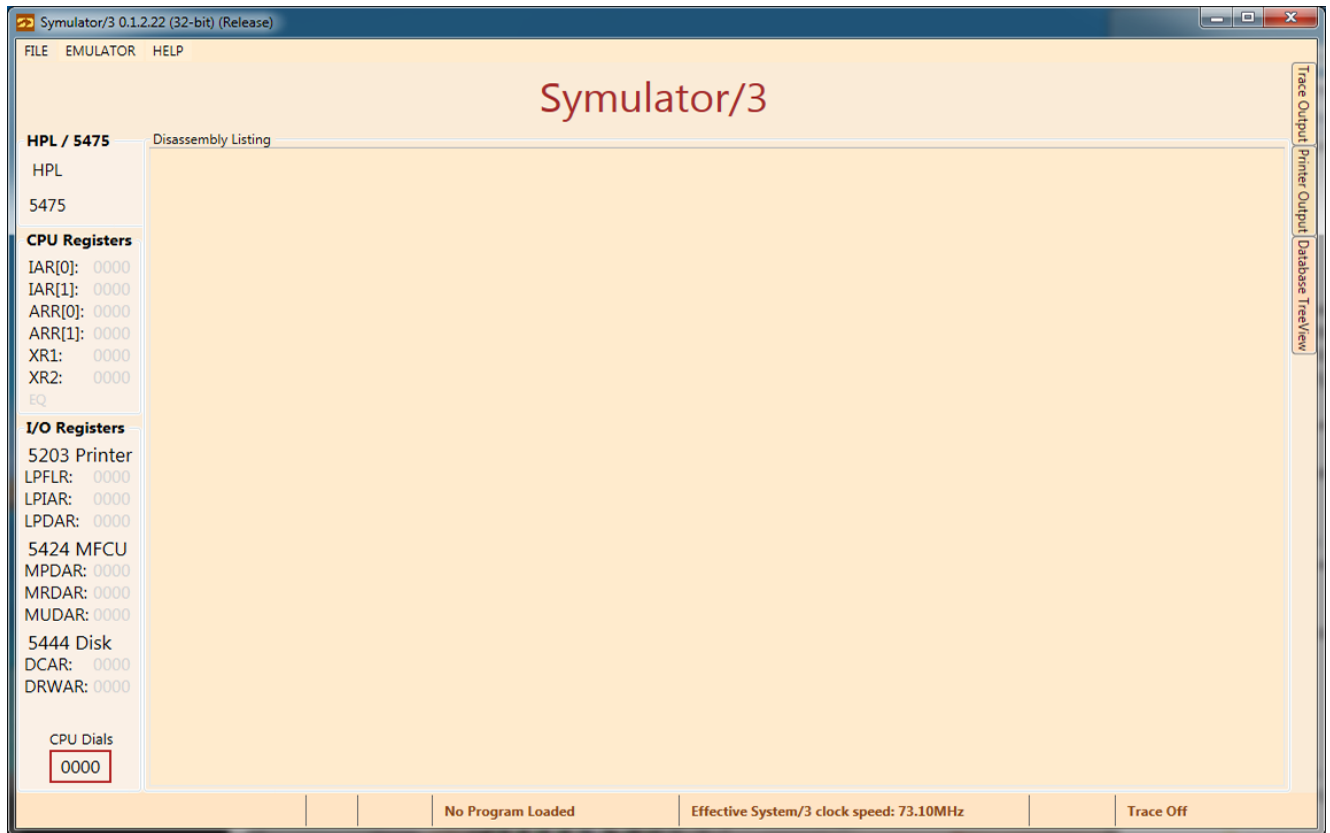


# Simulator/3 Visual Studio GUI

Simulator/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.

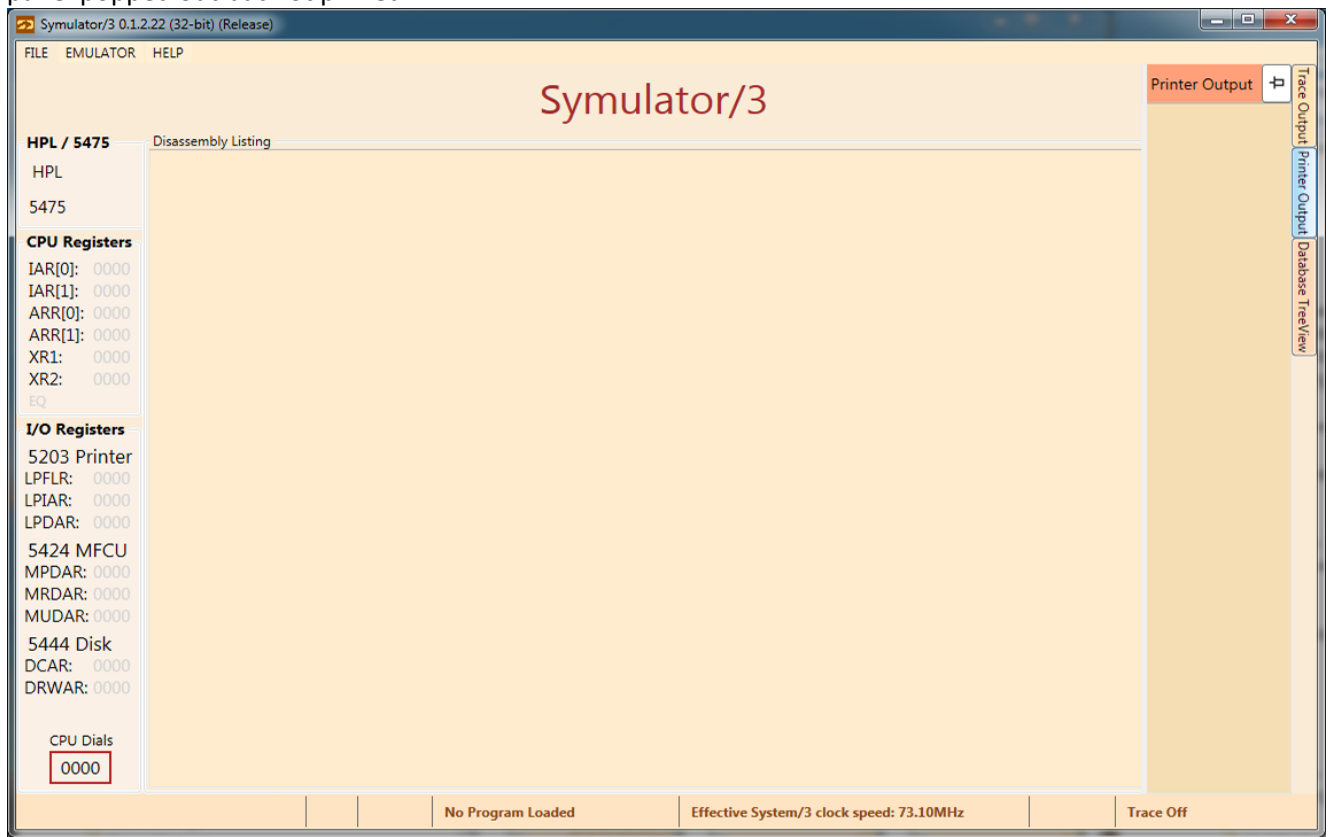


The first screenshot shows the UI in its initial state after having just been started. The left panel is fixed and shows the values in each of the CPU and I/O registers. The values show as grayed out and stay gray until a program has been launched. At the top are the displays for the 7-segment halt and 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.

