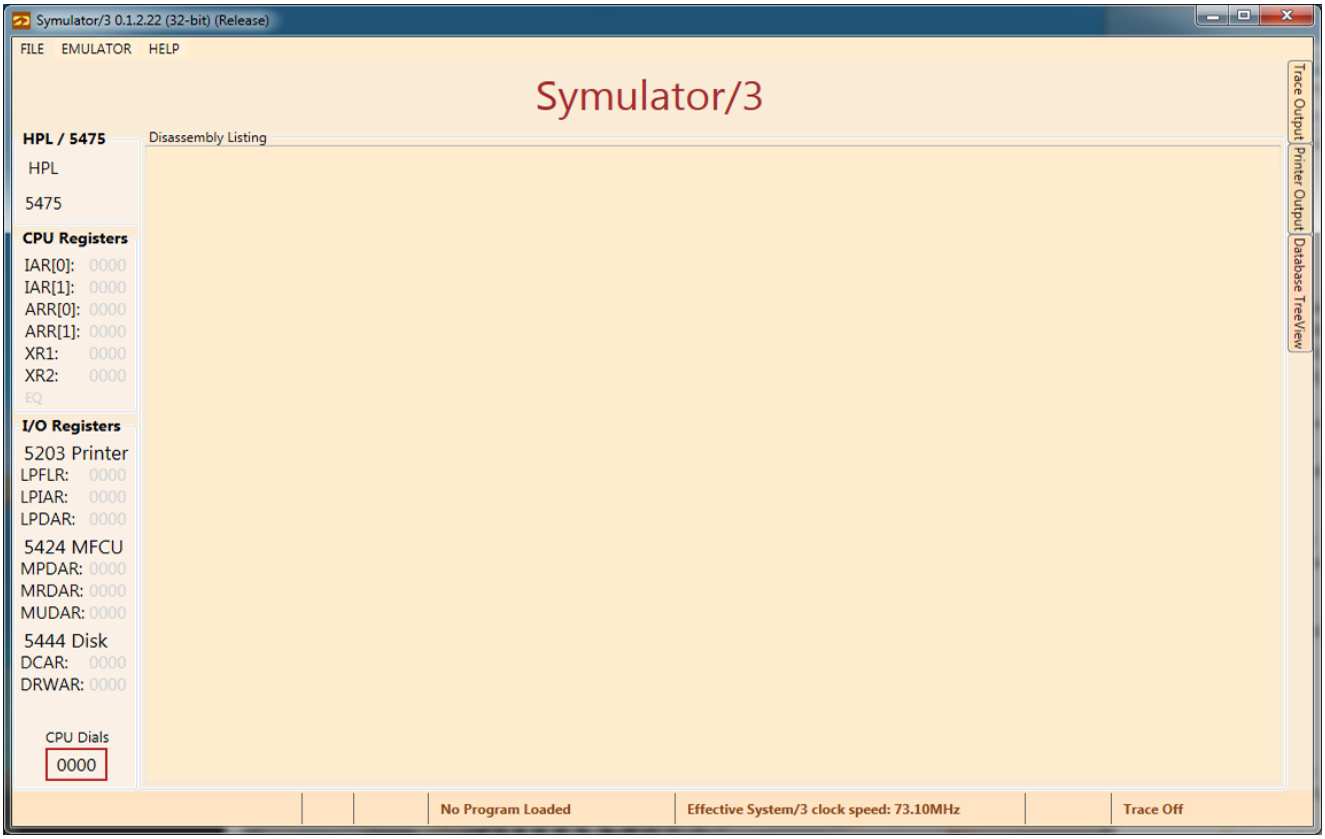
Symulator/3 Visual Studio GUI

Symulator/3 now has a UI wrapper based on code from “WPF 4.5 Unleashed” that mimics Microsoft Visual Studio. It is not yet complete but has sufficient functionality for publication.

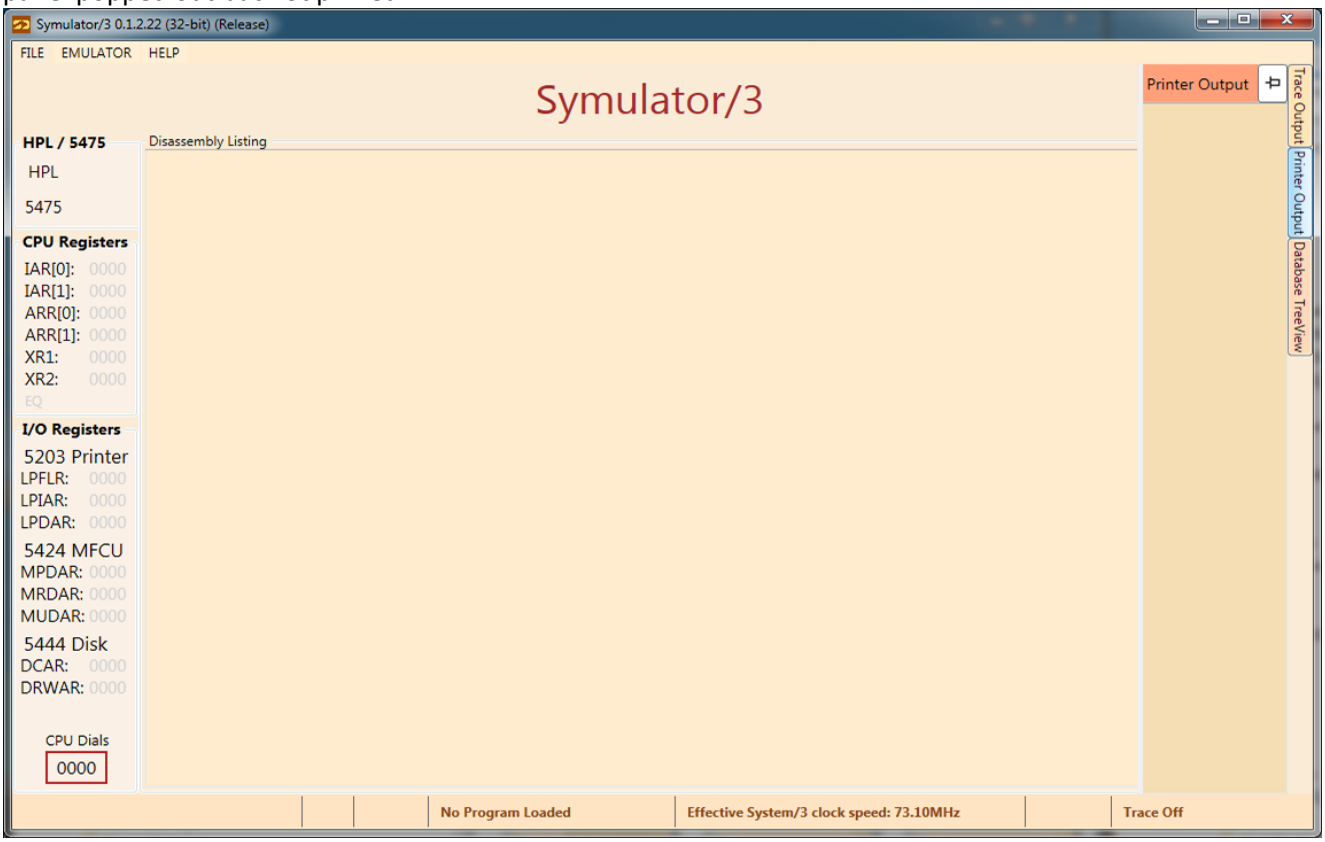


The first screenshot shows the UI in its initial state after having just been started. The left panel is fixed and shows the values in each of the CPU and I/O registers. The values show as grayed out and stay gray until a program has been launched. At the top are the displays for the 7-segment halt and 5475 keyboard column indicators. At the bottom is a control for setting the CPU console data dials which can either be entered directly from the keyboard or using the arrow keys. The up and down arrows rotate through all the available hexadecimal values for the digit just to the right of the caret. When the caret is at the far right end of the control, the up and down arrows rotate the values of the entire 16-bit value, wrapping from FFFF up to 0000 or 0000 down to FFFF.

