MODRATEC

A TEXT-BASED CODING SCHEME TO DEFINE INTERLOCKING

Version 1.1 - 20 November 2009

Introduction

MODRATEC has developed a textual coding language as a means of defining mechanical interlocking as applied to railway interlocking machines. This is preparatory to the release of SigScribe10 and will be applicable to SigScribe4 as at version 1.1.11.

It is envisaged that such coding be written in a plain text file and identified using the extension "itf" (itf = interlocking table file). Such files should be created using a text editor rather than a word processor. The default text editor in Windows is "Notepad". The default text editor in Macintosh systems is "TextEdit".

Structure

An "itf" file must contain **block**s of data separated by whitespace which may also surround line numbers.

Whitespace includes the space character itself, the line break character, the carriage return character, and the tab character. Thus blocks of data may be separated on one line by one or more space characters, or they may be placed on separate lines. These two representations of three blocks of data are equivalent.

```
1N:2N,3R,4R,5N 2N:1N,3R,4N,6N    3N:4B

1N:2N,3R,4R,5N
2N:1N,3R,4N,6N
3N:4B
```

By implication, a block of data must not itself contain any whitespace characters.

For the convenience of the author of an "itf" file, it is permissible to include a number (one or more numerals) within the whitespace separating blocks of data. Thus the following representations are also equivalent to those above.

```
1 1N:2N,3R,4R,5N
2 2N:1N,3R,4N,6N
3 3N:4B

1N:2N,3R,4R,5N     100
2N:1N,3R,4N,6N     101
3N:4B              102
```

Note that when using trailing line numbers, there must be at least one whitespace character following the last number.

When an "itf" file is interpreted, line numbers are ignored.

Note also that error reporting during the interpretation of an "itf" file may refer to line numbers. These refer to the logical blocks of data rather than any authored line numbers.

Comments

As at version 1.2.0 of SigScribe4, comments may also be included anywhere within an "itf" file. Comments must be surrounded by comment boundaries. /* is the start comment boundary and */ is the end comment boundary. See example below. When an "itf" file is interpreted, comments are ignored.

```
3N:4B /* FPL 3 locks points 4 both ways */
```

Permitted Characters

The following characters are the only non-whitespace characters permitted in an "itf" file unless enclosed within comment boundaries.

Numerals (0~9) ⇔ line numbers or lever/tappet numbers.
B ⇔ both ways.
N ⇔ normal.
R ⇔ reversed.
, ⇔ AND.
| ⇔ OR.
: ⇔ while.
; ⇔ while (interchangeable with ":").
() ⇔ enclose IF conditions.

Data Blocks

The **first data block** must contain an integer number being the number of levers/tappets associated with the interlocking being specified. A line number cannot be added before this first data block, otherwise the line number will be interpreted as the number of levers.

Each **subsequent data block** must contain just one "while" character, that is, either a ":" or ";". The ":" character will be used here, but it is totally interchangeable with ";".

There is a single data element to the left of ":". This is regarded as the "reference" element.

There are one or more elements to the right of ":". These are regarded as "driving" elements.

Both reference and driving elements contain a number followed by a single uppercase letter.

The number is that of the lever/tappet being locked.

For the **reference element**, the character has the following meaning.

- `N` - is free when normal.
- `R` - is free when reversed.
- `B` - is free both ways. *NOTE: this character **may not be used** in interlocking definitions, but may appear in a rationalised interlocking table.*

For a **driving element**, the character has the following meaning.

- `N` - is normal.
- `R` - is reversed.
- `B` - is either normal or reversed (both ways). *NOTE: this character **may be used freely** for driving elements.*

Simple "AND" Locking

Multiple driving elements in a list of simple locks are separated by "," characters which may be read as "and".

Here is an example of simple locking.

```
6
1 1N:2N,3R,4R,5N
2 2N:1N,3R,4N,6N
3 3N:4B
4 5N:4N,1N,2N
5 6N:4R,1N,2N
```

The first block says that the frame has 6 levers.

The second block, line 1, reads, lever 1 is free when normal (`1N`) while (`:`) lever 2 is normal (`2N`) and (`,`) lever 3 is reversed (`3R`) and (`,`) 4 is reversed (`4R`) and (`,`) 5 is normal (`5N`). Consequently, while lever 1 is reversed, it locks 2 normal, 3 reversed, 4 reversed, and 5 normal.

The fourth block, line 3, reads, lever 3 is free when normal while (`3N:`) 4 is either normal or reversed, i.e. both ways (`4B`). Consequently, while lever 1 is reversed, it locks 4 either normal or reversed.

"OR" Locking

OR locks are specified by using the "`|`" character instead of the "," as shown in the following example. *Note that OR and AND locks must not be specified in the same data block.*

```
1N:6R|7R|8R
```

This block reads, lever 1 is free when normal while 6 is reversed, or 7 is reversed, or 8 is reversed. Consequently, while lever 1 is reversed, it locks 6, 7 and 8 either normal or reversed.

Here is an example of an interlocking requiring OR locks.

```
18
 1  1N:6R|7R|8R
 2  6N:4R,18R
 3  7N:4R,16N,17N,18N
 4  8N:4R,5N,14N,15N,16R,17R,18N
 5  9N:4R,16N,17N,18N
 6 10N:4R,5N,14N,15N,16R,17R,18N
 7 11N:9R
 8 12N:10R
```

```
 9 13N:1R
10  9N:6N,7N,8N
11 10N:6N,7N,8N
```

Conditional "IF" Locking

Conditional locks are specified by enclosing IF elements within "()" characters as shown in the following examples.

```
2N:(3R,4N)6N

7N:(4R)3R,2N
```

The first example reads, lever 2 is free when normal while 6 is normal IF 3 is reversed and 4 is normal. Consequently, reversing 2 locks 6 normal but only if 3 is reversed and 4 is normal. If 3 is normal or 4 is reversed, there is no interlocking between 2 and 6.

The second reads, lever 7 is free when normal while 3 is reversed and 2 is normal IF 4 is reversed. Consequently, reversing 7 locks 3 reversed and 2 normal but only if 4 is reversed. If 4 is normal there is no interlocking between 7, 3 and 2.

Here is an example of an interlocking requiring IF locks.

```
7
 1 1N:4N,7N
 2 5N:3N,2N
 3 6N:4N,3R,2N
 4 7N:4B
 5 7N:(4N)1N
 6 7N:(4R)3R,2N
 7 4N:3R
 8 2N:3B
 9 2N:(3N)5N
10 2N:(3R)4B
11 2N:(3R,4N)6N
12 2N:(3R,4R)7N
```

Redundancy

There is no need to attempt to eliminate redundancy when writing an "itf" file. This will happen automatically when the file is interpreted. For example, `1N:2N` and `2N:1N` both describe exactly the same behaviour. It is safer and easier to include everything and to allow the interpreter to perform its task of rationalising a table.

PO Box 2205
Graceville Qld 4075
Australia
www.modratec.com