Sophisticated Debugging Features for Motorola's HCS12 Family are available on Nohau's Full-Featured Emulator

By: Doron Fael – Nohau

Nohau's second generation HCS12 full-featured emulator includes sophisticated debugging features to assist HCS12 users in their development, and allow getting products to market faster. Many of these debugging features are not available or are very limited when using the simpler BDM debuggers. These features are the topic of this article.

5V & 3.3V Operation

Motorola is in the process of releasing new HCS12 parts capable of running at either 5V or 3.3V external supply. This new capability is found in the new MC9S12B family, MC9S12C family, MC9S12E family, and MC9S12K family that already exist, and other HCS12 families that are currently in design stages.

The full-featured HCS12 emulator supports both 5V and 3.3V operation of the above-mentioned new parts, to the maximum allowed bus speed of 25MHz. In practice, the emulator performs to even higher bus speeds of 32MHz at 5V, and 30MHz at 3.3V. The large speed safety-margin means that at the maximum Motorola specified bus speed of 25MHz, the emulator has solid operation, and users can be sure there will be no speed-related misbehaviors caused by the emulator. The emulator also automatically adjusts its timing, as suitable for high speed, medium speed or low speed operation, which gives it another dimension of stability at all speeds.

Full-Featured Emulators operate in Expanded-Mode, so the external HCS12 bus carries address and data information to be recorded in a hardware trace and the shadow memory. Accesses to the internal HCS12 FLASH, EEPROM, RAM and SFRs (Special Function Registers) are allowed and always available to be recorded in the trace and the shadow memory on the emulator. In order to allow debugging of Single-Chip applications, a Port-Replacement-Unit (PRU) is used to regenerate the lost ports A, B, E and K, so from the target's and the user's point of view the emulator can operate at both Single-Chip mode and Expanded modes. The Port-Replacement-Unit (PRU) has CMOS voltage levels like the HCS12 MCU, and is designed and tested to work at both 5V and 3.3V and to the maximum specified bus-speeds mentioned above.

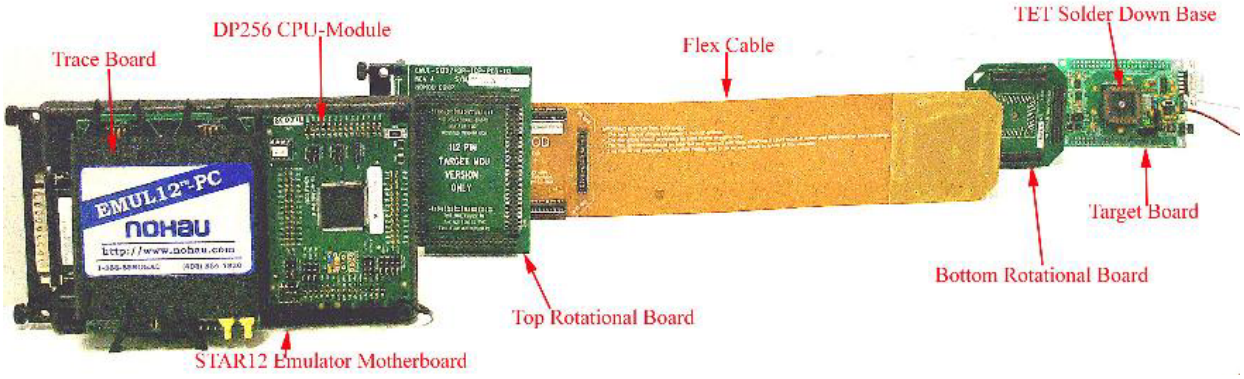

Figure 1 – An HCS12 DP256 Emulator & Target System with a Flex-Cable Adapter

Debugging & Testing all HCS12 Operating Conditions

The HCS12 full-featured emulator enables users to test and debug all of the HCS12 operating conditions, including some that are not possible, or are very limited when using BDM emulators. These include: Going through and out of Reset and COP Watchdog Reset, STOP and WAIT power-down modes, clock loss limp-home mode, and full internal PLL support for frequent and intensive speed changes. All these special events can also be recorded in the trace, including special trace frames to report active Reset, STOP or WAIT power down.