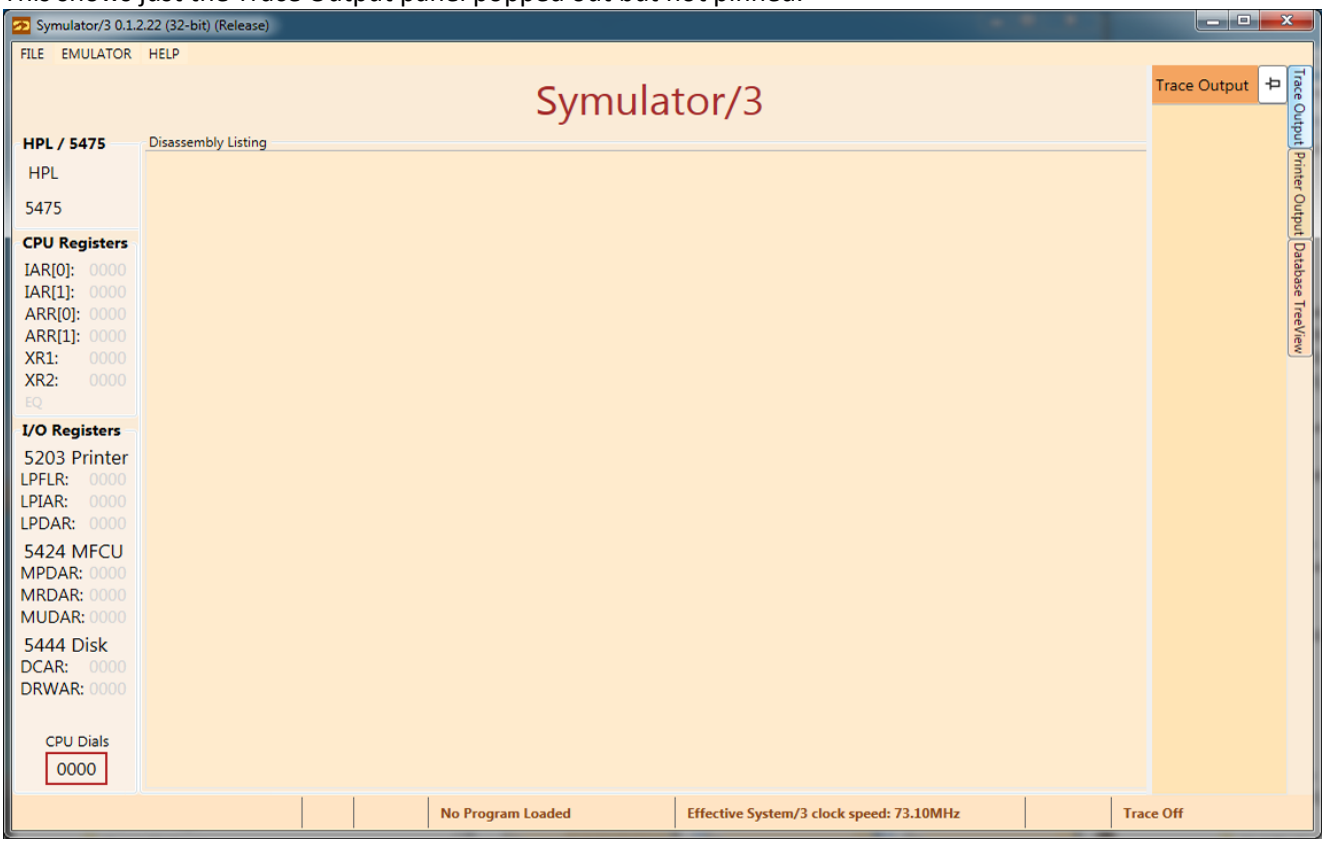
At the bottom is the status bar which contains panes to show:

- Program Name — The name of the program selected in the Database Treeview panel
- Program Card Count — The number of lines (cards) in the selected program
- Program Size — Size in bytes of the selected program
- Program State — Loaded, running, paused, stopped, …
- Processor Speed — Simulated IBM System/3 clock speed (physical CPU was 657.895kHz)
- Step Count — Count of CPU instructions executed since starting
- Trace State — Whether trace output appears in the “Trace Output” panel

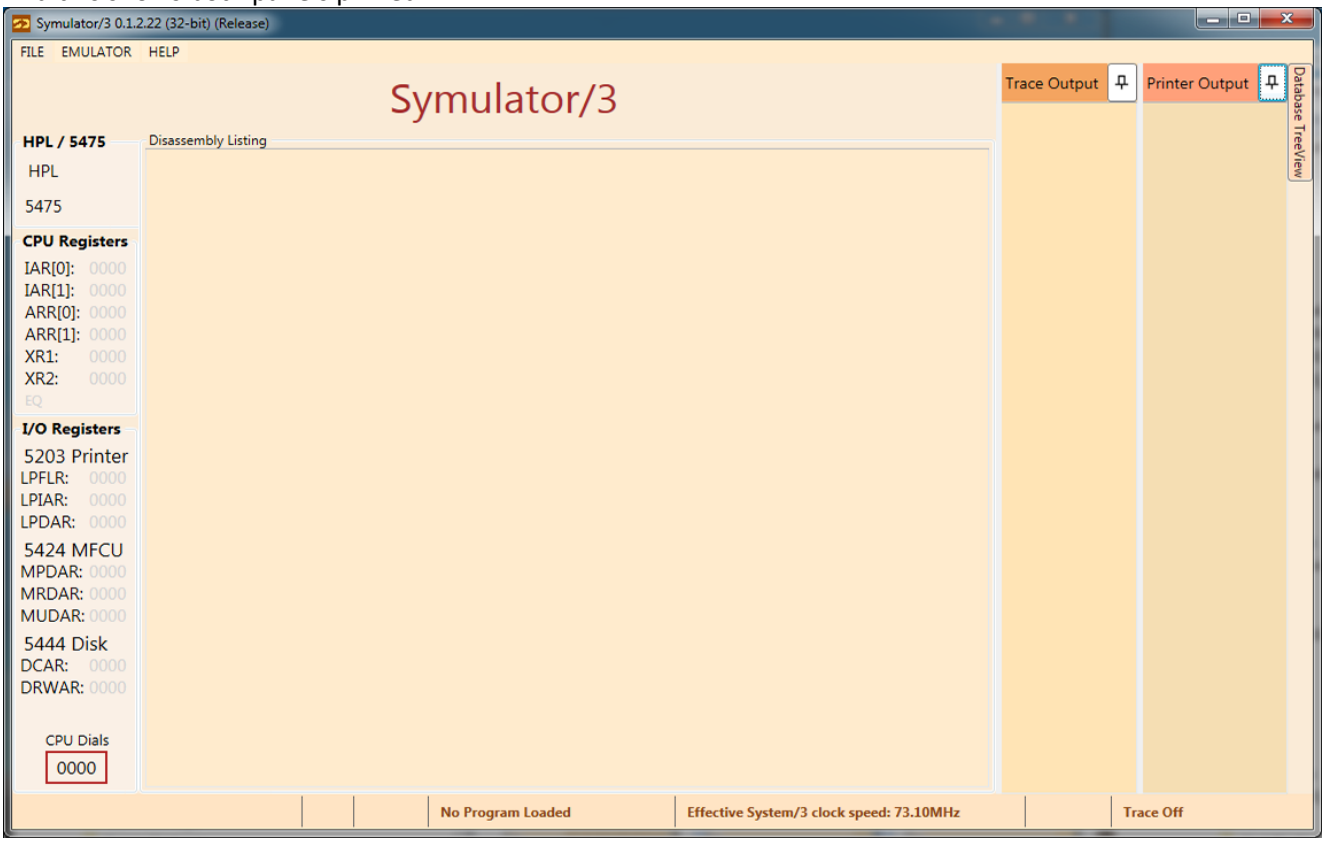
The UI has panels that pop out from the right side and a fixed status panel on the left side, and a status bar at the bottom. The pop-out panels can be pinned and unpinned, as in Visual Studio. Like Visual Studio, the pop-out panels can be shown temporarily by hovering the mouse over the title tabs. Once it shows, the push-pin button can be clicked on the keep the panel open, and clicked again to hide it. This shows the Printer Output panel popped out but not pinned.



