = "pmw00080"
l Status =4

FITEnd
```

Listing of purchase02.fit

```
FITini

[Header]

l NumGifts = 0
l NumMechs = 13
l NumVehicles = 23
l NumComponants = 28
l NumWarriors = 30

[Mech0]
//Commando (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM100100"
st TypeWFile = "PM100400"
st TypeJFile = "PM100300"

[Mech1]
//Firestarter (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM101100"
st TypeWFile = "PM101400"
st TypeJFile = "PM101300"

[Mech2]
//Stiletto (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM110100"
st TypeWFile = "PM110400"
st TypeJFile = "PM110300"

[Mech3]
//Raven (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM102100"
st TypeWFile = "PM102400"
st TypeJFile = "PM102300"
```

```
[Mech4]
//Hollander (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM103100"
st TypeWFile = "PM103400"
st TypeJFile = "PM103300"

[Mech5]
//Hunchback (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM104100"
st TypeWFile = "PM104400"
st TypeJFile = "PM104300"

[Mech6]
//Bushwacker (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM111100"
st TypeWFile = "PM111400"
st TypeJFile = "PM111300"

[Mech7]
//Centurion (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM105100"
st TypeWFile = "PM105400"
st TypeJFile = "PM105300"

[Mech8]
//Catapult (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM106100"
st TypeWFile = "PM106400"
st TypeJFile = "PM106300"

[Mech9]
//Jagermech (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM107100"
st TypeWFile = "PM107400"
st TypeJFile = "PM107300"
```

```
[Mech10]
//Awesome (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM108100"
st TypeWFile = "PM108400"
st TypeJFile = "PM108300"

[Mech11]
//Mauler (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM112100"
st TypeWFile = "PM112400"
st TypeJFile = "PM112300"

[Mech12]
//Atlas (IS)
l TypeAAvailable = 0
l TypeJAvailable = 0
l TypeWAvailable = 0
st TypeAFile = "PM109100"
st TypeWFile = "PM109400"
st TypeJFile = "PM109300"

//VEHICLES

[Vehicle0]
// Savannah Master
l NumAvailable = 0
st Filename = "pv31900"

[Vehicle1]
// J. Edgar Light Hover Tank
l NumAvailable = 0
st Filename = "pv21900"

[Vehicle2]
// Pegasus Light Scout Tank*
l NumAvailable = 0
st Filename = "pv21800"

[Vehicle3]
// Saracen Hover Tank
l NumAvailable = 0
st Filename = "pv21400"

[Vehicle4]
// Striker Light Tank
l NumAvailable = 0
st Filename = "pv21700"
```

```
[Vehicle5]
// Harasser Light Missile Tank
l NumAvailable = 0
st Filename = "pv20500"

[Vehicle6]
// Condor Heavy Hover Tank
l NumAvailable = 0
st Filename = "pv22100"

[Vehicle7]
// SRM Carrier
l NumAvailable = 0
st Filename = "pv20900"

[Vehicle8]
// Minelayer*
l NumAvailable = 0
st Filename = "pv22900"

[Vehicle9]
// Minesweeper*
l NumAvailable = 0
st Filename = "pv22800"

[Vehicle10]
// Bulldog Medium Tank
l NumAvailable = 0
st Filename = "pv20800"

[Vehicle11]
// Rommel Medium Tank
l NumAvailable = 0
st Filename = "pv20100"

[Vehicle12]
// Von Luckner Heavy Tank
l NumAvailable = 0
st Filename = "pv22000"

[Vehicle13]
// LRM Carrier
l NumAvailable = 0
st Filename = "pv21000"

[Vehicle14]
// Schrek PPC Carrier
l NumAvailable = 0
st Filename = "pv21300"

[Vehicle15]
// Swiftwind Scout Car*
l NumAvailable = 0
st Filename = "pv20000"
```

```
[Vehicle16]
// Refit Truck*
l NumAvailable = 0
st Filename = "pv32300"

[Vehicle17]
// Centipede
l NumAvailable = 0
st Filename = "pv33300"

[Vehicle18]
// Regulator
l NumAvailable = 0
st Filename = "pv33400"

[Vehicle19]
// Manticore
l NumAvailable = 0
st Filename = "pv33500"

[Vehicle20]
// Pilum*
l NumAvailable = 0
st Filename = "pv33600"

[Vehicle21]
// Alacorn*
l NumAvailable = 0
st Filename = "pv33700"

[Vehicle22]
// Ammo Truck*
l NumAvailable = 0
st Filename = "pv33800"

//COMPONENTS
//short range

[Componant0]
//Pulse Laser (IS)
uc ComponantID = 144
l NumAvailable = 0

[Componant1]
//Short-Range Missile Pack (IS)
uc ComponantID = 123
l NumAvailable =0

[Componant2]
//Laser (IS)
uc ComponantID = 143
l NumAvailable =0

[Componant3]
//Streak Short-Range Missile Pack(IS)
uc ComponantID = 125
l NumAvailable =0
```

