Upcoming Release

- nothing yet

This section covers the fixed bugs and added features as well as the known bugs and missing features in the latest available release from ASH WARE.

2.1 DevTool V2.75B / ETEC V3.01B (2023-Jul-22)

- Fix a bug wherein the 'Goto Line' (F4) run command was not working in source code files. It would instead behave like a plain 'Run' command. 'Goto Line' did work in script files. This has been broken since release 2.73A.
- Fix an issue with inter-target interrupts in multi-target simulations.
- Improve auto-complete behavior in the editor.
- Allow the Script Variable window to display variables from any target, not just the first one.
- For new eTPU projects, the default programming model for C files is now global scratchpad.
- Fix several issues with stepping by angle teeth. The step by angle tooth shortcut of Ctrl-A was broken because it conflicted with the built-in "select all" behavior - it has been changed to Ctrl-Shift-A. Also, step by N angle teeth did just not work but has now been fixed.

This section enumerates features added and bugs fixed in previous releases.

3.1 DevTool V2.75A / ETEC V3.01A (2022-Nov-21)

- Add support for the new S32K39x targets. This includes adding them to the new project part selection list, and adding support for the new linker -endian option that allows auto-generated host code to be output in a form that can support both big endian (MPC) and little endian (S32K) host processors.
- Add backward and forward context navigation to the editor windows - this is controlled via the arrow buttons and dropdown at the upper-left control button ribbon.
- Add backward and forward context navigation to the Waveform window, controlled by the new arrow buttons and dropdown near the left side of the control button ribbon. This is super handy for reverting zoom changes or range settings for discrete/analog data views!
- Fix a breakpoint hit count issue in mutliple target simulations.
- Significantly improve the simulation speed in some cases, particularly those where the Waveform re-draw was a bottleneck. Also improve GUI responsiveness when under high load simulations.
- Fix several minor Waveform issues in the areas of auto-scroll, cursor snapping to transitions, and autorange/ranging of float/fract discrete variables.
- Add a demo (AutoCode) for the new ASH WARE auto host code generation capability in the ETEC linker.
- ETEC: add the -fractFullPrec option to the compiler to enable full precision multiplication of signed fractional data. The historical behavior has been to truncate the LSB (faster).
- ETEC: add a new capability to auto-generate templates of host API code for all eTPU functions in a build, enabled by the new -awac linker option. See the new AutoCode demo for an example of how this works.
- ETEC: change performance, scratchpad and stack messages have been changed from warning-level to informational-level, as that much better categorizes what they are. They can still be individually disabled per message type, or the entire class of informational messages can be suppressed.
- ETEC: the linker now supports the capability to combine two independent sets of code each with their own entry table into one final image. The need for this is rare, but useful in the case where independent development teams each have control of one eTPU engine, one A and the other B.
- ETEC: the -ErrorFill linker option has been added to support the case where the user wants to use an error handler (default or custom), but does not want the unused portions of SCM to be filled with error-handling data.
- ETEC: add the -endian=<mode> option to the ETEC linker to support enhancements to the auto-generated host code for the new S32K39x parts, in which the Arm Cortex-M7 cores are little-endian. This new option can simplify support of eTPU API code across all target types.

3.2 DevTool V2.74A / ETEC V3.00A (2022-Jan-13)

- 32-bit register reads for display into "fast-text" windows such as Global Counters were failing and thus displaying incorrect data.
- Improved global error handler documentation in the Linker Manual.
- Add a write_wctl() script command that outputs collected WCTL data to a specified text file.
- For target-level nodes and below, global settings are now target settings and thus can be unique on a target basis. This really only affects System DevTool. Backwards compatibility is maintained.
- A demo for the new motor control set of eTPU code has been added to the DevTool install.
- ETEC: made numerous optimization improvements - in some cases ETEC now generates executable code over 20% smaller and faster than competing tools.
- ETEC: add a new option '-passParamByReg' which changes the calling convention in scratchpad mode to more closely match that of stack mode, which is more efficient because up to 3 parameters can be passed via register.
- ETEC: support the -d and -i options in the assembler to match the compiler. The -d option allows macro definitions to be passed via command line, and -i specified search paths.

3.3 DevTool V2.73B / ETEC V2.63B (2021-Sep-17)

- update the -ErrorLib- (disable default error handling library) ETEC linker option to output a minimal code image if no user defined fill opcode, unused entry thread handler, or unused code memory handler are defined. This results in a smaller array of opcodes in the output <>_scm.h file, or user-defined file output via the #pragma write facility, which is in some cases useful.

- fix several minor issues w/ regards to breakpoints and the new Breakpoint Window.

- add MPC5743R and MPC5745R to supported microprocessor list (same as existing MPC5746R, just added for completeness)

3.4 DevTool V2.73A / ETEC V2.63A (2021-Mar-17)

- Add a new Breakpoint Window in DevTool. This allows the easy viewing of all breakpoints in a project and provides advanced capabilities such as hit count break control, conditional breakpoints, and breakpoint actions which are like the existing action tag capability, but associated with a breakpoint rather than a tag in the source code, thereby providing finer-grained control. The breakpoint conditional expression can be particularly useful in multi-channel debugging - for example, if you have 6 channels of the same function but

only want to break on the instance on channel 17, just add the conditional expression "chan == 17" to the breakpoint. This feature is considered to be at beta-level and so may see tweaks and enhancements over the next few releases.

- Add AN4908 engine control library host code and System DevTool example to the install.

- Fixed a bug in handling of Uniform Naming Convention (UNC) style pathing was not supported properly in certain cases.

- Fixed an ETEC linker bug in which a false 'out of memory' error can occur when linking in the Engine Scratchpad mode. Note that this is seen in certain cases when building NXP's latest/upcoming Motor Control library. (Note that more detail on this bug can be found in the ETEC bug notes - see https://www.ashware.com/etec-release-notes) Note that the following rather cryptic message reports this issue. "*Internal diagnostics has detected the following bug: A field's bit pattern is overflowing it's # of bits*".

- Fixed a bug in which the menu option View->Timers triggers an errant, "unable to open the following window ...". Support for this window has never been available in the DevTool tools architecture but somehow this menu option was enabled without actually being supported.

- Fixed a number of small issues with System DevTool and its handling of host code: improved syntax highlighting, code references, corrected #include search.

- Fixed issues with using #include in Vector files. This should work in all cases now.

- Fixed an ETEC linker bug wherein the #pragma write ::ETPUenginedatasize (and ::ETPUenginememsize) macros were reporting the wrong values (global data getting included in the value). More info can be found in the ETEC bug notes.

3.5 DevTool V2.72E / V2.72D / ETEC V2.72C (2020 Dec 15)

eTPU, Version 2.72 Build E

- Fixed a bug in which if two waveforms are showing the same signal, then one of the signals will be turned to 'off' if the Waveform Signal Options dialog is opened.

- Fix a bug in which a breakpoint on a line that contains a MDU instruction (multiply, divide, or multiply-accumulate) the breakpoint gets hit twice in that the simulator will stop on both the first-hit opcode associated with the instruction as well as on any loop associated with the 'MDU-BUSY' flag. Note that while this is is annoying in the GUI, it did not affect the simulation model.

- Explicitly support the C99 fixed width types int8_t, int16_t, int24_t, int32_t, uint8_t, uint16_t, uint24_t, uint32_t, fract8_t, fract16_t, fract24_t, ufract8_t, ufract16_t, ufract24_t. This means that in existing code, typedefs creating types of these names will need to be removed or changed when moving to this release or newer. Additionally, a new system header file, ETpu_StdInt.h, has been added that presents additional fixed-width integer definitions.

- Fix a linker bug in which an errant error message 'cannot change seq, Optimizer::OptRegRegUpStreamSeqIf_C_V_FalseFixup()' is reported and the link fails when a combination of rare things line up just right AND a function is called just once.

- Fix an optimizer code-generation bug that occurs in certain combination of conditions (thank you Chris for reporting this.)

- Fixed a breakpoint issue with the very last instruction in a file under when the 'Legacy Mode' coding style is used and had a thread afterward's (typically a 'dangling else') that had no associated code because it forwarded the entry table entry address to another file.

- A newly-introduced annoying 'feature' is that in certain cases the simulator will wait an extra 40 seconds to launch. This occurs when your computer is connected to the internet, but the internet is not connected to the internet. This was seen when logged into GOGO on an airline flight, but the airplane's GOGO network was not connected to the internet. The case has been fixed and the delay reduced to a reasonable time span.

- For System DevTool, support eTPU channel interrupts (and global exceptions) triggering interrupt service routines on the Host target. Previously, interrupts had to be polled by reading the CISR registers. A new enhanced UART open source git project uses this feature in its host API demo.

- Improve tool compatibility with minor changes and updates to the ETEC standard headers.

- Add append_csv() script command to enhance csv file capabilities in the simulation scripting.

- Added an HSR field to the ASHchannel capability that provides symbolic access to channel hardware field from within the simulation environment.

MC33816, Version 2.72 Build E

- Added support for the High Side drivers on the gnd-side of the H-Bridge. [mbr3]

- Fixed an issue with the STRONG bias current. It was backwards in that a strong bias was being modeled as having greater resistance and has been fixed such that it is now less.

- Fixed the 'slew rate' field of the Low-Side Driver window when viewing LS7. It now shows the correct slew rates for the (0x18F) setting. (Note the slew rate and corresponding R_ON are not actually modeled ... but it is nice to see the correct values.

- Added the missing scripting enumerations for the LS7 driver's slew rates, Ls7Slewrate_1500Vus, Ls7Slewrate_300Vus, Ls7Slewrate_50Vus and Ls7Slewrate_25Vus.) Thank you Maurizio.

- Fixed a bug in which the filter delay was immediately firing coming out of reset when the configuration resulting in a HS or LS output being in the non-default state. [mbr1]

- Fixed the assembly and simulation of the MC33816 'chth' instruction. The order of the feedback select field in the instruction is incorrect in the MC33816 documentation. [mbr2]

Fixes from (Previous) Versions V2.72 Builds D and C

- Fixed a bug that occurs when moving a project between computer systems. The problem is with how floating windows are restored that would (otherwise) be outside the visible region of any monitor. Previously, the would be brought into view by making them visible as a tiny sliver on the edge of a monitor. Because they were just a sliver it was easy to miss them entirely. They are now restored fully visible within the primary monitor.

- Fixed a bug in which the 'Goto Time' function was not saving/restoring in the environment file correctly.

- Improved the MC33816 simulation speed. There were two issues. 1) A slow analog data storage. 2) Waveform display of analog waveforms, especially when zoomed all the way out. The improvements depend entirely on the simulation case, however in one particularly significant case the simulation speed was reduced from 110 seconds to 15 seconds.

- Fixed a memory leak in the scripting environment related to script array variables.

- Fixed an ETEC linker issue wherein a generated auto-struct could be incomplete.

3.6 DevTool V2.72C / ETEC V2.62C (2020 March 14)

These release notes for builds A and C have been combined.