Let's look at COP-Watchdog-Reset for example: The internal HCS12 COP-Watchdog is used to detect software runaways, and to Reset the HCS12 in case a software runaway is detected. In order to avoid timeout, the COP-Watchdog needs to be serviced periodically by the user code. When the code seriously misbehaves, it is very likely the COP-Watchdog will not be serviced and will therefore timeout and Reset the HCS12 in order to get it out of the bad place it ran to. As mentioned above, the full emulator allows this sequence of Reset and un-Reset, and records it in the trace (See Figure 2).

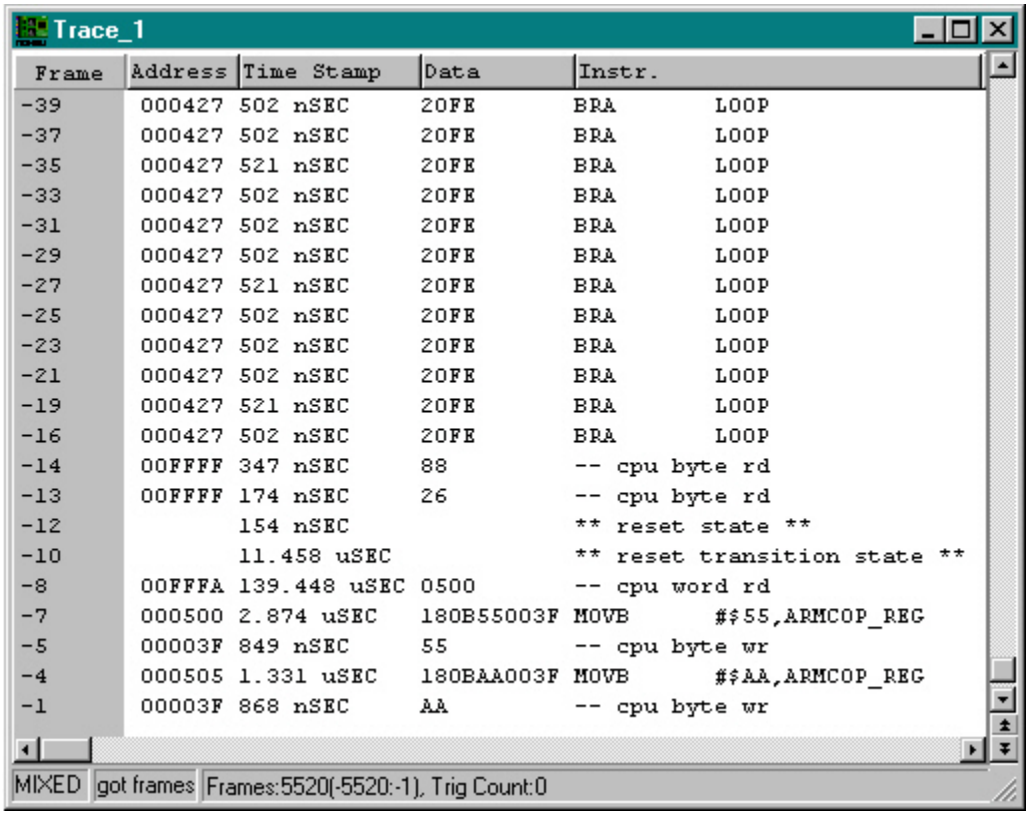

Figure 2 – Trace recording of a COP Watchdog Reset and un-Reset sequence

Frames –39 to –16 show that the program is stuck in an endless loop. The COP Watchdog is obviously not being serviced which leads to a Reset. Frames –14 and –13 are transition into the Reset and thought to be CPU reads. Frame -12 shows the Reset-state taking place. Frame –10 shows the transition out of Reset. Frame –8 is the COP-Watchdog Reset vector fetch from address FFFA. The fetched COP Watchdog Reset vector is 0500, and it specifies the address (0500) where execution should be resumed. Frames –7 to –1 show the resume of code execution at the specified address of 0500. Then a breakpoint that was previously placed on address 050A is met and user code execution is suspended to examine the state the HCS12 has reached