```
[Componant4]
//Heavy Flamer (IS)
uc ComponantID = 147
l NumAvailable =0

[Componant5]
// Heavy Autocannon (IS)
uc ComponantID = 102
l NumAvailable =0

//medium range

[Componant6]
//  Autocannon (IS)
uc ComponantID = 101
l NumAvailable =0

[Componant7]
// Large Laser (IS)
uc ComponantID = 140
l NumAvailable =0

[Componant8]
// Large Pulse Laser (IS)
uc ComponantID = 142
l NumAvailable =0

[Componant9]
// Particle Projector Cannon (IS)
uc ComponantID = 145
l NumAvailable =0

//long range

[Componant10]
//Light Autocannon (IS)
uc ComponantID = 100
l NumAvailable =0

[Componant11]
// Long-Range Missile Rack (IS)
uc ComponantID = 120
l NumAvailable =0

[Componant12]
// Light Ultra Autocannon (IS)
uc ComponantID = 103
l NumAvailable =0

[Componant13]
// Large ER Laser (IS)
uc ComponantID = 141
l NumAvailable =0
```

```
[Componant14]
// ER Particle Projector Cannon (IS)
uc ComponantID = 146
l NumAvailable =0

[Componant15]
// Gauss Rifle (IS)
uc ComponantID = 104
l NumAvailable =0

// Sensor

[Componant16]
// Sensor (Basic) (IS)
uc ComponantID = 13
l NumAvailable =0

[Componant17]
// Sensor (Intermediate) (IS)
uc ComponantID = 14
l NumAvailable =0

[Componant18]
// Sensor (Advanced) (IS)
uc ComponantID = 15
l NumAvailable =0

[Componant19]
// Beagle Probe (IS)
uc ComponantID = 37
l NumAvailable =0

[Componant20]
// Guardian ECM (IS)
uc ComponantID = 38
l NumAvailable =0

[Componant21]
// Light LBX AC(IS)
uc ComponantID = 107
l NumAvailable =0

[Componant22]
// LBX AC(IS)
uc ComponantID = 108
l NumAvailable =0

[Componant23]
// Hvy LBX AC(IS)
uc ComponantID = 109
l NumAvailable =0

[Componant24]
// Light Gauss(IS)
uc ComponantID = 99
l NumAvailable =0
```

```
[Componant25]
// Long Tom Cannon(IS)
uc ComponantID = 160
l NumAvailable =0

[Componant26]
// X Pulse(IS)
uc ComponantID = 139
l NumAvailable =0

[Componant27]
// Rail Gun
uc ComponantID = 98
l NumAvailable = 0


//pilots

[Warrior0]
//Beast
st Profile = "pmw00051"
l Status =0

[Warrior1]
st Profile = "pmw00052"
l Status =4

[Warrior2]
st Profile = "pmw00053"
l Status =4

[Warrior3]
st Profile = "pmw00054"
l Status =4

[Warrior4]
st Profile = "pmw00055"
l Status =4

[Warrior5]
st Profile = "pmw00056"
l Status =4

[Warrior6]
st Profile = "pmw00057"
l Status =4

[Warrior7]
st Profile = "pmw00058"
l Status =4

[Warrior8]
st Profile = "pmw00059"
l Status =4
```

```
[Warrior9]
st Profile = "pmw00060"
l Status =4

[Warrior10]
st Profile = "pmw00061"
l Status =4

[Warrior11]
st Profile = "pmw00062"
l Status =4

[Warrior12]
st Profile = "pmw00063"
l Status =0

[Warrior13]
st Profile = "pmw00064"
l Status =4

[Warrior14]
st Profile = "pmw00065"
l Status =0

[Warrior15]
st Profile = "pmw00066"
l Status =4

[Warrior16]
st Profile = "pmw00067"
l Status =0

[Warrior17]
st Profile = "pmw00068"
l Status =4

[Warrior18]
st Profile = "pmw00069"
l Status =4

[Warrior19]
st Profile = "pmw00070"
l Status =4

[Warrior20]
st Profile = "pmw00071"
l Status =4

[Warrior21]
st Profile = "pmw00072"
l Status =4

[Warrior22]
st Profile = "pmw00073"
l Status =4
```

```
[Warrior23]
st Profile = "pmw00074"
l Status =4

[Warrior24]
st Profile = "pmw00075"
l Status =4

[Warrior25]
// Blaze
st Profile = "pmw00076"
l Status = 0

[Warrior26]
st Profile = "pmw00077"
l Status =4

[Warrior27]
st Profile = "pmw00078"
l Status =4

[Warrior28]
st Profile = "pmw00079"
l Status =4

[Warrior29]
st Profile = "pmw00080"
l Status =4

FITEnd
```

Appendix 5: Step 6 Listings

Listing of masterdef.txt after adding one component and one mech:

```
FITini

[General]
st PurchaseFile="solopurchase"
st GameStateName=(null)
l MissionNumber=00
f LastScenarioTime=0.000000
f LastLogisticsTime=0.000000

[Planet]
l Setting=0

[ResourcePoints]
ul numPoints=500000

[Warriors]
ul NumWarriors=0

[AssWarriors]
ul NumAssWarriors=0

[Mechs]
ul NumMechs=1

[Mech0]
st Profile = "PM109400" // Atlas-W
l NumAvailable = 1


[AssMechs]
ul NumAssMechs=0

[Vehicles]
ul NumVehicles=0

[AssVehicles]
ul NumAssVehicles=0

[Components]
ul NumComponents=1

[Componant0]
// ER laser
uc ComponantID = 152
l NumAvailable = 4

FITend
```

Listing of filelist.txt after adding Lynx as a pre-hired MechWarrior

```
c masterdef.txt
c Lynx.txt
c initstore.txt
```

Listing of masterdef.txt after adding Lynx as a pre-hired MechWarrior

