Extended Basic Tools (EBT)

User Manual

Version 1.0 22 Aug 2020

(Record of Revisions located in Appendix 1)

Introduction to EBT

Extended Basic Tools (EBT) is a set of extensions for the Coridium BASICtools IDE which Coridium Corporation provides for free when used with any of the Coridium SBC Boards. EBT is provided for free and relies on the existence of BASICtools. Without BASICtools EBT is, by design, materially non-functional. The scope of this document is related to EBT and it is not intended to be a User Manual for Coridium's BASICtools IDE (BT). This work assumes that one has a basic to good understanding of BT and how to use it.

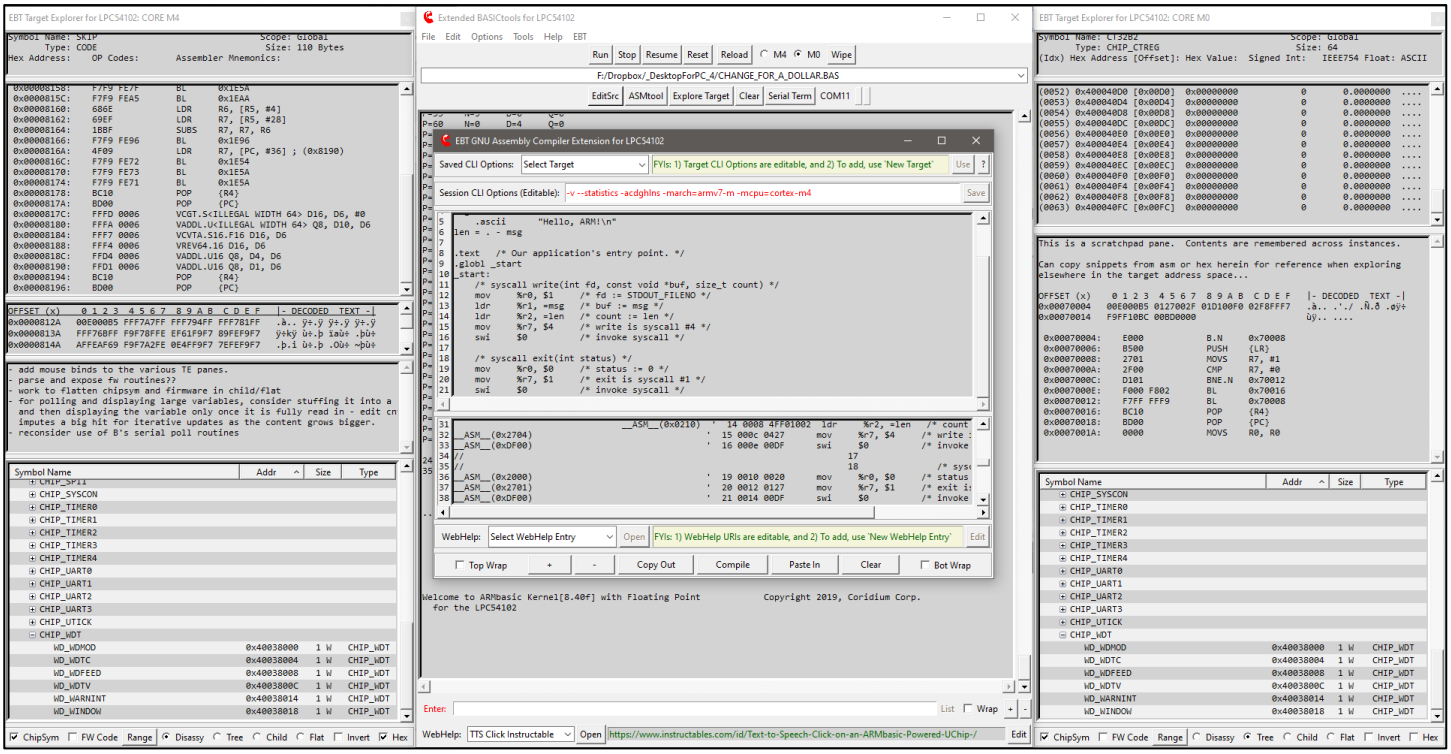


Figure 1: Extended BASICtools with Dual Target Explorers (one instance per core) and ASMtool Modules activated

Background/History/Inspiration/Dev Intro (a single-page wall of text – feel free to skip it…)

Good day. My name is Tod Wulff, an aerospace & defense professional that is also a bit of a geek at heart.

Hailing from the era of dial-up BBS, 8-bit Microcontrollers, Kaypro/Commodore/Tandy/TI-994A personal computers, when Radio Shack stores were plentiful (the good ole days), one of my first hobby embedded projects was working with a MEK6800D2 Motorola Microprocessor Training Kit, which I had purchased while working as a co-op at the MSU EE labs (after completing my High School Electronics VoTech training in Southern Lower MI). That project involved my prototyping the Radio Shack SP0256 NARRATOR™ SPEECH PROCESSOR onto the MEK6800D2, wiring it up and programming the 6800 to get it to emit pseudo-speech (those who've worked with the SP0256 based HW know exactly what I am alluding to). It worked wonderfully and I progressed down the path of cutting my teeth in embedded microcontrollers and Assembly. After High School, life got in the way, Military, War, Spouses, Children, entering the civil sector, starting a career, etc. all added up to my shelving my hobby in favor of pursuing the endeavors of life in a Western culture (here in the US).

Skip forward 20 years, coming to the inevitable time where the kids are maturing to the point that the Bride and I are severely distracting to their social status (and life in general), the mortgage, vehicles, college bills are slowly being paid down, earnings getting better with advancements, and my having enough spare time to the point where I could start to refocus on some selfish interests, I picked back up on the hobby electronics gig. Anyways, given my lineage and history, I sought out and found a dev environ that I quickly bonded with - ARMbasic - BASIC was my first love and this fit the bill of not only reacclimating myself to programming, but working with hardware that was vastly more powerful than what I had started with decades earlier, and thus the journey began. Having done some windows scripting for a couple years prior, using AutoHotKey (AHK), I decided to try to integrate a home-brewed AHK-based IDE, intended for use with early Coridium goods, into Notepad++, my editor of choice back then (remaining so thus far). That endeavor was only partially successful.

This was circa 2006-2009. Then, for reasons well beyond our control, life changed (as it had for many during those years – housing crisis). Hobbies shelved - focus on a new career, recovering from financial struggles (was heavily vested in the real-estate domain and we took it in the shorts and the youngins were just getting to the point where College funding was an imperative). Basically, Life and First-World problems (we are really blessed, considering the challenges and toils that folks in other parts of the world struggle with on a daily basis) manifested themselves and ... the hobby got shelved. I picked back up briefly on it in 2011-2012-ish then was met with another career change - hobby shelved yet again.

Fast forward another decade and ... I am back and, Good Lord Willing, hopefully for the duration (until I take that proverbial dirt nap and start pushing daisies up from below). So, here we are. Wow - Arduino (what is that weird word?) had stormed the market. Makers?? What the heck are they?! ... :) My friends at Coridium Corp (proprietors of ARMbasic and ARM-based microcontroller dev boards) had remained steadfast and true. Now, instead of the LPC2xxx series of controllers, there is this new (to me) entity of ARM, and Cortex M0/M3/M4, and Arduino, and ... WOW! The culture has morphed quite a bit, and in many a great way. Peeps are collaborating remotely and, indeed, globally. Hardware is getting amazingly fast and powerful, and ARMbasic, having matured and steadfastly hardened with employment across many different families of silicon, is a thing of beauty for me and many others.

EBT is, comparatively, somewhat recently birthed, with initial code being drafted in November 2018, and a mature alpha coming into existence by February 2019. Further development to EBT was sporadic until July 2019 when I was tasked professionally on a temporary duty assignment that ended up being 9 months long (so much for 'temporary'…). Returning home in April/May 2020, EBT dev was resumed slowly and continues into today, and the foreseeable future.

One initial intent for EBT (not being called such when conceived, but rather 'Tweaked BT' [sigh]) was to provide a means to have a workflow that somewhat automated the inclusion of ARM Assembly constructs into an ARMbasic user app. The next extension was to change the default color scheme from white to gray, to match how Notepad++ was set up, to provide visual continuity, if you will. Very quickly, it took on a life of its own and has been updated over the course of the last year and a half, pseudo-continuously. The initial roots of EBT can be found in the ASMtool module within EBT. Being an Android junkie and somewhat spoiled with customization/theming, coupled with being picky about how the tools used on a regular basis are customizable to one's liking, to include workflow aspects that conform to what is perceived as one's needs or desires, caused EBT to morph into what is perceived as a substantive set of extensions to BT.