DevTool

- Improved support for the upcoming NXP motor control release (Build C).
- Added multi-dimensional array support to script command. **This unleashes data-driven testing and is arguably the most significant features ever added!** See below.

```
F64 v1aLeft = vShortLeft - IPos * rShortLeft;
F64 vTermPosLft = (vShortLeft - BATT_VOLTS - DIODE_VOLTS);

F64 testArray[NUM_TESTS][NUM_DATA] =
{
       // funcAddr,                Sense,Signature,Threshold,Drvs, Resistance,
              Final,TerminalVolts,Volts1A,Volts1B,Volts2A,Volts2B,           Diags,
    {  1,_AW816CL_enable_0000_drivers_,  0,1,0xE51C, 0.100, 0x0000, LOAD_OHMS,
              0-BAT_2DIODES, 0-DIODE_VOLTS,BAT_1_DIODE,BIAS_VOLTS,BIAS_VOLTS,0x111111, },
    { -1,_AW816CL_enable_0000_drivers_,  0,1,0xE51C,-0.100, 0x0000, LOAD_OHMS,
              BAT_2DIODES,   BAT_1_DIODE,0-DIODE_VOLTS,BIAS_VOLTS,BIAS_VOLTS,0x111111, },

    {  1,_AW816CL_enable_0101_drivers_,  0,1,0x9AAC, 0.100, 0x0102,
LOAD_OHMS+POS_LS_OHMS+NEG_LS_OHMS, 0,VLowsPosA,VLowsPosB,0,0,
         0x100100, },
    { -1,_AW816CL_enable_0101_drivers_,  0,1,0x9AAC,-0.100, 0x0201,
LOAD_OHMS+POS_LS_OHMS+NEG_LS_OHMS, 0,VLowsNegA,VLowsNegB,0,0,
         0x100100, },
    {  1,_AW816CL_enable_1010_drivers_,  0,1,0xAA90, 0.100, 0x1020,
LOAD_OHMS+POS_HS_OHMS+NEG_HS_OHMS, 0,VHighsPosA,VHighsPosB,BATT_VOLTS,BATT_VOLTS,
         0x011011, },
    { -1,_AW816CL_enable_1010_drivers_,  0,1,0xAA90,-0.100, 0x2010,
LOAD_OHMS+POS_HS_OHMS+NEG_HS_OHMS, 0,VHighsNegA,VHighsNegB,BATT_VOLTS,BATT_VOLTS,
         0x011011, },
    {  1,_AW816CL_enable_0001_drivers_, -1,0,0xAB7D,+0.100, 0x0002,
LOAD_OHMS+NEG_LS_OHMS,             0-BAT_1_DIODE,   0,BAT_1_DIODE,0,0,
         0x100100, },
```

```
    { -1,_AW816CL_enable_0001_drivers_, -1,1,0xAB7D,-0.100, 0x0001,
LOAD_OHMS+NEG_LS_OHMS,             DIODE_VOLTS,   0,0-DIODE_VOLTS,0,0,
         0x100100, },
    {  1,_AW816CL_enable_0100_drivers_, -1,1,0x633D, 0.100, 0x0100,
LOAD_OHMS+POS_LS_OHMS,             0-DIODE_VOLTS, 0-DIODE_VOLTS,0,0,0,
         0x100100, },
    { -1,_AW816CL_enable_0100_drivers_, -1,0,0x633D,-0.100, 0x0200,
LOAD_OHMS+POS_LS_OHMS,             BAT_1_DIODE, BAT_1_DIODE,0,0,0,
         0x100100, },
    {  1,_AW816CL_enable_1000_drivers_,  0,1,0xE5D9, 0.100, 0x1000,
LOAD_OHMS+POS_HS_OHMS,             0-DIODE_VOLTS,
BATT_VOLTS,BAT_1_DIODE,BATT_VOLTS,BATT_VOLTS,   0x011011, },
    { -1,_AW816CL_enable_1000_drivers_,  0,1,0xE5D9,-0.100, 0x2000,
LOAD_OHMS+POS_HS_OHMS,             BAT_1_DIODE, BATT_VOLTS,0-
DIODE_VOLTS,BATT_VOLTS,BATT_VOLTS,   0x011011, },
    {  1,_AW816CL_enable_0010_drivers_,  0,1,0x33CF,+0.100, 0x0020,
LOAD_OHMS+NEG_HS_OHMS,             0-BAT_1_DIODE, 0-
DIODE_VOLTS,BATT_VOLTS,BATT_VOLTS,BATT_VOLTS, 0x011011, },
    { -1,_AW816CL_enable_0010_drivers_,  0,1,0x33CF,-0.100, 0x0010,
LOAD_OHMS+NEG_HS_OHMS,             0+DIODE_VOLTS,
BAT_1_DIODE,BATT_VOLTS,BATT_VOLTS,BATT_VOLTS,   0x011011, },
};

for(cnt = 0; cnt<NUM_TESTS; cnt++)
{
    // Table values
    S32 index = 0;
    S32 start_dir      = (S32) testArray[cnt][index++];
    S32 funcAddr       = (S32) testArray[cnt][index++];
    S32 sense_dir      = (S32) testArray[cnt][index++];
    S32 sense_valid    = (S32) testArray[cnt][index++];
    S32 signature      = (S32) testArray[cnt][index++];
    F64 desiredAmps    = testArray[cnt][index++];
    S32 driverStates   = (S32) testArray[cnt][index++];
    F64 resistance     = testArray[cnt][index++];
    F64 terminalVolts  = testArray[cnt][index++];
    F64 load_volts_1a  = testArray[cnt][index++]; // Initial bridge's left voltage
    F64 load_volts_1b  = testArray[cnt][index++]; // Initial bridge's right voltage
    F64 load_volts_2a  = testArray[cnt][index++]; // Final bridge's left voltage
    F64 load_volts_2b  = testArray[cnt][index++]; // Final bridge's right voltage
    S32 vdsVsrc        = (S32) testArray[cnt][index++];
    if( funcAddr == 0) break;

    F64 startAmps = 0.0;
    //-----------------------------------------------------
    if( start_dir == 1)
    {
        // Start with the positive-side turned on
        write_spi_data16( _AW816DA_IMM_TestFuncAddr_, _AW816CL_enable_1100_drivers_>>1);
        write_spi_data16( _AW816DA_IMM_Signature_, 0x0000);
        F64 posTimeConstant  = LOAD_MICRO_HENRIES / (LOAD_OHMS+POS_HS_OHMS+POS_LS_OHMS);
        wait_time( 5 * posTimeConstant );
```

- Improved reload of certain windows, improved window scroll position across project close and then re-open.
- Fixed a recently-added bug in which breakpoints and bookmarks below a deleted line of code or script would not stick with the correct line (incredibly annoying).
- Improved GTM ELF/Dwarf 3.0 support (Note GTM simulator not released to production.)
- Fixed a bug in which in certain rare conditions a failing simulation run could.
- Improved Universal Naming Convention (UNC) file name support (Build C).

MC33816 Features and Fixes

- Added and H-Bridge external circuit model.
- Added several flavors of high-side and low-side RL and R drivers. These use the HS and LS drivers to drive and can be either regulated (current measured through a sense resistor) or unregulated (current is not measured.)
- Implemented MC33816 trace capabilities.
- Fixed a MC33816 Injector bug in which the 'Shorted Device' fault condition on the injector was not being simulated correctly.
- Converted the MC33816 Simulator's 'Configuration', 'DACs', 'Flags', 'Registers', and 'Wait Rows' to be the new 'editable' style window. This means (for example) that a DAC command value can be modified, thereby resulting in a change to the DAC's output voltage.
- added scripts 'exponential_approach()' and 'natural_log()' to aid it certain calculations.
- Fixed a bug with the auto-completion of certain script commands. The auto-complete parameter list for certain parameter types shared by both the eTPU and MC33816 were listing the eTPU and not the MC33816.
- Improved certain circuit windows to show the current threshold of the current-sense comparator. This is VERY handy.
- Implemented the code coverage script commands for the MC33816. Previously, these were supported only in the eTPU.

Generic CPU Simulator Features and Fixes

- Added a generic CPU model that supports 'Internal' build. This is effectively a generic 'C' simulator that is central to our upcoming GTM simulator but quite useful for all of our simulators (eTPU, MC33816, etc.)
- Implemented the CPU call stack that had been available in Mtdt. It is now available in DevTool.
- Added feature to detect needed (but needed) GNU installation for generic CPU simulator.

Installation

- Fixed a installation bug that can occur when the user's 'My Documents' are is inaccessible (rare).

3.7 DevTool V2.70A / ETEC V2.61A (2019-Mar-06)

DevTool - eTPU Simulation Model

- Fixed a eTPU Model bug with TPR buffering in high rate mode. The missing tooth count was lost on this transition with the result of incorrect behavior when entering high rate mode with a non-zero missing tooth count.

DevTool - Debugging

- Adding a disable-able warning when code is executing out of the error handler ... FINALLY! Interestingly, this warning helped ASH WARE identify about a dozen errors in various sample code and internal test suite code here at ASH WARE.

- Sped up regression test simulations in which large log files are generated. Speedup depends on the number of messages in being written to the log file, but a 30X was found in our test case.

DevTool - Scripting

- Added script commands for verifying pulses, transitions, and pin levels over a time-window. This is a far more exhaustive tests that when using 'verify_output_pin() because the latter just verifies the pin state at a snapshot in time, whereas the new commands verify the pin state over entire region of time and extra transitions in unexpected locations will fail.

- Add ability to 'cast' in the scripting environment. This is a significant improvement in many situations, the most obvious one being that script variables can be floating point (F64) can be passed as parameters to (say) fract or S8 or such. Authors note: this was really holding me back in some work I was doing. When we added this capability I felt how one of those little hyperactive dogs feels who has been caged up all day and, arriving at the park, my owner takes the leash off.

- set_code_label_warning("MyCodeLabel"); script commands which allows you to generate a warning if code hits a specified label. This allow you to instrument your code in such a way as to detect errant code paths which are generally associated with subtle and difficult-to-detect errors in the entry table.

- Fixed several bugs associated with scripts contained in files that were #included from the main script file. For example, breakpoints and 'goto cursor' debug capabilities now work.

- Added a print_to_trace_ex1(); script command that allows the trace to be associated with a user-specified channel. Wow, this makes it really easy to instrument your code by inserting the following text into key spots in your source codeyour source code.

```
// @ASH@print_to_trace("PWMF.%d setting period %d to %d\n", chan,
period, frame_synch.new_period );
```

- Added a print_to_trace_ex1(ChanNUm, Message); where the ChanNum has been added to play better with the powerful filtering capabilities of the trace window.

- Support the read_time() script command in backdoor scripting (when script commands are accessed by 'c' code running on a simulated host processor.

DevTool -Trace

- Enhanced the already excellent (in the authors opinion) trace window by adding the ability to move Waveform Window cursors to future match times once the MRLE has been set. This is really effective. The window has also been cleaned up quite a bit making it easier to view with a combination of adding helpful information and removing extraneous information and also improving how things are lining stuff up in columns.

- Fixed a large number of small issues with the Trace Window which cumulatively are a significant improvement to the trace window.

- Fixed the trace capability ... the write_mer was not generating a trace if the mrle was already set. Pretty significant problem actually.

DevTool - Waveform Window

- Changed the look and feel of the waveform window and fixed a large number of small issues that cumulatively are a significant improvement in the waveform window. Also the 'usability' has been improved. We have noticed when working with customers that they would not become aware of the many powerful waveform window capabilities. So the emphasis has been on making these 'hidden' capabilities more obvious to users. It is expected that even power-users will be surprised to suddenly 'discover' some existing powerful capabilities.

- Made it far easier to grab (and then 'drag') the waveform's vertical cursors using the mouse.

- Fixed an issue in which windows associated with eTPU channels could have the incorrect channel name when first launched. Note that this can also affect the names of windows that have an associated eTPU channel such as the channel frame and channel hardware windows, etc.

- Fixed a recently-broken problem in which script variables could not be viewed as a waveform nor added to the watch window.

- Fixed an annoying 'feature' in which the waveform window would auto-scroll to the end of the waveform when beginning to edit a file.

- Fixed a serious bug in which the simulator would lock up if right-clicking twice quickly in the waveform pane of the window window.

- Fix waveform view of fract types (display as floating point).

- Improve re-finding of discrete symbols following a code rebuild and reload.

- Improved auto-scroll behavior in several situations, thereby better reading the user's mind (would have written 'better reading the user's hidden desires' but our editor nixed this description noting, 'this is not a gothic romance novel'.

- Allow script variables (only while running, since scripts are executed as if interpreted) to be added to waveform and watch window. This may be a fix to a recently-broken feature in certain cases.

- Fix display of _Bool type.

- Added popup menu items to move one waveform underneath a specific other waveform, or to the very top, or underneath all the non-off waveforms.

DevTool - Local Variable Window

- For only retrieve chan variables if the execution unit in that engine chan active.

- Improved the variable ordering.

- Fix a bug where target selection was getting overridden by the active target.

DevTool - Other Windows

- Provide a way to clear the output window in popup menu (customer feature request.)

- Re arranged and improved the 'eTPU Channel Hardware' window and added the missing (and important) Function Mode (FM) bits.

- Numerous improvements to drag & drop (e.g., dropping variables from source code into watch window) including fixing features and solved several issues where a DevTool crash could occur.

- Fixed '_Bool' display of channel variables (used in several debug windows and tooltips)

- Fixed an annoying bug in which the eTPU channel name in certain windows can go to a previously used, but no longer used name.

- Fixed a nasty (but rare) bug which, in certain circumstances, a waveform window name could not be changed, but instead was insisting in staying the same as a previous name on that same eTPU channel.

- Fix several issues in watch watch and waveform window's view of channel hardware fields.

- Fix a bug in the channel frame window in which in certain unusual cases the wrong variable can be found because wrong function number is checked (complicated).

- Fix an issue with the display of channel frame variables in the local variable window, when the channel changes.

- Add a popup menu option to add a symbol to the watch window

DevTool - Editor

- Fix a crash that occurred when a pre-processor keep-out is detected AND there are 0 characters past the keep-out start.

- Vastly improved "Find Code References' capability in Script Commands files (when a variable is selected, right clicking brings up a code reference option that locates all use of the script variable.)

- Added 'Type' information to variable tooltip in Source Code and Script files. This is handy in large projects where the variable type might not be remembered.

- Fixed an auto-complete crash under certain unusual circumstances such as empty file.

- Improved (but not totally fixed yet) some issues with syntax highlighting in which syntax highlighting would stop and the affected text would appear black.

- Fixed several related bugs in which the redo/undo buffer can get out of synch (bad!) and in which the 'Line Information' indicators (red bubbles and similar indicators on left side of editor) can move to the wrong line of source code. These occur when typing/pasting/indenting/etc., when there is selected text that gets overwritten, especially when the last character in the text is a newline.

DevTool - Search Utility

- Fixed a bug in which the 'search' utility would not find text if the 'whole words only' option was set, AND some non a-zA-Z0-9_ text were either at the beginning or end of the search text. Authors note: we had to fix this bug because I was running out of swear words.