Going through and waking up from the HCS12 STOP and WAIT Power-Down modes can also be executed by the Nohau emulator and be recorded in the trace (See figure 3). In order to allow this, the emulator has to overcome the following HCS12 CPU obstacles: 1) The HCS12 BDM interface is not available for communication during the power-down modes. Therefore the emulator needs to suspend BDM communication when it detects power-down modes, and resume BDM communication after it detects the end of the power down mode. 2) There are 4 sub cases of the WAIT power-down and 2 sub-cases of the STOP power-down in regard to the clocks behavior. In some of these sub-cases the bus speed before and after the power-down mode may be different, and the emulator needs to adjust the BDM communication rate accordingly. The full emulator automatically handles with these obstacles, and allows going through and waking up from all the possible Power-Down modes (See figure 3).

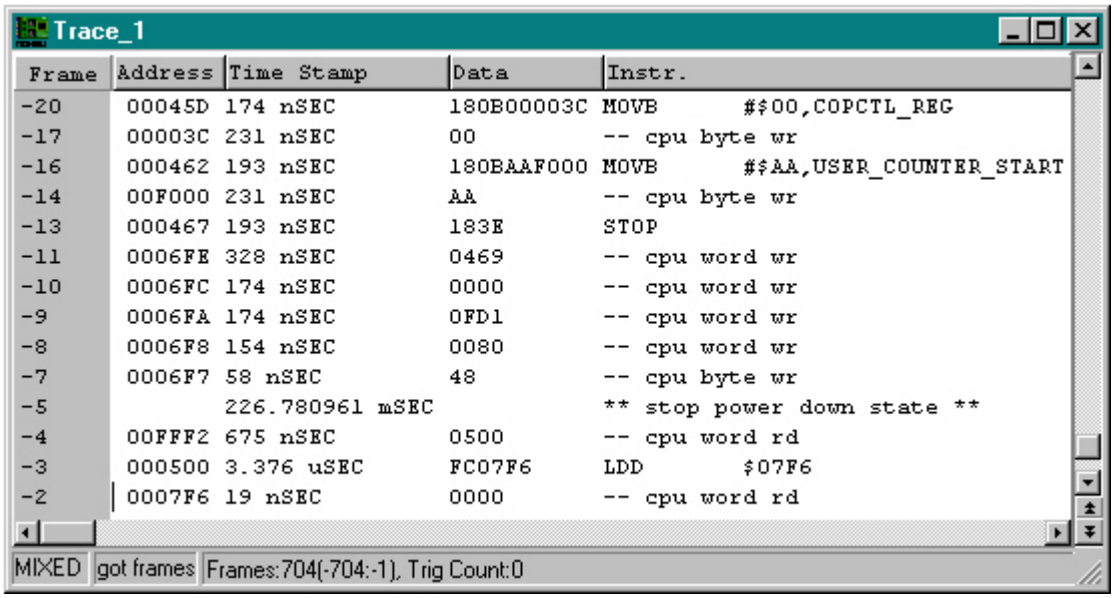

Figure 3 – Trace recording of a STOP Power-Down and an Interrupt Wakeup sequence

Frames –20 to –14 are general instructions being executed. Frame –13 is the execution of the STOP instruction, which will eventually take the HCS12 into the STOP Power-Down mode. Frames –11 to –7 are stores of all the HCS12 registers on the Stack as a result of the STOP instruction. Frame –5 shows the STOP Power-Down state taking place. Note that the trace also shows the power-down has lasted a long period of 226mSEC (in compare to nano-seconds that the other instructions take to execute). An external interrupt is used to wake-up from the STOP Power-Down, and Frame –4 is the External Interrupt Vector Fetch from address FFF2. The specified interrupt vector is 0500 which specifies the address where execution should be resumed. Frames –3 to –2 show the resume of code execution at the specified address of 0500. Then a breakpoint that was previously placed on address 0503 is met and user code execution is suspended to examine the state the HCS12 has reached