Note: Overt care to not confuse 'Extensions' with 'Enhancements' is employed. The former is an appropriate descriptor, while using the latter to describe EBT would be presumptively inappropriate as it is, admittedly, a quite subjective term...

Ok, 'nuf with the diatribe - the purpose of this document is to explain EBT and its use - let's get to it.

The best way to see and begin to understand EBT is to 'install' it and use it. Given it is an extension to an existing toolset, there are prerequisites needing to be fulfilled in order to make use of same.

EBT Prerequisites

BASICtools (BT) for ARM downloaded and installed, on an x86 32-/64-bit Windows dev box. BT can be download from here. An archive of the current (at time of drafting) complement of EBT files can be downloaded from here. This User Manual (online .pdf version) is also linked to in EBT's WebHelp MiniTool, detailed later in this document.

NOTE: At its root BT is, and by inclusion EBT is also, a set of TCL scripts that provide a UI framework and, when coupled with compilers, disassemblers, etc., comprise a complete and extended ARMbasic Integrated Development Environment. TCL is a powerful cross-platform interpreted scripting language. Sadly, not being a Linux or iOS type, no effort or assertions are made regarding EBT's functionality when installed on a machine that is not a Windows box. For those users who are on Windows boxes, assertions that it ***may*** work are made (see license/warranty on first page). For *nix-based OSes, EBT's author is not personally interested in progressing through the curve to ensure that EBT works with a BT installation on those machines. However, there is no opposition to someone taking the time to check and advise if EBT works thereon or not. If code changes/additions are needs to enable EBT to work on same, and if you, the reader of this document, are able and willing to tackle modifying EBT's set of scripts to work on Linux or iOS dev boxes, it is endorsed and will be supported, as long as the effort is reasonable and not too disruptive to other endeavors (Selfish? Maybe. | Practical? Absolutely.).

Once BT-proper is installed, the next step is to get the current build of EBT downloaded (link above) and extract same (preserving directory structure) into the Coridium BASICtools directory (often located at %ProgramFilesx86%/Coridium/ (the default offered by the installer)). Administrator approval will likely be queried for by the OS.

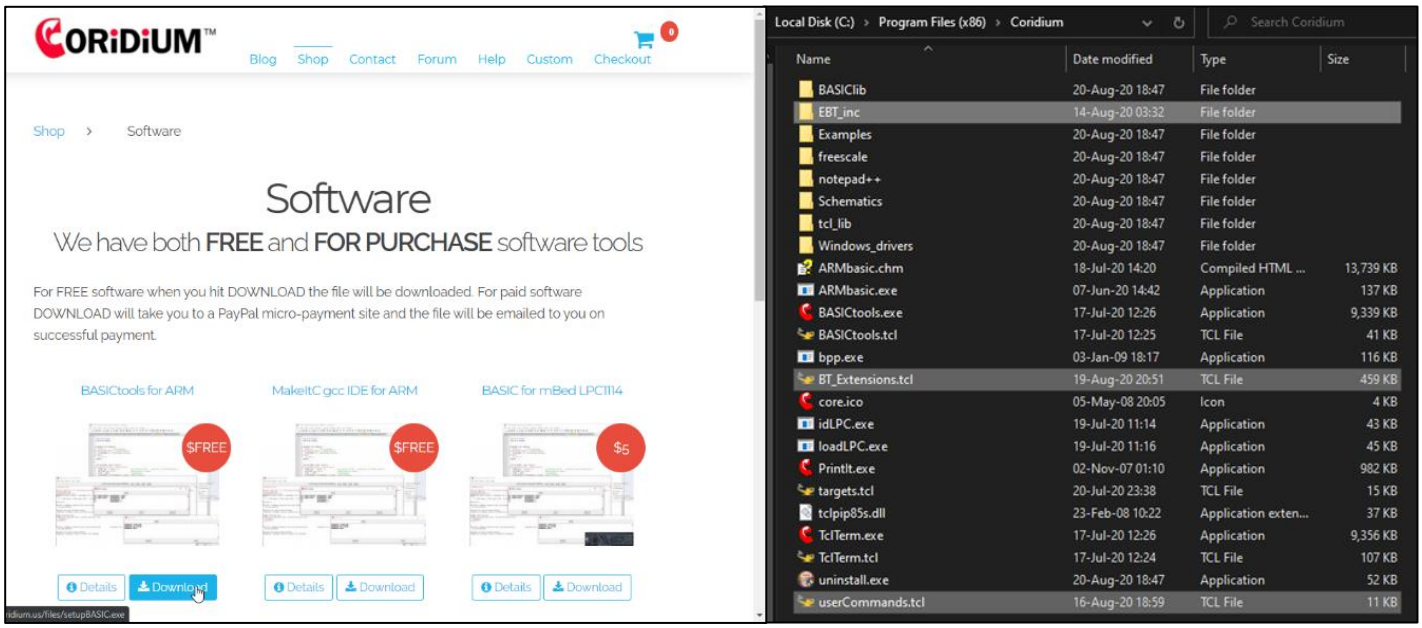


Figure 2: BASICtools Download Page - EBT files extracted into Coridium folder

EBT consists of two .tcl scripts in the root directory of the Coridium Install – BT_Extensions.tcl and userCommands.tcl. Additionally, there are a number of utility binaries, support scripts, and other support files of varying types, which are included in the EBT archive. These other files, when extracted to the BT folder, should end up in the EBT_inc subdirectory.

The userCommands.tcl script is organically 'sourced' in by BT, when it is started. The original intent was to enable BT's user devs to have custom commands be intercepted from the embedded target and be able to have the IDE/host dev box programmatically react to same. An example is either clearing the BT console or responding to a BELL ASCII character being sent by the target to cause the host system to emit sounds through the PC speaker.

This capability of being able to run customized user drafted .tcl scripts at BT's startup was latched onto and heavily employed to Extend BT's core capabilities and UI. In a nutshell, userCommands.tcl checks if an BT .ini saved setting reflect that EBT is turned on and, if so, launches BT with the EBT extensions enabled. If the EBT .ini is missing (i.e. first run), or the enablement of EBT is set to 'off', then BT-proper is launched and EBT is not started, enabling a stock Coridium BASICtools IDE experience.

Speaking of the .ini settings file. BT and EBT each have their own .ini. This was done to help ensure that if EBT is disabled, that there is the smallest possibility of EBT impeding the operation of BT. This philosophy was carried forward in EBT in that EBT doesn't materially alter the BT-proper menus/functionality, but rather has its own menu entry/hierarchy and functionality. The intent is to have EBT extend BT's capabilities, never impeding and keeping material alteration of same to a minimum. Extension of, not enhancement to, is the theme here.

And while we're talking about other directories, as additional information, BT's configuration .ini file is stored in %AppData%/Roaming/Coridium/ (EBT's is in the EBTini subfolder therein). Temp files, used during various phases of the tool chain's endeavors, possibly being touched on later herein, are stored in %AppData%/Local/Temp/Coridium/.

Once BT is installed, and EBT's archive is extracted into the Coridium program folder, BT should be started. Once BT is started, to start EBT, one would click on the ***Tools|Debug|Extended Debug*** menu option that Coridium was kind enough to code into BT. If things are as they should be, BT will source EBT's scripts and one will be met with EBT's default UI.