- Increased the 'Search File' dialog's limit of 200 files and 1000 lines to one million files and 10,000 lines.

DevTool - Build

- Fixed a subtle and nasty bug when using external build (code is being built outside the simulator) and an auto-generated file is open in in the simulator that are generated using #pragma write. In this case it was subtly possible to overwrite the new auto-generated files with the old auto-generated file as follows. When the file is deleted on disk by the build tools, a dialog box opens up asking if you want to re-save the file that was deleted. If you hit the obvious 'ok' button, the newly generated file would be lost because it was getting overwritten by the earlier version open within the simulator. Then the debug environment would get out of synch with the actual code in really bad ways such as variable offsets being incorrect. The fix is to detect this specific case and reload the newly generated file assuming the version that was open in the simulator had not been modified. Here is the specific change. '*Detect case where a file open in the editor has been deleted from disk, then after prompt for user action, it has been re-generated. In such a case, if file was unmodified, then just reload from disk, and if modified, give user choice on taking new file from disk or saving modifications.*'

- Added a post-build .bat file capability in which a user-specified .bat file gets executed at the conclusion on an internal build.

- Fixed several minor internal build' issues which would result in things such as an errant error message in the second eTPU engine if the build failed that was tied to the first eTPU engine.

- Batch build arguments were not being formatted in a usable way (do not prepend with “-d=” as would be done for compiler).

DevTool - Help

- Significant cleanup, re-layout and overall improvement of our help manuals.

- Fixed -man and -man=<ManualName> command line arguments to now work (had been broken to due move from .CHM to .PDF help manuals)

DevTool - Misc

- Fixed a 'Most Recently Used' (MRU) issue in which 'Startup Scripts' were being categorized 'Recent Other File' types rather than as script file times. Annoying because the MRU list would be filled with startup script files, and the desired recently-used (say) .bat and .txt files were getting pushed out of the list.

- Added the option of selecting either '// LINE-STYLE COMMENTS' or '/* Block-Style Comments */ when creating sample files. Note that many company's coding standards don't only allow line comments.

- Improved the ETEC 'sample' files to adhere to BKM's. For instance, enabling of the global timers comes AFTER timer configuration and channel enablement, TDL negation at start of thread, CPR is set AFTER all init hsr's are set (so all inits executes prior to any pending link events) etc.

- Fixed a crash that can occur in certain rare cases when the main DevTool app is resized.

- Improved panel resize algorithm when the main DevTool app is resized.

- Fixed source code search issue when '.' is specified in the source code search dialog. Only affects external builds.

- Fix a DevTool backwards compatibility issue with MtDt, with regards to source code search paths. This corrects a change that was inadvertently made in the previous release.

- Fix help info for -bd and -ccd command line options

- Set the original file name as the starting point when renaming a file or doing a 'Project save-as' operations. Not a huge issue, it but was annoying.

- Fix detection of end of constant tables in code coverage.

ETEC - Compiler/Linker

- Fixed a bug with handling of legacy (if-else) standard entry tables – the issue could affect assignment of vectors with don’t care pin values.

- Fixed a bug wherein eTPU class method/thread re-definitions were not getting caught at compile time

- Fixed an inline assembly bug in which, in certain cases, variables could fail to be recognized within the assembly. In this case, a compilation error would occur and now it properly assembles without issue.

- Added warnings on invalid #pragma writes.

- Improved ETEC Compiler (eTPU) documentation on the ETEC mode, especially w/r to grouping channels, using multiple entry tables per group, use of member functions that allow a common function to access channel variables, etc.

- Increased the allowed single line length in C pre-processed files (and scripts - this improves DevTool behavior also.) The actual increase is a little hard to know, as there is a parser depth setting that is somewhat mysterious. But significantly bigger than before.

- Improved clarity of several eTPU ETEC linker error messages.

- Fix a compiler bug in certain rare cases in which it was deleting the .eao file (normal output) in modes where it must not do that (this update does not affect normal compilation but only as ETEC is used for the DevTool code reference capability).

- If a structure has an alignment such that the modulo 4 of its size is 3 (unusual; in the last 4-byte word of the structure only the first 3 bytes are used), and an assignment copy from one object of the type to another is done, that last word does not copy correctly. This bug has been fixed.

- Fixed a bug wherein if typedefs of array types are used for the type definitions of function parameters, compilation will fail

3.8 DevTool V2.60C / ETEC V2.60C (2018-Aug-13)

User Manuals

Internal links in the PDF user manuals are now functional and the User Manual's '.pdf' files are now optimized for electronic viewing instead of for printing.

Worked around the Microsoft Windows' but in which the '.chm' help files would not open when the installation is on a non-local drive by using '.pdf' files instead of '.chm' files.

ETEC Toolset (compiler, linker, etc)

Fixed a compiler bug wherein variables declared with a typedef of the _Bool type were not getting allocated properly.

Fixed a compiler bug with string constants as direct arguments of function calls.

Fixed an issue with the line number listing when errors occur in the 'Worst Case Latency' (WCL) file.

Fix a bug in which the last line of a .WCL file could be ignored in certain cases.

Fixed a linker bug. When a pointer to an array was declared using 'extern' but was never actually instantiated, it could trigger an errant 'fatal error' message (instead) of an appropriate 'cannot resolve' message.

DevTool Simulator

Fixed a bug in the Project File in which, when multiple source code search options are specified (and certain other rare configurations), the project file fails to store correctly. Note that this was fixed between Version 2.50 Build B and Build C. It was reported by a customer and led the quick turn of 'Build C'.

Enlarge parsing buffers to support script files with very large lines (post macro-expansion). Previously limit was generally 16K characters; new limit is 10-16 times greater.

Local variable window in multi-target simulations always tracked active target rather than following selection – fixed.

Ensure source code search path backwards compatible with legacy MtDt product.

Fixed a bug in which the linker's 'Analyses' window was not getting reloaded into DevTool following a build.

Miscellaneous User Manual improvements

Improved the Waveform Window redraw rate in certain cases.

Added a -vd=<VectorDefine>=<Val> capability to DevTool that allows a #define to be passed to the vector file. Typically used for conditionals in the vector file (#ifdef <SomDefine) or test-specific frequencies.

License Server

The License Server has been ported to VS2017 and its release version updated to 2.21A. No functional changes.

Fixed two newly-introduced bugs in the license server installer

Fixed an issue that can occur when installing the ASH WARE license server.

3.9 DevTool V2.50A / ETEC V2.50A/ MtDt V4.91A (2018-Jan-01)

DEVTOOL (SIMULATION FRAMEWORK)

- **Added ability to modify values certain types of windows such as the 'Registers', 'Configuration', etc. This is considered to be the most significant missing feature that our previous simulation framework ('Mtdt') supported that 'DevTool' did not.**
- **Added a dropdown selection button in the GUIs top-right corner for ALL OPEN FILES (in the top portion) and ALL OPEN WINDOWS (in the bottom section. This has been a significant issue because these windows can be very difficult to find.**

- Fixed a bug that makes DevTool unusable for multi-target debugging in the case where the number of targets goes from '1' to more than 1 by tweaking the build batch file in the 'Settings' window. Kind of unlikely to hit, but if you do it is nasty.
- The auto-generated list of global variables available in the watch window is now sorted in alphabetical order starting from the variables middle character and working sideways. Just joking. Sorting from the first character.
- Detect enum literal duplicates in scripting language (they cause confusion and interfere with code references to some extent)
- Fixed a minor issue (but it was bugging me) that occurs when one double-clicks on an environment file from within Windows Explorer. It was properly opening the project in DevTool (Yes!) but was also opening the environment file as a text file in the editor. It no longer opens specifically the environment file, but will open other text files.
- Fixed an annoyance when changing executable image files. In certain cases (.cod/.McsLog, etc) it gives preference to a file type that matches the current file type.
- In the 'tab' only the filename is listed instead of the fully-qualified filename in certain cases. This allows many-more windows to be seen in the TabControl.

Editor

- Fix block indent/un-indent to use the spaces per tab setting rather than hard-coded 4.
- Fixed a syntax-highlighting crash that occurred when a preprocessor keep-out is detected AND there are 0 characters past the keep-out start.
- Fix an auto-complete related tooltip crash.
- Fix an auto-complete bug wherein the auto-complete dialog was not being shown at some times when it was supposed to be shown.
- Fixed a situation that was creating a 'bug' warning when right-clicking a header file that can occur in a GNU-like directory structure environment.Cleaned up the message that complained about the situation when a user-defined file has been deleted that was references in an project's environment file, and disabled a downstream message that was (falsely) indicating that there is a bug.

Backdoor Scripting

- Fixed a bug in 'backdoor scripting' (script commands that target an eTPU that are called from within 'c' code running in a simulated CPU with a system simulation model [CPU plus eTPU.])
- Support the read_time() script command in backdoor scripting.

State Machine

- State machine state value in the Waveform window could go wrong for one clock cycle because it was doing a state lookup and a program counter that was being flushed.

ETEC ETPU COMPILER TOOLSET

'C' Compiler (Part of the eTPU Compiler Toolset)

- **Provide a way to explicitly locate global symbols, including into areas above the normal 1K global area, which forces indirect access. This eliminates one of the biggest weaknesses remaining in our C compiler. A number of customers have requested this feature.**
- Fix an inline assembly issue with passing certain symbols (essentially type int16) from C to the assembler that aren't directly supported by the assembler. This has worked in ETEC mode, but was not working in eTPU-C mode (eTPU function, if-else array).
- The type alignment for arrays was not getting set, which caused a false compile error when an extern'd array was being touched in code.
- Test-only flags (such as channel hardware or CC flags) were getting left marked as addresses in the operand stack, which caused the conditional expression operator code generator to fail.
- If array indices were expressions of smaller type (e.g. sint8), the indexing multiplication could be incorrectly generated (only of array element size > 0x7f bytes)
- The duplicate expression optimization could fail in the case where the duplicate operation register gets released within a loop
- Prevent crash case on unknown enumeration used as function argument.
- Fixed a bug in the 'settings' window that was causing stale information from the previous project when the project file is changed.
- When another application is blocking DevTool from being visible, and a file changes on disk, when switching to the DevTool application, the dialog blocks DevTool from becoming the new application, instead, after the dialog is closed, DevTool stays as the non-topmost.

CPP Preprocessor (Part of eTPU Compiler Toolset)

- Fix issue handling malformed #include
- Fix and overly aggressive invalid #include check that was breaking a CPP test.

eTPU Linker (Part of eTPU compiler Toolset)

- Threads are not allowed to be called from other 'c' code and this was generally being properly enforced. However, if a fragment called a thread a bug allowed this unsupported construct to be allowed. To make matters worse, in certain cases this unsupported behavior resulted in linker crash. This bug has been fixed such that the unsupported behavior results in a link-error in all cases.

- Fixed a bug in the generation of the listing file that results in an exception of no code was generated during the assembly process

INSTALLER

- Fixed an installer bug in which the 'Disable Demo Installations' checkbox was not working.
- Fixed an installer bug in which in certain (rare) situations it would (silently) fail to install in Windows 10.

3.10 DevTool V2.40A / ETEC V2.44A / MtDt V4.90A (2017-May-02)

- Fully support simulation of full speed TCR1 (at system frequency) and T2/T4 timing including errata warnings.
- Fixed an issue/bug with the source code search window (used on 'external' builds ... e.g., inside a .bat file) in which source code files STILL could be opened for reading/editing/debugging even after the source code search path had been made 'find-able.' Previously, the source code search dialog would only take after the project was re-opened. Now it is available immediately.
- Improved the multi-file search capability for external directories such that it now has a 'stop' button. Previously, once started the search was not interrupt-able' such that once started it could not be stopped, even if it had hopped on a comet going to the Pleiades. Authors note - I once gave a hitchhiker a ride who claimed to from the Pleiades.
- Fixed issues with verify value script commands when using binary syntax in the address (previously only hex was supported, see below)

```
verify_val_int("*((int*)0x1)", "==", 0x1234);  // worked
verify_val_int("*((int*)1)", "==", 0x1234); // failed
```

- Improved warning messages in certain cases such as including the channel number action unit identifier in certain channel hardware warnings and including the data address in certain bad data accesses.
- Fixed a message issue in the 'Counters' window that was indicating an odd angle mode setting combination (AM=01b and tcr2clk=110b) was invalid when it is actually just ... odd.
- Fixed name-clash reporting issues in the waveform window in which the name assigned by the window clashes with the name assigned in the vector file thereby making such issues easier to resolve.
- Fixed a startup marker issue in which markers are not visible at startup in certain situation because they were being hidden by text-search markers.
- Vastly improved simulation speed in the case where the simulation is single-stepped and was getting hugely bogged down by an invalid variable reference is in the watch window.
- Fixed a erratic scrolling issue that was occurring when source code file has more than 4000 lines of text.
- Improved error message in some script errors such that more detail than just 'parse error' is provided.

- Fixed a bug that would cause a huge delay in certain cases when the popup menu is opened in a source code window.
- Ignore #pragma directives in the scripting environment; this makes the tool more flexible in including shared source code into scripts.
- Fixed some issues when using non-standard file source code file suffixes (e.g., <FileName>.cetpu instead of <FileName>.c).
- Improved Compiler Documentation.