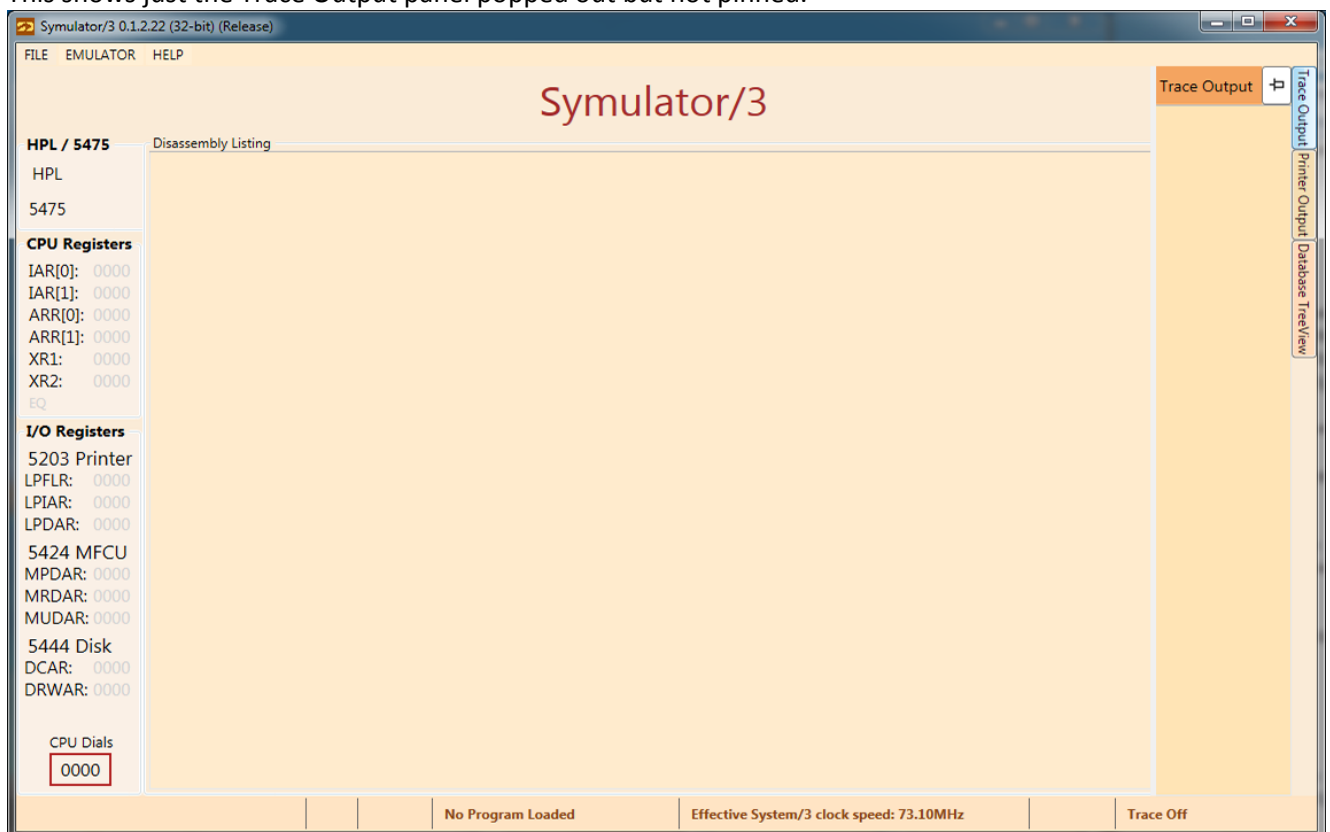
At the bottom is the status bar which contains panes to show:

- 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

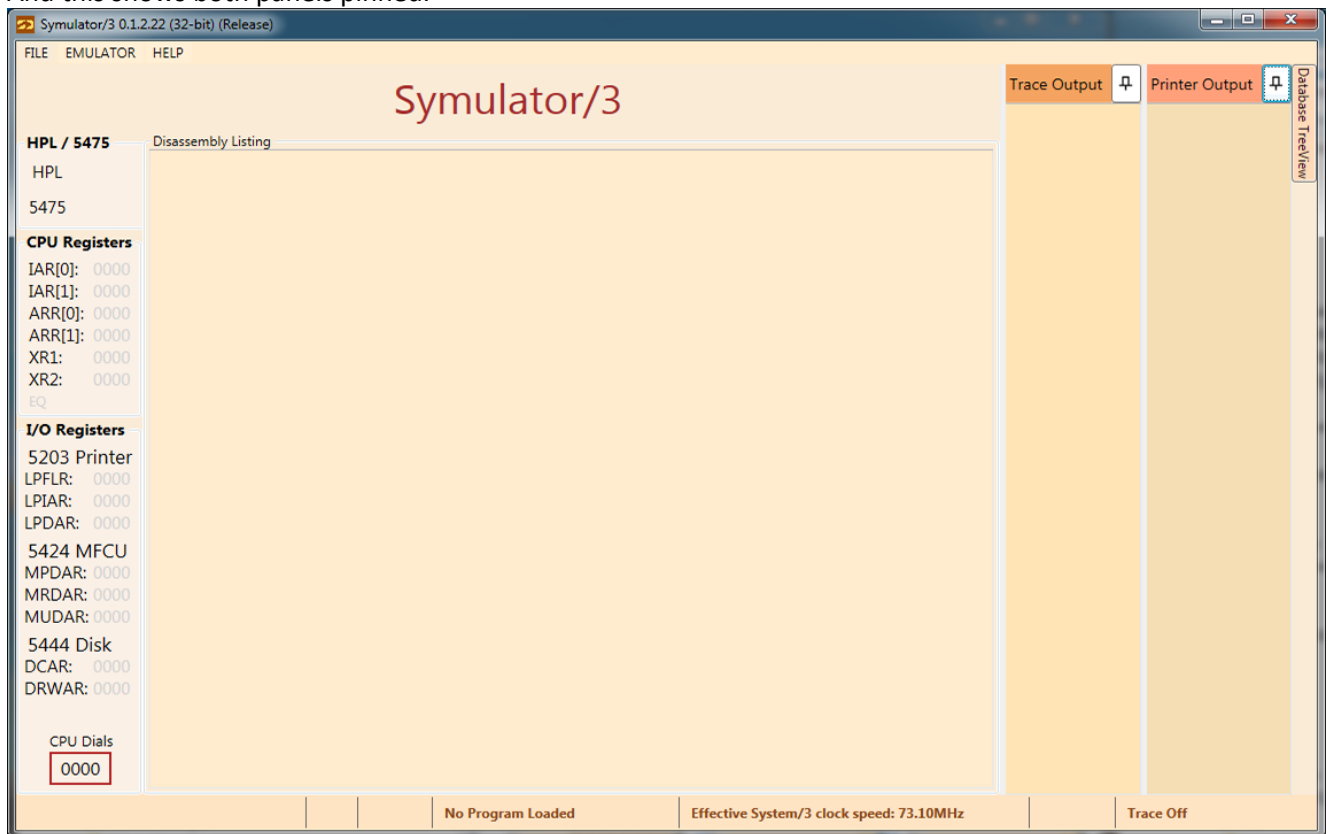
The UI has panels that pop out from the right side and a fixed status panel on the left side, and a status bar at the bottom. The pop-out panels can be pinned and unpinned, as in Visual Studio. Like Visual Studio, the pop-out panels can be shown temporarily by hovering the mouse over the title tabs. 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.