Emulation RAM, an Unlimited Number of Breakpoints and Shadow RAM

The full-emulator is equipped with 1MByte of paged emulation RAM + 64K of non-paged emulation RAM, that can be used to replace the internal FLASH and EEPROM, to allow an easy debug process, and to develop applications for future larger and smaller derivatives before the actual silicon becomes available. Executing from the internal FLASH and using the internal EEPROM is also allowed and may be useful for late debug stages. An unlimited number of hardware and software breakpoints are available on the full emulator, as opposed to only 2 or 3 hardware breakpoints in the case of a BDM emulator. A fully non-intrusive 64K Shadow RAM also exists to allow viewing memory and SFR variables and structures writes during run-time, and is available to be displayed to the user under all operating conditions.

The GUI (Graphical User Interface) Software

Seehau is the Macro based GUI software used to control the emulator. It includes on-line editing of C source files and re-compiling. It allows to start/stop, single-step, set trace triggers, examine/modify memory and C variables, and perform all other emulator functions. Variables, structures and arrays can be easily viewed and modified using the Inspect/Watch window, or by holding the mouse on the variable name. Updates to Global and Static variables, structures, and arrays can be viewed in the Inspect/Watch window in run-time non-intrusively. New features and bug fixes when implemented become available with software upgrades. The FPGAs on the emulator are also configured by the PC and may be upgraded by new software releases, available for download on the web. Data can be displayed graphically in real-time. Multiple windows can be opened, each individually configurable. Seehau is configurable to the user preferences with the Macro (scripting) language. Buttons can be created and attached to macros, commands or menus. These commands operate down to the layer that directly communicates with the emulator or can be made into higher-level commands with subroutines. These macros can use “IF...Then” statements and Boolean logic with the integrated Visual Basic for Applications (VBA) compatible interpreter. A built-in Macro editor and recorder provide an easy method to create and maintain custom macros.

