Running HP 2000 Time-Shared BASIC on SIMH

J. David Bryan, 9-Dec-2018

The HP 2100 simulator for the 21xx and 1000 series of machines supports execution of the HP 2000 family of Time-Shared BASIC operating systems. The 2000A and 2000E products run on a single CPU, whereas the 2000B, 2000C, 2000F, and 2000 Access products use a dual-CPU configuration. The former present no special problem for execution on SIMH, but the latter require some considerations to run reliably.

The Dual-CPU Hardware Implementation

The HP 2000B, C, F, and Access versions split the TSB operating system into two parts running on separate CPUs. The primary CPU, designated as the *System Processor*, runs the BASIC interpreter for up to concurrent 32 users. The secondary CPU, designated as the *I/O Processor*, handles I/O through the terminal multiplexers. An HP 12875A Processor Interconnect kit provides communication between the SP and the IOP. The kit consists of two bidirectional 16-bit parallel interfaces installed in each CPU. One interface in each CPU is designated as the output interface, and the other is designated as the input interface. Cables cross-connect the interfaces between the CPUs.

To start these TSB systems, a configured IOP program is loaded into the I/O Processor, and the CPU is run. Then a configured SP program is loaded into the System Processor and is run. The system operator interacts briefly with the SP startup routine, which ends with setting the current date and time. The SP and IOP communicate across the interprocessor link, and the system is then ready to accept user logins.

The essential aspect of system startup is that the IOP is running before the SP attempts to communicate with it. If the IOP is not running or is otherwise non-responsive, the SP startup routine halts with the ***MUST RELOAD IOP, LOAD ABORTED*** error message.

Simulation of the Dual-CPU Configuration

SIMH supports execution of the dual-CPU TSB versions. Two SIMH instances are used—one to run the SP software, and the other to run the IOP software. The **IPL** device simulates the Processor Interconnect kit, with network connections between the two instances serving as the interconnecting cables. SIMH also provides the special IOP firmware that is required with the 2000 Access version.

Starting the TSB system under simulation has the same requirement as in hardware: the IOP instance must be loaded and running before the SP instance attempts to communicate with it. In addition, the use of networking to simulate the CPU interconnection imposes an additional requirement: the network link between the SIMH instances must be established before either the IOP or SP begins program execution. These requirements complicate the use of automated simulator startup command files to bring up a TSB system.

Starting a Time-Shared BASIC System on SIMH

To start a dual-CPU TSB system successfully on SIMH, the following sequence of operations must be observed:

- The SP and IOP SIMH instances must be started, and their I/O devices must be configured to match the requirements of the SP and IOP software.
- The network connections between the IPL input device of one instance and the IPL output device of the other instance must be established.
- The IOP software must be loaded into the IOP instance, and the CPU must be run.
- The IOP software must complete its initialization work and begin monitoring the interprocessor link for communication requests.
- The SP software must be loaded into the SP instance, and the CPU must be run.

In hardware, the first two items above are accomplished when system power is applied. The remaining three items are guaranteed by requiring the system operator to load and start the IOP before starting the SP. The physical time required for the operator to attend to the SP guarantees that the IOP initialization routine has completed before SP communication occurs.

Manual configuration of the SP and IOP instances for each execution is tedious and error-prone. A better approach is to have separate SP and IOP command files that configure the CPU and I/O device simulations, set up the network connections for the IPL devices, and load the corresponding software. With such files, a direct simulation of the preceding sequence would require the user to:

- start an instance to process the SP command file, then
- start an instance to process the IOP command file, then
- start IOP program execution, and then
- start SP program execution.

The delay inherent in manually starting SP program execution guarantees that the IOP initialization routine has completed before the SP attempts to communicate.

Starting the system in this way is inconvenient. It is preferable to run a single command file that handles all of the subsidiary actions, including starting the SP and IOP programs. The problem with such an approach is guaranteeing that the startup sequence requirements are met.

Problems with Automatic Startup Command Files

The Processor Interconnect kit interfaces in the two instances are connected by attaching the IPLI and IPLO devices to user-specified network ports. One instance is

configured to listen on the ports, and the other is configured to connect to those ports. For example, the SP command file might contain:

```
attach -l ipli 4000
attach -l iplo 4001
```

...while the IOP command file contains:

```
attach -c iplo 4000
attach -c ipli 4001
```

For the connections to be established, the first instance must execute the listening commands before the second instance executes the connecting commands. If this does not occur, the connecting commands will fail, and attempts by the SP and IOP to communicate will cause network errors that result in simulation stops. The ***attach –l*** command may also specify the ***–w*** (wait) switch, and this will cause the first instance to wait for a connection to the listening port from the second instance. Still, regardless of whether the switch is specified or not, the listening commands must be executed first.

