ADI thanks Guy Lavoie for creating this very helpful FAQ document.

C-Max 2.0 FAQ

Version 1.5

Important: To get started using C-Max 2.0, please see Q15 and A15 in the text below.

Q1: What is the System Map and how do I use it?

A1: The System Map is one of the major new features in C-Max 2.0 . A bit of time invested in creating and maintaining it will make programming much easier, and your programs will be almost self-documenting. Start by creating a new Project (see question about projects below) and then create your System Map as follows: Click on Project → System Map. You will see a new window with an expandable list starting with "Ocelot" at the top. If your controller is a Leopard, right click on the word "Ocelot" and choose the right controller type (once you have chosen Leopard, you cannot set it back to Ocelot). Then click on each module address that you have in your system and once again, right click to choose the type of module at that address. Note the Bobcat types: Bobcat-T = temperature, Bobcat-H = humidity, Bobcat-A = ascii(serial), etc. Now, you will notice that to the left of each module that you add (and also under the master controller on the line labeled "Local") there is a small box with a "+" sign. Clicking on a "+" will expand that module's list of resources, like i/o ports, variables, timers, touch buttons, etc. You can now edit each of those resource's labels and give them meaningful names. If X-10 address B/9 is the hall light, then you can enter "Hall Light". The process of editing resource names is similar to editing file names in Windows; click once to select the item, a second time to highlight the whole name, and optionally a third time to position the cursor within the existing name. Once you are done, choose Project → Save Project and this will save your newly created map. You will now see those names right in the program text! If you already had a program loaded in the editor, you will need to reload the program to see the new names. You will also see those names in the appropriate list boxes when adding/editing lines.

Q2a: When I load a program created with a previous C-Max version, there are strange program lines like "IF 1, Button Ocelot is pressed"...

or

Q2b: I start up C-Max and I want to create a new program, but I cannot create certain types of program lines (like for I/O modules) because there is no button at the bottom of the control wizard to enter it.

A2: This will happen if you load a program containing lines referring to expansion modules or Leopard touch buttons without having first defined your System Map, or have incorrect module-type entries. Create and save the System Map as explained in the previous question above. Once finished, reload your program and all the lines should appear normally. Whenever you are asked to look at or work on a C-Max program written by someone else, make sure you ask for the .prj file as well as the already familiar .tch and .pgm files.

Q3: After defining my controller as a Leopard, I look at the System Map and in addition to all the familiar object types, I also see "icons". What are these?

A3: This relates to the new C-Max 2.0 feature allowing you to display graphical icons (that you can design yourself) on Leopard buttons To learn how to design and load icons, refer to the application note describing the process in detail.

Q4: What exactly are projects?

A4: C-Max 2.0 introduces the concept of "projects" which aims to make file management easier by keeping the files for a given controller (the "pgm", "tch", and any "bmp" files for icons) together in a directory. This allows a user to load and save all these related files together, saving time.

Create your project by clicking on Project → New Project. Browse to a directory or create a new one (recommended) where you want to keep the files associated with this project. It is best that you create a seperate directory for each project (using the "new directory" tool in the file browser if necessay when you click on "New Project" to first save your project). The first thing you should do is create the System Map (see question above) and then save your project (click on Project →Save Project). Now create a program or load an existing one, browsing to older, existing directories if necessary. Do the same to load or create a touchscreen file. Now, all you have to do to save everything is click on Project --> Save Project at any time and all the files loaded or created will now be saved in the same directory as where you created the new project, and the pgm/tch files will now be given the same name as the one you named the project earlier. Any icon bmp files referenced in the System Map will also be copied there. You can load a complete project in the same manner by clicking on "File" --> "Open Project" , browsing to your project's directory and selecting the appropriate "prj" file. No need to browse and load your program and touchscreen files separately any more. Note that you can still save program and touchscreen files individually if you want, useful for things like creating templates that will be used in several projects. The prj file itself contains the System Map definitions.

Q5: There is a new “skip to line …” instruction. How exactly does it work?