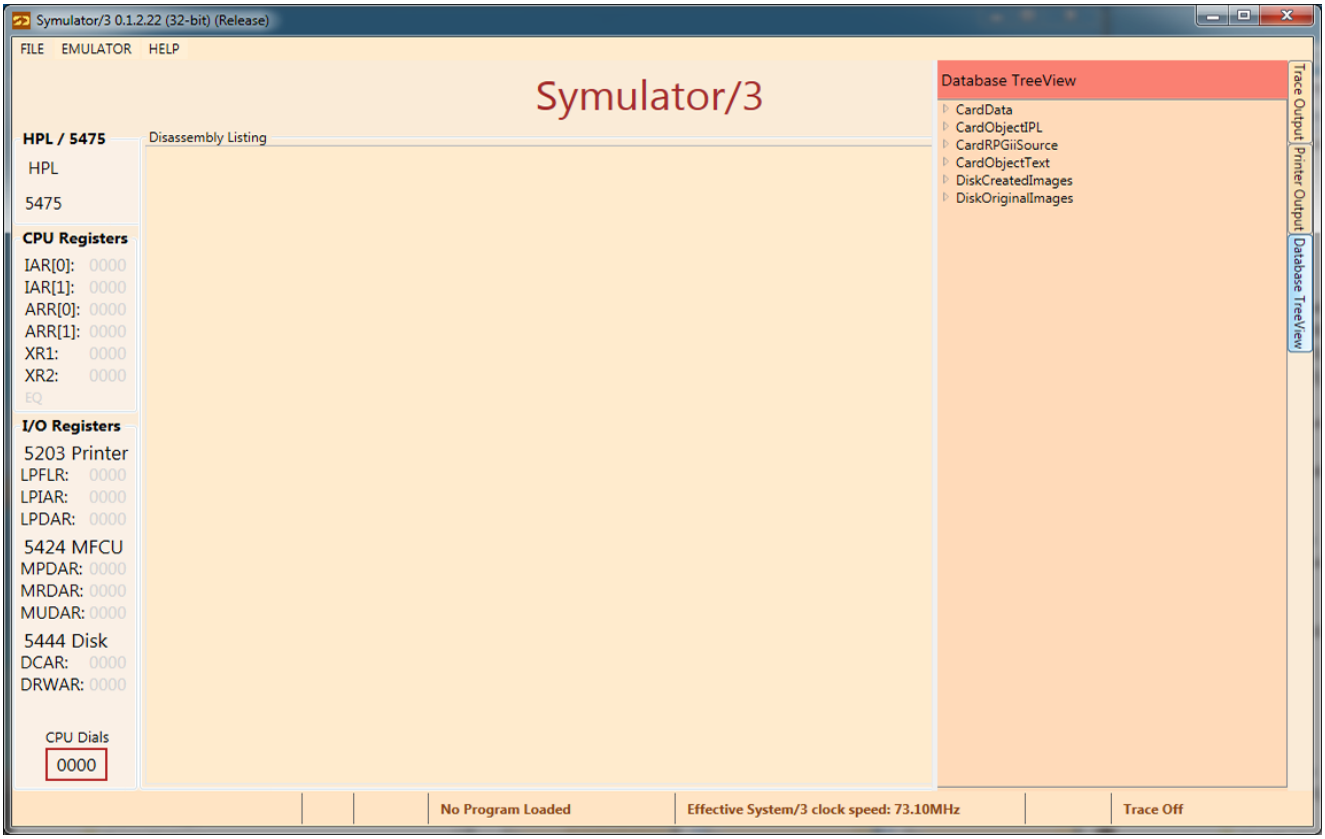
This shows just the Trace Output panel popped out but not pinned.



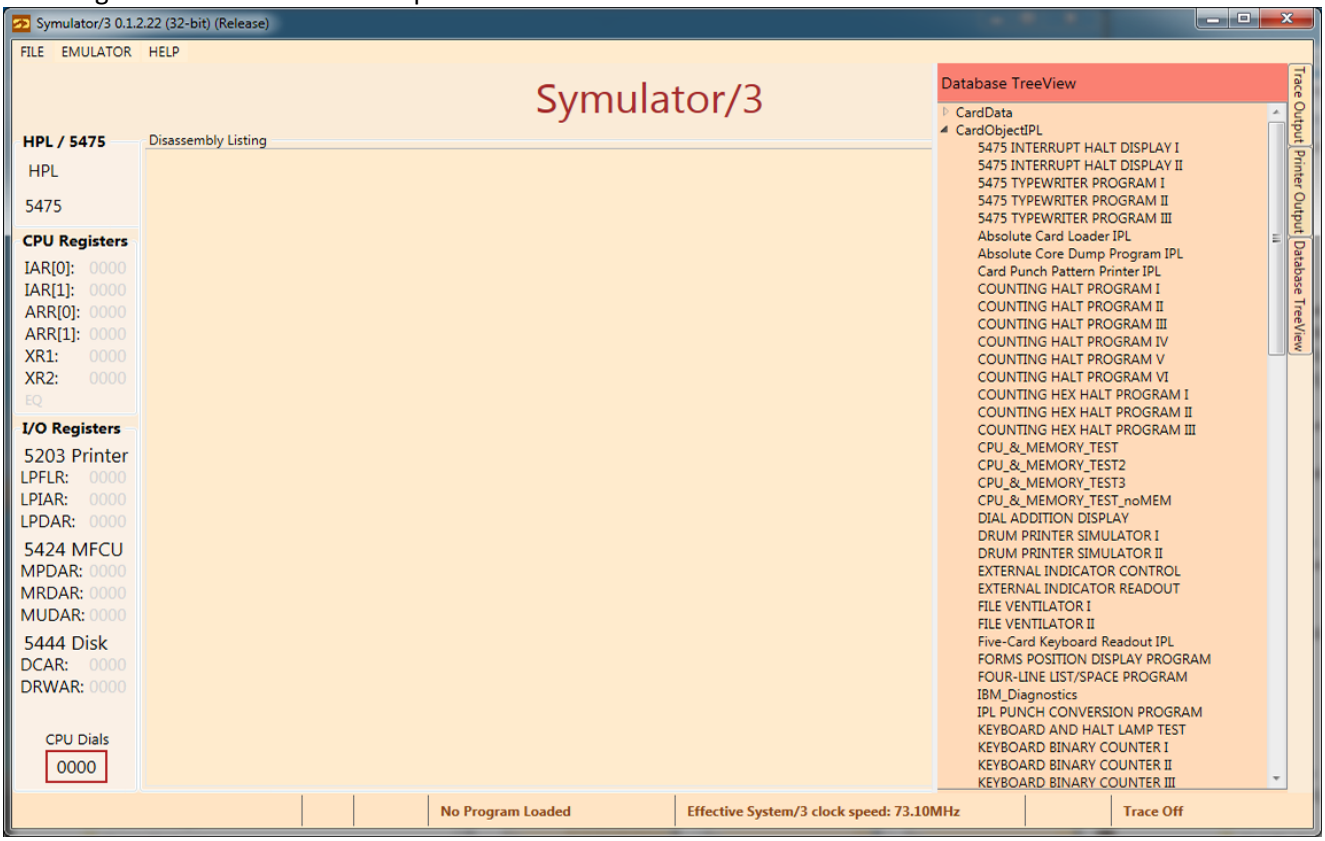
And this shows both panels pinned.