Given SP and IOP command files as outlined above, two options for TSB startup requiring only one user action are possible. One option is to start each instance with its associated command file from a host operating system shell script. The other is to start one instance with its associated command file and have that instance start the second instance with its command file.

As an example of the first approach, a UNIX Bash startup script might contain:

```
xterm -e 'hp2100 iop.sim' &
hp2100 sp.sim
```

The equivalent Windows CMD script would be:

```
start hp2100 iop.sim
hp2100 sp.sim
```

One problem is that the two instances execute their command files asynchronously, so the requirement that the IOP starts first cannot be guaranteed. A second is that the order in which the network connections are attempted is not guaranteed.

For the second approach, the user starts the first SIMH instance manually, specifying the SP command file:

```
hp2100 sp.sim
```

...and at some point within the SP file, there is a SIMH spawn command:

```
! start hp2100 iop.sim
```

...that starts a second instance of the simulator to execute the IOP command file. This approach eliminates the need for the host system shell script and guarantees that commands in the SP file preceding the spawn of the IOP instance are executed before the IOP commands. For example, if the ***attach –l*** commands are issued by the SP prior to spawning the IOP instance whose command file contains the ***attach –c*** commands, then the network connections are guaranteed before the IOP program runs. However, this still does not guarantee that the IOP starts first or completes its initialization before the SP program attempts communication. If the host operating system blocks the IOP instance from executing due to higher priority contending processes, the SP command file will continue through SP program loading and execution, which will fail when IOP communication is attempted.

The problem is exacerbated with the advent of multi-core PCs. With a single-core host machine, the SP and IOP instances must execute alternately. When the IOP instance is started, it may receive enough CPU time to load and initialize before it is preempted and the SP resumes execution. Processor contention, either from the host OS itself or from other programs executing concurrently, must suspend both SIMH instances. This improves the probability that the IOP will complete before the SP.

With multi-core machines, the SP may execute either concurrently with the IOP or while the IOP is blocked by processor contention. Successful system startup then becomes much more sensitive to the timing of operations within the respective SIMH command files.

There is no general solution to these problems because there is no way to guarantee that the SP and IOP instances will not be preempted by the host operating system. In hardware, the IOP takes a deterministic time to execute its initialization code. In simulation, this time will vary from some minimum lower limit to essentially an unbounded upper limit, depending on host system load. The best approach is to increase the probability that the IOP completes before the SP and therefore that the system startup will succeed.

Improving Successful System Startup Probability

While the required sequence of startup operations cannot be guaranteed, the probability of following the proper sequence may be improved with these actions:

- The IPL attachment in the SP command file should be done as early as possible, preferably just after I/O configuration and before loading the SP program.

- The IPL attachment in the IOP command file should be done as late as possible, preferably after loading and just before running the IOP program.

- The ***–w*** switch to the ***attach –l*** command should be used to pause the SP command file until the IOP command file performs the ***attach –c*** command to establish the IPL network connections.

For the SP to wait for the network connection, the IOP instance must be started before the IPL attachment is done. If SP attachment is performed immediately after spawning

the IOP instance, and if the IOP attachment is performed late in the command file after I/O configuration and IOP program loading is complete, then it is more likely that the SP will be listening when the IOP attempts to connect. This meets the first two requirements for a successful startup.

If the SP attachment is specified with the ***–w*** switch, then the SP will pause until the IOP connects. If the IOP attachment is done after IOP program loading, and the program is run immediately thereafter, then it is more likely that it will complete initialization by the time the SP completes loading and begins running its program. This meets the remaining three startup requirements.

With these principles in mind, the SP command file might look like this:

```
set CPU 2100,32K,FP

set IPLI SC=10
set TTY  SC=12
set CLK  SC=13
...

set IPLI ENABLED
set PTR  DISABLED
set PTP  DISABLED
...

! start hp2100 iop.sim

attach -L  IPLI 4020
attach -LW IPLO 4021

attach -E DS0 system.7905.disc

deposit S 000000
boot DS0

assert T=102077
go
```

...where the ***start*** command above is used to spawn an asynchronous HP2100 instance. The IOP command file might contain:

```
set CPU 2100,32K,IOP

set CLK  SC=10
set MUX  SC=11
set IPLI SC=14
...

set IPLI ENABLED
set BACI DISABLED
set MPX  DISABLED
...

attach MUX 1054
```

```
attach –E PTR configured-iop.abin
boot PTR

assert T=102077
detach PTR

attach -C IPLO 4020
attach -C IPLI 4021

deposit P 000002
reset
go
```

Subject to unexpected preemption, the IOP is likely to complete its initialization before the SP completes loading and execution. The probability of success may be increased, if necessary, by adding a short ***sleep*** in the SP command file just before the final ***go*** that initiates SP program execution.