Threads Window Improvements

- Fixed two issues with the Threads window:
- Store on the earliest rather than the latest. (Makes resetting and re-running to worst case thread optimally fast.
- Include the 'RAMS' in the Worst Case Thread Length determination such that two threads with the same number of steps but different number of RAMS are no longer considered to be the same length. Instead, the WCT with the most ram accesses is considered worst case.

3.11 DevTool V2.30G / ETEC V2.43G / MtDt V4.89G (2017-Jan-11)

Added Support for a Trace Window

- A new Trace Window is now available with tons of cool features.
- Increased the size of the trace buffer 10X, ..., from 200K data logs to 2 million logs.
- Fixed a bug in the trace data when recording thread starts. The entry table is a 5 bit field but bit 5 was being masked such that it was always a zero. So, for instance, if the entry index was 0x14, it would be listed as 0x04 instead.
- Fixed a bug in the recording of TCR2 counter increments in which the TCR2 counter was not deterministic when initially running a simulation versus when resetting and re-running the simulation. Specifically, an errant 'tcr2=0' trace log was being generated when resetting and rerunning a simulation.
- Fixed an issue with the trace data where the fetching of the special breakpoint opcode can inadvertently get saved into the trace buffer. For example, consider the following two cases. A simulation is run, a breakpoint is hit, and the user hits continue such that the simulation then continues to the end of the test. In a second case, the breakpoint is removed and the test is run to completion. In the first case, an extra opcode fetch will appear in the trace represented the breakpoint was hit. This issue has been fixed such that the breakpoint opcode will no longer appear in the trace data. The trace data will be identical in the two examples.
- Fixed an issue with the trace data where the thread's 'Worst Case Thread Length' information can be stored to the wrong channel in cases where the channel register was changed (but not restored) during the thread. In this case the thread's WCTL is calculated relative the new channel number rather than the original channel number. For example, if a thread began on channel 5, but then the 'chan' register was written during the thread to '8', the thread was counting toward's channel 8's WCTL data, etc, which was wrong. This affect both the view in the trace window as well as the data exported to a trace file.

- Fixed an issue with the trace data where a thread-start preload can appear twice in the trace data. This affect both the view in the trace window as well as the data exported to a trace file.
- Fixed an issue with the Mtdt trace where the opcode fetch was the same trace type as the memory read. This allows (say) disabling logging of instruction fetches while still retaining memory reads, thereby making more effective utilization of the window space. The opcode fetch is now considered to be a separate trace log type thereby allowing the disabling of relatively un-interesting but plentiful opcode fetches while retaining the far more important data reads. (note that this is changed from Mtdt. Mtdt traces dumps may not be identical to DevTool trace dumps due to this issue.
- Changed the DevTool trace behavior. Previously (and in Mtdt) the separator separated instructions and also TCR1 and TCR2 increments. It is no longer being used to separate TCR1 or TCR2 increments.
- The 'Divide' trace log now only generated prior to execution of each opcode. It no longer separates log types such as TCR1 increment, TCR2 increment, etc.
- The 'Pin toggle' indicates when the unusual case occurs in which an output pin is commanded to a new value but it was already at that value. E.g., 'pin was command high but was already high'.
- The trace window lists the channel name, class or eTPU Function name, and the thread or label name. Note that in legacy style eTPU functions a label should be placed at the very first line of a thread in your source code for a variety of reasons, including to make the trace window work better.
- Fixed an issue in which in certain rare conditions a spurious HSR clear event could be generated following a reset-and-restart of a simulation.

Watch Window Improvements

- The symbolic expression parsing capability has been greatly improved and overhauled. This change improves the capabilities of script commands such as verify_val(), write_val(), etc., and also allows more complex symbolic expressions to be processed in the Watch Window.
- Expandable symbols in the Watch Window allow selection of a child in the Symbol Name dropdown listbox.

Waveform Window Improvements

- After modifying settings to Waveform Window and committing to those changes by hitting the 'Ok' button, the window doesn't always get immediately updated, instead sometimes the simulation must run a bit more in order for the new settings to appear in the window. This has been fixed.
- When trying to rerun a simulation to the current time by double clicking on the 'Current Time' field in the waveform window, in certain situations the simulator would reset to zero then would not actually run to the original current time but (instead) would give an errant message saying it cannot go to the requested time. This has been fixed.
- When trying to reset and various times by dragging and dropping from the waveform window's time labels, in certain situations the request was being ignored. This has been fixed.
- The state of the angle mode counter can now be viewed in the Waveform Window as an enumeration.
- Fixed a bug in which the value of the HSR (being considered part of the 'Channel Flag') was not visible in the waveform window as a horizontal colored line when the other channel resources such as LSR, MRL_A, etc are enabled for viewing.

- Effectively added ability to Save/Restore waveform window 'Views' using the Trace window's markers capability. When a marker is created in the trace window it stores the waveform window's 'View' (cursors, borders, etc). When jumping back to a previously-set marker, the waveform window's view at the time the marker was originally created is restored. Clicking between multiple markers instantly changes between the multiple waveform window 'Views'.

Scripting and the Script Window Improvements

- Script File Octal compatibility. Script files had been ignoring the leading '0's and octal numbers were being incorrectly interpreted as decimal. However, in 'C', leading zeroes signify an octal number. So this has the following compatibility repercussions in the unlikely case that existing script command files use octal. 1) Octal numbers will now be interpreted as octal. For instance, 014 will now be correctly parsed as '12 decimal' instead of '14 decimal'. Illegal octal numbers are now interpreted as zero. For instance, '08' is illegal since the octal character set is 0...7 and therefore the '8' digit is not allowed.
- verify_wctl() script command. In the 'verify worst case thread length' script command the number of RAM accesses is now considered in the cases where the number of instruction steps is equal such that (say) a thread with 10 instructions and 3 rams is considered to be worse than a thread with 10 instructions and 5 rams. Previously, the number of RAMS was not considered in this case thereby yielding incorrectly optimistic results in some cases.
- Script file variables can now be referenced in the print_to_trace() script command, and from the AshPrint feature.
- The script commands write_val(), etc. could incorrectly write data to memory if the target symbol was a bitfield - this has been fixed.
- Constant expressions in the scripting environment were not handling signed values properly because each constant, if signed, was first being converted to unsigned. With this correction, a constant expression such as (-9 / 3) will correctly evaluate to -3.

Floating Windows

- This is the ability to float windows independent of the main application. It is especially helpful when using multiple monitors. This capability has been completely overhauled with numerous improvements and bug fixes.

Editor Windows

- Significant improvements to editor syntax highlighting and usability have been made, particularly when used in conjunction with floating windows.
- Undo/redo will return a file to a clean/saved state when an undo/redo action returns the file to a match of the last file save.
- The code reference capability has been improved, however, note that code reference updates are only made when a project dependency file is saved, and then updated based upon each file on disk, not the open editor version (if different).

Miscellaneous

- Added the ability to build code using a copy of the ETEC Compiler Toolset that has been copied but not installed on a computer. This is important for organizations that want to configure the toolset on multiple computers without having to go through the installation process on every computer
- Added a new '-Minimize' option used in regression tests such that DevTool launches and runs 'Minimized' such that it is now possible to both run a regression test while continuing to work on your computer. Note that this can have the added benefit of increasing the simulation speed which in one case was measured at 25% faster though it very much depends on the specifics of each simulation so results will vary.
- Added a command line option '-exe=<MyCodeImageFile>. Allows the project's code file to be changed.
- Fixed a bug in the memory window's 'Goto Next Bookmark' logic that was preventing it from working in some cases.
- Mixed source/assembly can only be viewed for files of the currently active target. This restriction has been mistakenly lifted in the last release, leading to several issues.
- If a file is removed from the build, it could leave old dependency information in place leading to a false report of a dependent file changing during a simulation session, ending the simulation session. This problem has been resolved.
- Under certain rare cases the C preprocessor used to find macro references and definitions for the Code Reference features was crashing, which could bring up a crash dialog that (temporarily) hangs the DevTool IDE : 'eTPU Pre Processor, part of the Embedded eTPU C (ETEC) toolset has stopped working'
- Improved the 'Toggle Mixed Assembly' button so that it does a better job finding the Top-Most source code window to toggle. For instance, it will toggle the top-most source code window mixed-assembly/source-only view even if the active window is the trace window (as long as there is just a single top-most source code files.)
- The resolution of symbol information in the Memory Window, as shown in the cursor Tooltip, was buggy - incomplete or incorrect information could be shown in some cases, or worse, hangs could occur. This has been fixed.
- Network license checkout reliability has been improved.
- Fixed an issue with the Network License
- Added ability to remove a Worst Case Latency (WCL) file from the internally built link. The WCL file could be added or changed just fine ... but once added it could not be removed. This is now fixed in that the WCL file can now be removed.
- Fixed a problem in which an avalanche of Behavior Verification failures can significantly slow the simulation when running regression tests. A limit of 25 behavior verification failure that get logged to the log file has been implemented. Note that this does not disable the user from becoming alerted. Instead, the avalanche of dialog boxes alerting the user can still be disabled in the Messages Options dialog to keep the annoying avalanche of failures from. This change only affects the log file.
- Build is now terminated on the first compilation unit that experiences a compilation error.
- The test name passed in the -tn=<SomeName> does not appear in the log file when using the -AutoBuild but not the -AutoRun command line options. (NOTE: that command line options -lf5=<Sim.Log> and -q also must be used to get a log file.)

- A pre-build Windows' Console (.bat) file can be specified in the internal build's build folder. This .bat file executes prior to the build and can be use for things like (say) a 'Clean.Bat'.
- Essentially all cases of temporary file creation have been eliminated - temporary files should no longer get created or left behind when using DevTool.
- Renaming of a state machine is now properly supported. The rename will also updated text in the state machine's associated source files, as well as the associated source file names.

3.12 DevTool V2.20J / ETEC V2.42J / MtDt V4.88J (2016-Jun-22)

General

- The 'Make' capability has been extended to non-source code, 'Script Commands' and 'Vector' files such that a change in these files or files they depend on (e.g., an included header file) such that changes in the dependent file will also cause DevTool to leave 'Simulation' mode and enter 'Edit' mode. This is important because it causes the latest version of these dependent files to be used in the simulation.
- Fixed miscellaneous false diagnostic warnings.
- Improved 'Most Recently Used' (MRU) be behavior when windows are closed ... such that the last-viewed window becomes the front-most instead of some random window.
- Fixed bug in language support in the Turkish (and possibly other) languages. Would trigger a 'Fatal Bug' at startup effectively preventing DevTool from opening when the computer's culture (not language!) was set to 'Turkish'.
- Updated to latest 'AN4907' Engine Demo.

Project/Environment Files

- Improved message displayed when the less important 'Environment' file could not be loaded (as opposed to the much more critical project file!)
- When DevTool is minimized, then closed (while still minimized) and the environment file is saved, then when the DevTool is re-opened, it opens really small. Basically the size and location of the minimized window were being restored, rather than the normally-sized, pre-minimized size and location. This has been fixed.

Text Editing Windows

- Generally improved the 'Code Reference' feature (ability to right click on a variable or #define and have auto-scroll to the file & line of the originating reference.)
- Improved speed of opening very large files caused be slow syntax highlighting routine.
- Improved source code and script commands file auto-complete in certain cases.

- Improved the 'goto cursor' function in certain cases when no code is associated with the current line ('find nearest valid line' logic improved.)
- Fixed a bug in the editable windows in which drag and drop was incorrectly causing the text to be deleted from the source.
- Fixed miscellaneous issues associated with the viewing & debugging of source code files when being viewed in mixed source/assembly mode.
- Fixed issues with undo/redo associated with cut & paste.
- Implemented a workaround to the 'Visual C' bug in which files saved from the Visual C editor were not being automatically detected as having been modified and therefore were not getting automatically reloaded into DevTool.
- Implemented a workaround to the 'McAfee Anti-Virus' bug in which certain Anti-Virus versions (not the latest) would falsely indicate that editable files parsed during code build (.c, .h, etc) were being written. This caused the file to be reloaded into the IDE, incorrectly treated as 'Dirty' (having changed) beginning an endless re-build cycle that effectively disabled DevTool. Note that the latest McAfee Anti Virus version at the time of this writing (2016 June 14) does not exhibit this bug.
- Source code files that change on disk that have not been modified within DevTool (are not 'dirty') are now automatically reloaded into DevTool without triggering a dialog box. Note that if the file had been modified by the user within DevTool, and the version on disk is detected as having changed, then the user is queried if DevTool should retain the changes within DevTool or use the version on disk (by reloading within DevTool) instead.

Code Building

- Fixed a bug in which target with it's build disabled would get (incorrectly) enabled by -AutoBuild command line option. This causes problems in situations where two targets share code such as a dual eTPU where eTPU_B uses the build from eTPU_A in that it would actually build both eTPU_A and eTPU_B when running a regression test from the command line but not in development. It was done this way to cause (say) demos that use the GNU compiler to build the GNU code, but this was not the right way. Instead, the '-enableCodeBuild=<target>' command line argument should be used instead to cause a disabled target build to actually build.
- Added a '-EnableCodeBuild=<target>' command line tag for DevTool tests that have their target builds set to 'Disabled' in order to enable the target builds. Note that this is now needed now that the -AutoBuild bug is now fixed. See previous bug.

