DJSX 虚拟机使用指南

Version 1.3

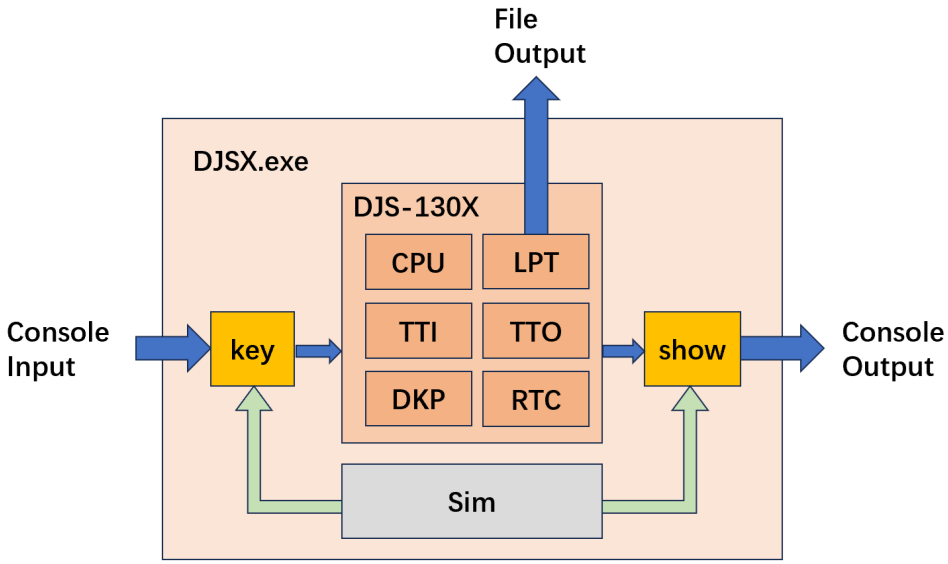
2024/9/27 By H.J.Xie

1. 简介

DJSX是DJS-130X的开源虚拟机，可运行Data General公司编写的Unmapped RDOS系统。软件代码使用C++编写。如果想快速体验，键入如下代码即可启动RDOS：

```
do default.set
```

虚拟机的结构如下：



虚拟机程序主要由4部分组成，即**DJS-130X**、**Sim**、**key**、**show**。下面说明四部分分别的作用：

DJS-130X: 主要的仿真对象，其中模拟了支持DJS-130基本指令集和乘除法扩展的CPU，配备了64KB内存（机器最大能支持的内存容量），此外模拟了常用且必要的五种外部设备：

- **LPT**: 行式打印机（Line Printer）。打印设备，通过绑定指定文件，可以将行式打印机的输出储存在指定文件中。
- **RTC**: 实时钟（Real-time Clock）。实时时钟。理论上提供四种实时频率，但在仿真模拟时，实时钟与指令执行条数挂钩。比如执行完1000条指令触发一次，因此并不是与现实时间相同的频率。
- **TTI**: 电传打字机输入（Teletype Input）。输入设备，接收从key模块传来的输入。
- **TTO**: 电传打字机输出（Teletype Output）。输出设备，将CPU发送的信号传递给show模块。
- **DKP**: 磁盘管理（Disk Manager）。连接4个虚拟磁盘（dp0~dp3）。系统盘应装载在dp0作为启动盘（类似C盘）。

Sim: 模拟器控制模块，按下指定的按键换出Sim界面（默认为F1键）。当出现`Sim>`的提示时，意味着此时DJS-130X的仿真暂停，并可以输入Sim指令执行各种操作。

key: 输入转换模块。由于DJS-130X上运行的RDOS系统接收的键盘输入ascii码与现在默认的键盘输入码不同，因此需要转换，根据设定好的转换规则，将控制台输入进来的按键码转换为另一个或一组按键码。同时key模块也负责检测按键是否为指定的唤起Sim界面的按键。

show: 输出转换模块。将TTO的输出按照设定的转换规则进行转换。比如RDOS系统中，会将`<EM>`（十进制：25）作为退格的控制符，而系统控制台接收`<BS>`（十进制：8）作为退格。因此我们可以为show模块加入`<EM>-><BS>`的转换规则，从而在控制台上正确显示字符。

2. Sim指令

2.1 load

load指令用于把指定文件加载进指定虚拟磁盘。DJSX一共拥有4个虚拟磁盘（dp0~dp3），每个虚拟磁盘占用8MB内存，实际可用空间6.2MB，模拟了六片磁盘组。指令示例如下：

```
load dp0 urdos.bin
```

所示指令为把urdos.bin文件读入到dp0（也就是第0号磁盘）中。

2.2 boot

boot指令用于打开DJS-130X虚拟机。该指令会将64Byte的引导程序装入内存单元000~037号（八进制），并让CPU从停机状态转入执行状态。指令使用示例：

```
boot
```

2.3 halt

halt指令用于使CPU停机，并清空目前的CPU状态。注意并不会退出DJSX.exe程序，因此你可以再次使用boot指令重启DJS-130X。使用示例：

```
halt
```

2.4 run

退出Sim界面，继续DJS-130X的仿真。使用示例：

```
run
```

2.5 save

save指令用于把指定虚拟磁盘的内容保存到指定文件。指令示例如下：

```
save dp0 mysave.bin
```

所示指令把dp0磁盘的内容保存到mysave.bin文件中。注意：虚拟机所读写的虚拟磁盘是储存在内存中的，如果想保存结果，必须用save指令保存到指定文件。（也就是说，如果你只想体验一下DJS-130X，就算你把系统搞崩了，只要不用save指令覆盖原来的磁盘文件，则不会有什么影响）。另外，如果要保存磁盘，注意正确退出，如果只是用halt指令停机，然后save，那么当下次你再次load这个文件进入磁盘，运行RDOS系统，它会显示`Partition in use - type C to continue`，这意味着你没有正常退出。