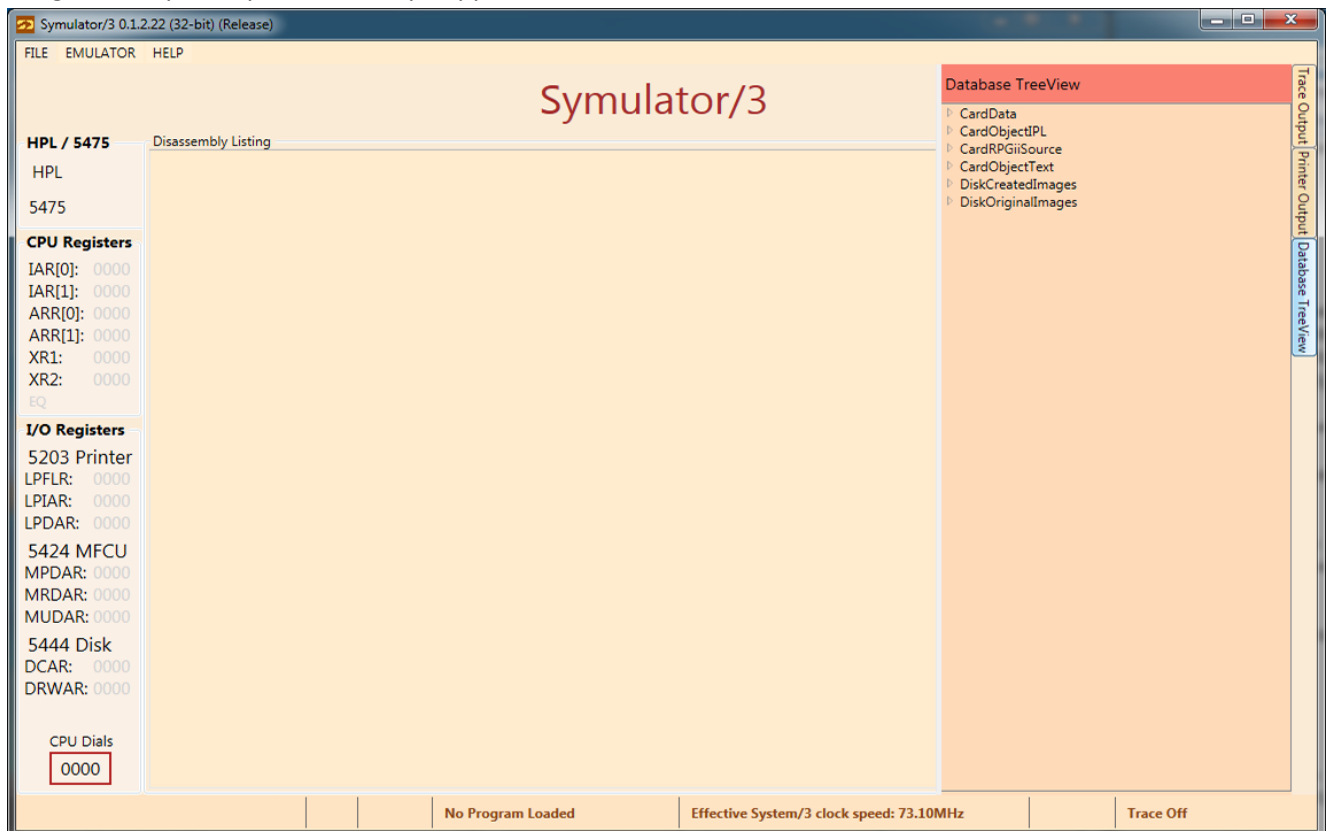
This shows just the Trace Output panel popped out but not pinned.