The third pop-out panel is a tree view listing the contents of the Access database containing the text files from all the punched cards that were read to .txt files. Each of the nodes in the tree represents a table in the database and can be expanded (shown in the next screenshot) to reveal the table's contents. This panel can't be pinned as there's no need since it's only needed when making a selection of a program to load or a file to assign to the primary or secondary hopper.



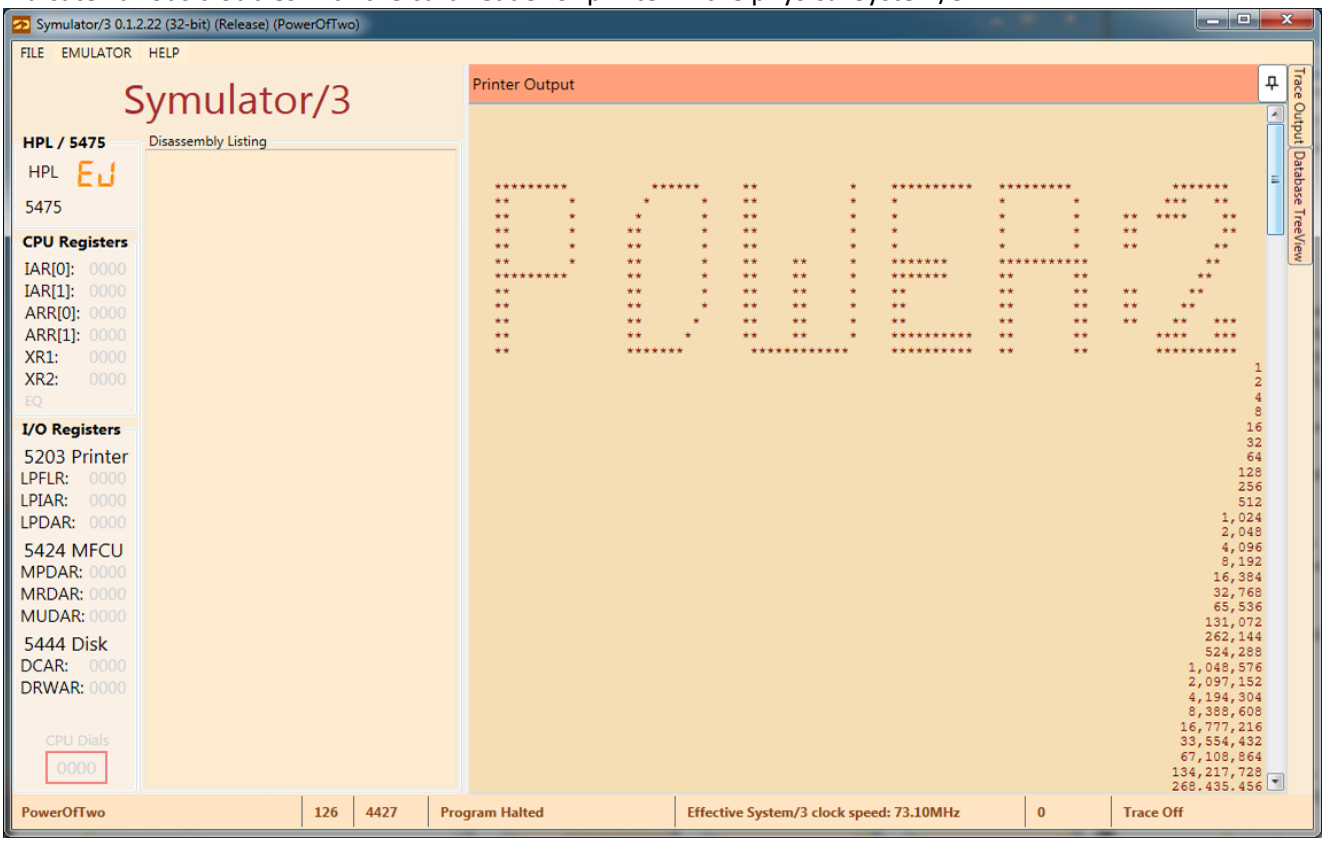
This shows the Database TreeView panel with the CardObjectIPL node expanded to show all the IPL-format programs in the database. Each program can be run by double-clicking on it, or by right-clicking on it and selecting the “IPL Load FreeRun” option.



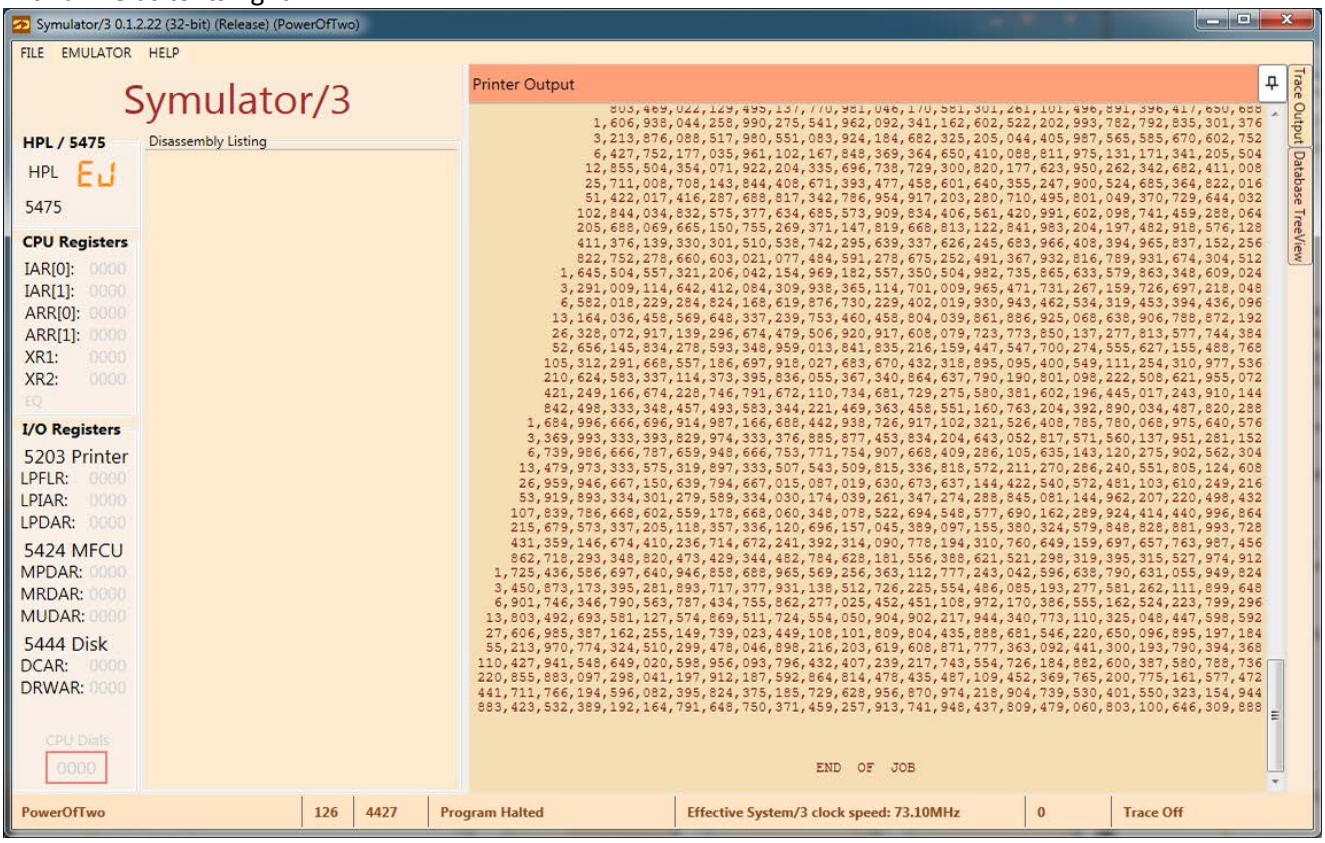
The options available in the context menu vary from one node to another. This is a complete list of the context menu options, none of which are available from every node or file/program:

- "Output to Printer" Show in Printer Output panel
- "Show Card Punch Pattern" Show the pattern of card punches in Printer Output
- "Load Program and Hex Dump" Show hex dump in the Disassembly main window
- "Load Program and Disassemble" Show annotated machine instruction list
- "IPL Load FreeRun" Show disassembly listing and run without interruption
- "IPL Load Break" Show disassembly listing and break on first instruction
- "Show Text Cards as Hex" Show hexadecimal listing of each card in Printer Output
- "Boot From Disk Image" Loads the selected System/3 OS image on disk
- "Clear Primary Hopper" Unload any file previously loaded in primary hopper
- "Clear Secondary Hopper" Unload any file previously loaded in secondary hopper
- "Clear Both Hoppers" Unload any file loaded in either hopper
- "Load in Primary Hopper (if empty)" Load the selected file in the primary hopper
- "Load in Secondary Hopper (if empty)" Load the selected file in the secondary hopper
- "Load in Primary Hopper (replace)" Load the selected file in the primary hopper
- "Load in Secondary Hopper (replace)" Load the selected file in the secondary hopper
- "Load in Primary Hopper (append)" Load the selected file in the primary hopper
- "Load in Secondary Hopper (append)" Load the selected file in the secondary hopper

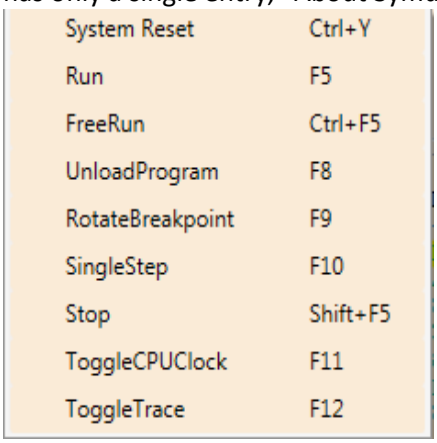
This shows the top of the completed output from double-clicking on and running "PowerOfTwo" in the "CardObjectText" node. Note the "EJ" in the "HPL" section in the left panel. This is the end-of-job halt that signifies that the selected program has run to successful completion. There are other values that would indicate various troubles with the card reader or printer in the physical System/3.



This is the bottom of the PowerOfTwo output listing. The program relies on 6 numeric fields from right to left. The rightmost field is initialized to 1 before printing. With each line printed, the value is doubled and then compared to its previous value. If the new value is *less* than the previous value, it is clear that arithmetic overflow has occurred and the next field on the left is initialized to 1 before printing. Then it is doubled along with all fields to its right.

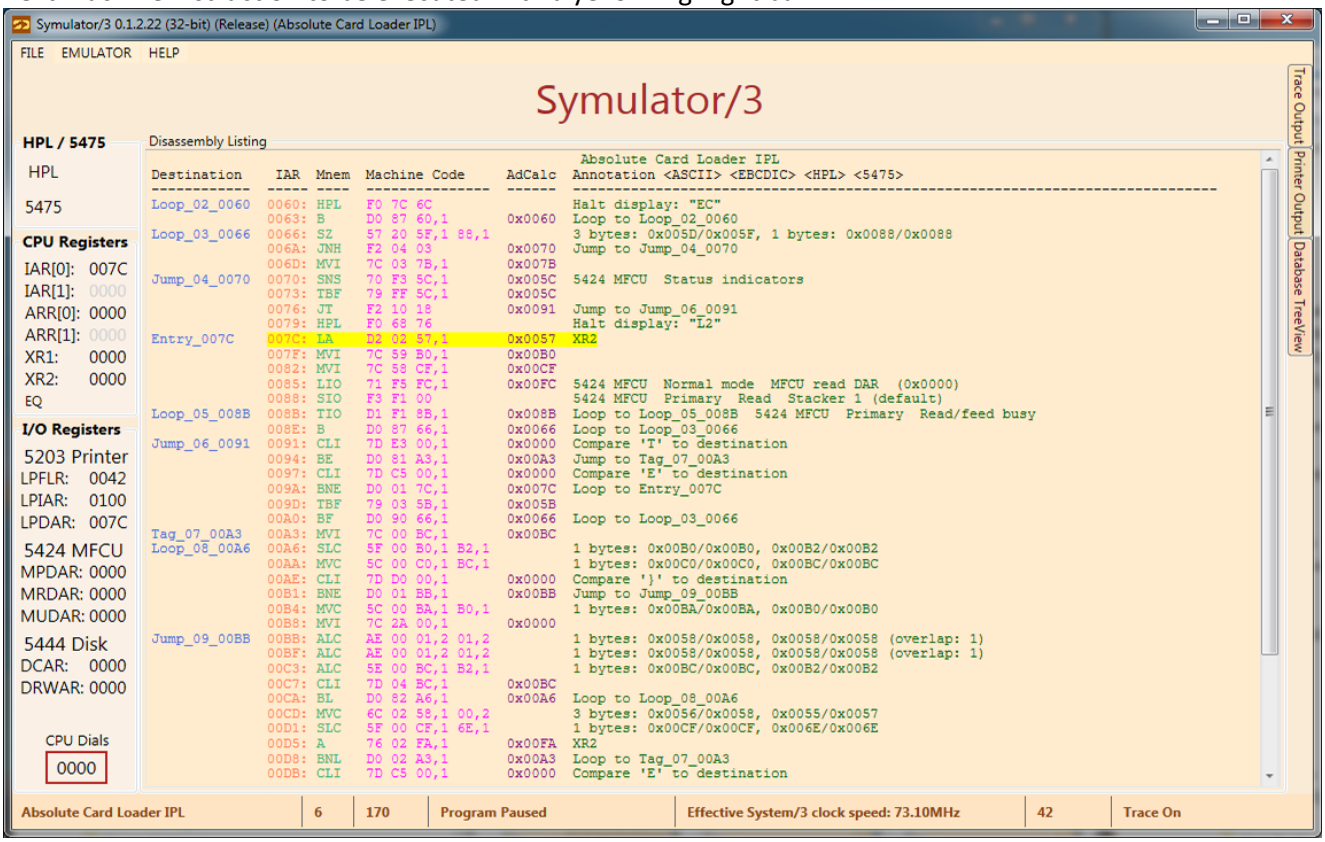


The menu bar has 3 options: File, Emulator, and Help. File has only a single entry at present, “Exit”. Help also has only a single entry, “About Symulator/3”. The Emulator option has several entries:



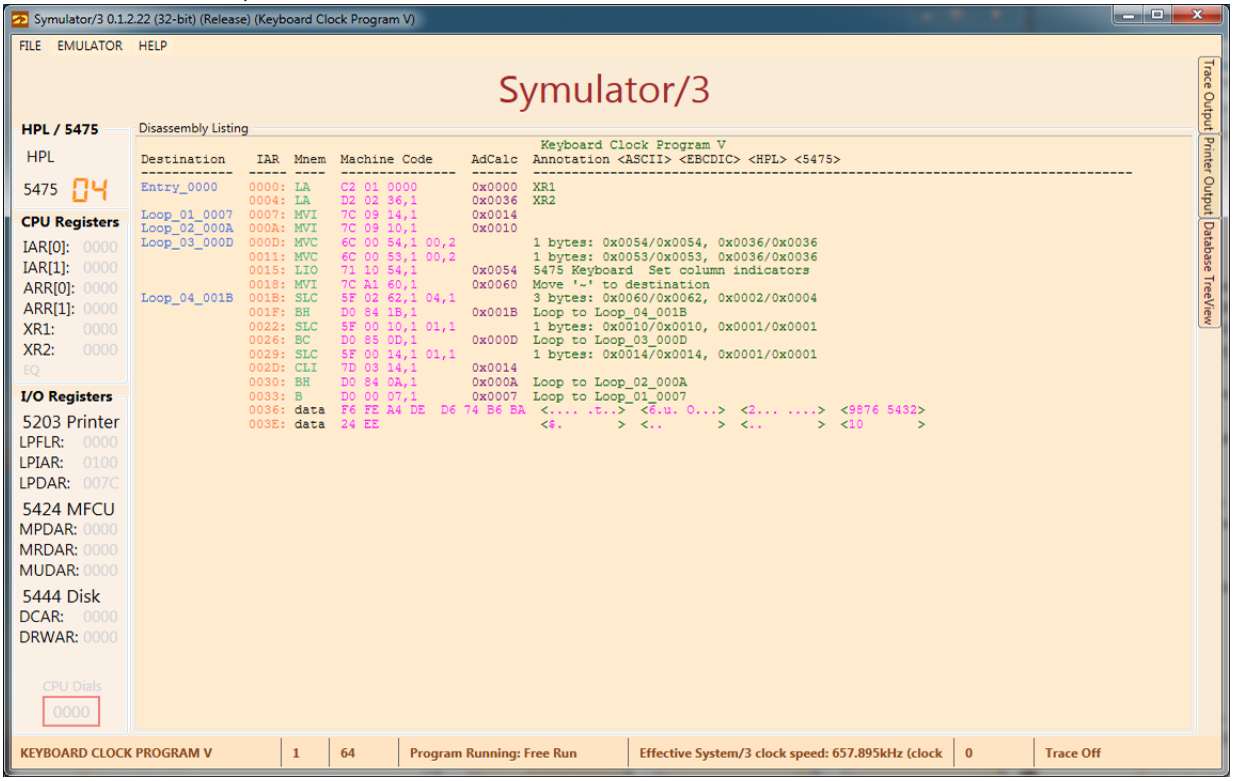
- System Reset stops program execution and resets the IAR, ARR, and condition register
- Run starts or resumes program execution with breakpoints (breakpoints not yet implemented)
- FreeRun starts or resumes program execution without breakpoints
- UnloadProgram performs a System Reset and removes the program from memory
- RotateBreakpoint not yet implemented
- SingleStep executes a single instruction and updates the UI
    - Shows all new register values and HPL/5475 values
    - Updates statusbar panes
    - Sets the current instruction bar to the next instruction to be executed
- Stop interrupts program execution and resets the instruction pointer to the entry point
- ToggleCPUClock switches between full emulator speed and actual System/3 CPU speed
- ToggleTrace enables and disables output to the Trace Output panel

This shows a disassembly listing in the main Disassembly Listing window with the program paused and the next machine instruction to be executed with a yellow highlight bar:

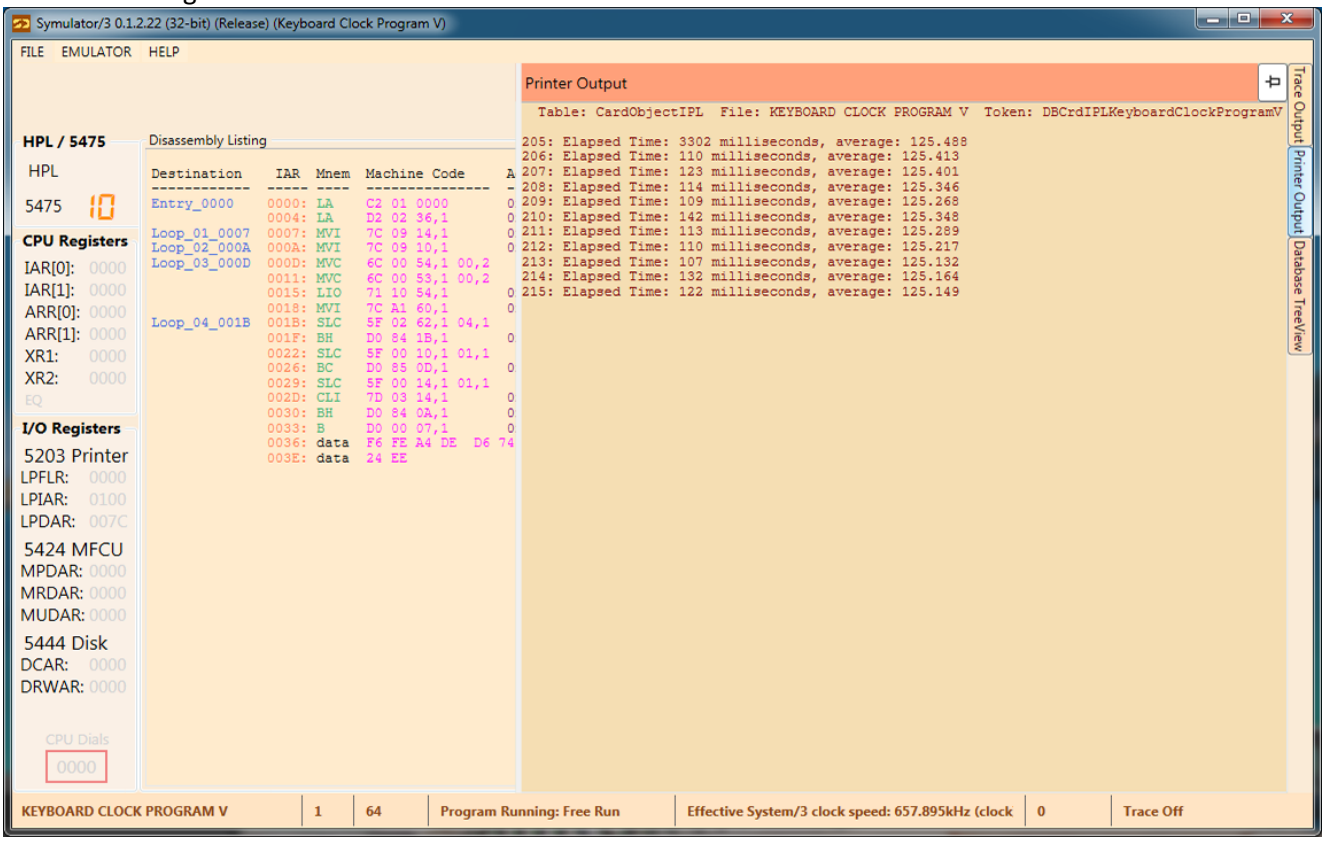


This shows a program running in “FreeRun” mode. Note the 5475 value showing the count of seconds passed. Also note the status bar, showing:

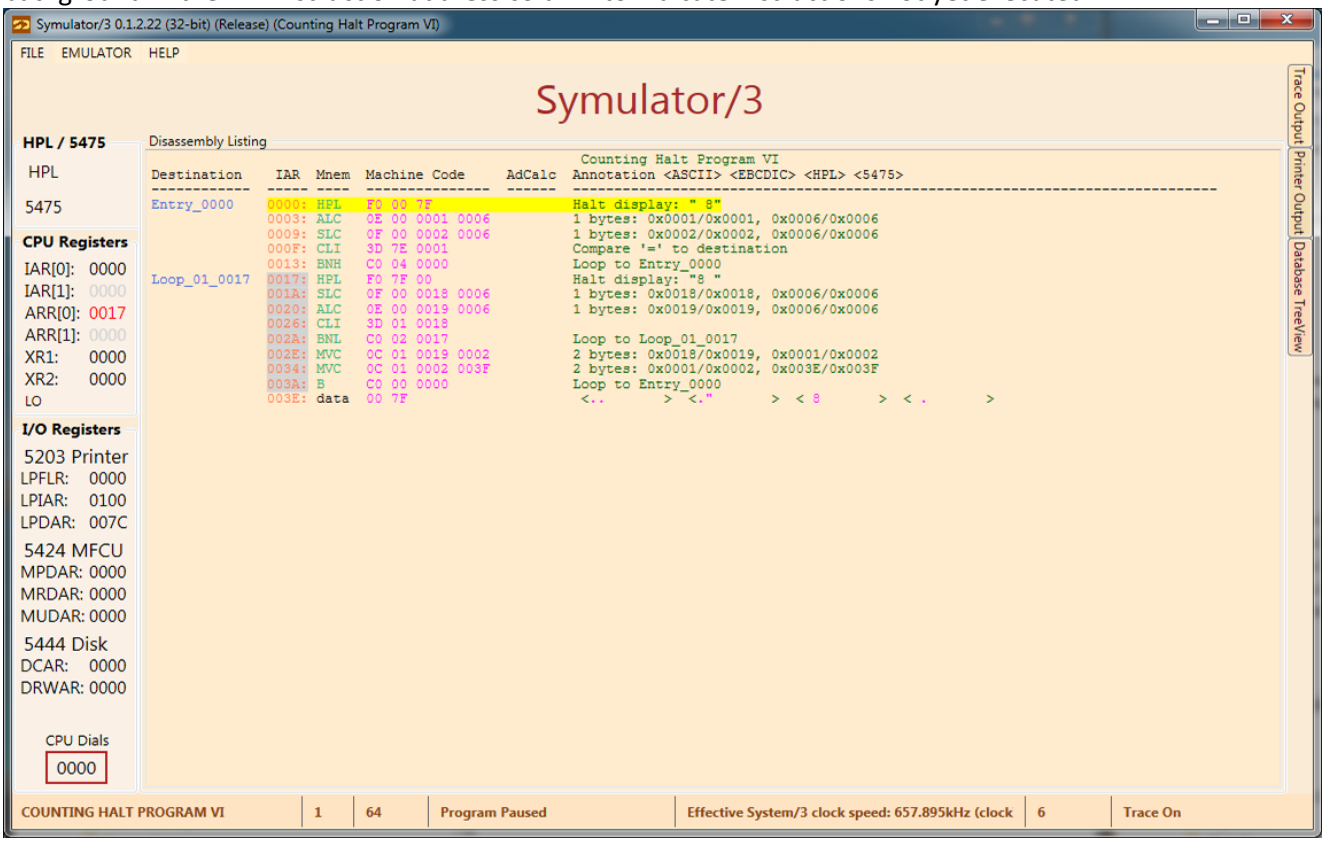
- The name of the program loaded, also shown in the caption bar at the top
- The number of cards in the original program deck
- The size of the loaded program in bytes
- The program state (“Free Run” in this example)
- Effective CPU clock speed
- Trace status, on or off



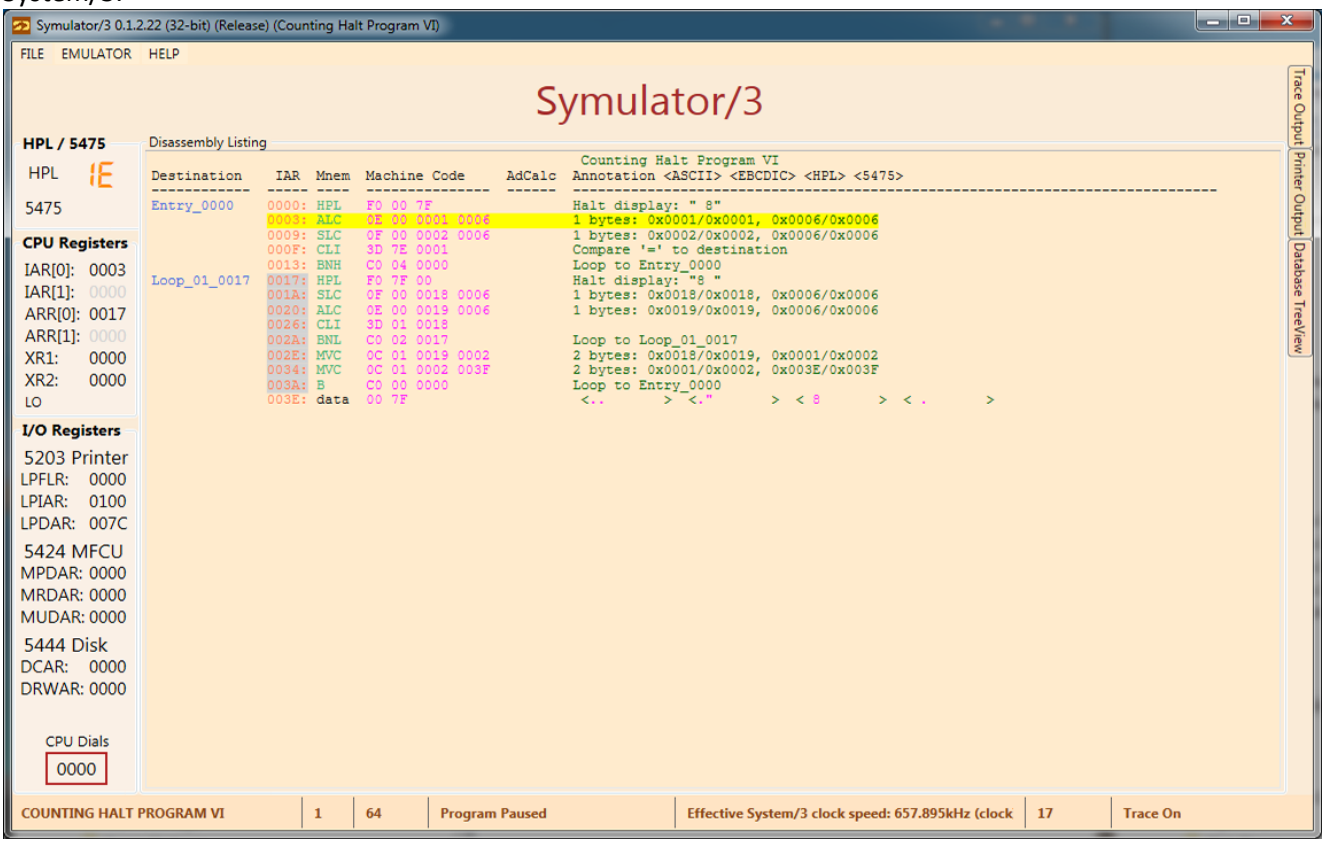
This shows the same program with the “Printer Output” panel popped-out but not pinned, showing output with each change in the 5475 column indicator value.



This shows the “COUNTING HALT PROGRAM VI” program loaded after a 6 machine instructions have been executed. Note that the value of the ARR (Address Recall Register) is shown in red characters to indicate that the value was changed after the last instruction execution. Also note that some of the lines have a gray background in the IAR instruction address column to indicate instructions not yet executed.