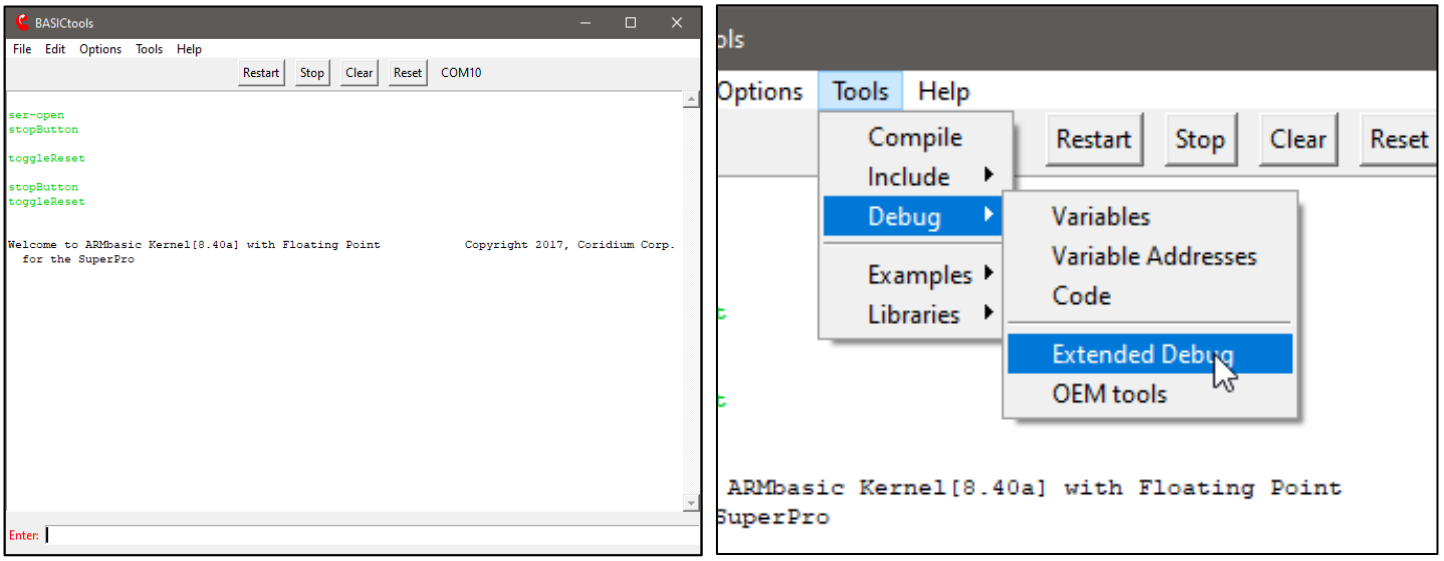


Figure 3: BASICtools Default UI – Starting EBT from BT proper

To further explain EBT functionality herein, screen shots and descriptive text will be used to illustrate the various aspects of using EBT. Also, be advised that there are two 'submodules' that EBT offers for dev's use – ASMtool and TargetExplorer (TE). Both of those tools offer substantive functionality which warrant their own addendums to this EBT User Manual. Those will be drafted and released as supplements hereto as quick as practical.

With that, let's get down to capturing tool images and explaining as much as practical about the feature sets and functionality of EBT's extensions to BASICtools.

EBT UI ELEMENTS – Existing, New, and Extended

The following image/table depict the elements that comprise the main EBT UI and provide details regarding the controls, their functions, and if functional differences which may exist between themselves and the BT-proper counterparts (if any).

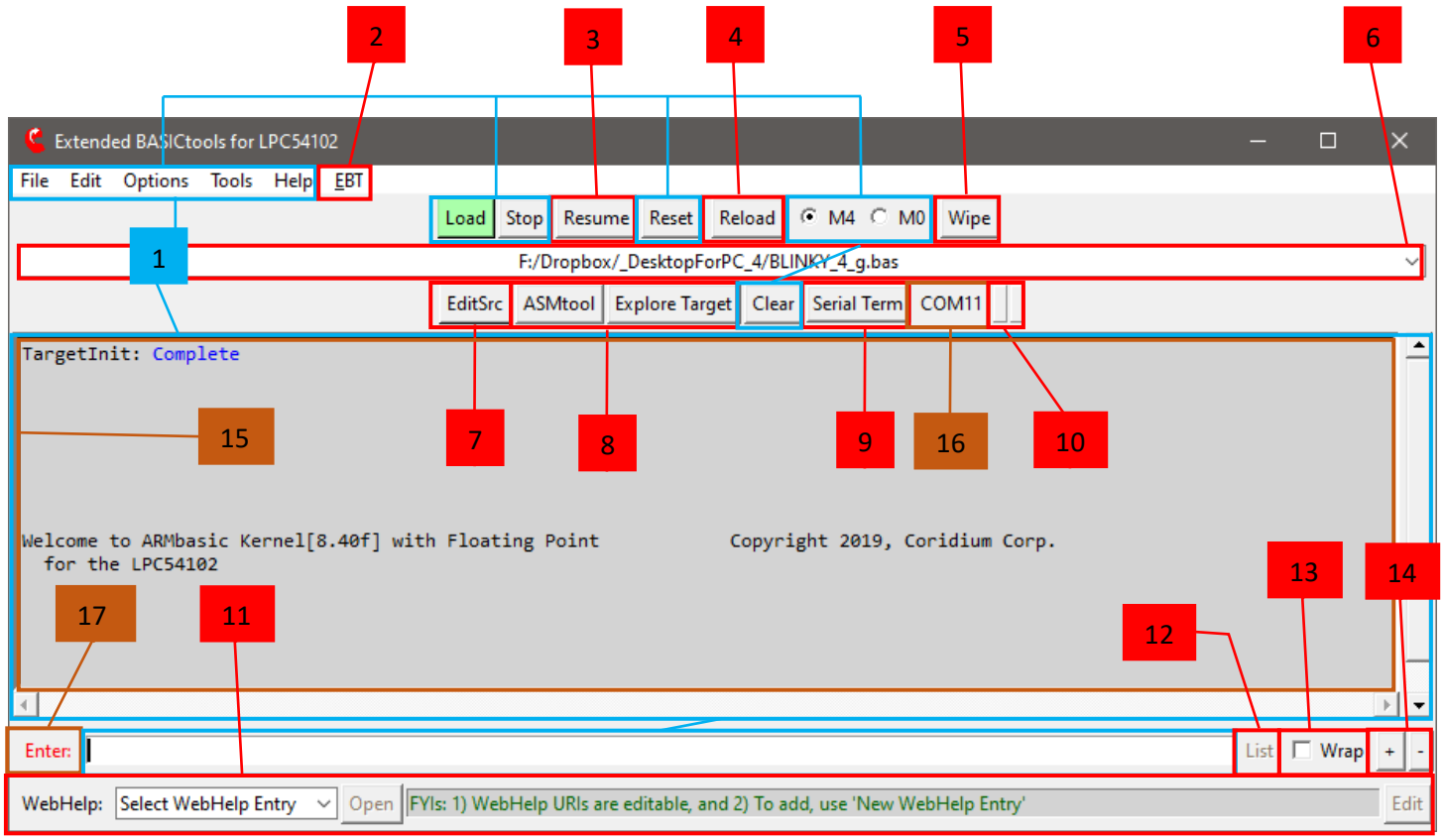


Figure 4: Extended BASICtools Default UI

EXISTING BT UI W/ MINOR OR NO CHANGES	EBT ELEMENT W/ NEW FUNCTIONALITY	BT ELEMENT W/ EXTENDED FEATURES

ITEM	DESCRIPTION
1	Existing **Stock BT UI elements** with functions that are largely unfettered (color/font/label/content excepted :-)
2	**EBT Menu** – will be detailed on a dedicated page(s) herein
3	**Resume** Button – simulates the user typing a carrot (^) – useful after a 'STOP' debug command
4	**Reload** Button – mirrors the BT Reload menu function, with the User App in the DDL (#6) being used
5	Core **Wipe** Button – to overtly erase ARMbasic user app from the MCU – on Core M0, codes an endless loop
6	Historized **Recent** User App Drop Down List – similar to Recent Files in BT File menu
7	**Edit** Source File Button – opens the DDL (#6) selection in the editor configured during BT setup
8	**ASMtool** and TargetExplorer Module Buttons – each module to be detailed in dedicated manual addendums
9	**Serial Terminal** Button – opens a TclTerm stand-alone serial terminal (for debug/monitor)
10	Serial **Traffic 'LEDs'** – Red = Host Tx, Green = Host Rx, both double as buttons to unlock disabled controls if needed
11	**WebHelp** MiniTool – will be detailed on a dedicated page(s) herein
12	**List** Button – becomes enabled with iterative compilation activities, used to list the WIP User App code
13	**Wrap** Toggle – to turn on and off wrapping of text in the console
14	Increase**(+)**/Decrease**(-)** Font Size – for real-time adjustment of BT/EBT UI element text
15	BT/EBT **Target Console** – this is also the drag and drop target for user ARMbasic apps
16	BT/EBT **Port Indicator** – in EBT is also a button that will re-enumerate the current serial instance
17	BT/EBT **Enter Label** – in EBT this is also a button to transmit a carriage return out the serial port

Table 1: Extended BASICtools UI Elements Index

EBT Menu

EBT's menu allow the user to access and/or control various aspects of EBT's functionality:

- Preprocessor Selection and various Preprocessor Settings for the selected EBT Preprocessor - FilePP.
- Menu selectable short-cuts to relevant explorer folders on the host dev box.
- Menu selectable short-cuts to various intermediate files generated by the tool chain during compilation.
- Control of various options that control EBT's UI, 'Windowing' on host OS, and functional behavior.
- Cessation of EBT with Restoration of stock BT functionality (imputes an automatic restart of the toolchain).
- On-demand Instantiation of TclTerm as an additional serial console (useful at times for debug/monitoring).
- Access to a composite pre-processed ARMbasic (AB) Source File – the toolchain's input to the AB compiler.