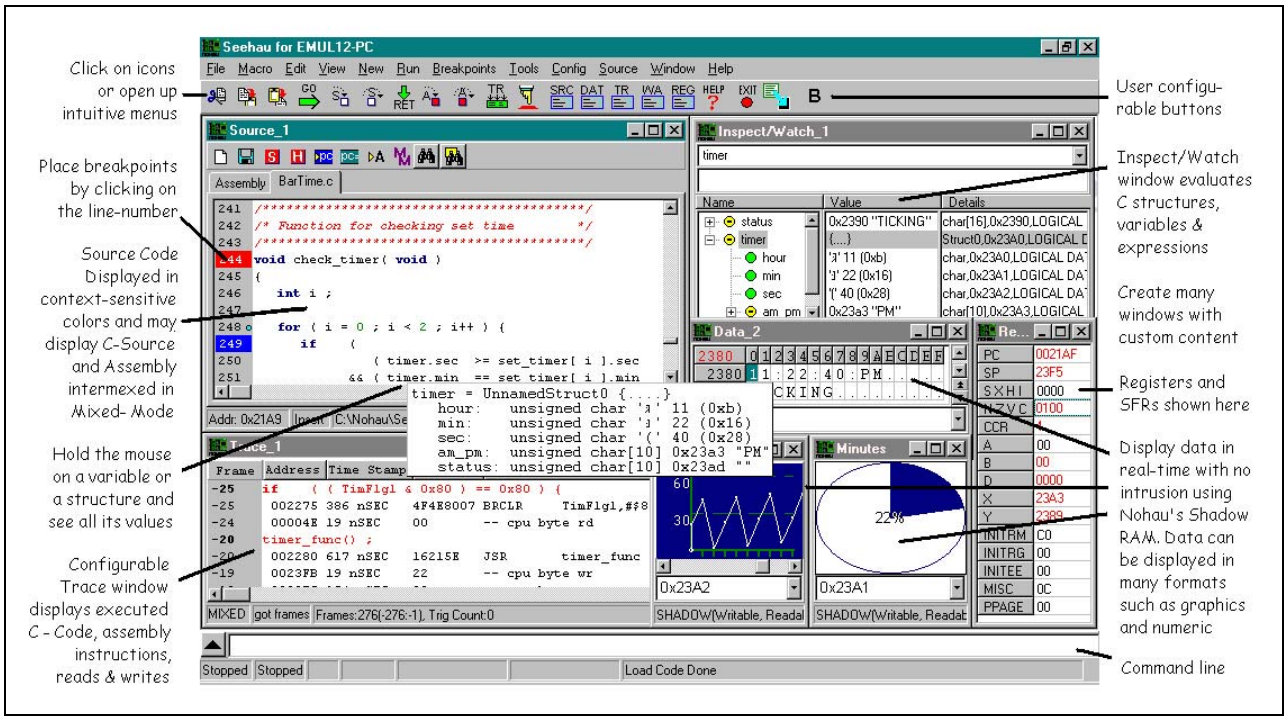

Figure 4 – The Seehau User-Interface for the Emulator

CPU-Modules and supported HCS12 derivatives

Most of the emulator components are general to all HCS12 parts. This includes the emulator motherboard, the trace, the communication cable to the PC, the flex-cable, and the Seehau GUI software. In order to change from one derivative to another a new CPU-Module may be required. New target adapter rotational boards may or may not be also required depending on the derivatives. The following CPU-Modules exist on the update date of this article (November 2003), to support all existing Motorola HCS12 derivatives (Even more CPU-Modules will be released when Motorola will release new HCS12 derivatives. Most of the HCS12 MCUs listed below have full emulator support available only from Nohau. Cost of CPU-Module - $995 - except for H256):

CPU-MC9S12B128 – Supports the B256, B128 and B64.
CPU-MC9S12C128 – Supports the C128, C96 and C64.
CPU-MC9S12C32 – Supports the C32.
CPU-MC9S12DP512 – Supports the DP512, DT512, DJ512 and A512.
CPU-MC9S12DP256 – Supports the DP256, DT256, DJ256, DG256, and A256.
CPU-MC9S12DT256 – Supports the DT256, DJ256 and DG256 (latest mask sets).
CPU-MC9S12DT128 – Supports the DT128, DJ128, DG128, DB128 and A128.
CPU-MC9S12DJ64 – Supports the DJ64, D64 and A64.
CPU-MC9S12D32 – Supports the D32 and A32.
CPU-MC9S12E128 – Supports the E256, E128, E64 and E32.
CPU-MC9S12H256 – Supports the H256 and H128, including LCD function of ports A, B, E and K.
CPU-MC9S12KG128 – Supports the KT256, KG256, KG128, K64 and K32.
CPU-MC9S12T64 – Supports the T64.

Setting the External Clock Frequency

Changing the clock fed to the HCS12 is easy and is made by typing in the desired external clock frequency to the configuration window of the Seehau user interface. The emulator is equipped with a PLL device, which is automatically programmed by Seehau to generate any requested frequency, to worst case inaccuracy of less than 1%. This emulator PLL generated frequency is used by default to drive the EXTAL pin of the HCS12 MCU, and should not be confused with the internal PLL of the HCS12.