```
FITini

[General]
st PurchaseFile="solopurchase"
st GameStateName=(null)
l MissionNumber=00
f LastScenarioTime=0.000000
f LastLogisticsTime=0.000000

[Planet]
l Setting=0

[ResourcePoints]
ul numPoints=500000

[Warriors]
ul NumWarriors=1

[Warrior0]
ul PacketNum = 0

[AssWarriors]
ul NumAssWarriors=0

[Mechs]
ul NumMechs=1

[Mech0]
st Profile = "PM109400" // Atlas-W
l NumAvailable = 1

[AssMechs]
ul NumAssMechs=0

[Vehicles]
ul NumVehicles=0

[AssVehicles]
ul NumAssVehicles=0

[Components]
ul NumComponents=1
```

```
[Componant0]
// ER laser
uc ComponantID = 152
l NumAvailable = 4

FITend
```

File Listing of Lynx.txt

```
FITini

// ---------------------------------------------------------------------
//
//      MechWarrior:  66   Name: Lynx  Callsign: LYNX
//
//      Creation Date: 12/15/98   By: jon marcus
//
// ---------------------------------------------------------------------



[General]
st Brain = "pbrain"
st Name =                                                   "Lynx"
l Rank =                                                    1
st Callsign =                                               "LYNX"
ul ID =                                                     66
l NameIndex =                                               15
l DescIndex =                                               316
ul HeadIcon =                                               0
l paintScheme =                                             68
st pilotAudio =                                           "pilotP"
st pilotVideo =                                           "PILOT15"
st Picture =                                              "pilot15.TGA"
b Assigned=FALSE
b NotMineYet=FALSE


[PersonalityTraits]
c Professionalism =                                         50
c Decorum =                                                 50
c Aggressiveness =                                          50
c Courage =                                                 50

[Skills]
c Piloting =                                                50
c Jumping =                                                 45
c Sensors =                                                 50
c Gunnery =                                                 55

[OriginalSkills]
c Piloting =                                                50
c Jumping =                                                 45
c Sensors =                                                 50
```

```
c Gunnery =                                                    55

[LatestSkills]
c Piloting =                                                   50
c Jumping =                                                    45
c Sensors =                                                    50
c Gunnery =                                                    55


[Status]
c Wounds =                                                     0

FITEnd
```

File Listing of filelist.txt (after assigned mechs)

```
c masterdef.txt
c Lynx.txt
c Mystique.txt
c Burnout.txt
c Cougar-A.txt
c Catapult-W.txt
c initstore.txt
```

File Listing of masterdef.txt (after assigned mechs)

```
FITini

[General]
st PurchaseFile="solopurchase"
st GameStateName=(null)
l MissionNumber=00
f LastScenarioTime=0.000000
f LastLogisticsTime=0.000000

[Planet]
l Setting=0

[ResourcePoints]
ul numPoints=500000

[Warriors]
ul NumWarriors=1

[Warrior0]
ul PacketNum = 0

[AssWarriors]
ul NumAssWarriors=2
```

```
[Warrior1]
ul PacketNum = 1

[Warrior2]
ul PacketNum = 2

[Mechs]
ul NumMechs=1

[Mech0]
st Profile = "PM109400" // Atlas-W
l NumAvailable = 1

[AssMechs]
ul NumAssMechs=2

[Mech1]
ul PacketNum=3  // Cougar-A

[Mech2]
ul PacketNum=4  // Catapult-W


[Vehicles]
ul NumVehicles=0

[AssVehicles]
ul NumAssVehicles=0

[Components]
ul NumComponents=1

[Componant0]
// ER laser
uc ComponantID = 152
l NumAvailable = 4

FITend
```

File listing of Mystique.txt

```
FITini

//--------------------------------------------------------------------
//
//      MechWarrior:  57   Name: Mystique  Callsign: MYSTIQUE
//
//      Creation Date: 12/15/98   By: jon marcus
//
// -------------------------------------------------------------------
```

```
[General]
st Brain = "pbrain"
st Name =                                   "Mystique"
l Rank =                                     0
st Callsign =                               "MYSTIQUE"
ul ID =                                      57
l NameIndex =                                6
l DescIndex =                                307
ul HeadIcon =                                0
l paintScheme =                              68
st pilotAudio =                             "pilotg"
st pilotVideo =                             "PILOT06"
st Picture =                                "Pilot06.TGA"
b Assigned = TRUE
b NotMineYet = FALSE

[PersonalityTraits]
c Professionalism =                          50
c Decorum =                                  50
c Aggressiveness =                           50
c Courage =                                  50

[Skills]
c Piloting =                                 35
c Jumping =                                  50
c Sensors =                                  35
c Gunnery =                                  40

[OriginalSkills]
c Piloting =                                 35
c Jumping =                                  50
c Sensors =                                  35
c Gunnery =                                  40