Each of these will be touched on, some in granular detail where appropriate, in the following dialog and images.

EBT Menu | EBT Source Code Preprocessors

BT-proper makes use of a customized version of CPP, which functions in a manner similar to CPP, and is called BPP, presumably an acronym for BASICpreprocessor (a reasonable assumption never thought to have affirmed). BPP is, by design and lineage, pretty focused at one thing, preprocessing source-code files with a rigid structure of functionality and capability. It is not intended to be expandable and is pretty limited in a myriad of other ways, as it was designed to do just one thing. Don't mistake these comments as non-appreciation. It does perform its designed functions pretty darn well. However, when one starts to request/expect CPP/BPP to do more, the wheels can depart the bus pretty quickly.

These limitations (was and still is perceive as such) was the initiating factor that caused the hunt for other preprocessors to begin. It didn't take to long to find FilePP. After testing FilePP, and eventually realizing just how extremely powerful such an extendible preprocessor solution was, it became evident that FilePP needed to become an EBT team member, and was integrated as one of the key extensions that EBT brought to the table for a dev to leverage in any AB dev effort.

It is acknowledged that, for the vast majority of AB dev work, BT's Stock BPP is more than adequate at preprocessing AB source file, doing so really well. However, it would be appropriate to acknowledge that there are some specific feature sets that FilePP empowers an AB dev with that simply can not be accomplished with the BPP tool as it exists today.

While in-depth details about what FilePP does, how it does it, and how to get it to do it, is well beyond the scope of this manual, the following listed items are some of FilePP's features that see regular employment by EBT's dev:

- Compile-time Maths
- Multi-Level Preprocessing Debug
- C comment Removal
- AB Comment Removal
- For-Next Loop Preprocessing
- Multi-Line Macros
- Selectable Boundary Macro Expansion
- Inner Literal Macro Expansion
- Regex-based Macro Definition & Expansion
- Multi-file Macro Definition, Build-up, Expansion

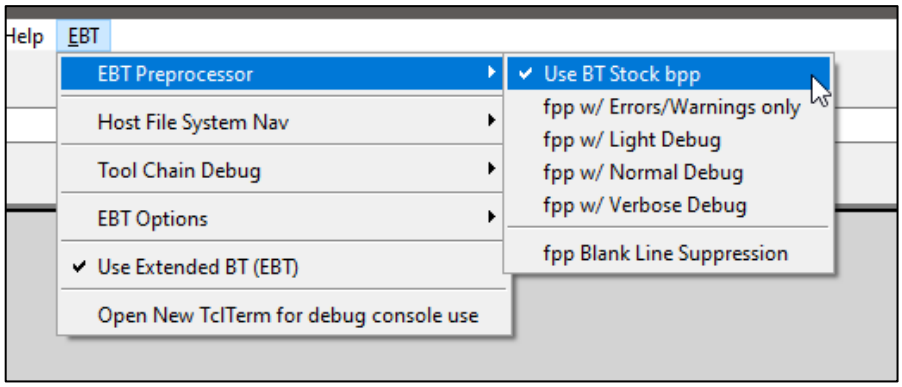


Figure 5: EBT Preprocessor Selection and Control

EBT Menu | Selectable Windows Explorer folders short-cuts

It should be pretty self-evident what these various menu selections do. If one doesn't grasp this, one may be well served to consider taking up knitting or some other hobby (kidding – if you get wrapped around the axle on these, just ask in the forums and someone should be able to help).

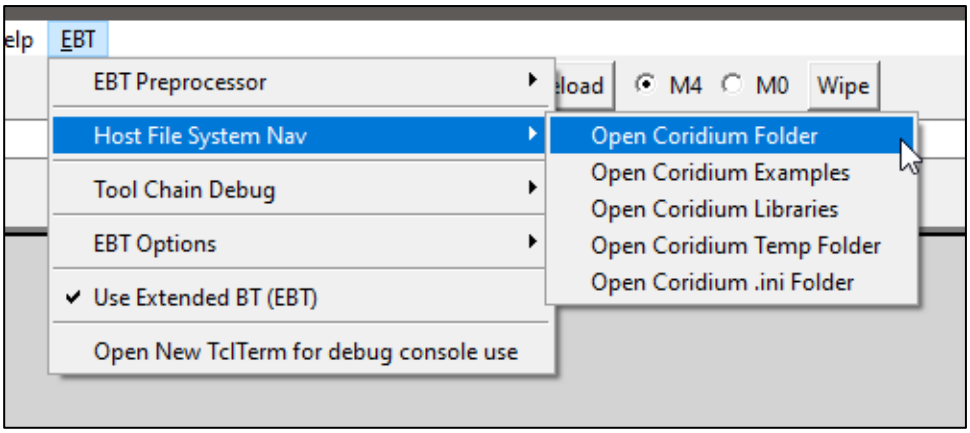


Figure 6: EBT Host File System Folder Short-cuts.

EBT Menu | Tool-chain Intermediate File short-cuts

As with the previous, and for anyone familiar with a typical tool-chain process, extrapolating what these menu-selectable options do should be pretty self-evident. Again, if one gets wrapped around the axle on these, just ask in the forums and someone should be able to help.

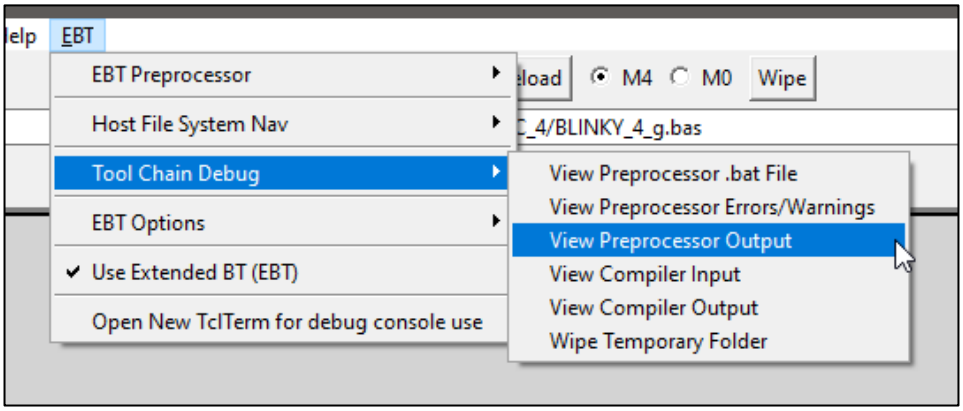


Figure 7: EBT Tool-chain Intermediate File Short-cuts.

EBT Menu | EBT's UI Options, Host-OS 'Windowing', and Optional Functionality

BT and EBT have 'normal' windows on a Windows OS Host box. By normal, one is inferring that the window size is adjustable, has a title, has an icon on the task bar, and can be manipulated to be on top or not, as the case may be.

Windows supports having Modal windows or Tool Windows, where the windows have title bars, but lack some of the dressing and control that 'normal' windows have. One of these differences are that tool windows don't normally have icons on the Windows Task Bar (WTB). Turning on tool mode for the Target Explorer (TE) removes the WTB TE Icons. If one elects to have their WTB configured to always combine the icons thereon, if TE windows were normal windows, and one of them were minimized, when one clicked on the WTB icon, one would then be offered thumbnails of the windows to click on to select which window is desired to be brought to the top and focused. It can be frustrating to have to do so.