2.6 key

key指令用于为key模块设定转换规则，指令使用示例如下：

```
key 97 65
```

所示指令将按键码97映射到65，即从小写`<a>`按键映射到`<A>`，则每次按下小写a键，TTI外设收到的将是大写A键信号。按键码的范围为0~65535，一般可见字符的按键即为对应的ascii码，控制按键则一般大于127，如果不确定按键对应的按键码，可以运行key_listen.exe程序，其会显示输入的按键键码。

key指令设定的规则是将一个按键映射到一个或一组按键，所以可以使用如下指令：

```
key 97 65,65,66
```

一组按键之间用逗号分隔（注意中间不要有空格）。所示指令的效果是当按下小写`<a>`键，相当于输入了三个按键：`<A><A><B>`

此外，key指令还可以指定控制键（即呼出Sim界面的按键），默认为`<F1>`键，若想将控制键映射到`<a>`键，则可以使用如下指令：

```
key 97 cmd
```

注意映射控制键只能单独映射，不能映射为一组，比如`key 97 65,cmd,66`将产生错误的行为。

如果有键码大于127且没有为其设置映射，将忽略此键码。

2.7 show

key指令用于为show模块设定转换规则，指令使用示例如下：

```
show 65 97
```

所示指令将输出的`a`映射到`A`，则TTO输出的大写A字符显示到控制台上都将变为小写a字符。字符的范围为0~255。同理，show指令也可以将一个字符映射为一组字符，所以可以使用如下指令：

```
show 65 97,98,97
```

则TTO每输出一个大写A字符，控制台都将显示为`aba`。

2.8 debug

debug指令用于开启/关闭debug模式。打开debug模式后，将显示CPU在运行时的PC、AC0~AC3的值，以及目前执行的指令以及反汇编。指令示例如下：

```
debug on
```

所示将开启debug模式，关闭则使用`debug off`即可

2.9 reset

reset指令将清空key和show模块设定的规则，但是控制键的映射不会被清除（因为必须要有一个控制键，且要保证用户可以按下该控制键）。此外会关闭debug模式。指令示例如下：

```
reset
```

2.10 input

input命令将指定的文件按照ascii格式输入到虚拟机中。该指令为输入大量的文本内容提供了便利。指令示例如下：

```
input test.txt
```

所示指令将test.txt内容自动输入到虚拟机中（可以认为输入从键盘变成了ascii文本，同样需要经过key模块的转换）。

2.11 output

output命令将LPT的输出绑定到指定文件。初始时LPT并没有绑定，此时LPT的输出会被丢弃。注意：绑定到的文件会先清空，如果内有主要内容请转存到其他文件。指令示例如下：

```
output lpt.txt
```

所示指令将LPT的输出定向到到 lpt.txt 中。

2.12 refresh

refresh命令将LPT的输出保存到绑定的文件。如果不使用refresh指令，则可能不会保存打印的内容。建议在使用LPT输出后使用该指令刷新。指令示例如下：

```
refresh
```

2.13 do

do指令将执行指定文件里的Sim指令。指令示例如下：

```
do default.set
```

所示指令会打开default.set文件并执行其中的指令

2.14 rem

rem为行注释。指令示例如下：

```
rem "This is a piece of comment."
```

该条指令没有任何效果。且Sim不会显示文件中的rem指令

2.15 end

end指令用于文件结尾。表示文件的结束。如果一个文件中只有Sim指令，则end指令不是必须的。指令示例如下：

```
end
```

2.16 exit

exit指令使虚拟机程序退出。指令示例如下：

```
exit
```

3. 一般流程

load --> key --> show --> boot --> run --> [进行软件操作] --> [操作系统退出] --> halt --> (save) --> exit

4. 控制台控制字符

控制台可以接收两个特殊的控制字符序列：`<SI>`（十进制：15）与`<DLE>`（十进制：16）。

- `<SI><fore><back>` 指定控制台文本输出颜色。其中`<fore>`为想指定的前景色（范围0~15），`<back>`为想指定的背景色（范围0~15)。
- `<DLE><_x_><_y_>` 指定控制台光标位置。其中`<_x_>`为想指定的X坐标（范围0~79），`<_y_>`为想指定的Y坐标（范围0~29)。

注意：控制台接收到这两个控制字符后，后续接收的两个参数字符不会经过show模块的转换。

5. 编译选项

在导入"djs130_sim.hpp"前，可以通过 `#define` 对编译的虚拟机程序进行定制。有如下四个选项：

```
//define DJS130_LOWSPEED before include "djs130_sim.hpp" to limit speed
#define DJS130_LOWSPEED
//define DJS130_SHOWINT before include "djs130_sim.hpp" to print ascii char in
int number
#define DJS130_SHOWINT
```

- `DJS130_LOWSPEED`

  定义该选项将限制虚拟机的速度，将仿真频率理论上降低到 20MHz，从而降低CPU的占用率。（在测试中CPU占用率低于2%）

- `DJS130_SHOWINT`

  定义该选项将在打出的每个字符后附带打出的ascii字符对应的值（十进制）。

在源码中提供了两个预设的版本：

DJS = 高速版本

DJS_ls = 低速版本

6. 跨平台

DJSX 尽可能地使用 C++ 标准库，除了使用 djsx_sim.hpp 库外，没有使用任何非标准函数。因此对于移植到其他平台，只需将djsx_sim.hpp重写即可。其中用到的非标准函数功能有：检测是否按键、获取键码、设置控制台字体颜色、设置控制台光标位置。