[LatestSkills]
c Piloting =                                 35
c Jumping =                                  50
c Sensors =                                  35
c Gunnery =                                  40

[Status]
c Wounds =                                   0

FITEnd
```

File Listing of Burnout.txt

```
FITini

//---------------------------------------------------------------------
//
//      MechWarrior:  72   Name: Burnout  Callsign: BURNOUT
//
//      Creation Date: 12/15/98   By: jon marcus
//
// ---------------------------------------------------------------------

[General]
st Brain = "pbrain"
st Name =                                                   "Burnout"
l Rank =                                                    1
st Callsign =                                               "BURNOUT"
ul ID =                                                     72
l NameIndex =                                               21
l DescIndex =                                               322
ul HeadIcon =                                               0
l paintScheme =                                             68
st pilotAudio =                                            "pilotV"
st pilotVideo =                                            "PILOT21"
st Picture =                                               "pilot21.TGA"
b Assigned = TRUE
b NotMineYet = FALSE

[PersonalityTraits]
c Professionalism =                                         50
c Decorum =                                                 50
c Aggressiveness =                                          50
c Courage =                                                 50

[Skills]
c Piloting =                                                40
c Jumping =                                                 45
c Sensors =                                                 75
c Gunnery =                                                 40

[OriginalSkills]
c Piloting =                                                40
c Jumping =                                                 45
c Sensors =                                                 75
c Gunnery =                                                 40

[LatestSkills]
c Piloting =                                                40
c Jumping =                                                 45
c Sensors =                                                 75
c Gunnery =                                                 40

[Status]
c Wounds =                                                  0

FITEnd
```

File Listing of Cougar-A.txt

```
FITini

//--------------------------------------------------------------------
//
//      Mech Profile:  201100   'Cougar A'  Type:  5
//
//      Creation Date: 2/17/98      Creator: david abzug
//
//      Spec Mech (Don't mess with this one)
//
// -------------------------------------------------------------------

[Header]
st FileType = "MechProfile"

[General]
st MechType = "Cougar A"
b Clan = True
ul ID =                                                  201100
ul Chassis =                                             17
l DescIndex =                                            412
st Name =                                               "Cougar A"
l NameIndex =                                            5
l NameVariant =                                          0
f CurTonnage =                                           35
st icon =                                               "na.gif"
c Status =                                               0
l ResourcePoints =                                       3500
l ChassisBR =                                            35
b Assigned = TRUE
b NotMineYet = FALSE
l Pilot = 1
b Required = FALSE

[Engine]
f Tonnage =                                             -9.5
ul Rating =                                              175
uc MaxRunSpeed =                                         27
uc JumpRange =                                           0

[Armor]
uc Type =                                                0
f Tonnage =                                              8

[MaxArmorPoints]
uc Head =                                                10
uc CenterTorso =                                         22
uc LeftTorso =                                           19
uc RightTorso =                                          19
uc LeftArm =                                             16
uc RightArm =                                            16
uc LeftLeg =                                             17
uc RightLeg =                                            17
```

```
uc RearCenterTorso =                                    8
uc RearLeftTorso =                                      7
uc RearRightTorso =                                     7

[CurArmorPoints]
uc Head =                                               10
uc CenterTorso =                                        22
uc LeftTorso =                                          19
uc RightTorso =                                         19
uc LeftArm =                                            16
uc RightArm =                                           16
uc LeftLeg =                                            17
uc RightLeg =                                           17
uc RearCenterTorso =                                    8
uc RearLeftTorso =                                      7
uc RearRightTorso =                                     7

[InventoryInfo]
uc NumOther =                                           13
uc NumWeapons =                                         5
uc NumAmmo =                                            3

[Head]
uc CASE =                                               1
uc CurInternalStructure =                               3
uc HotSpotNumber =                                      14
uc[2] Component:0 =                                     5,0
uc[2] Component:1 =                                     6,0
uc[2] Component:2 =                                     12,0
uc[2] Component:3 =                                     255,0
uc[2] Component:4 =                                     255,0
uc[2] Component:5 =                                     255,0

[CenterTorso]
uc CASE =                                               1
uc CurInternalStructure =                               11
uc HotSpotNumber =                                      9
uc[2] Component:0 =                                     0,0
uc[2] Component:1 =                                     7,0
uc[2] Component:2 =                                     255,0
uc[2] Component:3 =                                     255,0
uc[2] Component:4 =                                     255,0
uc[2] Component:5 =                                     255,0
uc[2] Component:6 =                                     255,0
uc[2] Component:7 =                                     255,0
uc[2] Component:8 =                                     255,0
uc[2] Component:9 =                                     255,0
uc[2] Component:10 =                                    255,0
uc[2] Component:11 =                                    255,0

[LeftTorso]
uc CASE =                                               1
uc CurInternalStructure =                               8
uc HotSpotNumber =                                      6
uc[2] Component:0 =                                     14,0
uc[2] Component:1 =                                     15,0
uc[2] Component:2 =                                     18,0
```