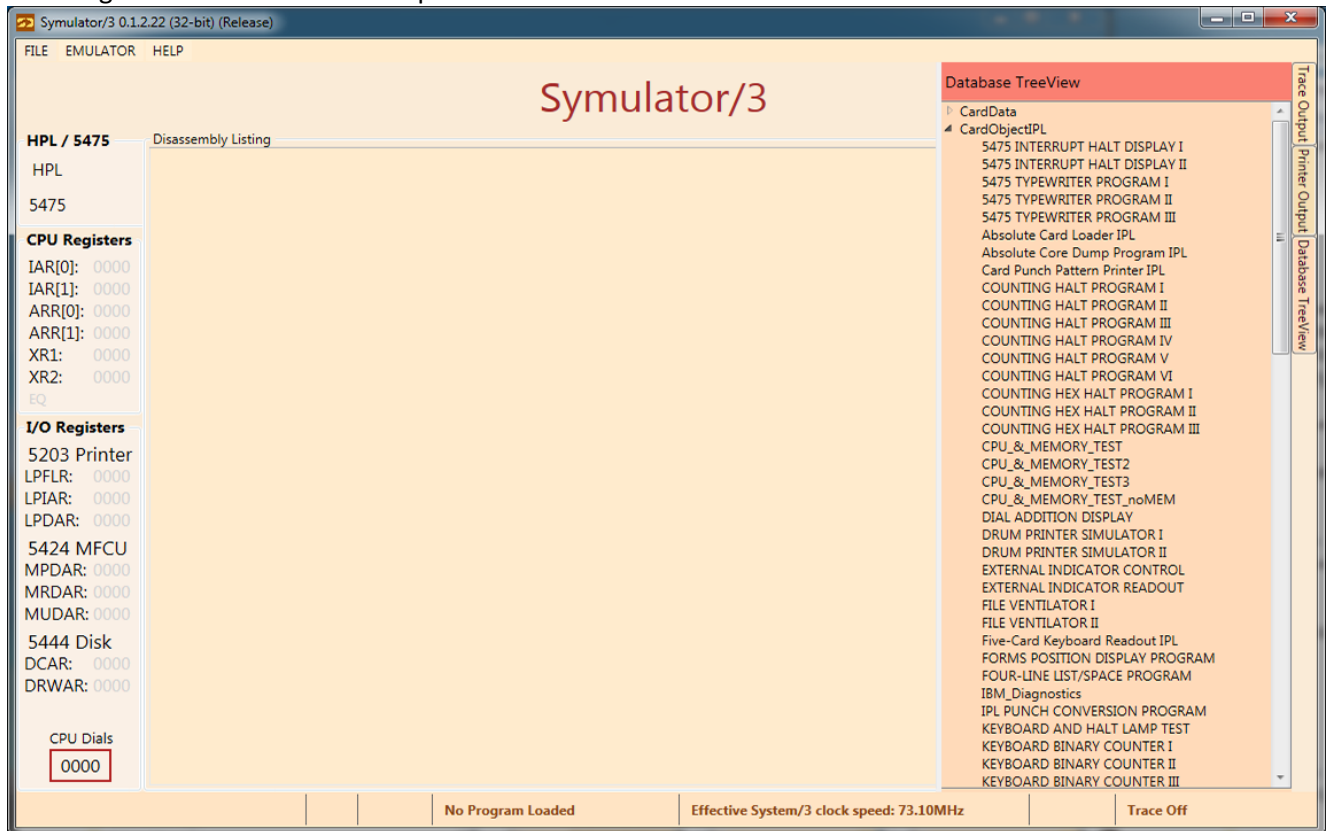
And this shows both panels pinned.



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.



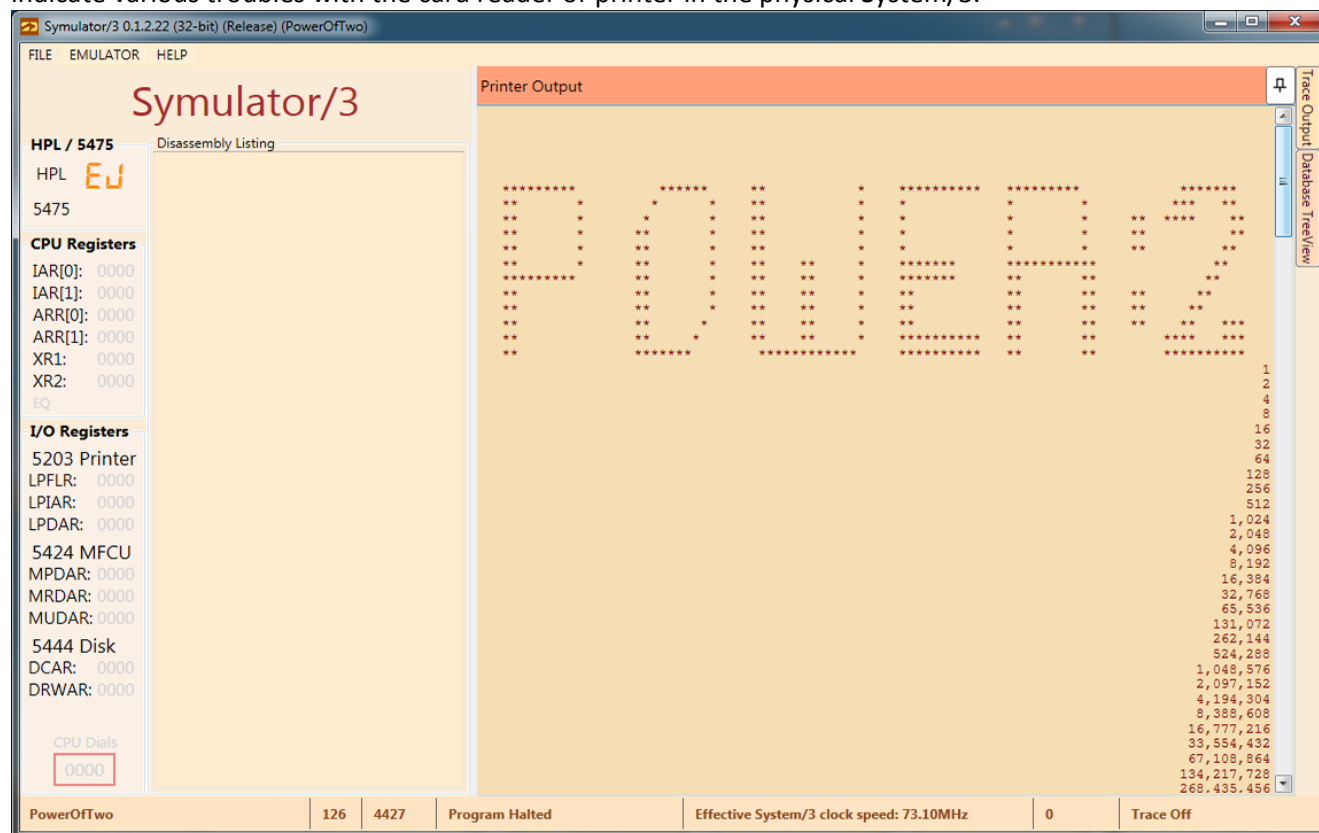
This shows the Database TreeView panel with the CardObjectIPL node expanded to show all the IPL-format programs in the database. Each program can be run by double-clicking on it, or by right-clicking on it and selecting the "IPL Load FreeRun" option.



