Model Railroad System

2.2.1

Generated by Doxygen 1.8.17

Chapter 1

Switch (Turnout) Abstract Types (Classes)

This folder contain a collection of Tcl code to implement switches (aka turnouts), using various actuator hardware. Included is OS detection and point position detection, along with code to operate switch motors.

1.1 Source Files

There are several Tcl source files in this directory. Each contains a SNIT **Abstract** data type (also known as a *Class*). This abstract data type encapsulates a single *switch* (turnout). All of these abstract data types a method named `occupiedp`, which returns a true or false result indicating whether or not the OS is occupied. Also included are a `pointsense` method which returns the state of the points and a `motor` method which operates the switch machine to move the points.

1.1.1 SR4 as actuator and pointsense with a MRD2U for OS detection

SR4_MRD2_Switch.tcl contains an abstract data type (SR4_MRD2_Switch) that implements switches using one half of a Azatrax SR4 for the actuator and pointsense and a Azatrax MRD2U Sensor for OS detection.

1.1.2 SR4 as actuator and pointsense with a Circuits4Tracks Quad Occupancy Detector with a SR4 (USB connected I/O board).

SR4_C4TSR4_Switch.tcl contains an abstract data type (SR4_C4TSR4_Switch) that implements switches using one half of a Azatrax SR4 for the actuator and 1/4 of a Circuits4Tracks Quad Occupancy Detector connected to another Azatrax SR4 for OS detection.

1.1.3 Chubb SMINI board as actuator and pointsense with a Circuits4Tracks Quad OD as OS sensor

C4TSMINI_Switch.tcl contains an abstract data type (C4TSMINI_Switch) that implements switches using a Chubb S↩
MINI board as actuator and pointsense with a Circuits4Tracks Quad OD as OS sensor.

1.1.4 CTI Yardmaster as actuator and Train Brain as pointsense with a Circuits4Tracks Quad OD as OS sensor

TB_Switch.tcl contains an abstract data type (TB_Switch) that implements switches using a CTI Yardmaster as actuator and a Train Brain for pointsense with a Circuits4Tracks Quad OD as OS sensor.

1.2 Common methods and functionality

All three types have a common structure. The constructors take the form:

```
typename objectname [optional options]
```

Eg:

```
SR4_MRD2_Switch switch1 -motorobj turnoutControl1 -motorhalf lower \
                      -pointsenseobj turnoutControl1 \
                      -pointsensehalf lower -plate SwitchPlate1 \
                      -ossensorsn 0200001234
```

There are a trio of common options, -previousblock, -nextmainblock, and -nextdivergentblock which are the names of the previous block in the forward direction and the name of the previous blocks in the reverse direction. Another trio of common options, -forwardsignalobj, -reversemainsignalobj, and -reversedivergentsignalobj are the names of signal objects. Also there is the option, -direction with sets the current operating direction for the block and can be forward or reverse. For blocks that only support traffic in one direction, use only the -previousblock and -forwardsignalobj options. The -direction defaults to forward. There are other object specific options that define how the sensor is accessed by the block object.

There are six common methods, four public and two private (the private methods should not be used by external code). The public methods are occupiedp, propagate, pointstate, and motor. The occupiedp method returns a true or false (logical) value that indicates whether the switch is occupied or not. The pointstate method returns the state if the points. The motor method activates the switch motor to move the points. The propagate method takes a signal aspect to 'propagate' to the previous block. The occupiedp method is typically called from the occupied command script associated with a piece of track work. The `pointstate` method is typically called from the state sense script associated with the switch track work. And the motor method is assocated with the normal and reverse scripts for the switch's switch plate on the CTC panel.

The two private methods, _entering and _exiting are used to implement special handling when entering or leaving a block. Presently, the _entering method sets the signal aspect and propagates signal aspects down to previous blocks and the _exiting method does nothing. These methods can be extended to add additional functionality, as needed.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 C4TSMINI_Switch Class Reference

Switch (turnout) operation using a Chubb SMINI board and a Circuits4Tracks Quad OD for OS detection.