```
uc[2] Component:3 =                                    19,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[RightTorso]
uc CASE =                                               1
uc CurInternalStructure =                               8
uc HotSpotNumber =                                      10
uc[2] Component:0 =                                    16,0
uc[2] Component:1 =                                    20,0
uc[2] Component:2 =                                    255,0
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[LeftArm]
uc CASE =                                               1
uc CurInternalStructure =                               6
uc HotSpotNumber =                                      3
uc[2] Component:0 =                                    8,0
uc[2] Component:1 =                                    1,0
uc[2] Component:2 =                                    1,0
uc[2] Component:3 =                                    1,0
uc[2] Component:4 =                                    13,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[RightArm]
uc CASE =                                               1
uc CurInternalStructure =                               6
uc HotSpotNumber =                                      0
uc[2] Component:0 =                                    9,0
uc[2] Component:1 =                                    2,0
uc[2] Component:2 =                                    2,0
uc[2] Component:3 =                                    2,0
uc[2] Component:4 =                                    17,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
```

```
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[LeftLeg]
uc CASE =                                               1
uc CurInternalStructure =                               8
uc HotSpotNumber =                                      8
uc[2] Component:0 =                                    10,0
uc[2] Component:1 =                                    3,0
uc[2] Component:2 =                                    3,0
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0

[RightLeg]
uc CASE =                                               1
uc CurInternalStructure =                               8
uc HotSpotNumber =                                      7
uc[2] Component:0 =                                    11,0
uc[2] Component:1 =                                    4,0
uc[2] Component:2 =                                    4,0
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0

// Inventory

// Other


[Item:0]
uc MasterID =                                           1  //Fusion
Engine

[Item:1]
uc MasterID =                                           4  //Arm
Actuator

[Item:2]
uc MasterID =                                           4  //Arm
Actuator

[Item:3]
uc MasterID =                                           5  //Leg
Actuator

[Item:4]
uc MasterID =                                           5  //Leg
Actuator

[Item:5]
uc MasterID =                                           6  //Cockpit
```

```
[Item:6]
uc MasterID =                                   16  //Basic
Clan Mech Sensor

[Item:7]
uc MasterID =                                   22
//Gyroscope II

[Item:8]
uc MasterID =                                   33
//Shoulder

[Item:9]
uc MasterID =                                   33
//Shoulder

[Item:10]
uc MasterID =                                   34  //Hip

[Item:11]
uc MasterID =                                   34  //Hip

[Item:12]
uc MasterID =                                   35
//LifeSupport

// Weapons


[Item:13]
uc MasterID =                                   151
//Large Pulse Laser
uc FacesForward =                               1

[Item:14]
uc MasterID =                                   130  //LRM/5
uc FacesForward =                               1

[Item:15]
uc MasterID =                                   130  //LRM/5
uc FacesForward =                               1

[Item:16]
uc MasterID =                                   130  //LRM/5
uc FacesForward =                               1

[Item:17]
uc MasterID =                                   155
//Flamer
uc FacesForward =                               1
```

```
// Ammo


[Item:18]
uc MasterID =                                          230  //LRM/5 Pack
l Amount =                                                  -1

[Item:19]
uc MasterID =                                          230  //LRM/5 Pack
l Amount =                                                  -1

[Item:20]
uc MasterID =                                          230  //LRM/5 Pack
l Amount =                                                  -1

FITEnd
```

File Listing of Catapult-W.txt

```
FITini

// ---------------------------------------------------------------------
//
//      Mech Profile:  106400   'CLPT-W'  Type:  2
//
//      Creation Date: 2/17/98      Creator: david abzug
//
//
// ---------------------------------------------------------------------

[Header]
st FileType = "MechProfile"

[General]
st MechType = "Catapult W"
b Clan = False
ul ID =                                                     106400
ul Chassis =                                                18
l DescIndex =                                               407
st Name =                                                  "CLPT-W"
l NameIndex =                                               2
l NameVariant =                                             1
f CurTonnage =                                              65
st icon =                                                  "Null"
c Status =                                                  0
l ResourcePoints =                                          6500
l ChassisBR =                                               65
b Assigned = TRUE
b NotMineYet = FALSE
l Pilot = 0
b Required = FALSE

[Engine]
f Tonnage =                                                 3.5
```

```
ul Rating =                                    260
uc MaxRunSpeed =                               18
uc JumpRange =                                 0

[Armor]
uc Type =                                      0
f Tonnage =                                    9

[MaxArmorPoints]
uc Head =                                      14
uc CenterTorso =                               25
uc LeftTorso =                                 22
uc RightTorso =                                22
uc LeftArm =                                   18
uc RightArm =                                  18
uc LeftLeg =                                   20
uc RightLeg =                                  20
uc RearCenterTorso =                           10
uc RearLeftTorso =                             8
uc RearRightTorso =                            8

[CurArmorPoints]
uc Head =                                      14
uc CenterTorso =                               25
uc LeftTorso =                                 22
uc RightTorso =                                22
uc LeftArm =                                   18
uc RightArm =                                  18
uc LeftLeg =                                   20
uc RightLeg =                                  20
uc RearCenterTorso =                           10
uc RearLeftTorso =                             8
uc RearRightTorso =                            8

[InventoryInfo]
uc NumOther =                                  13
uc NumWeapons =                                8
uc NumAmmo =                                   6