Scripting

- Implemented 'Action Tags' (also referred to as 'ASH' tags) in DevTool. These allow 'smart' commenting such as the underlined source code, below. In the watch window, the timer named 'TestTag' is now visible that shows the last traversal time, the total time spend traversing, the number of clocks it takes to traverse, and the number of times the code has been traversed.

```
if ( HostServiceRequest == PWM_INIT )
{
     DisableMatchesInThread();  // @ASH@timer_start("TestTag");
```

```
        OnTransA( NoDetect );
        OnTransB( NoDetect );
        Clear ( TransLatch );
        Clear ( LSRLatch );
        EitherMatchNonBlockingSingleTransition();
        Set (flag0);                    /*after init frame edge should be
    serviced 1st*/
        Flag = Flag | 1;  // @ASH@timer_stop("TestTag");
    }
```

- Implemented a set_link() script command to inject a link into a specific channel. This is actually quite helpful in testing functions in isolation that normally are used in conjunction with other functions.
- Fixed accidentally broken 'switch' construct support in script files.
- Improved stepping in script commands files in which a variable is initialized. Previously, had been being treated as two lines.
- Improved GUI behavior when a script commands file error occurs. The offending script command file pops to the front, and the offending line scrolls into view and is highlighted prior to opening of the dialog box.
- Add missing MRLE bits from the ASHChannel structure. Used in script commands such as write_val("@5.ASHchannel.MRLE", "1");
- Misc improvements to the search and replace dialog functionality including ability to search mixed assembly files.
- Removed the 'eTPU2-Only' text from various windows ... pretty much axiomatic ... very little these days still using eTPU1.
- Make the 'Scheduler', 'Host Interface' and 'Counters' windows default to open in the (wider) output 2 Tab-Control where they are easier to read.

Waveform Window

- When the 'Auto-Scroll' is selected it now immediately auto-scrolls. Previously it would not auto-scroll until the next single-step, or other such event.
- Allow the 'chan' variable to be viewed in the waveform window as an color-code enumerated type ... VERY COOL FEATURE!!!
- Fixed miscellaneous issues with the display of color-code enumerated types in the waveform window (a super cool feature, btw.)
- Fixed a bug that was causing a diagnostic warning associated with deleting the viewing of variables in the waveform window.
- Improved eTPU Channel Signal listing of MRL/TDL/etc. flags.

- When a named node changes in the vector file, if that same named node is being viewed in the vector file as-is (not aliased) then the name change in the waveform window follows the name change in the vector file.
- When the waveform window is displaying a node using the built in name, and there is no alias or node naming in the vector file, any newly-added node name in the vector file will be reflected in the name listed in the waveform window.
- If the waveform window is displaying a named node from the vector file, and the vector file is editted such that the node is no longer named, then the waveform window should display the built in window and not the no-longer-valid vector name.
- Any aliased name in the waveform window is not affected by changes in the vector file.
- Fixed a bug in the waveform window's dialog in which any chances made within the configuration dialog might not get restored when the user hits 'Cancel'
- No longer allow user-aliases that clash with built in names or named nodes in the vector file.
- Misc fixes to the 'Channel Frame' window.

Debug Windows (Watch/Channel Frame/Local Variable)

- Added ability to open and scroll the memory window to a variable's address. Oddly convenient for 2016.
- Fixed a bug which would cause invalid warning dialogs, and sometimes removal if temporarily invalid variables being displayed in the watch window.
- Improved debug window (watch, local variable, channel frame) handling of 'Address' and 'Type' columns when a variable is not longer valid (e.g., goes out of scope) such that these fields get listed as 'unavailable'. Previously they retained their last-good value which was confusing.
- Filter code addresses from waveform window's variable list (annoyance.)
- Improved 'typedef' handling in watch window structures. In certain cases this allows variables to be viewable in the debug windows that previously could not be viewed. IF YOU WERE HITTING THIS ISSUE, THIS CAN BE HUGE!
- In the Local Variable window, support locking on to a specific target.
- Allow “@<chan num/name>.” references to work in DevTool Watch window.
- Fixed the edit of the watch Value (and possibly other columns) such that the TextBox that pops up to enable editing has it's alignment set to the same as the underlying text field. (it was annoying for the alignment to change when editing)
- Save the expanded state (e.g, if a structure's members are expanded such that they are visible) of the watch window across project/open/save.
- Added tool-tip text to watch window.

- Fixed miscellaneous additional watch window issues.
- Added copy/paste capability to the 'Watch' window.
- 'Clear Watch' menu item now becomes grey when watches cannot be cleared.
- Remove display of drag's x/y coordinates from status window (diagnostic instrumentation had accidentely been left in the 'activated' state)
- The 'Add New Symbol' row could be drag & dropped ... now disabled. Note that this should always be the last row.
- The 'Add Symbol' row can no longer be selected.
- Multi-select drag & drop of watch elements now supported
- Various standard windows 'Cut'/'Copy'/'Paste' keystrokes were not implemented in the watch window, but now are.
- Fixed an issue in which the watch window was not being brought to a proper reset state when the project file is changed.

Memory Window

- Fixed a lockup bug in the new Memory Window that can occur when the mouse is hovered over an address of a symbol that is of size zero (e.g., could happen in complex structures/unions)
- Now possible to open the memory window and bookmark or scroll to the location of a variable, directly from any of the 'Local Variable', 'Watch', or 'Channel Frame' debug windows.
- Generally improve symbol references when hovering the mouse over the memory window.

3.13 DevTool V2.20A / ETEC V2.42A / MtDt V4.88A (2016-Jan-05)

- Vastly improved the memory window including panning, editing data, bookmarks, copy, paste, multi-select, DC to Daylight range, etc.
- Added ability to view the ETEC-generated 'Analyses' files which display information such as Worst Case Thread Length, Worst Case Latency, each source code file's code size, etc.
- PROBLEM: When the 'Auto-Build' fails as part of a regression (command line argument is -AutoBuild) such that the DevTool is automatically closed, it was not possible to tell WHY the build failed. SOLUTION: in this case, the output window is now sent to a file named 'OutputWindowDump.Log' that contains the error messages required to debug the build issue.
- Changed behavior when a target's build is disabled AND -AutoBuild is passed on the command line. It had been that this would cause a 'FAIL' however now (instead) the 'disable' is overridden and (instead) the build is enabled.

- Added a '-NoBuild' Development Tool command line option that prevents a code rebuild even if the code is out of date. This allows (say) a series of tests to be run on a set of source code without an executable code image getting re-built over and over again, thereby saving time. Note that this option adds a tad bit of risk that your code could potentially be out of date and would no longer work correctly if it were to be rebuilt.

  Question: Why not just disable the code rebuild by not passing '-AutoBuild' on the the command line? Answer - when '-AutoBuild' is not specified on the command line a 'Make' occurs in which a build may (or may not) occur depending on the many files' time stamps. However the make timestamps go into the environment file (yuck - this should be fixed) which generally don't get saved as part of a regression test suite. So generally, even if '-AutoBuild' is NOT specified on the command line, your code is likely going to get rebuilt on every test run. The '-NoBuild' option overcomes this by forcing your code to not be rebuilt.

- Added ability to display in the waveform window 'discrete nodes' (e.g., variables, registers, etc) in CPU code for system simulation. (Already available in the eTPU and MC33816.)

- Added code coverage to CPU (especially helpful for cases where code is unexpectedly not executing.)

- Enabled waveform discrete node capability for CPU32 target.

- Improved editor operation with regards to syntax highlighting, undo/redo, and general behavior.

- Improved the Output Window's 'Click-to-line' capability when doing external builds or internal builds in which the ETEC compiler/linker verbosity is at 5 or above.

- Added menu options and hot-keys for reloading of externally-built code (e.g., when built by an build utility not under the control of DevTool)

- Cleaned up Debug windows (watch/channel frame, local variable, etc) to use better font & formatting.

- Added address column to Debug windows (surprisingly helpful in certain cases.)

- Development Tool comes with a standard five 'Built In' panels, however it is possible to add additional panels called 'user defined' panels. When a user-defined panel is defined within another user-defined panel it can trigger a false diagnostics error, 'Attempt to close a built-in table control???' This has been fixed.

- Fixed a newly-introduced bug (Version 2.10) in which when dragging a vector or script commands file to the reference folder (all within the project tree) the file becomes hidden at it's original (non reference folder) location where the drag began.

  - Fixed a memory leak that in certain cases could cause an 'out of memory' error in certain cases after running the simulator for awhile.

  - Fixed numerous bugs including auto-reload file when changed on disk, occasional crash when changing project files, issues with dragging and dropping in project window, several situations in which 'internal diagnostics' messages were getting triggered, instances in which background processing could cause a sluggish editor.

  - Improved cleanup of temporary files that were getting left around in certain situations.

- Fixed a bug when a project file double clicked from windows explorer and a DevTool was already open.

3.14 DevTool V2.10C / ETEC V2.41C / MtDt V4.87C (2015-Nov-01)

- Fixed numerous (>45) minor 'nuisance' bugs since Version 2.10 Build A.
- Added ability to view a variable in the Waveform Window (shows value over time) directly from the source code window without having to (first) add it to the 'Watch' window.
- Numerous Watch Window improvements including the ability to view Script Variables in the watch window. (Version 2.10 Build B only)
- Variable debug windows (e.g., Watch Window, Local Variable window, Channel Frame Window, etc) now highlight variables (color red) that have changed since the last time the simulator was stopped.
- Added an persistent 'Auto-Fit' setting to the Circuit and State Machine windows in which they are maximally zoomed to fully fit into the entire window, every time the window size changes. (Version 2.10 Build B only)
- Fix a waveform window issues with multi-target environments that prevented easy change of waveform node names and other issues. Also fix some waveform display of data variable issues.
- In multi-target environments, fix a spurious pop-up dialog warning message that occurred when a window that didn't support the active target was updated.
- Improve support for multiple-target script and vector handling. Note that the recommended script configuration is a single script file associated with the first target in the project.
- Right-clicking on an in-scope variable during simulation now provides the option to add it directly to the waveform window for graphical viewing.
- The MC33816 Simulator has been released to production.
- Added an important new demo that illustrates end-to-end 'Moving to Hardware.' This demo co-simulates a Host CPU and in fact drives the MC33816 using only a series of optimized-for-speed SPI accesses. The MC33816 is fully controlled by the host CPU code using only SPI accesses, thereby fully illustrating every required aspect of running the MC33816 from the host. Of particular importance is the compiling of the MC33816's startup script (that fully initializes the MC33816 registers at startup prior to running the MC33816) into a series of optimized SPI block writes and word writes of the registers and any initialized data RAM variables, including banked data RAM variables. (Version 2.10 Build B only)
- Added a eTPU/MC33816 demo that illustrates co-simulation of an eTPU and MC33816. The MC33816 uses one of it's channels to drive four injectors. The four injectors are organized into two banks, and each MC33816 controls one bank of two injectors. The eTPU monitors an incoming CAM signal to generate the four 'START' pulses. (Version 2.10 Build B only)

- Added an _ENTRY tag to MC33816 Assembler. This is used to mark certain labels as code entry points such that no warning is generated for an unused code label. This is similar to the existing _ISR tag that (Version 2.10 Build B only)

- Complete Simulation to Hardware path is supported. All register initialization under simulation is done using a startup script, and this same startup script can be compiled in a 'C' compiler and converted to a SPI Block writes. The Data RAM's initialized values are output from the assembler into a data file that also can be compiled in 'C' and transferred across the SPI bus. Run time scripts, when written as SPI writes (the new way of scripting) can also be compiled in 'C' and written across the SPI bus. In conjunction, these capabilities allow the exact simulation input to be run on the hardware, thereby 'Connecting the Dots between Simulator and Hardware'. (Version 2.10 Build B only)

- Less than the guaranteed two-cycle availability on reads and writes as well as not keeping the SPI address register (spi_add) constant result in warnings as do invalid access addresses and access of invalid locations result in warnings. These warning can be disabled in the message dialog. Note that these violations actually generally work ok, the issue appears to be sporadic unavailability based on SPI bus activity resulting in occasional collisions.

- Support for use of the IR register in SPI Backdoor reads/writes now supported.

- The Bootstrap sequence is now supported in that the value of the 'Bootstrap_charged' register (0x1A5) reflects the injector connections. However, the simulation begins after the bootstrap sequence has already completed so the bootstrap_init_timer field contains 0x34 and the hsx_src_1V and hsx_bs_charged fields reflect the presence/absence of a charging path (due to presence of injectors, high side driver shorts or injector opens.)