A5a: This is another major new feature of C-Max 2.0 . It tells the program to skip ahead to the designated line if the THEN or ELSE line it’s on tests true. You can effectively create nested If/then logic with this command because any intervening lines are literally skipped over. Judicious use of this new capability can save you many AND statements and make your program more logical in appearance and flow, and execution time is also improved. Consider the case where you have a dozen or more “IF Time of Day” statements that you want to execute only if the alarm system is armed (to simulate an occupied house). Instead of adding an AND statement to look for the alarm being armed along with each time of day, look for the armed status first and if not armed, skip over all the statements looking for the time of day that are related to this occupation simulation routine. You will notice that you cannot jump backwards in your program. This would be contrary to the ladder logic “looping” programming model and could even get the program stuck in a tight loop. Remember that this is not meant as a “goto” statement to go and do things, but as a statement skipping function to get on with the task of looping as quickly as possible. This is also useful for resolving “order sensitive” code issues, like trying to create a touch button that toggles a variable between two states:

```
0001 - IF  Touch Object #1, Button  Leopard is pressed   //
0002 -   AND Variable #0 is = 0                          //
0003 -     THEN Variable #0 = 1                          //
0004 -     THEN Skip to line 8                           //
0005 - IF  Touch Object #1, Button  Leopard is pressed   //
0006 -   AND Variable #0 is = 1                          //
0007 -     THEN Variable #0 = 0                          //
0008 . . .
```

A5b: The “Skip to” command displays and accepts the line number where it will jump to if true, but internally it stores the destination line as relative offset from the current line. For example, if line# 10 specifys a skip to line #18, the actual code tells it to skip “8 lines ahead”. This has the desirable effect that if you cut/copy a program segment with such a statement (or import one as a saved snippet), the offset will be maintained and the new, calculated destination line will be displayed in the editor. With the example just mentioned, pasting that line on line 24 will show it as skipping to line 32. One thing to watch out for is if

you cut/copy or create a snippet containing a “skip to” statement where the destination line is beyond the end of the selected segment. When you paste such a program segment, the offset still be maintained but the line it happens to target for the skip may not be what you want in it’s new location, and could cause unexpected behavior. It is a good precaution to verify the destination lines of any “skip to” lines imported or moved and make sure that they point to proper program lines, and correct this if necessary.

Q6: When I insert or delete a line, nothing seems to happen for several seconds.

A6: When you click on Insert or Delete, watch the “Update” button in the lower left corner of the program window. You will see a rapidly incrementing number indicating the line number that the editor is currently moving to it’s new location. The updating will always start from line 1 because any insertions or deletions could affect a “skip to line” instruction if the line being inserted or deleted is between the “skip to” instruction and the line number it would skip to if true. In other words, C-Max diligently keeps track of the destination lines of all your “skip to” instructions and automatically adjusts for any lines added or removed.

Q7: The editor now shows 4096 lines instead of 2048. Can I really use that many? Is it just for the new Leopard II or also for my existing Leopard I or Ocelot?

A7: The answer is yes, you can use all those lines and yes, also on your existing Leopard I or Ocelot as well! The previous 2048 line limit was not a physical memory limitation but a performance one. It was determined that with a full 2048 lines that need to be interpreted, it takes about a second to loop through an entire C-Max program, which corresponds to the timer’s resolution (what use would be a one second resolution if you can only check it every two seconds?). The new “skip to line” instruction will allow a considerable execution time savings because the interpreter actually skips over the program lines, not spending any time on them. It is therefore assumed that large programs will make frequent use of the new skipping capability and therefore allows more lines to be present in the total program.

Q8: There is a new instruction called “Set Displayed Icon” under the THEN/ELSE statements. What does this do?

A8: This new command allows you to change the icon displayed on a Leopard button under program control. With this you can do many interesting things like simulate the action of a bar graph, guage, dial, or anything else that you wish based on a variable value, time of day, or any other condition. You could also create icons with text on them and appear to have button captions change depending on user input or selections. See the application note on icons for details on how to use this.

Q9: There is a new math operator labeled as “%”. What does it do?

A9: This is the new “modulo” function. Modulo corresponds to the remainder after an integer division. For example, 23 % 3 = 2 because 23 divided by 3 = 7 with a remainder of 2. This can be useful for a number of things, like number base conversions. For example, you want to log the time that an event occurs so you always want the current hour and minute in a variable. Capturing the Time of Day in a variable gives you the number of minutes since midnight, but you want an hours and minutes value. You could use the following code segment to do this (final value is in Variable #1):

```
0001 - IF Time of Day is > 00:00              //  look at Time of Day
0002 -     THEN Load Data to:  Variable #0    //  and capture in Var# 0
0003 -     THEN Variable #1 = Variable #0     //  and also copy to Var# 1
0004 -     THEN Variable #1 / 60              //  calculate hours
0005 -     THEN Variable #1 * 100             //  convert hours to HHxx
0006 -     THEN Variable #0 % 60              //  calculate “minutes” value
```

```
0007 -      THEN Variable #1 + Variable #0      // add to hours: HHMM
0008 -      ELSE Variable #1 = 0                // if midnight, then time = 0
```

Q10: Do I have to modify anything in my programs created with a version previous to C-Max 2.0?