Public Member Functions

- C4TSMINI_Switch (name,...)

  Constructor: initialize the switch object.
- occupiedp ()

  The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.
- pointstate ()

  The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.
- motor (route)

  The motor method sets the switch motor to align the points for the specificed route.
- propagate (aspect, from,...)

  Method used to propagate distant signal states back down the line.

Static Public Member Functions

- static validate (object)

  Type validating code Raises an error if object is not either the empty string or a SR4_C4TSR4_Switch type.

Protected Member Functions

- _entering ()

  Code to run when just entering the OS Sets the signal aspects and propagates signal state.
- _exiting ()

  Code to run when about to exit the OS.

Private Member Functions

- _settruedirection (option, value)

    A method to fake direction for frog facing switches.
- _gettruedirection (option)

    A method to fake direction for frog facing switches.

Private Attributes

- node

    SMINI node object.
- isoccupied

    Saved occupation state.

Static Private Attributes

- static _motorbits

    Motor bit values.
- static _pointsense

    Point sense bit values.
- static _routes

    Route check validation object.

3.1.1 Detailed Description

Switch (turnout) operation using a Chubb SMINI board and a Circuits4Tracks Quad OD for OS detection.

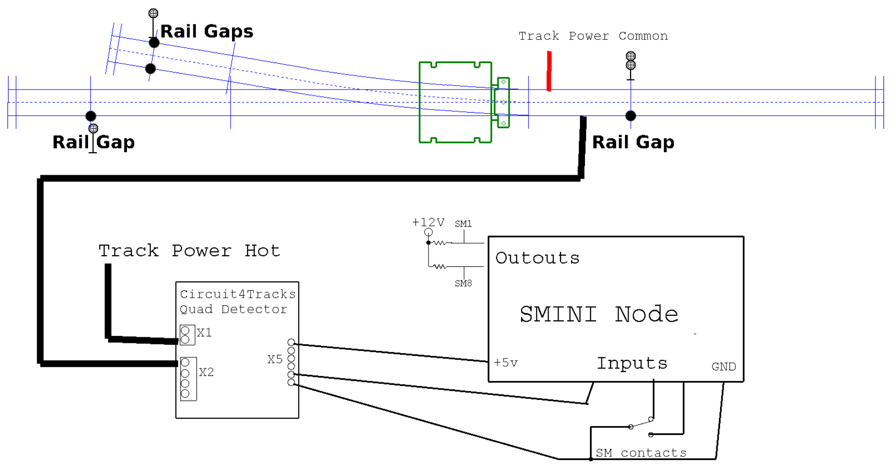

Figure 3.1 Switch controlled by a Chubb SMINI board with a Circuits4Tracks OS detection

Above is a typical switch (turnout) using an Chubb SMINI board to control a Circuitron Tortoise Switch Machine and to sense the point position and a Circuits4Track quad occupancy detector to sense occupation of the switch.

Typical usage:

```
# Connect to the cmribus through a USB RS485 adapter at /dev/ttyUSB0
CmriSupport::CmriNode openport /dev/ttyUSB0
# SMINI board at address 0
CmriSupport::CmriNode SMINI0 -type SMINI -address 0
# Switch 1 is controled by bits 0 and 1 of output port 0
# Switch 1 points are sensed by bits 0 and 1 of input port 0
# Switch 1 OS is detected on bit 0 of input port 1
C4TSMINI_Switch switch1 -nodeobj SMINI0 -motorport 0 -motorbit 0 \
-pointsenseport 0 -pointsensebit 0 \
-plate SwitchPlate1 \
-ossensorport 1 -osbit 0
# Switch 2 is controled by bits 0 and 1 of output port 1
# Switch 2 points are sensed by bits 2 and 3 of input port 0
# Switch 2 OS is detected on bit 1 of input port 1
C4TSMINI_Switch switch2 -nodeobj SMINI0 -motorport 1 -motorbit 0 \
-pointsenseport 0 -pointsensebit 2 \
-plate SwitchPlate2 \
-ossensorport 1 -osbit 1
```