Target Adapters

Special adapters exist to allow connecting to HCS12 target boards for any of the available derivative-specific footprints. For every footprint, 4 different adapter configurations exist. These configurations are combinations of with / without a flex-cable, and with the NQ/HQ/YQ or the TQ solder down base from Tokyo Eletech. Using direct connection between the header pins on the bottom of the emulator, and appropriate female headers that can be designed on the target, is also possible.

The flex-cable adapter is the most popular as it allows maximum flexibility, which is usually required. The flex-cable, which is used with this adapter, is designed with shielding and controlled impedance, and is tested to perform well at speeds in excess of 100MHz. A top rotational board is used to connect between the emulator system and the flex-cable. A bottom rotational board is used to connect between the flex-cable and the solder down base (See figure 1). The flex-cable can escape from the target at any of four directions: 0, 90, 180 and 270 degrees, using rotation of its connections with the two rotational boards. The Tokyo Eletech NQ/HQ/YQ solder-down adapter is also used by this adapter kit. The NQ part solders down to the target HCS12 QFP footprint. The YQ part is the emulator adapter cover, and allows to connect the NQ base to the bottom rotational board and the emulator system. The HQ part is a CPU cover, and allows the option to disconnect the target from the emulator and install a socketed HCS12 MCU on the target, to test target stand-alone operation (see figure 5). A cheaper Tokyo Eletech TQ adapter is made of one piece and also exists.

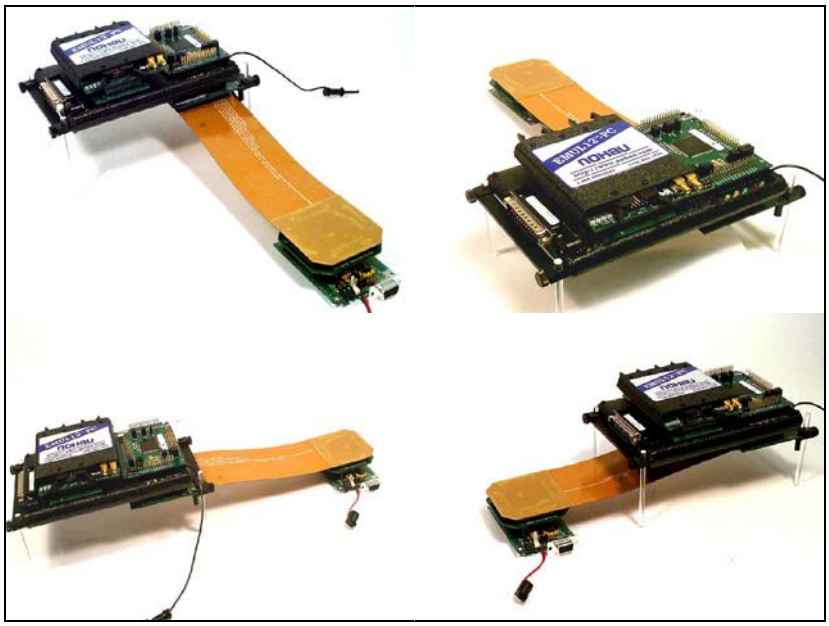

Figure 5 – The Flex-Cable can escape from the Target in 4 different directions: 0°, 90°, 180°, or 270°