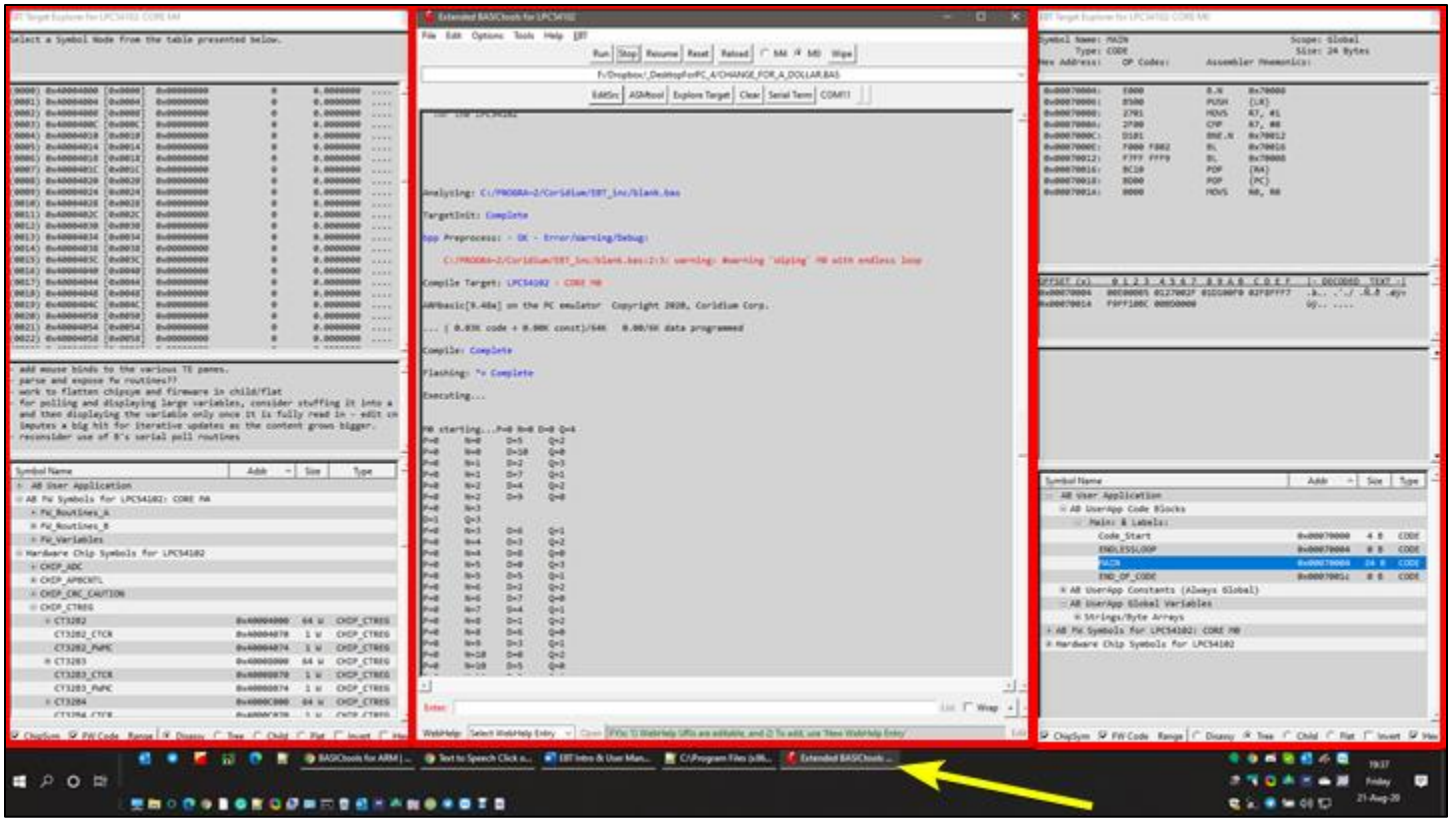


Figure 8: EBT Main UI Window with two TE 'Tool Windows' (note the single Task Bar Icon)

Thus, the option to make TE windows 'Tool' windows was coded into EBT. Additionally, it was determined that in some use cases, it might be desirable to have an option to force windows to keep either EBT's main window on top, or to be able to do so with the TE windows, regardless if they are 'normal' windows or tools windows. The options for indication and control of these window modes exists on the EBT Options menu.

With tool windows, typical observed behavior is to have tool windows minimize with the host-app's main window. The 'Minimize TE when EBT Minimized' option enables this behavior.

The 0x prefix option enables TE to display hex numbers with the non-standard 0x prefix notated onto a hex number. BASIC standard is often either $ or &H (ARMbasic is &H). For some users, the 0x prefix either is visually easier to employ, or is a habit of muscle memory borne from working with other dev languages. This is why the 0x option exists.

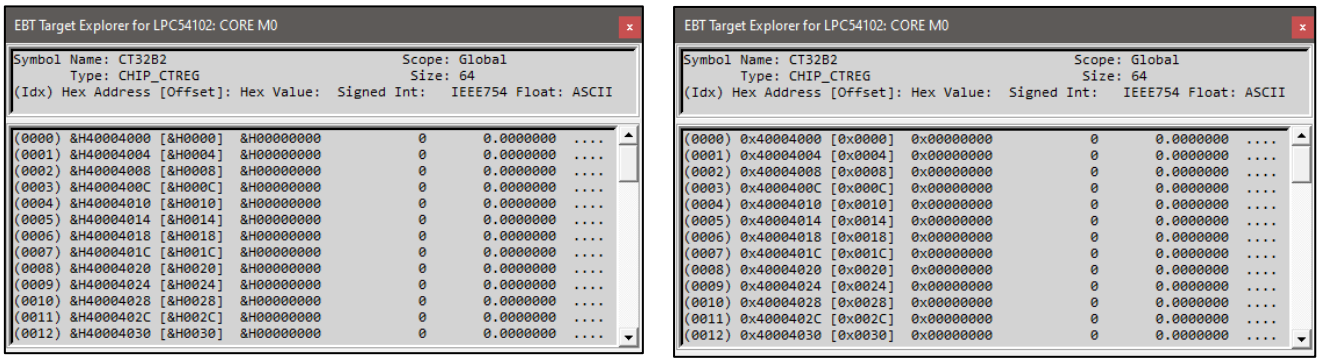


Figure 9: Stock AB &H hex prefix (left) vs. EBT's Optional 0x hex prefix (right)

Anther BT UI default behavior is to wait until the first time a target is programed before it updates the UI to reflect said target, not only in the title bar, where it depicts the controller with an expressive verbose name, but also on the UI, if the unit is a multi-core device (i.e. LPC4330, LPC54102, etc.), where the MCU Core selection radio buttons come into existence.

EBT's approach can be made to be a bit more aggressive in where it actively queries the target controller at startup and updates the UI accordingly. The reasoning for having EBT's optionally behave be this way is two-fold:

1) It enables the UI elements to be updated at startup
    a. admitted eye candy, but for those who have a bit of CDO (OCD in alphabetical order, as it should be…) in them, this can make for a lower-stress user experience, at the price of a bit of startup latency to EBT.
2) It enables EBT to perform some background operations related to the firmware options in TE.
    a. Note: These FW options within TE are still being worked on and are NOT mature yet (hence the 'placebo').

Another way in which to customize EBT is to modify the fonts used on the UIs. This is accessed by the Choose EBT Font option in the EBT Options Menu. Selecting same pops up a font-picker UI. It is relevant to note that the Monospaced fonts are highlighted in Red text on the font name selection control. Additionally, in the sample control, one can observe what is often salient to developers – Visual presentation of specific characters (Slashed Zero (Ø) vs. letter o/O, Upper-/Lower-case iI vs lL vs 1, 2 vs Z, 5 vs S, etc.). The author prefers Consolas due to its commonality and these visual elements. The font selection is for all of the UI elements that are generated by EBT. The menus and control text is the stock BT font.