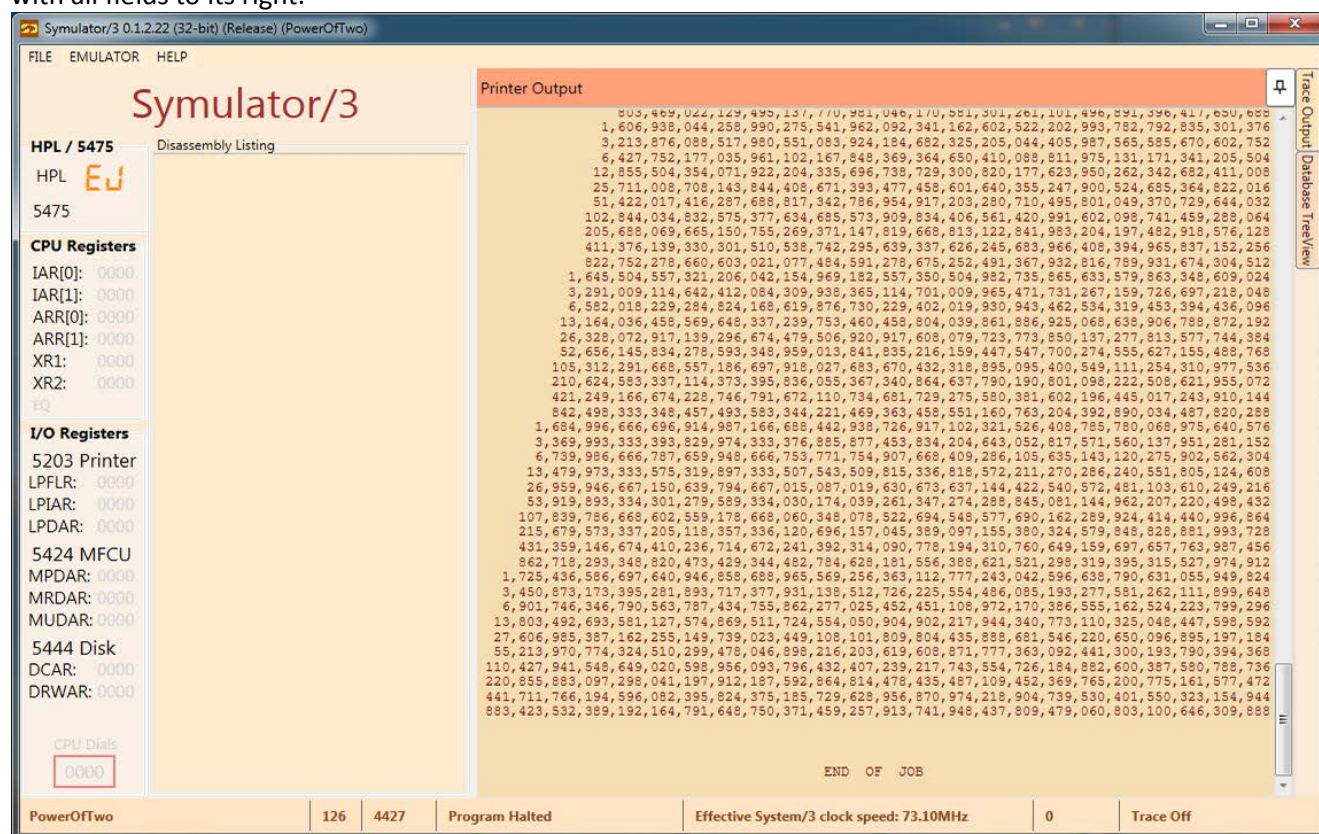
The options available in the context menu vary from one node to another. This is a complete list of the context menu options, none of which are available from every node or file/program:

- |   |   |
|---|---|
| • "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 “CardObjectContext” 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 Simulator/3". The Emulator option has several entries:

System Reset	Ctrl+Y
Run	F5
FreeRun	Ctrl+F5
UnloadProgram	F8
RotateBreakpoint	F9
SingleStep	F10
Stop	Shift+F5
ToggleCPUClock	F11
ToggleTrace	F12

- 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:

**Simulator/3**

**HPL / 5475**

**Disassembly Listing**

Destination	IAR	Mnem	Machine Code	AdCalc	Annotation <ASCII> <EBCDIC> <HPL> <5475>
5475	0060: HPL	F0 7C 6C			Halt display: "EC"
Loop_02_0060	0063: B	D0 87 60,1	0x0060		Loop to Loop_02_0060
Loop_03_0066	0066: SZ	57 20 5F,1 88,1	0x0070		3 bytes: 0x005D/0x005F, 1 bytes: 0x0088/0x0088
Loop_04_0070	006A: JNH	F2 04 03	0x0070		Jump to Jump_04_0070
Jump_04_0070	006D: MVI	7C 03 7B,1	0x007B		5424 MFCU Status indicators
Entry_007C	0070: SNS	70 F3 5C,1	0x005C		
	0073: TBF	79 FF 5C,1	0x005C		
	0076: JT	F2 10 18	0x0091		Jump to Jump_06_0091
	0079: HPL	F0 68 76	0x0091		Halt display: "L2"
Entry_007C	007E: LA	82 88 87,1	0x0087		XR2
Loop_05_008B	0082: MVI	7C 58 8F,1	0x00B0		
	0085: LIO	71 F5 FC,1	0x00CF		5424 MFCU Normal mode MFCU read DAR (0x0000)
	0088: SIO	F3 F1 00	0x00FC		5424 MFCU Primary Read Stacker 1 (default)
Loop_06_0091	008B: TIO	D1 F1 8B,1	0x008B		Loop to Loop_05_008B 5424 MFCU Primary Read/feed busy
	008E: B	D0 87 66,1	0x0066		Loop to Loop_03_0066
	0091: CLI	7D E3 00,1	0x0000		Compare 'T' to destination
	0094: BE	D0 81 A3,1	0x00A3		Jump to Tag_07_00A3
	0097: CLI	7D C5 00,1	0x0000		Compare 'E' to destination
	009A: BNE	D0 01 7C,1	0x007C		Loop to Entry_007C
	009D: TBF	79 03 5B,1	0x005B		
	00A0: BF	D0 90 66,1	0x0066		Loop to Loop_03_0066
Tag_07_00A3	00A3: MVI	7C 00 BC,1	0x00BC		
Loop_08_00A6	00A6: SLC	5F 00 B0,1 B2,1	0x00B0		1 bytes: 0x00B0/0x00B0, 0x00B2/0x00B2
	00AA: MVC	5C 00 C0,1 BC,1	0x00C0		1 bytes: 0x00C0/0x00C0, 0x00BC/0x00BC
	00AE: CLI	7D D0 00,1	0x0000		Compare 'J' to destination
	00B1: BNE	D0 01 BB,1	0x00BB		Jump to Jump_09_00BB
	00B4: MVC	5C 00 EA,1 B0,1	0x00EA		1 bytes: 0x00EA/0x00EA, 0x00B0/0x00B0
	00B8: MVI	7C 2A 00,1	0x0000		
Jump_09_00BB	00BB: ALC	AE 00 01,2 01,2	0x0058		1 bytes: 0x0058/0x0058, 0x0058/0x0058 (overlap: 1)
	00BF: ALC	AE 00 01,2 01,2	0x0058		1 bytes: 0x0058/0x0058, 0x0058/0x0058 (overlap: 1)
	00C3: ALC	5E 00 BC,1 B2,1	0x00BC		1 bytes: 0x00BC/0x00BC, 0x00B2/0x00B2
	00C7: CLI	7D 04 BC,1	0x00BC		
	00CA: BL	D0 82 A6,1	0x00A6		Loop to Loop_08_00A6
	00CD: MVC	6C 02 58,1 00,2	0x0058		3 bytes: 0x0056/0x0058, 0x0055/0x0057
	00D1: SLC	5F 00 CF,1 6E,1	0x00CF		1 bytes: 0x00CF/0x00CF, 0x006E/0x006E
	00D5: A	76 02 FA,1	0x00FA		XR2
	00D8: BNL	D0 02 A3,1	0x00A3		Loop to Tag_07_00A3
	00DB: CLI	7D C5 00,1	0x0000		Compare 'E' to destination