- Supporting many additional simulation configurations:

  SINGLE CORES:

  - Ch1Uc0
  - Ch1Uc1
  - Ch2Uc0
  - Ch2Uc1

  DUAL CORES:

  - Ch1Uc0, Ch1Uc1
  - Ch2Uc0, Ch2Uc1
  - Ch1Uc0, Ch2Uc0

  QUAD CORE:

  - Ch1Uc0, Ch1Uc1, Ch2Uc0, Ch2Uc1
- Note that there was no missing feature associated with testing (using for example 'jocr <SomeLabel> ocur;' or similar test with 'wait';. The manual is confusing, however there is no 'microcode enablement of these tests required for these tests to yield proper results based on the value of the registers Curr_block_access_1 (0x188) or Curr_block_access_2 (0x189) as described in the FSL MC33816, revision 4, dated 4/2014 on page 186.

3.15 DevTool V2.00E / ETEC V2.40E / MtDt V4.86E (2015-Sep-02)

- Fixed a bug in which grammatically-incorrect script commands were being accepted. In certain cases, script commands were allowed when the number of closed parentheses did not match the number of open parentheses. For example, the following was allowed even though it is invalid.

```
write_reg5(0x17), REG_CHAN);  // Missing close-parentheses
```

  Note that the above is now (properly) generating an error message. The proper script command is below.

```
write_reg5(0x17), REG_CHAN));
```

- Fix/improve handling of ‘overwrite vs. insert’ mode in editor windows. In overwrite mode, cursor becomes a solid box to make it obvious that the mode has changed.

- Fix a USB license dongle issue with MC33816 DevTool wherein when a project had circuit windows, the application could take a minute or more to open.

- Properly supporting registers 'Flags_source' (0x1C3), 'Flags_direction' (0x1C1) and ' 'Flags_polarity' (0x1C2) as well as the I/O pins and flags associated with those registers (FLAG0-FLAG2, START1-START6, IRQB, OA_1, OA_2, DBG, internal flags FLAG12-FLAG15.) As part of this implementation/fix the following issues were addressed:

  - Use of the input pins and output pins as 'flags' is now supported. This includes using flags to read input pins, write output pins. See MC33816 User Manual Rev 4, 2014-April page 96. This also includes the OA_x pins being used as input/output flags.

  - Fixed an issue with the Flags_polarity Register (0x1C2) not getting applied to the start pins as part of start management.

  - The IRQB pin was being driven with the IRQ state even when configured as a generic I/O pin. Note that for the IRQB state to be driven unto this pin, register 'Flags_source' (0x1C3) bit 9 must be a zero. However the default is for the IRQB pin to be a generic input pin.

  - When driving one of the pins associated with a flag (E.g., the IRQB pin is associated with FLAG 9) as a generic Output pin, then the pin follows the 'AND' logic of the four cores' version of flag 9. In other words, the pin is a '1' only if all four cores have executed the 'stf high b9'. instruction. However, the pin will be a '0' if ANY of the cores has executed the 'stf low b9;' instruction. Note that this logic can be easily missed because the internal flags for all four cores reset to all 1's.

  - When a pin is configured to be a generic input pin instead of it's normal function (e.g., Start 1 is a generic input pin) and the invert bit is set (Flags_polarity Register (0x1C2) bit 3 is set) then the value read by cp flags ir; or tested by jocr flag3; should be the inversion of the input pin, but it is not. The invert bit was having no affect in this case.

  - Flags_source Register (0x1C3) bits 1, 2, and 3 (corresponding to the FLAG0, FLAG1, FLAG2) are 'hard-wired' as 1's. However, these values were being allowed to be cleared to 0's which was not how it works. These bits now remain 1's, even if written to 0's.

- Renamed the 'I/O Pins' window to be the 'Flag Pins' window and greatly improved the window in terms of fully providing all the information required to determine what is going on with the pins in all cases. Moved

the 'DRVEN' and the 'RESETB' pins to the 'Config' window as they no longer are appropriate for the 'Flag Pins' window ... not actually being flags or following the patterns of those pins.

- Added documentation for the MC33816_SPI_SPACE 2 which can be used to access registers and the data RAM's.
- Provide labels as auto-complete options for ldjr1/2 instructions.
- Fixed a visualization issue with the Voltage seen in the Injector window. The VBatt Vds is showing the incorrect voltage. Note that this is just a visualation bug ... (importantly!) the simulation model is functionally correct.
- Added File 'StdDefMc33816.h' is missing the Err_ucXchY_1 Registers (0x162, 0x164, 0x166, 0x168). NOTE: This was actually fixed in the Version 2.00 Build 'C' version released on 2015-Aug-3 but was not documented in that release.

3.16 DevTYool V2.00C / ETEC V2.40C / MtDt V4.86C (2015-Aug-03)

- Added File 'StdDefMc33816.h' is missing the Err_ucXchY_1 Registers (0x162, 0x164, 0x166, 0x168). NOTE: Previous versions of these release notes fail to mention that this was actually fixed in this release.
- Added the ability to initialize global variables and databank variables.
- Added the ability to load the address of a variable into the 'ir' register using the 'LOAD_IR @<VariableName>; syntax. Note that the address of a variable, databank, and databank member can all be loaded.
- Add the ability to specify initial values for variables and databanks. Initial values are output in auto-generated 'C'-compliant files for use in script files and host code.
- Fix the output of databank type definitions to the ELF file, so that databank variables can be properly viewed in the watch window.
- Fixed the OutputWindow such that 'Click-To-Error' works even when the directory name has a '(', ')' or '-' character. These can occur when the user makes a second copy of a directory.
- In the watch window, when the <add symbol> marker is clicked on a dropdown combo box appears that provides register and global variable options for watch selection.
- Support code references (script function and variables) in script files.
- Fix an editor crash in the case where multiple lines are selected and then a keystroke is made.
- Remove an unhelpful 'invalid target designator' warning in the editor.

3.17 DevTool V2.00A / ETEC V2.40A / MtDt V4.86A (2015-Jun-24)

- **Misc**
  - Improved 'snappiness' of editing source code files by improving the performance of syntax highlighting.
  - Fixed a bug in which if a file is changed on disk and is reloaded into the simulator, it fails to get re-parsed if it was a script file, and fails to trigger a rebuild if it is a source code file.
  - On an automated test it is possible to override 'fake' a passing '0' exit code by closing using the files->exit menu. This now results in a '1' (fail) exit code.
- **Scripting**
  - Support enum type definitions and literal usage in the scripting environment.
  - Improved script file stepping and breakpoint handling such that when breaking at an instruction following a timing instruction (e.g. 'wait_time') the timing instruction fully executes before simulation is halted.
- **State Machines**
  - Added multiple-state/transition selection capability via dragging the mouse
  - Added ability to horizontally align all selected states and transitions (previously, only 'vertical align' was supported)
  - Added ability to create snapshot image files (.png, jpg, etc) of the state machine
  - Added ability to scroll view via mouse drag
  - Added ability to zoom-in and zoom-out
- **Multi-Target/Core Simulation**
  - Support co-simulation of an eTPU with a MC33816! (Added build batch file for 1 eTPU target + 1 MC33816 core, a second model with 1 eTPU and the full 4-MC33816 cores to follow)
  - Added a model for co-simulation of the eTPU and MC33816 together (e.g. allow eTPU's output channels to drive the MC33816's 'start' input pins.)
  - Added extended scripts for buffers, gates, 'OR', 'AND', 'XOR', 'NOR', 'NAND' and 'NXOR' to drive pins between targets. (Example: the eTPU's output pin 4 and output pin 5 could be connected to an 'OR' gate and the 'OR' gate's output could drive the MC33816's 'START3' pin.
  - Fixed multi-core/target debugging issues (e.g. stepping, breakpoints, goto cursor, etc) in the case where the cores are different executable image files (often the case.) The source level debugging is now updated relative to the active core so that things like single stepping/breakpoints/etc will occur for whichever core is active (but NOT for the inactive core!)

- Mixed assembly view is now done relative to the active core/target such that if a source code file is not used by the active target then in dis-assembly view will show there being no dis-assembly. This is consistent with the approach of making DevTool more 'active target/core' focussed such that information for any inactive cores/targets is less 'in the way' thereby reducing confusion.
- In multi-target simulation, fixed a problem in which only the first target/core's signals would display correctly in the waveform window.
- Changed the GUI's 'Multi-Target' behavior of script commands files when only a single target has a script commands file (which is generally the case). Single-stepping the script commands file can be done from ANY target and does NOT change the active target. Also, breakpoints, goto cursor, etc, work even if the script commands file is not in the active target.
- In multi-target simulation mode, added the ability on the 'Memory' and 'Wait Rows' (33816 only) windows to specify the target/core for which the window should render information.
- Fixed a bug in which the analog portion of the simulation would not occur unless Channel 2 Core 1 is enabled to execute code (33816 only). This includes simulation of circuits (DC/DC Converter, Injectors, etc, as well as an external gates placed for (for example) drive the start pins from an eTPU's output pins.
- Fixed a Waveform Window bug in which there was only a single copy of the 'wait', 'timer EOC', and 'Start Event' indicators. Each core now has it's own copy

- **Circuit Windows**
  - Added ability to edit fields (as applicable)
  - Redesigned ALL the schematics. The HS, LS, and Current Sense schematics have a complete makeover.
  - Fields that have changed since the last simulation pause, single step, goto cursor, etc, are highlighted red. Remainder are now blue.
- In the waveform window, fixed an issue with auto-ranging analog signals, esp. when highly zoomed.
- Fixed a bug in the 'Replace Text' dialog such that when the replacement text is a superset of the text being replaced, it doesn't try to replace the same text over and over again.
- Fixed a bug wherein the running of simulation after creation of a discrete node could sometimes cause a crash.
- When auto-running the simulator as part of a test suite, a failure to load a script file would halt the test suite (bad.) Instead, the DevTool now exits with an error code thereby allowing the test suite to proceed (but with errors.)
- The 'Timers' window is not function yet was showing up the 'View' menu. It is now removed from the view menu to avoid confusion. Note that the 'Timers' window allowed 'start' and 'stop' tags to be

embedded in the source code such that the amount of time it takes to traverse from one spot in the code to another can be measured. This will be enabled in a future version of software when it is correctly supported in DevTool.

- Fixed bug in which ALL feedback registers (0x180, 0x181,0x182, and 0x183) were getting reset to 0x4403. The four cores' feedback registers are now being correctly reset to 0x440C, 0x880C, 0x1030, and 0x43C0, respectively.
- When exporting SPI-commands files, made the 'Unsupported script command' message disable-able so that one can create the spi-commands file without having to click through zillions of dialog boxes.
- Improved several MC33816 error/warning messages in various cases (e.g. on an 'invalid immediate value' error would report 'missing jump destination' instead', etc.)
- Modified the MC33816 memory windows such that the address field shows word addressing rather than byte addressing.
- Removed the spurious warning when an output driver is configured as 'undef.' Also, added warning when a diagnostic comparator is queried using 'wait' or 'jocf'.
- Extended Buffers and Gates capability such they can 'drive'/'be driven by' the MC33816. For example, the MC33816's IRQB pin could drive (say) the flag0 input via placement of an logic device using the 'place_buffer();' command.
- Added the capability to drive the rest of the MC33816's binary pins via script commands (e.g. the OA_2 pin when configured to be a digital I/O pin, etc.)
- Added several missing pins from the MC33816's 'I/O' pins window. Also, added important details such as (e.g., OA_2 is configured as 'Analog Output', Input, or Digital output pin, etc.)
- Fixed a spurious diagnostic message that can occur when a row in the wait table is enabled that has not yet been initialized.
- Fixed a bug with the joslr and joslf instructions when testing the 'start1, start2, ..., start6' pin states. The instruction had been returning 'true' on matches on start pins matching a 'high' pins, whereas the 'low'pins needed to also match. Consider joslr Start12Set start12; This should only return 'true' if both start1 and start2 pins are high. The problem is that it should return 'false' if (say) start3 is also high. However only 'pins that should be 1' were being tested and the 'pins that should be zero' were not being tested.
- Fixed bugs associated with viewing multiple eTPU Channel variables as Discrete's in the waveform window at once. There were problems associated with auto-ranging and with saving/restoring the project.