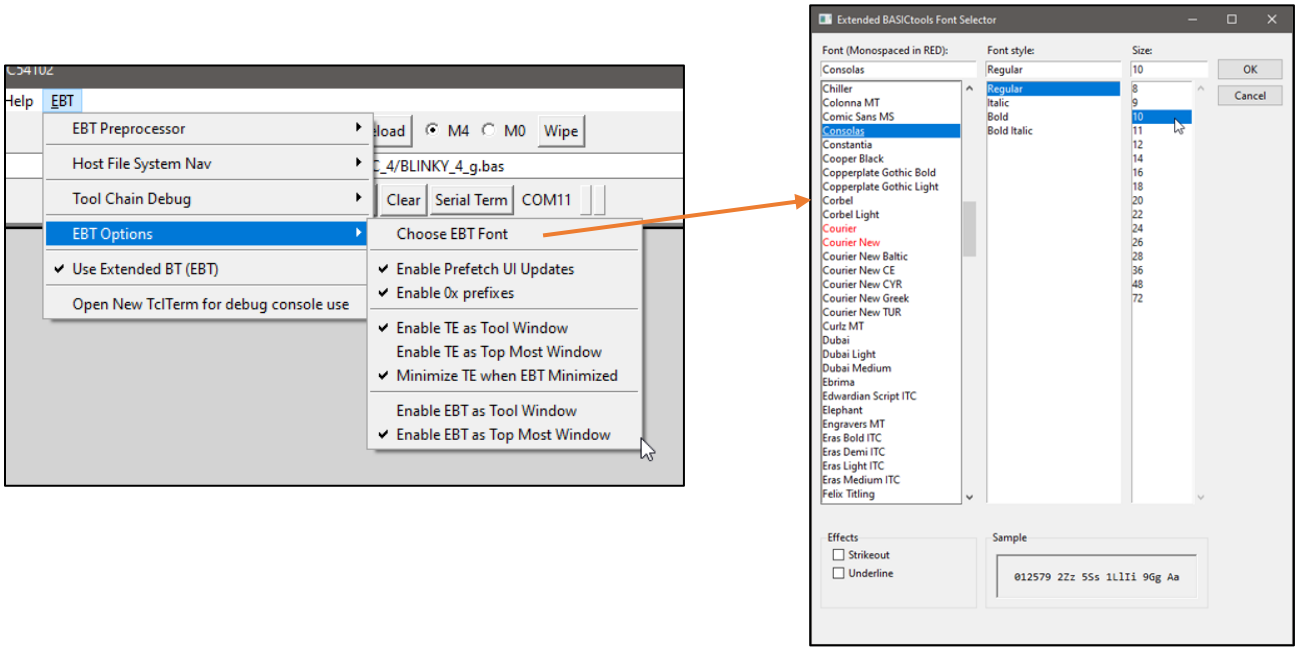


Figure 10: EBT Options - Window Modes, Prefetch, and Font Selection

EBT Menu | Use Extended BT (EBT)

When EBT is active, this option is enabled. It is provided on the EBT menu to enable the dev user to deselect and restore the Stock BT functionality. Deselecting this menu option will cause a restart of the toolchain, in which the stock BT UI will be restored upon restart.

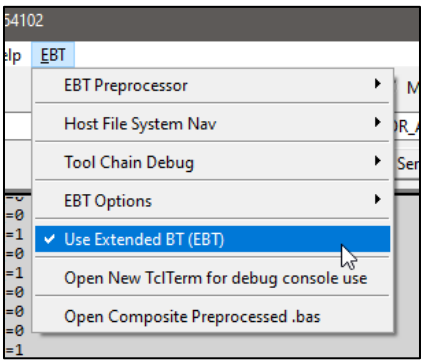


Figure 11: Use Extended BASICtools – Deselect to restore Stock BT-proper functionality

EBT Menu | Open New TclTerm Instance

This menu option opens up a separate instance of BT's TclTerm Serial Terminal Program. A dev may find this useful to have the target stream debug out of a separate serial port (possibly at uber high rates – author has had 1.2mbps debug streaming from a Coridium SBC Target. Same could also be used to enable serial IO comms with a 2nd core on a multi-core target (4330/54102/…) via a separate serial port (peripheral or bitbanged via GPIO).

Figure 11: Use Extended BASICtools – Desele

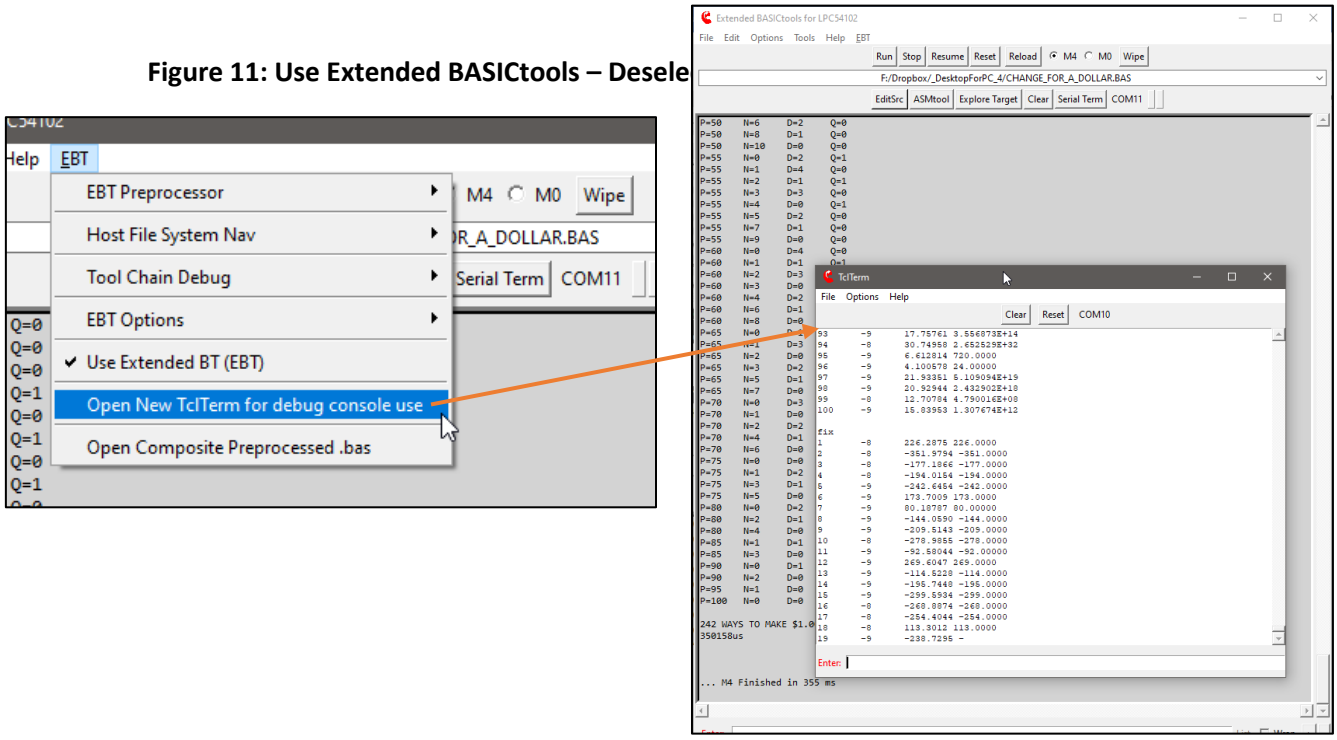


Figure 12: Open New TclTerm Instance (useful for Debug Console (or for 2nd-core comms) via other serial port)

EBT Menu | Open Composite Preprocessed.bas

This menu option, when available (disabled prior to compilation), opens up an instance of the configured editor to allow the dev to review the composite (after all BPP/FilePP preprocessing) to

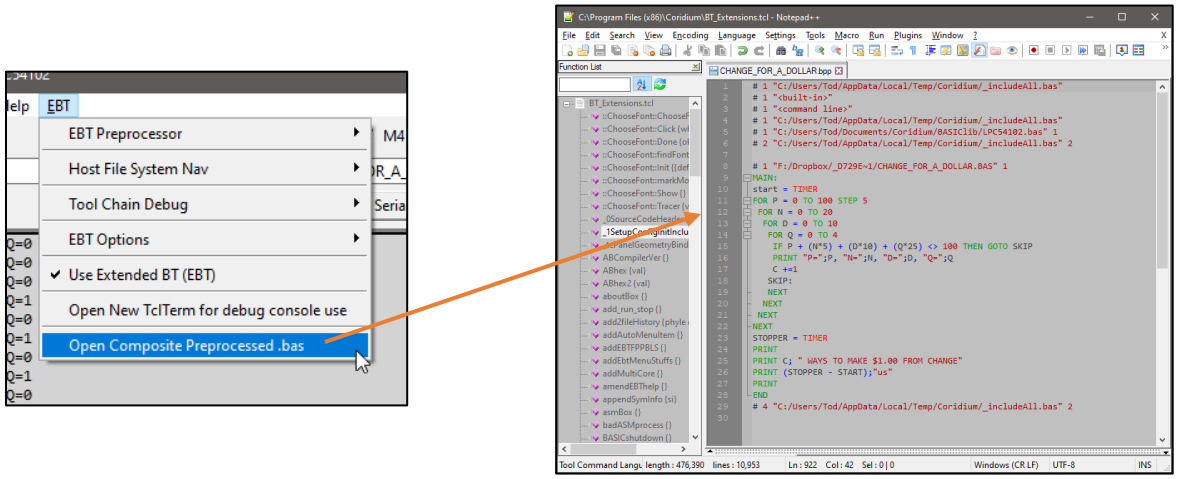


Figure 13: Open Composite Preprocessed.bas – available after sessions first compilation/flash load

EBT WebHelp MiniTool

EBT's main UI contains what is known at the WebHelp MiniTool at the very bottom of the window.