For the track work elements use "switchN occupiedp" for the track work elements' occupied script and use "switchN pointstate" for the track work elements' state script. For the switch plate use "switchN motor normal" for the normal script and "switchN motor reverse" for the reverse script.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Switch1
MainWindow ctcpanel invoke Switch2
MainWindow ctcpanel invoke SwitchPlate1
MainWindow ctcpanel invoke SwitchPlate2
update;# Update display
}
```

Author

Robert Heller <heller@deepsoft.com>

Definition at line 57 of file C4TSMINI_Switch.tcl.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 C4TSMINI_Switch()

```
C4TSMINI_Switch::C4TSMINI_Switch (
              name ,
               ... )
```

Constructor: initialize the switch object.

Create a low level sensor object and install it as a component. Install the switch's signals, motor, and point sense objects.

Parameters

name	Name of the switch object

Parameters

...	Options: • -nodeobj Cmri node object • -motorport Output port number for motor control. • -motorbit First (of two) motor control bits. • -pointsenseport Input port for point sense. • -pointsensebit First (of two) point sense bits. • -ossensorport Input port for OS sense. • -osbit This defines the input bit on the input port for OS sense. • -direction The current direction of travel. Forward always means entering at the point end. • -forwarddirection The *logial* forward direction. Set this to reverse for a frog facing switch. Default is forward and it is readonly and can only be set during creation. • -forwardsignalobj The signal object protecting the points. Presumed to be a two headed signal, with the upper head relating to the main (straight) route and the lower head relating to the divergent route. The upper head has three colors: red, yellow, and green. The lower head only two: red and green. • -reversemainsignalobj The signal object protecting the straight frog end. Presumed to be single headed (with number plate). • -reversedivergentsignalobj The signal object protecting the divergent frog end. Presumed to be single headed (with number plate). • -previousblock The block connected to the point end. • -nextmainblock The block connected to the straight frog end. • -nextdivergentblock The block connected to the divergent frog end. • -plate The name of the switch plate for this switch.

3.1.3 Member Function Documentation

3.1.3.1 _entering()

```
C4TSMINI_Switch::_entering ( )  [protected]
```

Code to run when just entering the OS Sets the signal aspects and propagates signal state.

3.1.3.2 _exiting()

```
C4TSMINI_Switch::_exiting ( )  [protected]
```

Code to run when about to exit the OS.

3.1.3.3 _gettruedirection()

```
C4TSMINI_Switch::_gettruedirection (
             option  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.

Returns

Either forward or reverse.

3.1.3.4 _settruedirection()

```
C4TSMINI_Switch::_settruedirection (
             option ,
             value  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.
value	Either forward or reverse.

3.1.3.5 motor()

```
C4TSMINI_Switch::motor (
             route  )
```

The motor method sets the switch motor to align the points for the specificed route.

Parameters

route	The desired route. A value of normal means align the points to the main (straight) route and a value of reverse means align the points to the divergent route.

3.1.3.6 occupiedp()

```
C4TSMINI_Switch::occupiedp ( )
```

The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.

Returns

Yes or no, indicating whether the OS is occupied.

3.1.3.7 pointstate()

```
C4TSMINI_Switch::pointstate ( )
```

The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.

If the state cannot be determined, a value of unknown is returned.

Returns

Normal or reverse, indicating the point state.

3.1.3.8 propagate()

```
C4TSMINI_Switch::propagate (
             aspect ,
             from ,
              ...  )
```

Method used to propagate distant signal states back down the line.

Parameters

aspect	The signal aspect that is being propagated.
from	The propagating block.
...	Options: • -direction The direction of the propagation.

3.1.3.9 validate()

```
static C4TSMINI_Switch::validate (
             object   )  [static]
```

Type validating code Raises an error if object is not either the empty string or a SR4_C4TSR4_Switch type.

Parameters

object	Some object.

3.1.4 Member Data Documentation

3.1.4.1 _motorbits