3.18 DevTool V1.01D / ETEC V2.30D / MtDt V4.84D (2014-Oct-02)

- Fixed the 'error color' (making it far lighter) such that the line of text were the build-error occurs can still be read. Previously, the 'error color' was so dark that it could not be read on certain monitors.
- The injector model multi-injector in a bank bug has been fixed. The diagnostic feedback comparator outputs are now being simulated correctly.
- The set_fault() script command now supports all planned faults including 'device open' and 'device short' for HS, LS, and INJECTORS. Injectors additionally support shorts (either injector side) to the BATTERY, to GROUND, and to the BOOST supply.
- The injector resistance and the resistance of the HS and LS driver's Rds-on is now modeled and can be specified via script commands.
- The ability to test the flags using the the 'wait', 'jocr', or 'jocf' instruction has been added and the flags register is now (correctly) global to all cores as it was previously (and incorrectly) modeled as being local to each core.
- In the watch window, fixed the display of banked variables. The watch window as calculating the wrong bank-address and displaying the wrong address's variables when displaying the bank-variables.
- Changed the verify_current() script command. The current sense current resistors are identified using the 'SENSE1' through 'SENSE4' syntax. Previously it was documented as such, however the actual script command used the 'VSENSE1' through VSENSE4'. ALSO, added the ability to verify the current through the high side and low side drivers (HS1-HS5, and LS1-LS7) using the 'verify current' script command. Note that previously only the ability to verify the sense current was supported.
- The help file can now be accessed for extended instructions (right-click and select 'Open help...').
- The instruction examples can now be copied to the Windows' clipboard (right-click and select 'Copy Example...').
- An auto-code helper capability has been added for writing MC33816 assembly. When starting to type in an MC33816 source code window, assembly instruction options appear in a floating list box, allowing auto-completion of if instructions and their parameters. More detailed help hints appear when the mouse is hovered over the auto-complete options.
- Improve source code syntax highlighting in cases where "/*" multi-line comments change to "//" line comments.
- Include files meant to be used in simulation script files (e.g.. StdDefMc33816.h, etec_sim_autodef.h) are now installed into an Include directory under the executable install path. When script files are processed, this directory is automatically searched.

3.19 DevTool V1.01C / ETEC V2.30C / MtDt V4.84C (2014-Sep-07)

- Changed the default font to 10pt Courier New. Users can now configure the font on a project-by-project basis in the IDE settings. Font changes affect all editor windows and the simulation register/status windows.
- Allow tabs within DevTool to be floated into their own window, and can later be re-docked into the main DevTool application. The floating/docking control is selected by right-clicking in the tab and selecting the float/dock option in the context menu. Window types that can be floated include the Waveform, Watch, Local Variable and Channel Frame window types, as well as all of the simulation register/status windows.
- Hovering the mouse over a variable in the code will show its current value (in-scope variables only for given execution point)
- The simulation detects a large number of suspicious situations (such as setting the VBoost DAC above 0xD0 ... not recommended per FSL literature.) These messages can be disabled within the Message Options dialog box. However, it was often difficult to figure out exactly which message to disable within the dialog box because there are so many possible messages and because the dialog box was poorly organized. The dialog organization has been improved and the messages themselves now generally indicate exactly where, within the messages dialog, to find the message to disable. The message ID is also now listed in the dialog in case the user wants to disable the message from the command line (when the IDE is launched ... useful for automated testing.)
- When compiling (or building in the MC33816) using the <alt F9> hotkey, changed files were not getting saved prior to the compile/build. This has been fixed.
- Fully-implemented the 'rstreg' instruction which was missing several of the TargetReg field values (e.g. 'all' was not supported.)
- Support for 'wait', 'jocr', 'jocf' testing of the the Vds and Vsrc shortcut diagnostic comparators (sc1v-sc3v, sc1s-sc3s, etc) as well as 'Operation Done' is now supported. (Note that previous versions supported 'Operation Done' using 'jarf' and 'jarr' ... just not 'wait', 'jocr', 'jocf')
- Vastly improved the DC/DC Converter (VBoost) window and added much needed missing information.
- Added detection and warning when multiple low side drivers in same bank are 'ON' at same time.
- Injector simulation model now functional for multi-bank simulations.
- Implemented most missing instructions (sto, jfbkf, etc.) See missing features list for remaining unsupported instructions.
- Script commands for simulating opens/device-shorts/shorts-to-battery, etc, have been implemented. Note that the 'set_fault();' script command is used which is a different script command than was originally documented and intended.

- The set_reset_source() script command has been added to allow users to configure the reset_source register. Any read of the register clears all its bits.
- The Circuit Windows did not draw correctly when the window is resized smaller such that the scroll bar appears. The draw function didn't work in this case and the window became illegible. This has been fixed.
- Vastly improved the Injector schematic such that much, much more important information is provided. Also, fixed numerous misinformation.
- Numerous validity checks now performed on the 'place_injector()' script including that VBoost High Side Driver must be connected to HS2 or HS4, that any shared current sense drivers also share the same current sense resister, that any shared high side driver must share both Boost and Battery high side drivers, etc.
- The DAC settling time register (0x1A9) and current filter registers (0x198, 0x199, and 0x19A) are now implemented. This prevents a wait instruction from returning 'true' when waiting for a DAC output state.
- Added a warning if the VBoost pin voltage exceeds the maximum allowed (72V.) Note that (similar to all messages) this warning can be disabled.
- Support simulation of the DAC Comparator filters (registers Current_filter12 Register (0x198), Current_filter34l Register (0x199), Current_filter4h4neg Register (0x19A), and Boost_filter Register (0x19D)) Dac4h4negFiltersAddr. Note that these filters de-glitch the comparator output by not allowing the comparator output to change unless the unfiltered output is stable for the number of clocks specified in the filter.
- Support simulation of the DAC settling time of 0.9 microseconds. The DAC output voltage is simulated as having linear slew from the previous DAC output voltage to the new DAC output voltage taking 0.9 micro-seconds.
- Support viewing of DAC1-DAC4L and DAC4Neg voltages in the waveforms window.
- Removed OA_1 and OA_2 pin voltage from being viewable in the waveforms window. Note that these voltages are not going to be simulated in the first production release which was confusing.
- When single stepping by hitting the F8 key at the 'wait' instruction, the simulator only advances by one clock, but instead should simulate until past the wait instruction.
- There is not warning (and should be) if the VBoost DAC is given a value outside its allowed range of 0x9 to 0xD0. See See MC33816 User Manual Rev 4, 2014-April page 31, very top of page.
- Fixed an MC33816 simulation bug wherein the create wait table entry relative (cwer) was incorrectly calculating the address when the offset was negative.
- Added a set of extended instructions to the ASH WARE MC33816 assembler that relieve the developer of figuring out whether they need to use far or relative addressing for jumps and when creating wait table entries. The new instruction names are JUMP, JUMP_ARITHMETIC, JUMP_CONTROL,

JUMP_STATUS, JUMP_START, JUMP_CONDITION, JUMP_FEEDBACK, JUMP_CORE_ID, CALL, CREATE_WAIT_ENTRY.

- When a multi-instruction cycle instruction is executed (such as the shri instruction, see instruction sequence below) then any subsequent ALU instruction (add, subtract, load immediate, swap, toc2, and toint) to that get executed while the mult-instruction cycle instrucion is still executing are supposed to be ignored. The bug is that (instead) the alu instructions get executed instead of ignored. For instance, the variable named 'Signature' in the sequence gets written with the value 0x9D97. whreas the 'ldirh' ALU instruction should be ignored, and is not.

```
ldirl 20h rst;
cp ir r2;
shri r2 4;
ldirh 9Dh _rst;  // Should be ignored, but was not
ldirl 97h _rst;  // Should be ignored, but was not
store ir Signature _ofs;
```

- [NOTE: This bug was still listed in the previous release even though it had actually been fixed.] Instruction ldcd does not write the output drivers, but it does perform the RAM operation.

3.20 DevTool V1.01B / ETEC V2.30B / MtDt V4.84B (2014-Jul-11)

These features have been added and bugs fixed in Version 1.01 Build B which was released on 2014 July 11.

- Enforced access protection of output drivers by certain instructions (e.b. 'bias') thereby causing the effect of the instruction to be blocked. These can be enabled by setting the appropriate bits in the 'Out_acc_ucX_chX' registers (0x184, 0x185, 0x186, and 0x187.)
- Asm816: Added various instruction extensions including an optimized constant load to the 'ir' register.
- Asm816: Added support for –I=<path> and –d=<macro> cmd line options.
- The Boost voltage comparator's filter is now modeled (0x19D). See MC33816 User Manual Rev 4, 2014-April section 6.2.3 pages 24.
- Fixed a bug in which, when switching project files, any non-standard panels in the previous project file were being retained by the new project.
- Fixed a bug in which when switching projects, windows used for viewing a file (e.g. source code window or script window) from the old project were not fully closing such that if the file on disk (from the OLD project) were to change, an annoying diagnostic message was firing.
- The build define (e.g.. "MPC5554_B") is now automatically injected into the script command environment as a macro.
- Fix a symbolic debug issue where when reading/writing a struct/union member value, if the member had a typedef'd type, the read/write could fail.

- In the target node settings of the project tree, settings for controlling the directory for object (intermediate) files and the directory for auto-generated code output from the state machine compiler have been added. The default is that object files get placed in a sub-directory of the project directory called "obj", and by default state machine auto-generated code files are output into the same directory as the final executable.

3.21 DevTool V1.01A / ETEC V2.30A / MtDt V4.84A (2014-Jun-16)

These features have been added and bugs fixed in Version 1.01 Build A which was released on 2014 June 16.

- Fixed a bug in which the file being built (usually a '.elf' file) cannot be renamed (when right on the file in the project window) if the .elf file does not exist to begin with. This makes sense (say) for a source code file (because there needs to be an existing source code file to rename). But this restriction makes no sense for an .elf file because it is generated by DevTool, AND it makes it squirrelly to rename the generated files.
- The Current Sense Block amplifier gain was ~20 mv/LSB. It is now now modeled correctly at 9.763 mv/LSB.
- Current sense block has a -0.25V amplifier output offset was not modeled and now is.
- Current Sense Amplifier bug in which the amplifier stops working altogether if the gain is changed (stgn instruction is executed) has been fixed.
- Diagnostics is mostly supported.
- SPI Loopback is partially functional. However, it acts just like SPI interface with the restrictions and core access limitations mostly not enforced. Also, the two-cycle availability on reads is not enforced.
- Watch window displays bank and global variable display.
- The ASH WARE assembler now produces the ciphered .bin file needed to load code into the device (using the SPI interface.) Note that for simulation an industry-standard .elf file is included.
- Fixed a bug in when testing 'ocur' or '_ocur' in a 'wait' or 'jocr' or 'jocf' instructions.
- Fixed a bug in which execution of the 'Change Opamp Gain' instruction ('stgn') would set the gain to 0.
- Fixed a bug in which the Amplifier settling time, which is supposed to be 2 micro-seconds max, is clock dependent and changes if the core's clock frequency changes or due pretty much anything such as a change in the system clock frequency or a change in the clock divide or if the core clock is 1/4 the system clock, etc.
- The MC33816 resets to the frequency of 16.778 MHz. The 'set_clk_period(41666666);' script command should be used to set the device to the more typical 24 MHz. In a multi-core simulation this script needed to be executed on every core (e.g. ch2_core0.set_clk_period(41666666);,

ch2_core1.set_clk_period(41666666);, etc). This has been changed such that a change to the system clock frequency on any core changes the system clock frequency for all cores.

- Fixed a 'duplicate DAC4L bug' in which the DAC4L value accessed across the SPI bus(e.g. using script commands) was not the same as the DAC4L value read/written in the UcReg's 'dac_osoc' when run in Ch1.Uc0, or 'dac_sssc' when run under Ch2.Uc0, etc.
- Fixed a bug in which the vector frequency is incorrect if the Ck_per register is set to anything except '1' (Core clock must be half of the system clock for the vector frequency to be correct.) This has been fixed.
- Waveform _CUR4L_FBK was swapped with _CURR4H_FBK.
- Fixed a bug in which the keyboard's 'UP' and 'DOWN' arrow keys were not changing the active waveform in the waveform window.
- Do not allow organizer nodes to be selected in waveform channel selection dialog.
- Fix syntax highlight issue with comments on same line as preprocessor directives.
- Apply syntax highlighting to new text that hasn't yet been highlighted, when the cursor moves. (text was being left un-highlighted longer than necessary)
- Make several improvements to waveform discrete nodes, including fixing an issue where channel frame variables could be unassociated with a channel making their data signal behave strangely.
- In the MC33816 IDE, added an MC33816 assembler manual to the installation available in .pdf format from the Windows menu or in .chm format from DevTool's help menu.
- Fixed a bug in which the circuit windows definition file was not getting loaded thereby preventing viewing of the circuit.
- Fixed a bug in which the single step (F8 or F7) would stall at the 'wait' instruction, progressing the simulation only one clock cycle for each step. When at a 'wait' a step now runs the simulation until the first instruction past the wait, even when it takes many clocks until the wait-ending event appears.Fix a bug that causes an exception when editing MC33816 state machine transition properties. If the transition condition and then the transition priority are modified, an exception occurs when exiting the dialog with an "Ok".

MC33816 Assembler 'ASM816.exe' Fixed Issues

- Added support for Cipher algorithm (non-ciphered .bin files would not work on the actual hardware)
- Added support for -msgStyle and -msgPath command line format (GNU, PSPAD, etc message format)
- Support for variables (sint16 and uint16) included indexed (banked) variables. Variable addresses are exported into the auto-defines header file for reference by the host code. Bank base index.
- Fully locates data including banked data. Macros support setting the bank address for defined banks.

- Added 'C-Like' preprocessor support.
- Removed 'missing preprocessor' work-arounds from state machine
- Fixed misc elf/dwarf executable image format issues.