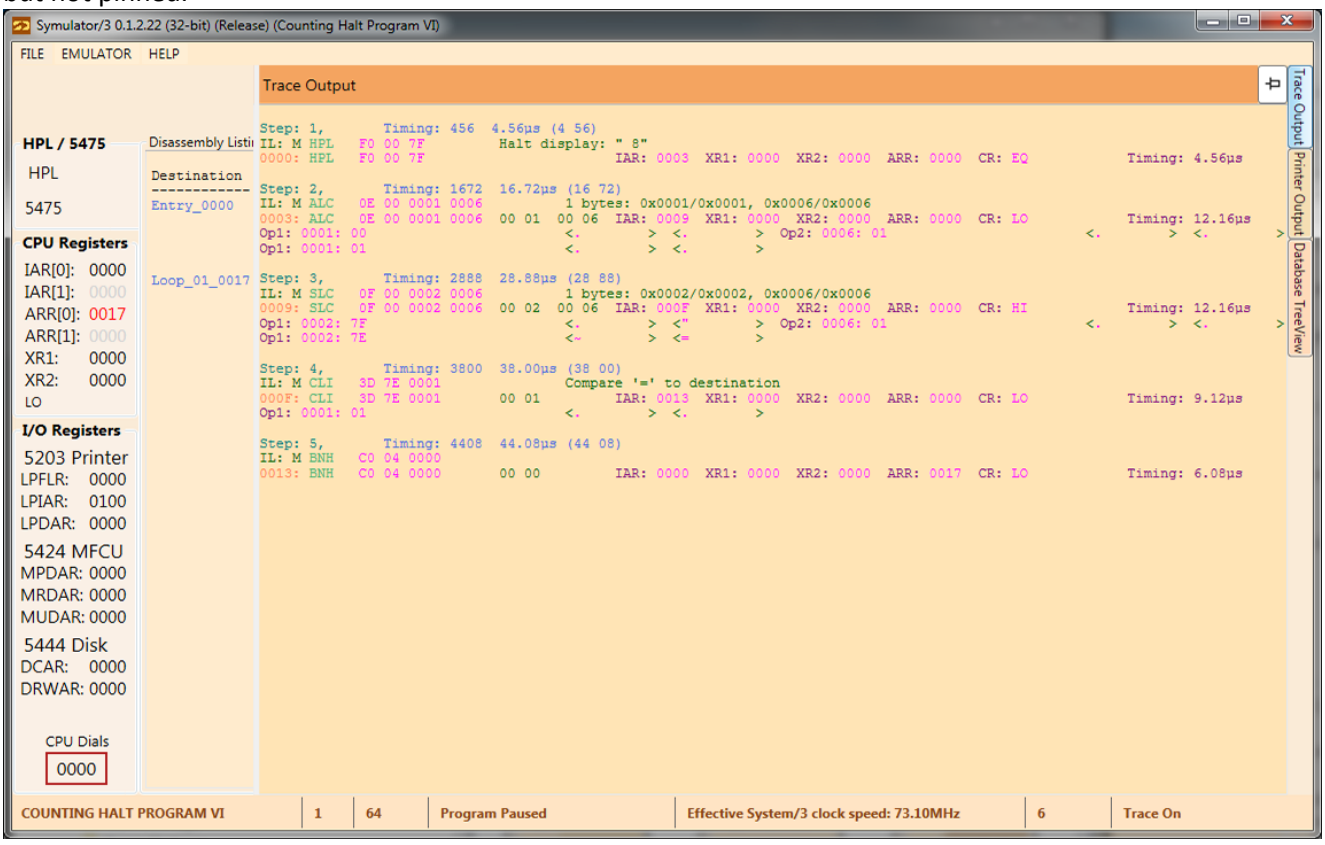
This shows a value in the “HPL” display which represents the actual CPU panel halt indicator on the physical System/3:



The columns in the Disassembly Listing window are:

- Destination: (blue) Disassembly-generated tag for jump or branch instructions consisting of 3 sections separated by underscores. This is generated by the disassembler when it encounters a jump or branch instruction, to indicate branch or jump target, shown in the Annotation. If the destination is referenced only by instructions of higher addresses, it's a "Loop". If only by lower addresses, it's a "Jump". If both, it's a "Tag". "Entry" is the program starting point.
  - Destination type: Loop, Jump, or Tag
  - The sequence of the destination, incremented with each jump/branch encountered
  - The address of the destination
- IAR: (orange) "Instruction Address Register" showing the instruction address
- Mnem: (green) The instruction assembly code mnemonic
- Machine Code: (magenta) The hexadecimal value of the machine instruction code
- AddCalc: (purple) The calculated address using the index register selected in the instruction
- Annotation: (dark green) Disassembly-generated comments explaining the instruction's function
  Annotation lines may wrap onto the following line

This show the same program after 6 instructions have been executed, with the Trace Output panel popped out but not pinned.

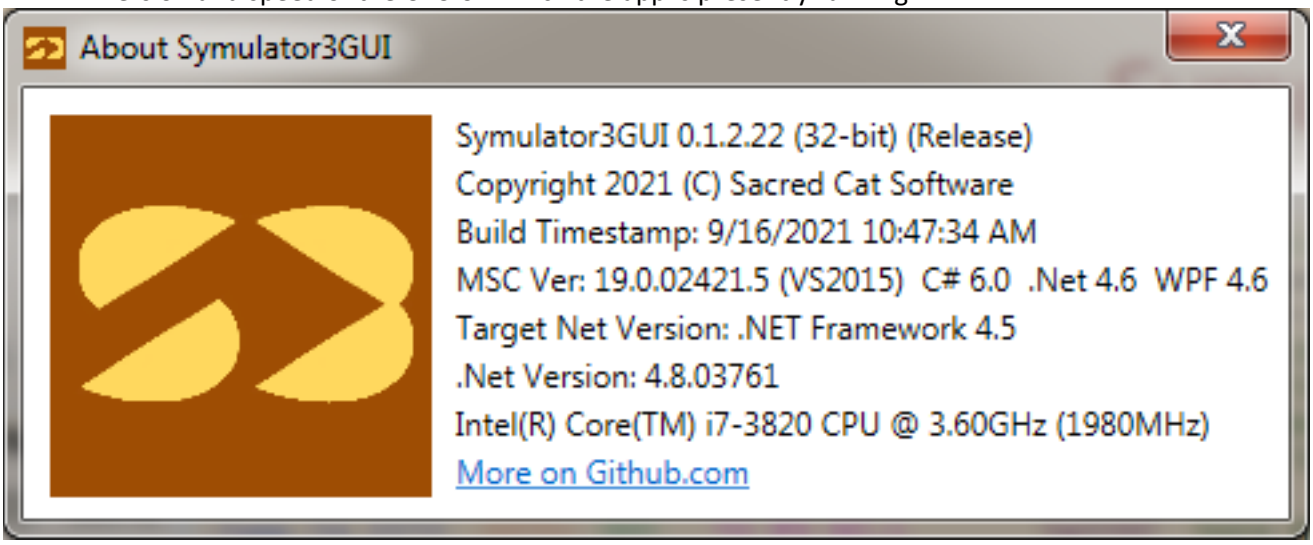


The Trace Output window shows 4 line formats for each instruction executed:

1. Top line shows the instruction count (teal) and the accumulated timing in microseconds based on the physical System/3 CPU clock speed (657.895 kHz)
2. Interrupt Level (dark green): “M” for the main level, or the number of the level, then the instruction mnemonic (bright green), the machine instruction hexadecimal code (magenta), then any annotation comments available (dark green)
3. Instruction address (orange), the instruction mnemonic (bright green), the machine instruction hexadecimal code (magenta), then the CPU registers shown in the left panel (labels in dark purple, values in magenta), then the timing in microseconds of the individual instruction based on the physical System/3 CPU clock speed.
4. If the instruction references operands in memory, this line shows operand 1, and if a second operand is used, it is shown further to the right. The data value is displayed in hexadecimal on the left, and in ASCII and in EBCDIC. If operand 1 is changed by the instruction, then the value after the instruction executes is shown in second operand line. Operand 2 is never changed by any instruction.

This shows the About box which shows several pieces of information:

- Version & build and build configuration
- Build date and time
- Version of the compiler used to create the build, and versions of C#, .Net, and WPF
- Required minimum .NET version for execution
- Full text of .NET version used to create the build
- Version and speed of the CPU on which the app is presently running



The complete source code for this app is on GitHub.

The System/3 machine instruction reference manual is also on GitHub.

My history with the IBM System/3 and why I bothered to create all this code is here.