[Head]
uc CASE =                                      0
uc CurInternalStructure =                      3
uc HotSpotNumber =                             2
uc[2] Component:0 =                            5,0
uc[2] Component:1 =                            6,0
uc[2] Component:2 =                            12,0
uc[2] Component:3 =                            255,0
uc[2] Component:4 =                            255,0
uc[2] Component:5 =                            255,0

[CenterTorso]
uc CASE =                                      1
uc CurInternalStructure =                      21
uc HotSpotNumber =                             5
uc[2] Component:0 =                            0,0
uc[2] Component:1 =                            7,0
uc[2] Component:2 =                            255,0
```

```
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[LeftTorso]
uc CASE =                                               1
uc CurInternalStructure =                               15
uc HotSpotNumber =                                      4
uc[2] Component:0 =                                    13,0
uc[2] Component:1 =                                    14,0
uc[2] Component:2 =                                    15,0
uc[2] Component:3 =                                    21,0
uc[2] Component:4 =                                    22,0
uc[2] Component:5 =                                    23,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[RightTorso]
uc CASE =                                               1
uc CurInternalStructure =                               15
uc HotSpotNumber =                                      1
uc[2] Component:0 =                                    16,0
uc[2] Component:1 =                                    17,0
uc[2] Component:2 =                                    18,0
uc[2] Component:3 =                                    24,0
uc[2] Component:4 =                                    25,0
uc[2] Component:5 =                                    26,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[LeftArm]
uc CASE =                                               1
uc CurInternalStructure =                               10
uc HotSpotNumber =                                      3
uc[2] Component:0 =                                    8,0
uc[2] Component:1 =                                    1,0
uc[2] Component:2 =                                    1,0
uc[2] Component:3 =                                    1,0
uc[2] Component:4 =                                    19,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
```

```
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[RightArm]
uc CASE =                                               1
uc CurInternalStructure =                               10
uc HotSpotNumber =                                      0
uc[2] Component:0 =                                    9,0
uc[2] Component:1 =                                    2,0
uc[2] Component:2 =                                    2,0
uc[2] Component:3 =                                    2,0
uc[2] Component:4 =                                    20,0
uc[2] Component:5 =                                    255,0
uc[2] Component:6 =                                    255,0
uc[2] Component:7 =                                    255,0
uc[2] Component:8 =                                    255,0
uc[2] Component:9 =                                    255,0
uc[2] Component:10 =                                   255,0
uc[2] Component:11 =                                   255,0

[LeftLeg]
uc CASE =                                               1
uc CurInternalStructure =                               15
uc HotSpotNumber =                                      6
uc[2] Component:0 =                                    10,0
uc[2] Component:1 =                                    3,0
uc[2] Component:2 =                                    3,0
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0

[RightLeg]
uc CASE =                                               1
uc CurInternalStructure =                               15
uc HotSpotNumber =                                      7
uc[2] Component:0 =                                    11,0
uc[2] Component:1 =                                    4,0
uc[2] Component:2 =                                    4,0
uc[2] Component:3 =                                    255,0
uc[2] Component:4 =                                    255,0
uc[2] Component:5 =                                    255,0

// Inventory

// Other


[Item:0]
uc MasterID =                                       1  //Fusion Engine
```

```
[Item:1]
uc MasterID =                                   4  //Arm
Actuator

[Item:2]
uc MasterID =                                   4  //Arm
Actuator

[Item:3]
uc MasterID =                                   5  //Leg
Actuator

[Item:4]
uc MasterID =                                   5  //Leg
Actuator

[Item:5]
uc MasterID =                                   6  //Cockpit

[Item:6]
uc MasterID =                                   13  //Basic
IS Mech Sensor

[Item:7]
uc MasterID =                                   21
//Gyroscope III

[Item:8]
uc MasterID =                                   33
//Shoulder

[Item:9]
uc MasterID =                                   33
//Shoulder

[Item:10]
uc MasterID =                                   34  //Hip

[Item:11]
uc MasterID =                                   34  //Hip

[Item:12]
uc MasterID =                                   35
//LifeSupport

// Weapons


[Item:13]
uc MasterID =                                 120  //LRM/5
uc FacesForward =                               1
```

```
[Item:14]
uc MasterID =                                   120  //LRM/5
uc FacesForward =                                1

[Item:15]
uc MasterID =                                   120  //LRM/5
uc FacesForward =                                1

[Item:16]
uc MasterID =                                   120  //LRM/5
uc FacesForward =                                1

[Item:17]
uc MasterID =                                   120  //LRM/5
uc FacesForward =                                1

[Item:18]
uc MasterID =                                   120  //LRM/5
uc FacesForward =                                1

[Item:19]
uc MasterID =                                    142
//Large Pulse Laser
uc FacesForward =                                1

[Item:20]
uc MasterID =                                    143
//Medium Laser
uc FacesForward =                                1

// Ammo


[Item:21]
uc MasterID =                                   220  //LRM/5 Pack
l Amount =                                      -1

[Item:22]
uc MasterID =                                   220  //LRM/5 Pack
l Amount =                                      -1

[Item:23]
uc MasterID =                                   220  //LRM/5 Pack
l Amount =                                      -1

[Item:24]
uc MasterID =                                   220  //LRM/5 Pack
l Amount =                                      -1

[Item:25]
uc MasterID =                                   220  //LRM/5 Pack
l Amount =                                      -1
```