The Effect of Throttling on System Startup

HP 2000 Time-Shared BASIC does not have a traditional localized idle loop that is executed while the operating system is otherwise unoccupied. Consequently, the SIMH ***set cpu idle*** command does not idle the simulator when running TSB, and therefore the SP and IOP instances take 100% of their available host CPU times.

It is tempting to use ***set throttle*** commands to reduce the loads on the host CPU. However, because throttling periodically preempts the SIMH process, achieving a successful system startup in the presence of throttling is even more problematic than it is otherwise. The throttling parameters must be set empirically, and the resulting system will be more sensitive than usual to host loading.

If throttling is used, startup probability may be increased by enabling IOP throttling only after initialization is complete. This may be accomplished by running at full speed until the *.COM.* routine is entered at the end of IOP initialization and then enabling throttling at that point. For example, if the entry point map generated during IOP program configuration lists the *.COM.* entry point at address 42634, then the command file may be modified as follows:

```
deposit P 000002
reset
go until 42634

set throttle 50%
go
```

This allows the initialization code to complete before throttling reduces the host CPU time available to the simulator.

The Race Condition in 2000 Access

The 2000 Access version has a race condition that manifests itself by an apparently normal boot and operational system console but no ***PLEASE LOG IN*** response to terminals connected to the multiplexer. The frequency of occurrence is higher on multi-core host systems, where the SP and IOP instances may execute concurrently.

The cause is this code in the SP disc loader (source files S2883, S5ISS, S7900, and S7905 on the Access source tape):

```
LDA SDVTR     REQUEST
JSB IOPMA,I     DEVICE TABLE
[...]
STC DMAHS,C   TURN ON DMA
SFS DMAHS     WAIT FOR
JMP *-1         DEVICE TABLE
STC CH2,C     SET CORRECT
CLC CH2         FLAG DIRECTION
```

The STC/CLC at the end normally would cause a second "request device table" command to be recognized by the IOP, except that the IOP DMA setup routine *DMAXF* (in source file SD61) has specified an end-of-block CLC that holds off the IPL interrupt, and the completion interrupt routine *DMCMP* ends with a STC,C that clears the IPL flag.

In hardware, the two CPUs are essentially interlocked by the DMA transfer, and the DMA completion interrupts occur almost simultaneously. Therefore, the STC/CLC in the SP is guaranteed to occur before the STC,C in the IOP. Under simulation, and especially on multi-core hosts, that guarantee does not hold. If the STC/CLC occurs after the STC,C, then the IOP starts a second device table DMA transfer, which the SP is not expecting. Consequently, the IOP never processes the subsequent "start timesharing" command, and the multiplexer does not respond to user login requests.

The simulator employs a workaround that decreases the incidence of the problem: DMA output completion interrupts are delayed to allow the other SIMH instance a chance to process its own DMA input completion first. This improves the race condition by delaying the IOP until the SP has a chance to receive the last word, recognize its own DMA input completion, drop out of the SFS loop, and execute the STC/CLC. The delay is initially set to one millisecond but is exposed via a hidden IPLI register, *EDTDELAY*, that allows the user to lengthen the delay if necessary.

The condition is only improved but not solved because ***sleep***ing the IOP does not guarantee that the SP will actually execute. It is possible that a higher-priority host process will preempt the SP, and that at the sleep expiration, the SP still has not executed the STC/CLC. Still, in testing, the incidence dropped dramatically, so the problem is much less intrusive.

Summary

The HP 2000B, C, F, and Access dual-CPU Time-Shared BASIC operating systems will run on the HP2100 simulator, but several internal aspects must be considered for successful startup and operation. These arise from the differences in behavior of dedicated hardware versus two simulator instances running on a multi-core host machine. Arbitrary host OS preemption of the separate SP and IOP instances means that the deterministic behavior of the original HP hardware cannot be guaranteed. However, following certain guidelines during system startup and operation will improve the probability of successful system execution.