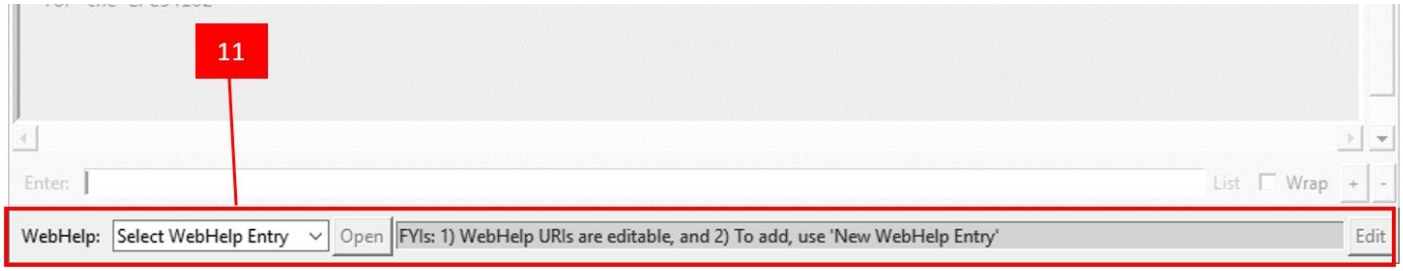


Figure 13: Open Composite Preprocessed.bas – available after sessions first compilation/flash load

This tool is intended to be a place in which links relevant to EBT, AB, MCUs, or whatever else a dev might find useful to have at their fingertips when working within EBT. As one can see in the following image, the author has a number of web resources that can be quickly opened by merely selecting same and then clicking the Open button alongside the DDL.

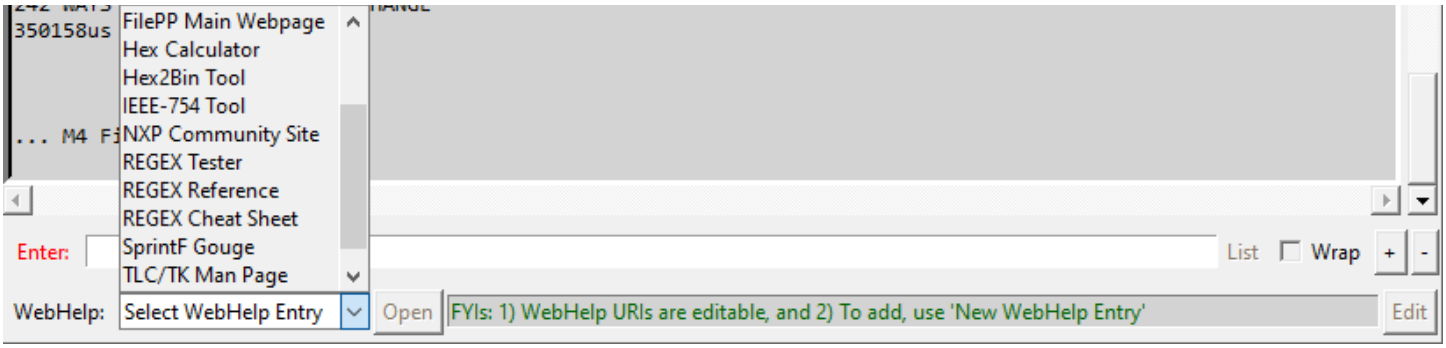


Figure 14: WebHelp MiniTool Drop Down List depicting the current entries available for selection

Figure 15: REGEX Tester selected and Open being clicked – will open a browser to regex101.com, as depicted

The author has programmed to allow new WebHelp entries to be coded in an easy and intuitive manner. The steps needing to be followed are detailed in the following images:

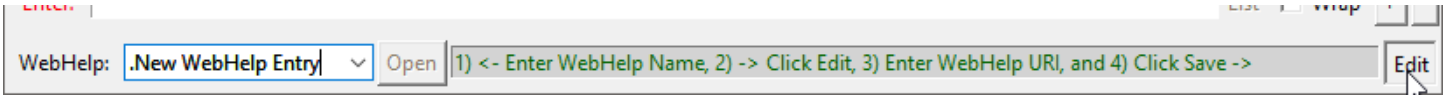


Figure 16: Step 1: Select New WebHelp Entry

Figure 17: Step 2: Clear the entry name and click Edit (changes to Save)

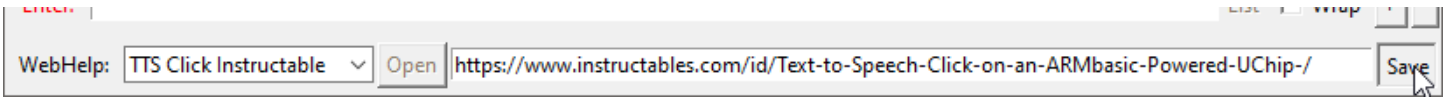


Figure 18: Step 3: Enter WebHelp Entry Name, Enter URL for the resource, and then click Save button

That is all there is to it. Anytime it is desired to visit that resource, just select it and then click the Open button alongside.

Figure 19: Selecting one of the entries in the WebHelp MiniTool

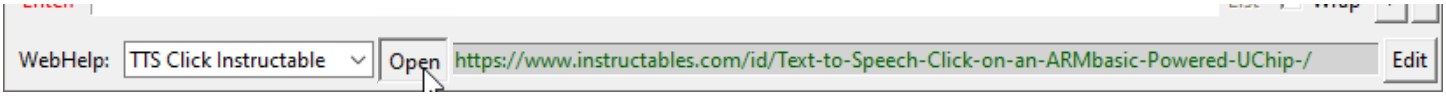


Figure 20: Clicking Open to launch a browser to open the selected resource

All EBT WebHelp MiniTool entries are backed up to the .ini file.

Also, the ASMtool, to be detailed in the following addendum (when completed), also has a WebHelp MiniTool. While the UI and controls are the same, the entries therein are a separate set (also backed up in the EBT .ini) given the context of using the ASMtool to help compile ARM Assembly is likely [much] different than when developing ARMbasic.

Conclusion

Thank you for taking the time to read about using the EBT toolset. If you have already tried EBT, or decide to do so, the author will be pleased. It is understood that it might not be to everyone's liking and that is OK. Trying to please everyone ensures a quick and painful trip to certain failure, which is not the intent here. If it does provide a few folks some usefulness, then it has achieved a goal.

