# Controlling Fischertechnik with Java

Axel T. Schreiner
Department of Computer Science
Rochester Institute of Technology
http://www.cs.rit.edu/~ats/talks/ft/ [pdf] [code] [code.zip]

Fischertechnik offered a parallel interface in 1984 and a serial interface in 1997. The interfaces control four digital outputs and eight digital and two analog inputs. There is an extension box which doubles the number of digital connections. By now another company provides a USB interface.

This talk describes Java classes to access the serial interface using the Java COMM API and a driver for Windows to access the parallel interface.

## Contents



## Links

| | |
|---|---|
| Fischertechnik | |
| German lecture notes | 2000 |
| Java COMM API | version 2 |

## Bytecodes

The serial interface has a byte protocol:

| | |
|---|---|
| `c1 mm` | set motors, |
| `tt` | read digital inputs |
| `c5 mm` | set motors, |
| `tt x1 x2` | read digital inputs and EX |
| `c9 mm` | set motors, |
| `tt y1 y2` | read digital inputs and EY |

The Java Communications API `javax.comm` can be used to access serial ports. Sun provides an implementation for Windows and Solaris and there is an implementation for Linux.

## ft.Interface

`ft.Interface` contains a thread that tries to keep an up-to-date copy of the interface state by polling over `javax.comm`.

`ft.properties` has to be in the current directory. It contains lots of properties to configure `ft.Interface`, e.g., it decides if the analog inputs are monitored.

```
C> cd ft
C> java -classpath .. ft.Interface COM1 100
8 true 1
100 in 20050 msec, 4 Hz
```

The speed is not overwhelming...

## javax.comm

| | |
|---|---|
| `CommPortIdentifier.getPortIdentifiers()` | returns `Enumeration`. |
| `portId.getName()` | returns names like `COM1` or `/dev/ttya`. |
| `portId.getPortType()` | returns `PORT_SERIAL` or `PORT_PARALLEL`. |
| `portId.open(...)` | returns `CommPort` (either `SerialPort` or `ParallelPort`) for exclusive use. |
| `port.getInputStream()` `port.getOutputStream()` | return byte streams. |
| `port.setSerialPortParams(...)` | and many others manipulate communication configuration. |

Only a single driver can be entered in the file `javax.comm.properties`.

## ft.comm

ft.comm implements a very rudimentary serial driver for Windows.

**ft.comm.Driver implements javax.comm.CommDriver**

initialize() is called by CommPortIdentifier and loads from ft.properties among other things names like COM1 which are used as port names and must be usable to get to the device.

getCommPort() is later called by CommPortIdentifier and has to deliver a port object for a name.

**ft.comm.SerialPort extends javax.comm.SerialPort**

SerialPort() gets the port name and creates the streams which getInputStream() and getOutputStream() deliver.

All other methods are blanked out — as far as possible. The speed improvement probably results from not implementing listeners.

**ft.comm.SerialInputStream extends java.io.InputStream**

SerialInputStream() gets the port name and connects to a Windows handle.

**ft.comm.SerialOutputStream extends java.io.OutputStream**

SerialOutputStream() gets the port name and connects to a Windows handle.

**ft.comm.dll**

*javah* creates the headers for the native methods which use Windows functions. The library is loaded by ft.comm.Driver at first access.

## Speed

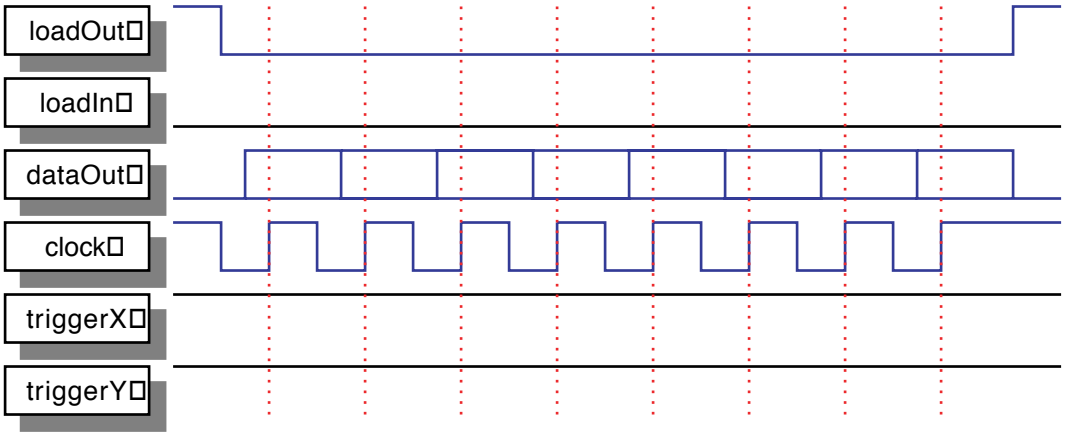
```
C> cd ft\comm
C> nmake -f nmakefile -t install
C> nmake -f nmakefile install

C> cd ft
C> java -classpath .. ft.Interface COM1 100
```