A10a: Yes, there are a couple of things that might require attention. If you have a Leopard program that looks for button 255 being pressed (the "touch anywhere" button) this has been moved to become button #0, modify your program accordingly.

A10b: If you display a bitmap on your Leopard's screen 0 to show on powerup or when the unit is idle, this capability has been merged with the new icon display facility, and you can now use a large icon instead to accomplish this. This will be icon # 0. See the application note on icons for more details.

A10c: If you have a screen text object on a Leopard screen with an embedded variable, this capability has been enhanced to allow formatting of the displayed variable in several ways. If you are familiar with the "printf" statement of the "C" language, will find the format string options very familiar. An application note describing these options is available. If you do not need the new formatting options or plan on making changes only later, simply add a "u" following the number after the percent sign (ex: edit a "%4" to become "%4u") to get the same behavior as before. Failing to do this will result in a blank variable field being displayed. If you've previously wanted to be able to display negative numbers, now you can by simply using a "d" instead of the "u". With a "u", a variable containing 65535 will now display a nice "-1". The variable formatting application note explains all the options available. Also see A12j below.

A10d: The internal command encoding has not only been added for the new commands, but some of the commands that already existed in versions prior to C-Max 2.0 are now coded differently and optimized. This conversion is done automatically for you when you load a pre-2.0 program. This means that a program created or edited with C-Max 2.0 is not backward compatible with previous versions, even if none of the new features are used. Just doing something as simple as loading a pre-2.0 program and immediately saving it without editing a single line will make that program incompatible with a pre-2.0 version of C-Max.

This also means that any programs that were created using an alternate 3rd party compiler will at least need to be loaded into the C-Max 2.0 program editor and saved again to be converted to the new C-Max 2.0 encoding format. It is up to the authors of such alternate compilers to revise and update their code to produce compatible program files and take advantage of the new capabilities in C-Max 2.0. ADI cannot make any commitment as to their compatibility or suitability.

Q11: How do I use the "Find" button in the program editor ? I click on it and just get an error message saying "Not Found"…

A11: Begin by doing a right click on the "Find" button and the shadow under the button will expand to a text field where you can enter the text you are looking for (neat trick huh!). Enter the text string (it is not case sensitive) to be searched and also check the "include comments" box if you want to search the comments column as well. Then click on the "Find" button and the next line containing the search string will be highlighted every time the button is pressed. Note that the search will always begin from the currently selected program line (just click once on the desired start program line) downwards. You must re-select the start line if you want to do a new search. Finally, do another right click on the "Find" button to re-hide the search string.

Q12: Are there other new features, fixes or time saving tricks that I should know about?

A12a: - Making program listings for text copying and pasting has just become much easier! Just click on: File → Print to File and you will have a nicely formatted listing of your program complete with the

comments column and the line numbers all neatly lined up. The code samples in this FAQ were created with this new feature.

A12b: -You can now capture the current month into a variable with the “load data to variable” command. This had not been implemented in C-Max 1.70

```
0001 - IF  Month is > January                  // Compare month
0002 -     THEN  Load Data to:  Variable #0    // capture if > January
0003 -     ELSE  Load Data to:  Variable #0    // + capture if = January
```

A12c: - The “IF Weekday…” command has been fixed. In C-Max 1.70, the editor would not allow you to compare the weekday against any variable # higher then 6. The rules for the variable contents (0 to 6) were also being enforced for the variable number itself.

A12d:- Inserting or deleting program lines always causes the entire program to update all it’s lines, which can become tedious to wait for if you want to insert or delete several lines. If you know approximately how many lines you want to insert, you can just highlight (by clicking and dragging your mouse) an equivalent number of existing lines, or even blank lines past the end of your program, and then right click and “copy”. You then select the line over where you want to insert the new lines and again right click and “paste”. Inserting all those lines will take the same time as inserting just one. Now edit these lines to what you want them to be. You can delete a group of lines in a similar manner by selecting the lines in question and then right clicking and selecting “cut”…nothing says you have to paste them somewhere else after!