3.22 DevTool V1.00B / ETEC V2.23B / MtDt V4.83B (2014-Mar-12)

- Selecting vector file aliases for waveform channels was failing, resulting in a blank waveform button for the signal. The workaround was to select the underlying node (e.g. "_tcrclk" rather than "Crank"). This has been fixed, however, the configuration of waveform channels will need to be re-done with this release.
- The editor response while syntax highlighting has been improved.
- Fixed a problem where the user was unable to disable message for HSR when HSR already set.
- Right-clicking on a local variable or channel frame window column header could cause a crash-type failure - this has been fixed.
- The legacy project file import capability of DevTool has been greatly improved. Simulator message options get imported, logic analyzer nodes get read into the waveform window, and watch variable are transferred into the DevTool watch window.
- Settings to reload/reparse script and vector files on every reset have been added, as well as File menu options to explicitly reload and parse each file type.
- Hid a un-implemented 'All Targets' menu item from the breakpoints menu.
- Added a 'Reset at <time>' message to the output window so that the window does not go (disturbingly) completely blank when the simulator is reset in simulation mode.
- In the 'Build' menu added 'reload script' and a 'reload vector' options.
- Added options for ALWAYS re-reloading the script and vector files on reset. Note that the default is to reload only when these files have been edited in DevTool. (to adjust, in the project window, right click the vector/script files and select 'settings') Rational: this option is needed if the user has changed an included '.h' file.

Crashes/Hangs

- Fixed a lockup problem that could occur when right-clicking the mouse in the waveform window.

Help Documentation

- Help documentation improved, though still shortcomings.
- Help buttons in many dialogs have now been connected to the appropriate help documentation.
- Removed 'Mtdt' information from the DevTool manuals that was not applicable to DevTool. (Note that there are still known issues in this area.)

Spurious Diagnostic Messages

- Fixed spurious diagnostic message that occurs when a text file that is open in the IDE changes on disk such that it gets reloaded.
- When the mouse is hovering over a panel divider and the user right-clicks, a spurious fatal diagnostic error occurs. This has been fixed.

Installer

- When installing DevTool, no longer change Mtdt project associations to DevTool. Instead, these associations stay with Mtdt.

3.23 DevTool V1.00A / ETEC V2.23A / MtDt V4.83A (2014-Feb-24)

The following features and bugs were added/fixed in this version.

Note that this release was given a '1.0' version designation because the DevTool eTPU Simulator is released to production. However, the MC33816 simulation is still pre-production.

- Added popop hint text added to MC33816 assembly code that describes the MC33816 opcode and in some cases the opcode fields.
- MC33816 assembler's -version command line option now produces the correct version.
- An errant 'The default ETEC (eTPU) compiler installation cannot be found ..." message is getting printed when running the MC33816 simulator. The MC33816 simulator does not use this eTPU compiler so no such message is appropriate. Fixed.
- Complete redesign of the 'Text Search' dialog including adding ability for search to span multiple files. Many issues and shortcomings fixed and addressed including redo/undo tracking issues fixed.
- Waveform window was not allowing the cursor and other times to be drag-and-dropped into the wave view area to run to the specified time. Fixed.
- Fixed a problem in which the 'Goto State' was not working in the state machine window.
- When attempting to begin a simulation, but the build fails, then the output window is popped to the front so that the offending error can be readily viewed. This is good. However, if a script or vector file fails to load then the Output window could remain hidden, making it difficult to tell what the specific problem is with the script or vector file. This has been fixed.
- Fixed a bug in which when a file is changed outside of DevTool such that DevTool reloads the file, if the file was in mixed-asm-source view mode at the time of the reload, an errant diagnostics message was firing.
- Fixed a bug in which the code coverage boxes that show whether code has been traversed or not would sometimes not fully span the height of the code window.
- Fixed the '-bd=' command line option. (overrides the build script ... typically used to change the simulation model)

3.24 DevTool Version 0.82 Build F (2014 Feb 3)

The following features and bugs were added/fixed in this version.

- The following all appear to happen when the Current sense values are very large Waveforms "Auto range buffer" calculates an incorrect value which is less then "Auto range view"
- State Machine Compiler now is a stand-alone command line tool thereby allowing incorporation into an automated regression test. (Does not affect on GUI.)
- Added ability to specify the Channel Number (or in some cases Function Number) in windows such as the Channel Hardware window so they can be tied to the active channel (which changes for each thread) or a specific channel which does not change.
- Fix a crash that can occur if a needed .net service pack is not installed. Now detecting needed .net Service Pack, alert user, and gracefully shut down.
- Redirect 'Un-recoverable Error Reporting' to ASH WARE and no longer Microsoft.
- Added detection in the installer for .net.
- Added an Hour-Glass cursor and a moving status image that appear when building (compiling, assembling, linking, etc) so DevTool does not appear to be hung during lengthy builds.
- Fixed a crash that can occur in certain cases in the Waveforms window when the 'Auto range buffer' is used.
- In the Waveform window's waveform dialog, the auto-range calculation was not always correct because the very latest value was not always included in the calculation; this has been corrected.
- The ASH WARE MC33816 Assembler is now the default.
- Using the ASH WARE MC33816 assembler, multiple assembly (dfi) files can be built into a single executable image.
- Multiple MC33816 cores can now load the same executable code image. Note that only a single core can have 'build' enabled for each executable image. In words, only one executable core can have 'auto-build' enabled. Disable cores from building by right clicking on the project's 'target' node and selection 'Disable Build.'

3.25 DevTool Version 0.82 Build E (2013 Jan 13)

The following features and bugs were added/fixed in this version.

- In the Watch window, moved the '[register xyz]' information to the 'Type' column and not the 'Value column where it was a bit confusing.
- In the Watch window, moved the '[register xyz]' information to the 'Type' column and not the 'Value column where it was a bit confusing.

- Change the 'Edit Mode' of the 'value' field to occur on a single mouse click (a double-click had been required)
- Fixed issue in which swithing between mixed-source assembly and source-only view of a file could cause the file to be considered 'dirty' thereby triggering an (unneeded) rebuild when doing a 'make.'
- Added a 'NOP' to any new state created by the MC33816 Graphical State Machine tool. Previously there was no NOP and the issue was that the assembler generates an error on a line with a label but no opcode.
- Fixed an issue in which the 'Dongle.exe' and 'License.exe' utilities were not being installed on a License Server installation.
- Click-to-line on the Output Window was not working for errors reported by the ASH WARE State Machine Compiler when acting on the DFI files in the MC33816.
- Right clicking in space in the state machine window, pop now supports a 'Add State' capability.
- Deleting a state bug in MC33816 was leaving orphan transitiions.
- The delete keyboard key can now be used to delete states and transitions.
- Enabled the TrackWhell in the Waveforms window (changes selected channel.)
- When selecting the (unsupported) 'Threads' window it would crash the app.
- On the Edit menu, the 'block indent' and 'block un-indent' had the wrong hotkeys listed AND did not work.
- In the 'Options' menu the 'Waveform->Activate Right Cursor' submenu was not working.

3.26 DevTool Version 0.82 Build C (2013 Jan 7)

The following features and bugs were added/fixed in this version.

- Added recognition of .h, .map, .lst, and .report files as 'text files' such that when they are double clicked in the project's 'Reference' folder they open up in the editor. Unrecognized file types are no longer ignored when double clicked. Instead a dialog opens asking the user if they should be opened in the editor.
- Header files (.h) now getting syntax highlighted.
- When loading a project file the eTPU's default type (etpu1/etpu2) and code size are detected and these become the default type and code size passed to the ETEC Compiler.
- Fixed a bug in which the 'Click-To-Line' capability of the Output Window (and other cases) would scroll to the wrong line when the file was being viewed in 'Mixed-Assembly' view.
- Fixed bugs in which a view of a source code file in mixed-assembly mode was not getting saved/restored correctly in the project file and sometimes caused a spurious diagnostic warning. Also, improved text-selection persistence when toggling between mixed-assembly and source-only view.

- When first non-whitespace character typed in the editor is a '#' (preprocessor) character then auto-align to the left hand side of the editor.
- When selecting a line by holding the END and SHIFT buttons down it seems like the trailing newline characters get included in the selection when they should not.
- Fixed bug in which if the first character typed into an editable file window is a backspace or delete (and perhaps a couple others) then the keystroke is lost, though the run-mode does switch to 'edit' such that the second keystroke is not lost.
- Fixed the toolbar's 'Goto Cursor' button.
- The popup menu that appears when editing a file or state machine now contains the 'Save' option.
- Improved the mouse behavior (selecting text, etc.) on the far-left side of the editable window where the Coverage boxes are seen when simulating.
- If file viewed in DevTool has a completely different path (down to root of drive) display the absolute path, not a (HUGE) relative path.
- Added support and documentation for disabling warning(s) from the command line using the '-ws=„...'
- Fixed issue with MC33816 'Direct Injection' and 'DC-DC Converter' demo not loading into non-licensed computer.
- Changed nomenclature 'Stop' to be 'Pause' instead to avoid confusion. Modified several buttons to match. (Thank you Tim!)
- Right-click on the build batch file should include make/build option(s)
- Fixed issue in which extremely large listings generated by external 'Mk.bat' style builds (I.E. when code is built by an external batch file) could cause DevTool to hang.
- Support 'Click-To-Line' support for listings where a path is 'relative' to the 'CWD'. For example 'Error ..\.. \WebApps\SomeFile.c line 37'
- Fixed issue with specifying the .elf file following a creating of a new project. The .elf file cannot be specified
- New project problem. BC build - can't build with external batch file because executable is not set yet, and can't set executable because it hasn't been built yet!
- Fixed certain cases in which a dialog box would appear as a separate program within Windows.
- Added a verify_version_ex()' script command for DevTool to check DevTool and ETEC versions. Note that the old verify_version() script command still works, but generates a warning, AND it checks the equivalent Mtdt Version rather than the new DevTool version which restart at version 1.0.
- On auto-run mode when a command line parameters is used as part of a regression test, ..., if the test is interrupted and a different project is opened, the command line parameters used initially to launch the

regression test are retained and stay in effect, whereas they should have been erased when a different project file was loaded.

- In an eTPU Target's settings, click on the 'Tools Directory' dropdown. Then click (Some Hotkey ???) This causes some sort of race condition and locks up DevTool. Bizarre. Note that this bug was not intentionally fixed (is it really gone?) However, some other changes were made to fix another bug that apparently fixed this bug also. In any case, can no longer reproduce.
- Fixed a bug in which, other than using drag & drop, it was not possible to add files to the reference folder.
- Fixed a crash that can occur when closing a panel. Note that DevTool consists of panels, within panels, within panels. Panels with child panels cannot be closed but this was not being enforced in all cases which could lead to a crash.
- Fixed a bug in which if partway through a simulation a change is made to the project file that should cause a code rebuild, the code doesn't (always) rebuild, thereby allowing the simulation to continue using a stale code build.
- Added an MRU approach to closing windows such that if a window is closed, the last-viewed window becomes active.
- After a re-build, script file needs to load on reset!! (because defines file may have changed)
- Files in the project (vector, script, 'c', .dfi, etc) can be renamed by right-clicking on them.
- Fixed a general weakness in that it was difficult to create and use header files or other generic files with non-supported file extension. This is now supported by both the main menu and project's popup menu.
- Fixed a bug in which files that are opened for viewing (e.g. '.LST' files) that are supposed to automatically be reloaded on a link were not (in certain situations) being automatically reloaded.
- Added ability to generate CodeCoverage, CodeLoad, SymbolTable and ExecutionTrace files from the Files, Write menu.
- Fixed an issue with creating new script files where the file would get created but would not get added to the project. Also there were spurious messages when creating new (template or blank) vector and primary script files that have been fixed.

The purpose of this section it to help invite customer feedback in order to help ASH WARE prioritize development of these features. The following features are planned for future releases.

- Memory Tool not implemented. Memory tool supports things like dumping memory to a dis-assembly or image file. [Workaround: use same feature via a script command, 'dump_file();']
- The 'Help->Register Computer' menu item is missing. [Workaround: Open an explorer window and copy the license file, 'AshWareComputerKey.ack' from the installation directory into your email program (as an attachment) and email to ASH WARE.]
- The 'Threads' Window is not yet supported. The new script command write_wctl() does provide access to the data though.
- Complex breakpoints (available in Mtdt) have not been enabled in DevTool yet. [A breakpoint window with many enhanced capabilities now exists]
- The Timers are not supported (timers embedded into code addresses)
- Ability to continue to edit during long builds (currently, an Hour-Glass appears and DevTool is disabled during long builds)
- Support the ability to edit code without ending simulation mode. Does not include ability to compile and run patches on the fly - to get the changes the simulation must be re-started after a re-build.
- A disable-able warning that gets issued when channels have overlapping channel frames
- Verification errors appear in the output window in a click-to-line format.
- Spell check within comments and strings
- Support a separate edit mode and simulation mode IDE configuration. Currently only one mode is supported and no window layout changes when switching between the modes.
- It would be nice to store the Code's 'Make' information in it's own file rather than using the environment file since in many cases the environment file is not read/saved such as when using the '-NoEnvFile' command line option on a regression test.
- Un-install if failing to remove the desktop icon and is failing to remove the the environment variable used to find the software (e.g. 'ETEC_BIN').