**CPU Dials**

0000

**Status Bar:** Absolute Card Loader IPL | 6 | 170 | Program Paused | Effective System/3 clock speed: 73.10MHz | 42 | Trace On

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

**Symulator/3**

FILE EMULATOR HELP

**HPL / 5475**

Disassembly Listing

Destination	IAR	Mnem	Machine Code	AdCalc	Annotation
Entry_0000	0000: LA	C2 01 0000	0x0000	XR1	
	0004: LA	D2 02 36,1	0x0036	XR2	
Loop_01_0007	0007: MVI	7C 09 14,1	0x0014		
Loop_02_000A	000A: MVI	7C 09 10,1	0x0010		
Loop_03_000D	000D: MVC	6C 00 54,1 00,2			1 bytes: 0x0054/0x0054, 0x0036/0x0036
	0011: MVC	6C 00 53,1 00,2			1 bytes: 0x0053/0x0053, 0x0036/0x0036
	0015: LIO	71 10 54,1	0x0054		5475 Keyboard Set column indicators
	0018: MVI	7C A1 60,1	0x0060		Move '-' to destination
Loop_04_001B	001B: SLC	SF 02 62,1 04,1			3 bytes: 0x0060/0x0062, 0x0002/0x0004
	001F: BH	D0 84 1B,1	0x001B		Loop to Loop_04_001B
	0022: SLC	SF 00 10,1 01,1			1 bytes: 0x0010/0x0010, 0x0001/0x0001
	0026: BC	D0 85 0D,1	0x000D		Loop to Loop_03_000D
	0029: SLC	SF 00 14,1 01,1			1 bytes: 0x0014/0x0014, 0x0001/0x0001
	002D: CLI	7D 03 14,1	0x0014		Loop to Loop_02_000A
	0030: BH	D0 84 0A,1	0x000A		Loop to Loop_01_0007
	0033: B	D0 00 07,1	0x0007		
	0036: data	F6 FE A4 DE D6 74 B6 BA			<.... .t...> <8.u. 0...> <2... ..> <9876 5432>
	003E: data	24 EE			<#. > <.. > <.. > <10 >

**CPU Registers**

IAR[0]: 0000  
IAR[1]: 0000  
ARR[0]: 0000  
ARR[1]: 0000  
XR1: 0000  
XR2: 0000  
EQ

**I/O Registers**

5203 Printer  
LPFLR: 0000  
LPIAR: 0100  
LPDAR: 007C  
5424 MFCU  
MPDAR: 0000  
MRDAR: 0000  
MUDAR: 0000  
5444 Disk  
DCAR: 0000  
DRWAR: 0000

CPU Dials  
0000

KEYBOARD CLOCK PROGRAM V | 1 | 64 | Program Running: Free Run | Effective System/3 clock speed: 657.895kHz (clock) | 0 | Trace 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.

**Symulator/3**

FILE EMULATOR HELP

**HPL / 5475**

Disassembly Listing

Destination	IAR	Mnem	Machine Code	A
Entry_0000	0000: LA	C2 01 0000		
	0004: LA	D2 02 36,1		
Loop_01_0007	0007: MVI	7C 09 14,1		
Loop_02_000A	000A: MVI	7C 09 10,1		
Loop_03_000D	000D: MVC	6C 00 54,1 00,2		
	0011: MVC	6C 00 53,1 00,2		
	0015: LIO	71 10 54,1		
	0018: MVI	7C A1 60,1		
Loop_04_001B	001B: SLC	SF 02 62,1 04,1		
	001F: BH	D0 84 1B,1		
	0022: SLC	SF 00 10,1 01,1		
	0026: BC	D0 85 0D,1		
	0029: SLC	SF 00 14,1 01,1		
	002D: CLI	7D 03 14,1		
	0030: BH	D0 84 0A,1		
	0033: B	D0 00 07,1		
	0036: data	F6 FE A4 DE D6 74 B6 BA		
	003E: data	24 EE		

**CPU Registers**

IAR[0]: 0000  
IAR[1]: 0000  
ARR[0]: 0000  
ARR[1]: 0000  
XR1: 0000  
XR2: 0000  
EQ

**I/O Registers**

5203 Printer  
LPFLR: 0000  
LPIAR: 0100  
LPDAR: 007C  
5424 MFCU  
MPDAR: 0000  
MRDAR: 0000  
MUDAR: 0000  
5444 Disk  
DCAR: 0000  
DRWAR: 0000