A12e: - When accessing the Controller Access screen to look at the status window or X-10 event window (under Monitor X-10), you will see that the messages now appear at the top of the window and any previous messages are now pushed downward. This avoids needing to use the scroll bar to see any new messages once the window gets full.

A12f: - You can now trap I/O Errors (if the master can no longer access an expansion module that was present when it was powered up) and know what the Adnet address of the non-responding module is. You could then log the error through the serial port or display an error message on the screen if you have a Leopard, or take any other action. Coding the trapping action is as simple as:

```
0001 - IF  I/O Error Occurs                    //if I/O error
0002 -     THEN  Load Data to:  Variable #0    //put Adnet mod# in V 0
```

You also need to modify parameter 9 in your master to specify how many consecutive unsuccessful attempts must be made before the error condition is trapped (a value between 3 and 5 should be good). Note that if the module reappears on the bus following such an error, it will then again be accessed, without the master needing to be rebooted. You may also want to adjust parameter 8 to reduce the time wasted waiting for an unresponsive module, a value between 5 and 10 for this parameter should be sufficient. Lastly, if several modules become inaccessible at one time (due for example, to a broken bus wire between the master and a string of modules) then the highest Adnet module address of the missing units will be the one reported.

A12g: - A new “bus” command allows the master to directly command a slave to transmit an ASCII string just as if it was a serial bobcat. This can be handy if you have slave Leopards or Ocelots where the serial port is generally not being used other then when a program is being loaded into the unit. To use this, make sure that both the master and any slave(s) are running the new executive that is part of C-Max 2.0 . Note that unlike a serial bobcat, you cannot load the ASCII strings into a slave controller over the bus from the master; this must be done using the slave’s own serial port.

A12h: - You can now obtain the sunrise and sunset times for the current day (as of midnight) and store them into variables. This has been implemented as a special case of using the “Load Data to Variable” instruction to capture the result of an “IF Time of Day…” instruction. If your “IF Time of Day…” instruction compares the current time against a constant or a variable then it works as before and capturing

the result will get you the current time in minutes since midnight. If however you compare the time against sunrise or sunset (by selecting the “Sunrise / Sunset” tab in the control wizard) then using “Load Data to Variable” will capture the sunrise or sunset time instead. Like when capturing the current time, this value will be in minutes after midnight, but you can easily use the routine shown in A9 above to convert it to a HHMM format. You can then display this value on your Leopard screen or use it for other calculations as you desire.

A12i: The control wizard attempts to be as context sensitive as possible. For example, the inputs of a SECU16 or 16I can be read as being on/off or as analog values and both types of usage are listed for possible custom naming in the System Map. On/off mode items are listed as “Input #” while the corresponding analog mode inputs are listed as “Analog #”. In the command wizard an “IF Module/Point…” command will only let you choose among the “input #” while the “IF Module/Parameter…” commands will confine you to the “Analog #” choices.

A12j: The new formatted variable capability described in A10c above is also available for ASCII strings transmitted by the controller’s serial port and also with an ASCII (serial) bobcat. This is a big improvement over ASCII strings in previous versions which did not even offer embedded variables of any kind. Sending strings containing values like time, date, temperatures, etc. is now possible. Note that an ASCII bobcat must have a firmware version of 6 or higher to support the embedded variable capability. Detailed instructions on how to use embedded variables in ASCII strings are shown in the Formatted variables application note.

Q13: Are there still any outstanding issues in regards to the beta versions?

A13: As listed below:

A13a- “Mouse over” of named items in the System Map will show the generic name for that item. This is already done for X-10 items and will be done for the others.

Q14: When I program a “Transmit X-10…” command, I cannot select an X-10 device I created in the System Map, I have to choose the House and Unit/Command codes manually.

A14: That is the intended functionality. The vast majority of X-10 commands are used to turn devices on and off and the “X-10 Quick ON/OFF…” is best suited for this. The “Transmit X-10” command allows you to send “raw”, single X-10 commands that are not necessarily related to your regular devices. For example, the original "Preset Dim" command can be sent whereby you would then follow that command with a house code/unit code pair that is actually selecting one of 256 dim levels. This is used by some models of the RCS thermostats.