```
[Item:26]
uc MasterID =                                                220  //LRM/
5 Pack
l Amount =                                                  -1
FITEnd
```

File listing of filelist.txt (with assigned vehicles)

```
c masterdef.txt
c Lynx.txt
c Mystique.txt
c Burnout.txt
c Cougar-A.txt
c Catapult-W.txt
c Swiftwind.txt
c initstore.txt
```

File Listing of masterdef.txt (with assigned vehicles)

```
FITini

[General]
st PurchaseFile="solopurchase"
st GameStateName=(null)
l MissionNumber=00
f LastScenarioTime=0.000000
f LastLogisticsTime=0.000000

[Planet]
l Setting=1

[ResourcePoints]
ul numPoints=500000

[Warriors]
ul NumWarriors=1

[Warrior0]
ul PacketNum = 0

[AssWarriors]
ul NumAssWarriors=2

[Warrior1]
ul PacketNum = 1

[Warrior2]
ul PacketNum = 2

[Mechs]
ul NumMechs=1
```

```
[Mech0]
st Profile = "PM109400" // Atlas-W
l NumAvailable = 1;

[AssMechs]
ul NumAssMechs=2

[Mech1]
ul PacketNum=3  // Cougar-A

[Mech2]
ul PacketNum=4  // Catapult-W


[Vehicles]
ul NumVehicles=1

[Vehicle0]
st Profile = "PV20800"
l NumAvailable = 1

[AssVehicles]
ul NumAssVehicles=1

[Vehicle1]
ul PacketNum = 5

[Components]
ul NumComponents=1

[Componant0]
// ER laser
uc ComponantID = 152
l NumAvailable = 4

FITend
```

File Listing of Swiftwind.txt

```
FITini

//---------------------------------------------------------------------
//
//      Ground Vehicle Profile:  20000   'Swiftwind'  Type:  426
//
//      Creation Date: 5/14/98      Creator: Mike Lee
//
//
//---------------------------------------------------------------------

[Header]
st FileType = "GroundVehicleProfile"
```

```
[General]
st Crew =                                                    "PCREWA"
ul Chassis =                                                  426
st Name =                                                    "Swiftwind"
l NameIndex =                                                 20
l DescIndex =                                                 517
f CurTonnage =                                                5
st icon =                                                    "na.gif"
c Status =                                                    0
b Assigned = TRUE
b NotMineYet = FALSE
b Deployed = FALSE
b Required = FALSE


[Engine]
f Tonnage =                                                   1
ul Rating =                                                   1
uc MaxMoveSpeed =                                             21

[Armor]
uc Type =                                                     1
f Tonnage =                                                   1.5

[Front]
uc CurInternalStructure =                                     1
uc MaxArmorPoints =                                           6
uc CurArmorPoints =                                           6

[Left]
uc CurInternalStructure =                                     1
uc MaxArmorPoints =                                           5
uc CurArmorPoints =                                           5

[Right]
uc CurInternalStructure =                                     1
uc MaxArmorPoints =                                           5
uc CurArmorPoints =                                           5

[Rear]
uc CurInternalStructure =                                     1
uc MaxArmorPoints =                                           4
uc CurArmorPoints =                                           4

[Turret]
uc CurInternalStructure =                                     1
uc MaxArmorPoints =                                           5
uc CurArmorPoints =                                           5

[InventoryInfo]
uc NumOther =                                                 4
uc NumWeapons =                                               0
uc NumAmmo =                                                  0