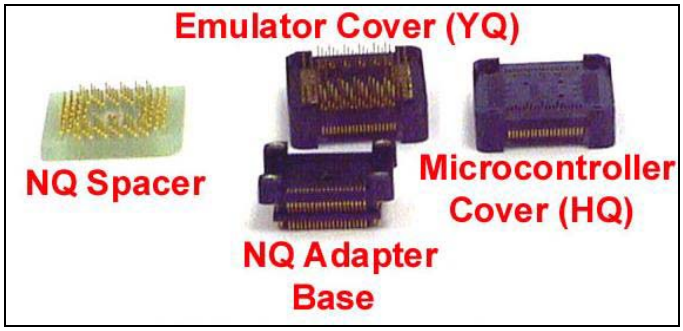

Figure 6 – The Tokyo-Eletech NQ/HQ/YQ Solder-Down Adapter

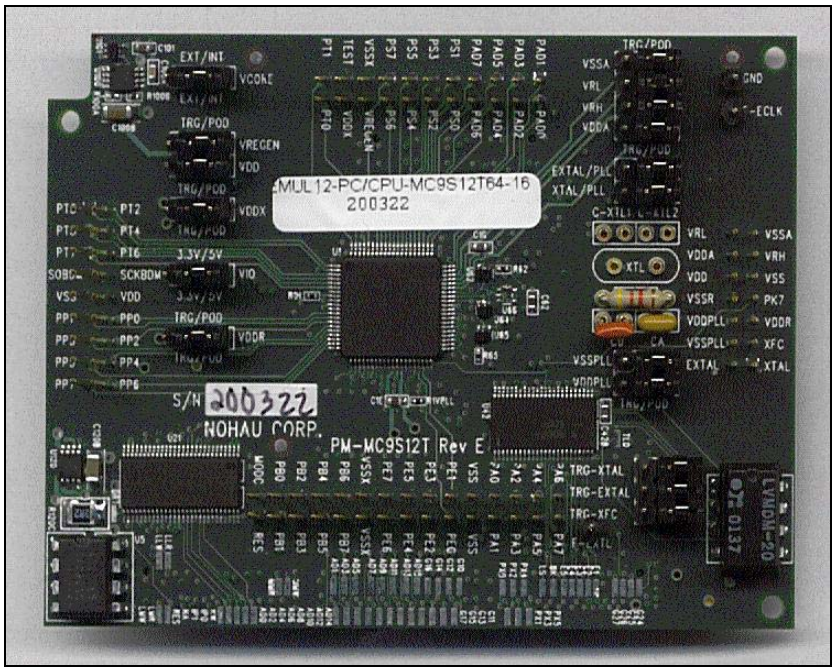

Figure 7 – Typical HCS12 CPU-Module (MC9S1T64 shown here)

The CPU-module has all the derivative specific signals going from the emulator to the target, available for convenient probing using logic analyzer, scope, or the trace, on four 0.1" headers. Jumpers exist on all the HCS12 voltages to allow supplying these either from the emulator power supply or the target power supply. The VIO jumper allows for either 5V or 3.3V supply to the HCS12. Both internal and external 2.5V generation for VDD and VDDPLL is possible. An external 2.5V voltage regulator exists on the CPU-Module to supply these voltages when configured for external supply. Special pin sockets for the crystal and for the PLL loop-filter components exist on the CPU-Module next to the HCS12 MCU, in order to minimize the susceptibility of these analog signals, and to prevent the need for them to travel all the way to the target.

The Power of the Trace

The trace is optional, and adds the ability to record up to 131,000 frames of instruction execution, data cycles, and other CPU activity. The trace is equipped with triggers and a filter. The filter can focus the trace to record only specific information, which is of interest for the current debugging stage. For example it can be programmed to record only the execution of a certain function, record only writes of data different than x to addresses y and z, etc. The triggers allow recognizing complex events. For example: Trigger after complex condition A is recognized, then complex condition B is recognized and then complex condition C is recognized 10000 times. When the trace triggers, it can be programmed to only stop itself and display its recorded content, or alternatively to also stop the HCS12 execution. The trace has the ability to start and stop many times, and be re-programmed without ever stopping or slowing down the HCS12 execution. This ability is very important to allow debugging of applications that cannot be stopped (for example motor control), and therefore must be debugged using only the trace-recorded information. Triggering from external devices (logic analyzer, scope, etc.) is also possible.

Conclusion

The Nohau HCS12 full-featured emulator, brings sophisticated yet easy to use debugging features to the aid of HCS12 developers, to help in fully testing and finding elusive bugs, and to shorten the time to market.

For more information on the web see: http://www.icetech.com/emul12pc.html
Or email: support@icetech.com