CPU Dials  
0000

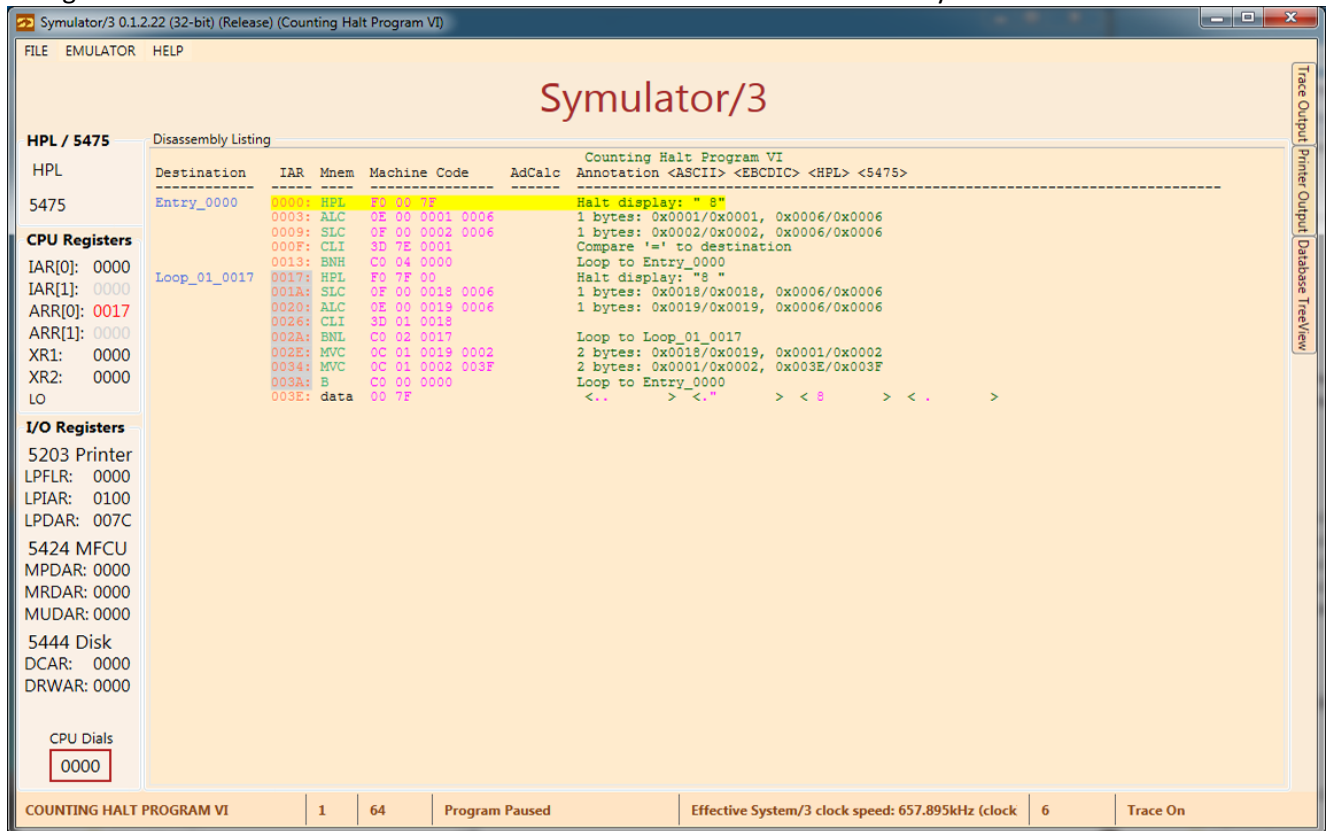
**Printer Output**

Table: CardObjectIPL File: KEYBOARD CLOCK PROGRAM V Token: DBCrdIPLKeyboardClockProgramV

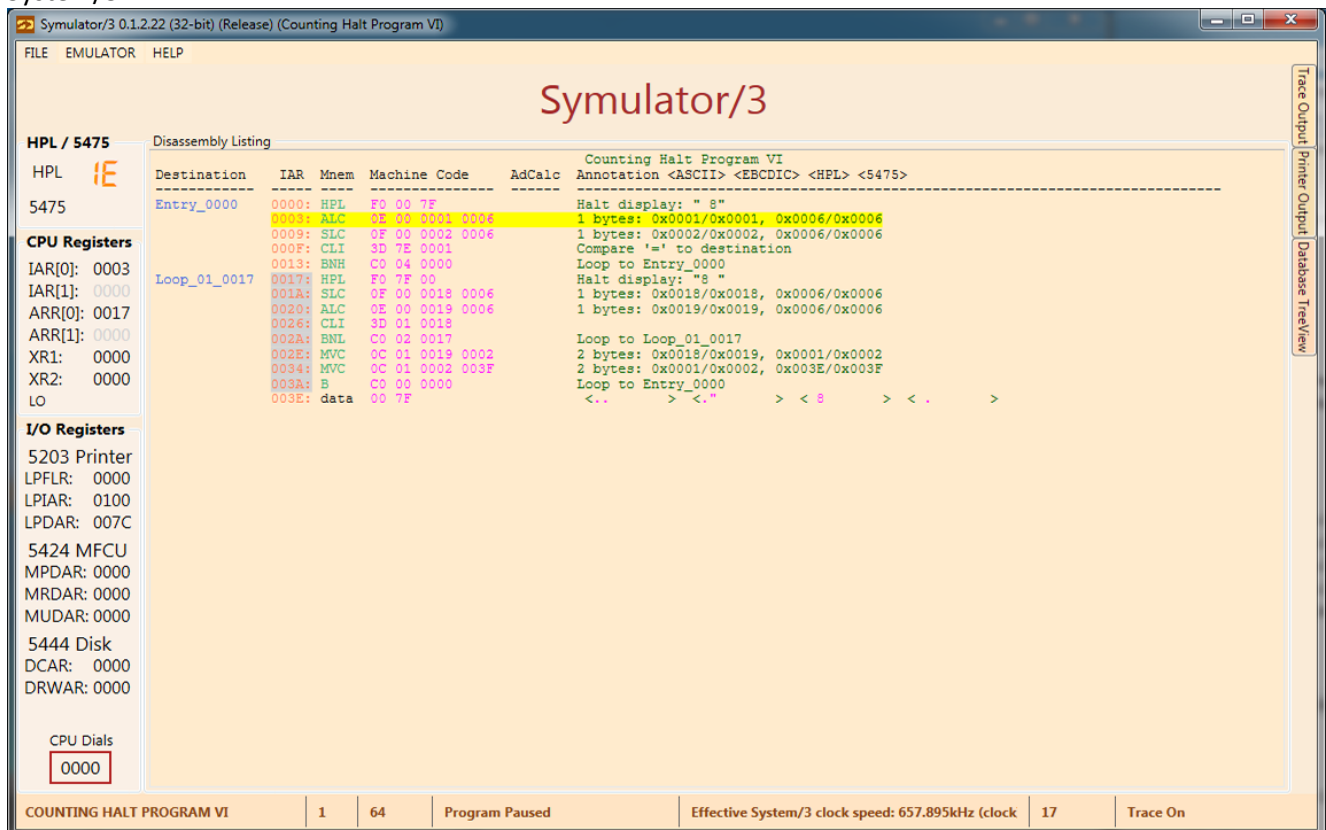
205: Elapsed Time: 3302 milliseconds, average: 125.488
206: Elapsed Time: 110 milliseconds, average: 125.413
207: Elapsed Time: 123 milliseconds, average: 125.401
208: Elapsed Time: 114 milliseconds, average: 125.346
209: Elapsed Time: 109 milliseconds, average: 125.268
210: Elapsed Time: 142 milliseconds, average: 125.348
211: Elapsed Time: 113 milliseconds, average: 125.289
212: Elapsed Time: 110 milliseconds, average: 125.217
213: Elapsed Time: 107 milliseconds, average: 125.132
214: Elapsed Time: 132 milliseconds, average: 125.164
215: Elapsed Time: 122 milliseconds, average: 125.149