// Inventory
```

```
// Other

[Item:0]
uc MasterID =                                                   1

[Item:1]
uc MasterID =                                                   6

[Item:2]
uc MasterID =                                                   35

[Item:3]
uc MasterID =                                                   18

FITEnd
```

Appendix 6: The Data Tables

Game State Table:

START_GAME	0
MAIN_SCREEN	1
START_CAMPAIGN	2
LOGISTICS_SCREEN	3
BRIEFING_SCREEN	4
GENERAL_ORDERS_SCREEN	5
SCENARIO_RESULTS_SCREEN	6
IN_SCENARIO	7
IN_SMACKER_MOVIE	8
NEXT_SEGMENT	9
START_SMACKER_MOVIE	10
START_SCENARIO	11
EXIT_GAME	16

Player Hire-able MechWarriors Table:

51	Beast	76	Blaze
52	Hawk	77	Thunder
53	Siren	78	Paingod
54	Rooster	79	Firestorm
55	Isis	80	Vixen
56	Gunman	86	Siren (Ex)
57	Mystique	87	Rooster (Ex)
58	Skater	88	Isis (Ex)
59	Spice	89	Gunman (Ex)
60	Baron	90	Skater (Ex)
61	Fiend	91	Countess (Ex)
62	Ronin	92	Scarab (Ex)
63	Gator	93	Mantis (Ex)
64	Rebel	94	Dragon (Ex)
65	Hunter	95	Blaze (Ex)
66	Lynx	96	Thunder (Ex)
67	Hitman	97	Goblin (Ex)
68	Countess	102	Spice (Ex)
69	Scarab	103	Baron (Ex)
70	Outlaw	104	Fiend (Ex)
71	Falcon	105	Ronin (Ex)
72	Burnout	106	Gator (Ex)
73	Goblin	107	Rebel (Ex)
74	Mantis	108	Hitman (Ex)
75	Dragon	109	Burnout (Ex)

MechWarrior Store Hiring Status Table:

0	Available for hire
1	Hired
2	Dead
3	Dismissed (sold back)
4	Unseen

Component ID Table:

ID	Component Name
13	Sensor (Basic) (IS)
14	Sensor (Intermediate) (IS)
15	Sensor (Advanced) (IS)
16	Sensor - Basic (Clan)
17	Sensor - Advanced (Clan)
37	Beagle Probe (IS)
38	Guardian ECM (IS)
42	ECM Suite (Clan)
43	Active Probe (Clan)
98	Rail Gun
99	Light Gauss (IS)
100	Light Autocannon (IS)
101	Autocannon (IS)
102	Heavy Autocannon (IS)
103	Light Ultra Autocannon (IS)
104	Gauss Rifle (IS)
107	Light LBX AC (IS)
108	LBX AC (IS)
109	Hvy LBX AC (IS)
110	Ultra AC/5 (Clan)
111	Ultra AC/10 (Clan)
112	Ultra AC/20 (Clan)
113	Gauss Rifle (Clan)
116	Clan Lt LBX (Clan)
117	Clan LBX (Clan)
118	Clan Hvy LBX (Clan)

ID	Component Name
120	Long-Range Missile Rack (IS)
123	SRM Pack (IS)
125	Streak SRM Pack (IS)
130	LRM/5 (Clan)
133	SRM/2 (Clan)
135	Streak SRM/2 (Clan)
139	X-Pulse Laser (IS)
140	Large Laser (IS)
141	Large ER Laser (IS)
142	Large Pulse Laser (IS)
143	Laser (IS)
144	Pulse Laser (IS)
145	Particle Projector Cannon (IS)
146	ER Particle Projector Cannon (IS)
147	Heavy Flamer (IS)
150	Large ER laser (Clan)
151	Large PS laser (Clan)
152	ER laser (Clan)
153	PS laser (Clan)
154	ER PPC (Clan)
155	Flamer (Clan)
160	Long Tom Cannon(IS)

Mech ID Code Table:

	IS Mechs		Clan Mechs
100	Commando	200	Uller
101	Firestarter	201	Cougar
102	Raven	203	Hunchback IIc
103	Hollander II	204	Vulture
104	Hunchback	205	Loki
105	Centurion	206	Thor
106	Catapult	207	Mad Cat
107	JagerMech	208	Masakari
108	Awesome	209	Shadow Cat
109	Atlas	210	Nova Cat
110	Fire Falcon	211	Turkina
111	Bushwacker		
112	Mauler		

Mech Types Table:

	Mech Type
1	Armor (A)
2	**invalid**
3	Jump Jet (J)
4	Weapons (W)

Defined Mech Variants Table:

Mech & Type	Variants
Awesome-A	1
Atlas-A	50
Uller-A	50, 60, 70
Cougar-J	50
Vulture-A	1
Vulture-J	50
Thor-W	1
Mad Cat-J	50, 60
Masakari-A	50
Masakari-J	50
Shadow Cat-A	50
Shadow Cat-W	50
Turkina-J	50
Turkina-W	50

Vehicle IDs and Variants Table:

	Vehicle Name	**Var.**	**Variant Name**
200	Switfwind		
201	Rommel		
202	MobileHQ		
203	Fuel Truck		
204	Elem. Carrier		
205	Harasser		
206	APC	50	APC
208	Bulldog		
209	SRM Carrier		
210	LRM Carrier		
211	Mobile Artillery	50	Mobile Artillery
212	Artillery Loader		
213	Shrek		
214	Saracen		
215	Towed Turret		
216	AeroSpace Spotter		
217	Striker		
218	Pegasus		
219	J. Edgar		
220	VonLuckner		
221	Condor		
222	Ordnance Truck		
224	Salvage Rig		
227	Light Transport	01	Armored Car
228	Mine Sweeper		
229	Mine Layer		
304	Semi Tractor		
305	Flatbed	01	Armored Flatbed
306	Farm Truck	01	Armed Farm Truck
307	Car		
308	Sports Car		
309	Sports Car		
310	Car		
311	Hover Limo		
312	Limo		
313	Sports Car		
314	Car		
315	Car		
316	Sports Car		
317	Car		
318	Car		
319	Savannah Master		
320	Sports Car		

	Vehicle Name	Var.	Variant Name
321	Salvage Truck		
322	Refit Truck		
323	Refit Truck		
324	Ambulance		
333	Centipede		
334	Regulator		
335	Manticore		
336	Pilum		
337	Alacorn		
338	Ammo Truck		

Briefing Text Commands Table:

Code	Description	Example Use
`%c`	Center text line	
`%n`	New line	
`%%`	Percent ('%') character	
`%fc#`	Select font color (1-6)	`%fc2`
`%t##`	Tab over ## pixels	`%t12  %t03`
`%fs`	Select small font	
`%fm`	Select medium font	
`%fl`	Select large font	

Briefing Text Usable Colors Table:

1	Red
2	Yellow
3	Green (bluish green really)
4	Cyan
5	Dark Grey
6	White