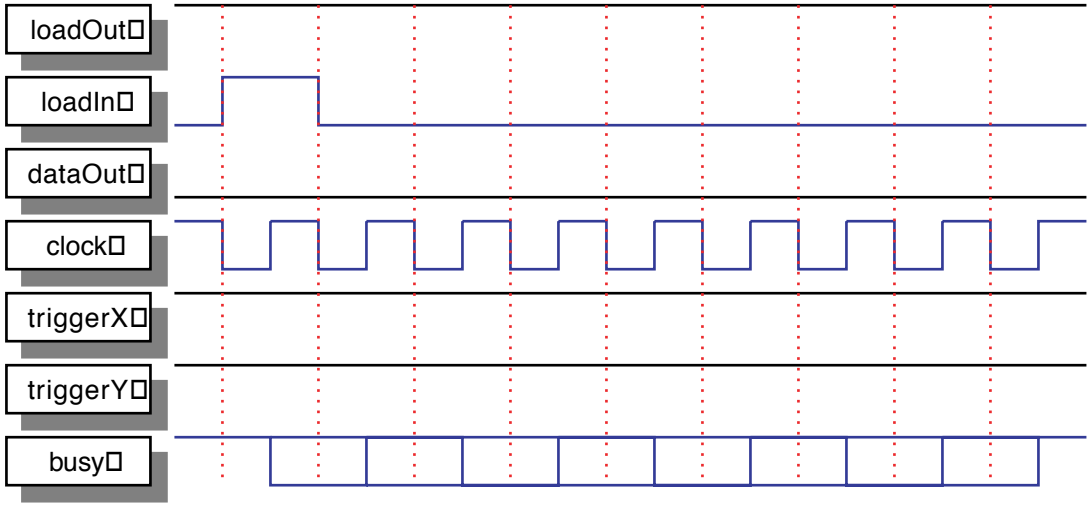
The speed is now about 130 Hz.

## Parallel Interface

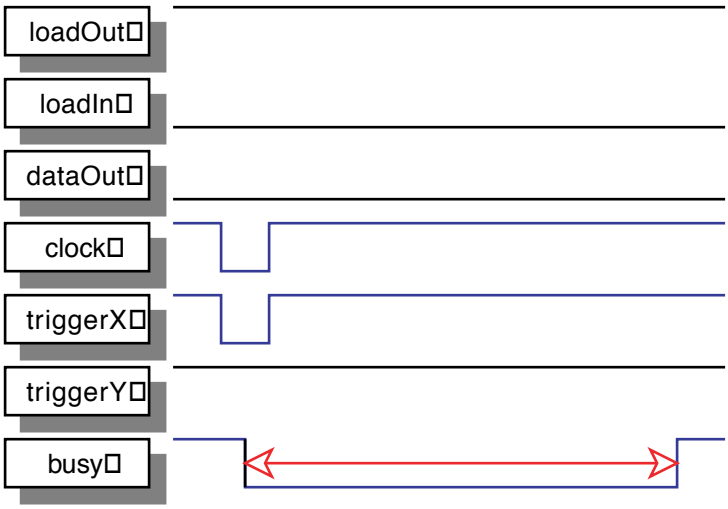
The parallel interface originally had many outgoing and very few incoming wires. Fischertechnik defined a way to transmit serial bits. The different wires can be influenced over time by writing the hardware port of the interface.




Input is read from the `busy` wire while the `clock` wire is toggled:



Analog inputs must be timed:



## Parallel Driver

The serial byte protocol can be implemented by a fake serial driver which accesses the parallel port:

```
C> cd ft
C> java -classpath .. ft.Interface 3bc 10000
```

The speed can be as high as 4500 Hz.

**`ft.comm.ParallelOutputStream extends java.io.OutputStream`**

`ParallelOutputStream()` is called first with the port name and records an index to access the device.

**`ft.comm.ParallelInputStream extends java.io.InputStream`**

`ParallelInputStream()` is called first with the port name and records an index to access the device.

**`parallel.c`**

The native methods are implemented in C as part of `ft.comm.dll`.

`write` has to send two bytes and calls `ftOutput()` with the port address and the second byte. This function toggles the data lines.

`read` can receive one or three bytes, depending on the last write. `ftInput()` and `ftAnalog()` toggle the data lines and watch the `busy` line.

## Performance

| Computer | System | Comm API | Interface | Speed |
|---|---|---|---|---|
| Toshiba 430CDT | Windows 98 SE | Sun | serial | 4 Hz |
| Pentium 120 MHz | JDK 1.2.2_005 | `ft.comm` | serial | 97 Hz |
| | HotSpot 2.0rc2 | | parallel, idle 1 | 298 Hz |
| | Linux 2.2.13 | *rxtx-1.3-13* | serial | 99 Hz |
| | JDK 1.2.2 (Sun) | | | |
| Sony PCG-F190 | Windows 98 SE | Sun | serial | 5 Hz |
| Pentium II 366 MHz | JDK 1.3.0 | `ft.comm` | serial | 130 Hz |
| | HotSpot 2.0rc2 | | parallel, idle 10 | 4500 Hz |

# Controllers and Views

`ft.Interface` can be viewed as a model encapsulating the state of a Fischertechnik interface. The package `ft` contains numerous other classes which can be used as observers and building blocks for graphical interfaces.

```
C> java ft.Diagnose COM1
$ java ft.Diagnose /dev/ttyS0
```