Q15: How do I go about upgrading my current C-Max 1.x software to C-Max 2.0 ?

A15: No matter how you obtained C-Max 2.0 (Internet download or hard media), follow the instructions given to install the software. For the downloaded version, you will get a single large file that will start a self-contained decompression utility when executed. It will default to decompress in C:\temp but you may specify another directory if desired. Once decompressed, go to that directory and run the “Setup” program. The installation program will overwrite any existing installation of C-Max. Any programs and touchscreen files you may already have will be safely preserved. If for whatever reason you wish to keep a copy of your previous C-Max version and any programs or touchscreen files created by it (because once modified with version 2.0, they will no longer be backward compatible, see A10d above), then copy the entire contents of the “C:\Program Files\ADI\Adicon2500” directory (or wherever you had installed it) to another directory before installing C-Max 2.0 .

Note that some things like the install path are saved in the Windows Registry, so if you decide to install the new version in a different directory then the old one, the Registry settings will now point to the new directory. This can cause unexpected behavior if you then attempt to use the old version. For example, starting up the old version and reloading the executive will locate the flash512.bin file using the path of the new version (ie: you will be reloading the new executive version instead of the old one…).

The first thing that you must do once C-Max 2.0 is installed is to reload the executive, to make your controller compatible with the new instructions available in version 2.0 . If you obtained your Leopard or Ocelot with version 1.70 of C-Max, you may have never done this procedure before. Here is a quick explanation of the purpose of the executive and why it needs to be reloaded: The C-Max program editor produces a compressed version of your program, which is then downloaded into the controller. The controller then runs it's own command interpreter to execute the commands in your program. Any new commands offered in a newer version of C-Max needs to have the equivalent new commands added to the internal interpreter so that it will know how to execute them. Reloading the executive thus equates to loading the new command interpreter into your controller.

To reload the executive, follow these steps: Start up C-Max 2.0 and click on Comms → Comms Setup to verify that the correct serial port is selected to communicate with your controller. Also verify that the "Programming Mode" radio button (at the bottom of the screen) specifies "PC Programming". With that done, click on "OK" and after the window closes, click on Comms → Attach to Controller. Once on the Controller Access screen, click on Controller Utility → Reload Controller Executive to start the executive reload process. You will see a progress bar near the bottom of the Controller Access screen showing the reload's progress as well as status messages in the status window (left side of Controller Access screen). When finished, you will get a message window telling you to reload your program. You are now ready to use C-Max 2.0. The first thing you should do at this point is read questions 1 through 4 of this FAQ to create your System Map and turn your existing program and any touchscreen files into a project. Also read question 10 to see if any adjustments need to be made in your existing program(s) before they are downloaded into the controller.

Q16: What does the word "Controller" that I see here and there exactly mean?

A16: The first ADI product that was used with C-Max was called the "CPUXA" and this is the term that was used to designate the device in all versions of C-Max prior to 2.0 . Since that time the Ocelot and two versions of the Leopard have been introduced that also use C-Max, so the more generic term "controller" has replaced "cpuxa" to refer to all of these…controllers!

Q17: I edited or created an "IF Variable…" program line and scrolled quickly down the variable list box to get to one of the higher numbered variables, but the line now shows "IF Data for Module…." ?

A17: If you scrolled very quickly down the list of variables, you may not have noticed that there are actually two series of variables; the conventional variables numbered 0 to 127 followed by a "Data for Module" series also numbered from 0 to 127. This second series gives you a way to access a bobcat's raw data value easily by just specifying it's Adnet module address as an extended variable. You can either directly compare against this value or save it in a conventional variable by using the "Load Data to Variable…" instruction.

Q18: Every time I want to start a new project, I need to redefine the System Map. Is there a way to have a default System Map that I could then use as a template?

A18: There is no "default" map as such but you can easily implement an equivalent strategy. Start by first defining a System Map with all the definitions that you want to reuse from one project to the other and then save it as a project with a name like "default" or other obvious term. You can then load this project every time you start a new project and then use the "Save As" menu choice to create a new project. It is a good

idea to do the “save as” right after starting the new project to avoid overwriting the default project by mistake. Another advantage to doing it this way as opposed to having a hardcoded default project is that you might be maintaining several controller installations and each one can have it’s own “default” Map.