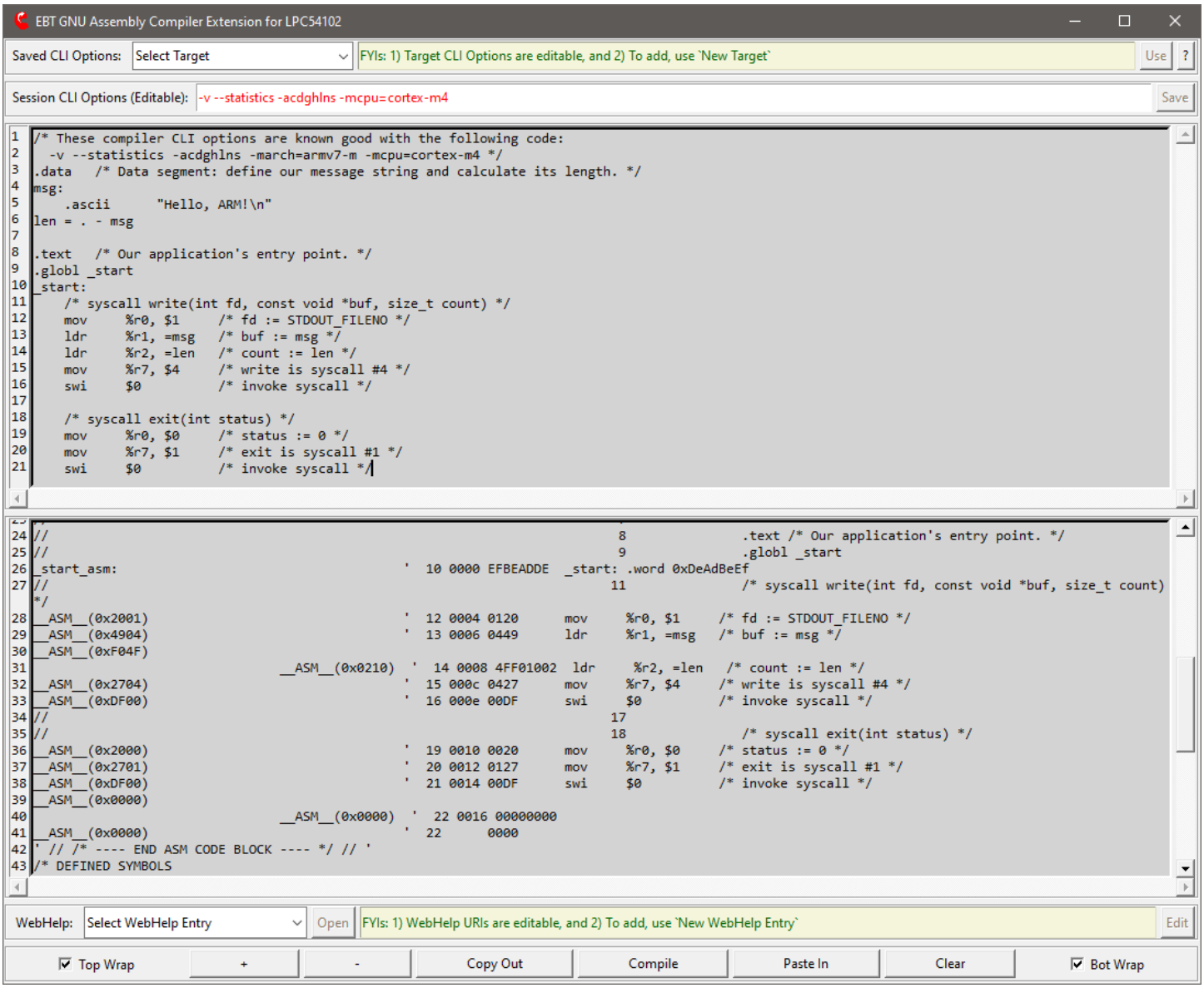
If you experience trouble with EBT there are three options available to you to try to secure assistance:

1) Posting to thread on the Coridium Forums where EBT was announced,
2) cracking open the tcl sources and see if you might glean a better understanding of what is transpiring and how it might be able to be resolved, or
3) Joining the Telegram ARMbasic Channel (https://t.me/ARMbasic) and seeing if anyone there might be who can offer input or assistance.

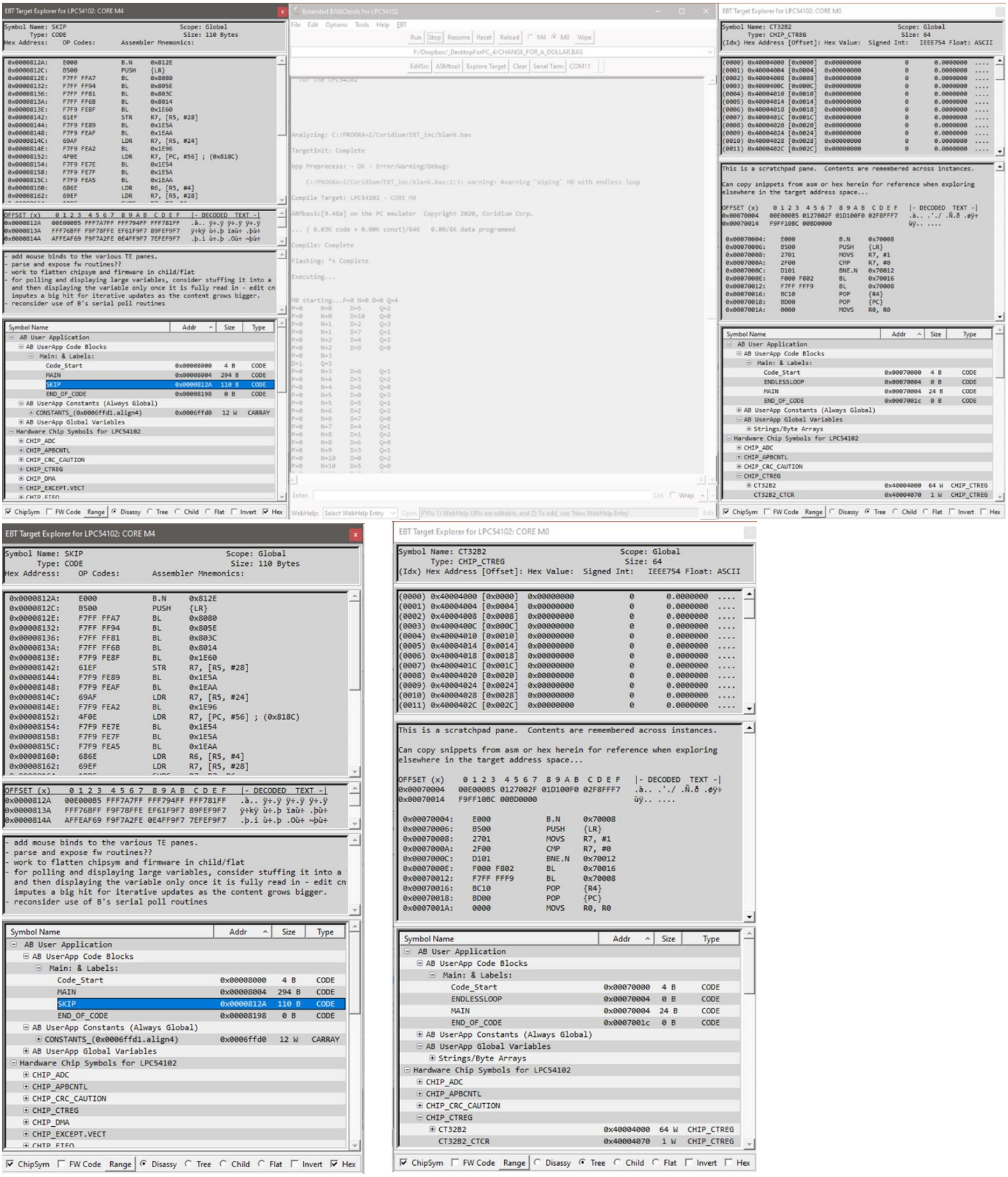
The author bids you well and hopes each and every one of you experience success. Take care and Happy Coding!

-t

Addendum 1: EBT ASMtool … PLACE HOLDER for ASMtool Manual



Addendum 2: EBT Target Explorer (TE) … PLACE HOLDER for TE Manual



Appendix 1: Document Control and Record of Revisions

Document Control	
Document Number	0235_1800_EBT.um.001
Document Title	Extended Basic Tools (EBT) User Manual
Authored By	Tod Wulff, Chief Hack @ The House of Wulff
Source Location	DropBox Cloud Repo
Published Location	http://bit.ly/EBT_UM (Links to the .pdf version)
Acknowledgements	Bruce Eisenhard, Founder/HMFIC @ Coridium Corp Gary Zwan, ARMbasic/Embedded Dev Compatriot

Record of Revisions			
Version Number	**Date issued**	**Revisor**	**Reason for Revision**
0.1	21 Aug 2020	Tod Wulff	Initial draft document
1.0	22 Aug 2020	Tod Wulff	Initial Release