KEYBOARD CLOCK PROGRAM V | 1 | 64 | Program Running: Free Run | Effective System/3 clock speed: 657.895kHz (clock) | 0 | Trace Off

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:

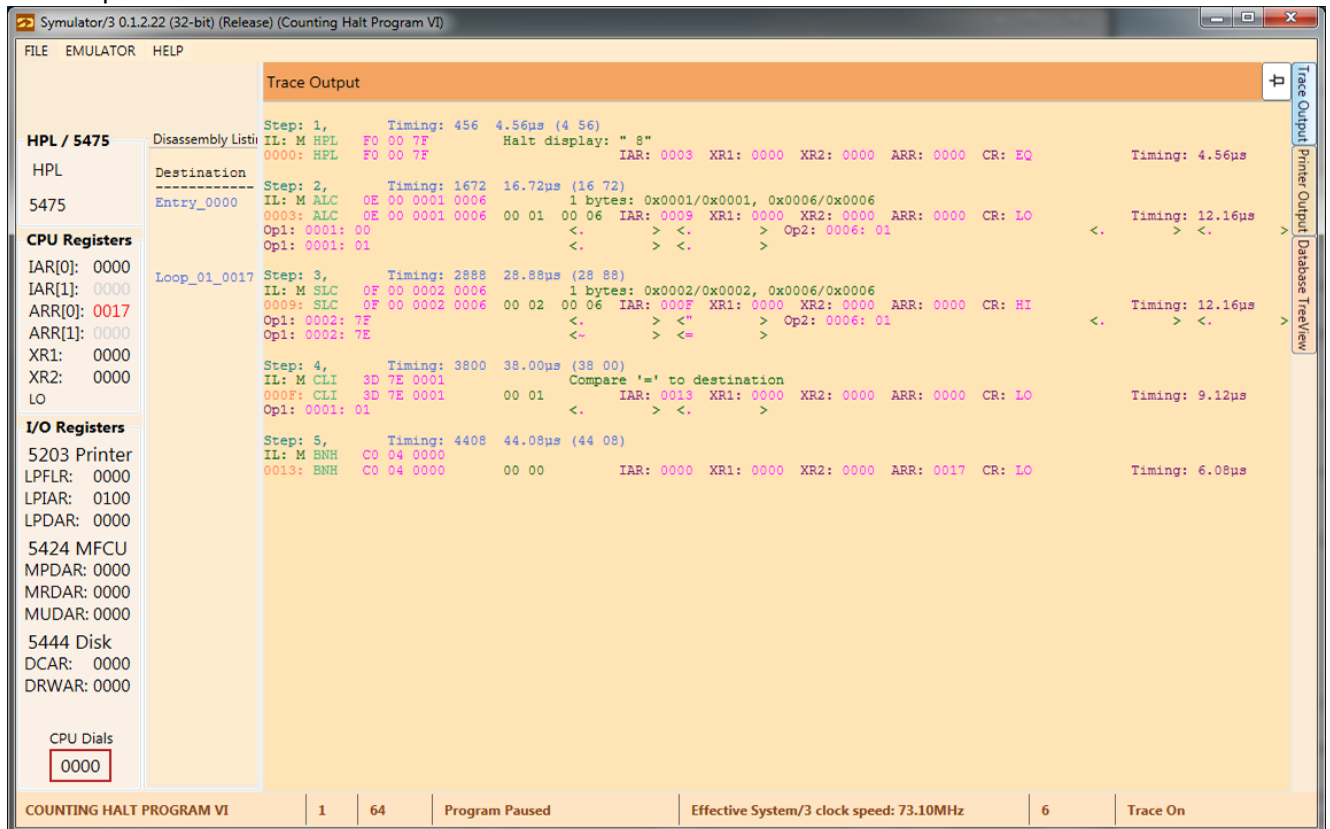




The columns in the Disassembly Listing window are:

- 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.

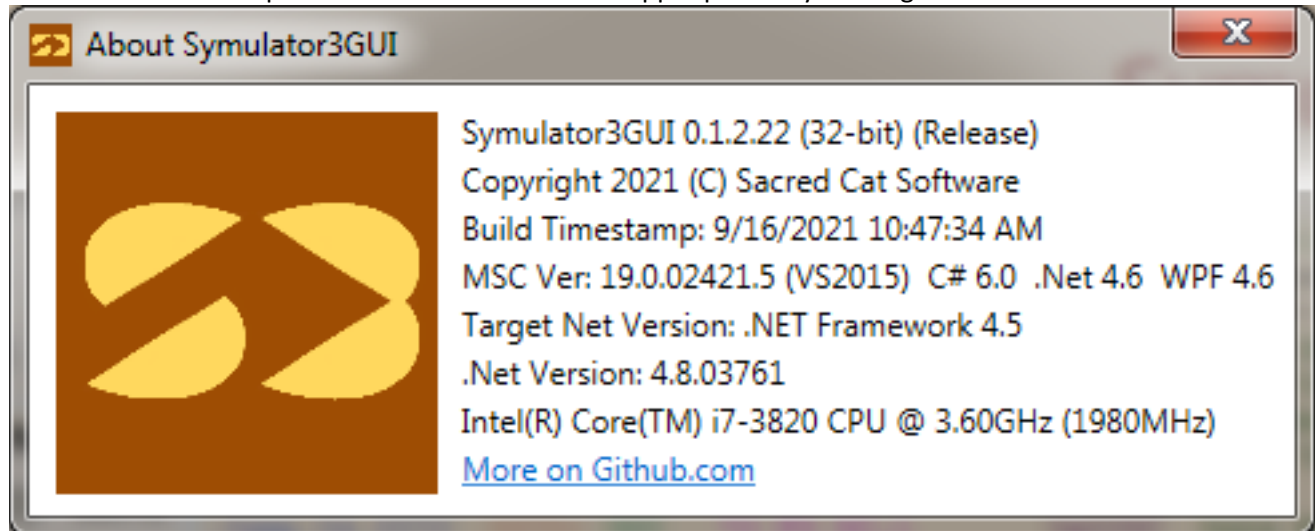


The Trace Output window shows 4 line formats for each instruction executed:

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](#).