```
C4TSMINI_Switch::_motorbits  [static], [private]
```

Motor bit values.

Definition at line 86 of file C4TSMINI_Switch.tcl.

3.1.4.2 _pointsense

```
C4TSMINI_Switch::_pointsense  [static], [private]
```

Point sense bit values.

Definition at line 90 of file C4TSMINI_Switch.tcl.

3.1.4.3 _routes

`C4TSMINI_Switch::_routes [static], [private]`

Route check validation object.

Definition at line 165 of file C4TSMINI_Switch.tcl.

3.1.4.4 isoccupied

`C4TSMINI_Switch::isoccupied [private]`

Saved occupation state.

Definition at line 82 of file C4TSMINI_Switch.tcl.

3.1.4.5 node

`C4TSMINI_Switch::node [private]`

SMINI node object.

Definition at line 78 of file C4TSMINI_Switch.tcl.

3.2 SR4_C4TSR4_Switch Class Reference

Switch (turnout) operation using 1/2 of a SR4.

Public Member Functions

- SR4_C4TSR4_Switch (name,...)

  Constructor: initialize the switch object.
- occupiedp ()

  The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.
- pointstate ()

  The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.
- motor (route)

  The motor method sets the switch motor to align the points for the specificed route.
- propagate (aspect, from,...)

  Method used to propagate distant signal states back down the line.

Static Public Member Functions

- static validate (object)

    Type validating code Raises an error if object is not either the empty string or a SR4_C4TSR4_Switch type.

Protected Member Functions

- _entering ()

    Code to run when just entering the OS Sets the signal aspects and propagates signal state.

- _exiting ()

    Code to run when about to exit the OS.

Private Member Functions

- _settruedirection (option, value)

    A method to fake direction for frog facing switches.

- _gettruedirection (option)

    A method to fake direction for frog facing switches.

Private Attributes

- motor

    Motor device (SR4 outputs)

- pointsense

    Point sense device (SR4 inputs)

- ossensor

    SR4 object.

- isoccupied

    Saved occupation state.

Static Private Attributes

- static sensemap

    Sensor bit mapping to sensor functions.

- static _routes

    Route check validation object.

3.2.1 Detailed Description

Switch (turnout) operation using 1/2 of a SR4.

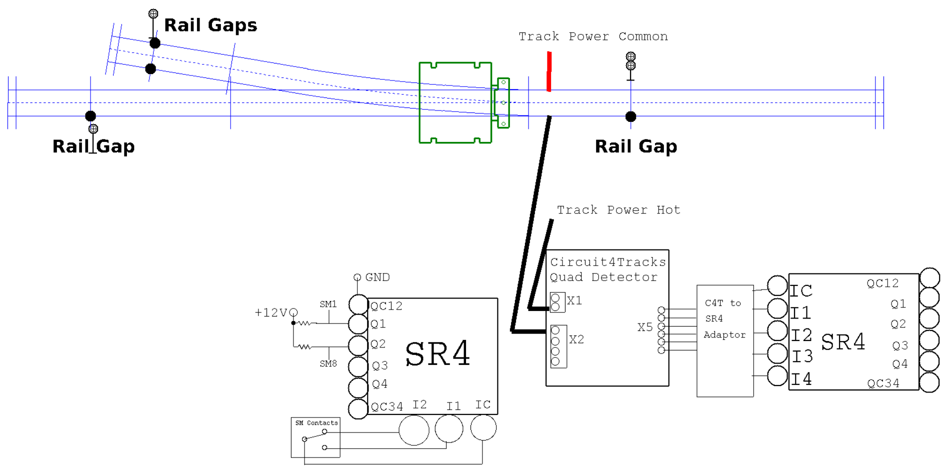

Figure 3.2 Switch controlled by a SR4 with a Circuits4Tracks OS detection using a second SR4

Above is a typical switch (turnout) using an Azatrax SR4 to control a Circuitron Tortoise Switch Machine and to sense the point position and a Circuits4Track quad occupancy detector and a second SR4 to sense occupation of the switch.

Typical usage:

```
SR4 turnoutControl1 \
-this [Azatrax_OpenDevice 0400001234 $::Azatrax_idSR4Product]
SR4 quadsense1 \
-this [Azatrax_OpenDevice 0400001235 $::Azatrax_idSR4Product]
# Disable inputs controlling outputs.
turnoutControl1 OutputRelayInputControl 0 0 0 0
quadsense1 OutputRelayInputControl 0 0 0 0
# Switch 1 is controlled and sensed by the lower 1/2 of turnoutControl1
SR4_C4TSR4_Switch switch1 -motorobj turnoutControl1 -motorhalf lower \
-pointsenseobj turnoutControl1 \
-pointsensehalf lower -plate SwitchPlate1 \
-ossensorobj quadsense1 -bit 0
# Switch2 is controlled and sensed by the upper 1/2 of turnoutControl1
SR4_C4TSR4_Switch switch2 -motorobj turnoutControl1 -motorhalf upper \
-pointsenseobj turnoutControl1 \
-pointsensehalf upper -plate SwitchPlate2 \
-ossensorobj quadsense1 -bit 1
```

For the track work elements use "switchN occupiedp" for the track work elements' occupied script and use "switchN pointstate" for the track work elements' state script. For the switch plate use "switchN motor normal" for the normal script and "switchN motor reverse" for the reverse script.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Switch1
MainWindow ctcpanel invoke Switch2
MainWindow ctcpanel invoke SwitchPlate1
MainWindow ctcpanel invoke SwitchPlate2
update;# Update display
}
```

Author

Robert Heller <heller@deepsoft.com>

Definition at line 57 of file SR4_C4TSR4_Switch.tcl.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 SR4_C4TSR4_Switch()

```
SR4_C4TSR4_Switch::SR4_C4TSR4_Switch (
            name ,
             ... )
```

Constructor: initialize the switch object.

Create a low level sensor object and install it as a component. Install the switch's signals, motor, and point sense objects.

Parameters

name	Name of the switch object
...	Options: • -motorobj Object (SR4) that controls the motor. • -motorhalf Which half: lower means Q1 and Q2, upper means Q3 and Q4. • -pointsenseobj Object (SR4) that senses the point state. • -pointsensehalf Which half: lower means I1 and I2, upper means I3 and I4. • -ossensorobj Object (SR4) that senses occupation (via the C4T) • -bit This defines the input bit on the SR4 for this block as an integer from 0 to 3, inclusive. This option is read-only and can only be set at creation time. The default is 0. • -direction The current direction of travel. Forward always means entering at the point end. • -forwarddirection The *logial* forward direction. Set this to reverse for a frog facing switch. Default is forward and it is readonly and can only be set during creation. • -forwardsignalobj The signal object protecting the points. Presumed to be a two headed signal, with the upper head relating to the main (straight) route and the lower head relating to the divergent route. The upper head has three colors: red, yellow, and green. The lower head only two: red and green. • -reversemainsignalobj The signal object protecting the straight frog end. Presumed to be single headed (with number plate). • -reversedivergentsignalobj The signal object protecting the divergent frog end. Presumed to be single headed (with number plate). • -previousblock The block connected to the point end. • -nextmainblock The block connected to the straight frog end. • -nextdivergentblock The block connected to the divergent frog end. • -plate The name of the switch plate for this switch.

3.2.3 Member Function Documentation

3.2.3.1 _entering()

```
SR4_C4TSR4_Switch::_entering ( )  [protected]
```

Code to run when just entering the OS Sets the signal aspects and propagates signal state.

3.2.3.2 _exiting()

```
SR4_C4TSR4_Switch::_exiting ( )  [protected]
```

Code to run when about to exit the OS.

3.2.3.3 _gettruedirection()

```
SR4_C4TSR4_Switch::_gettruedirection (
             option  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.

Returns

Either forward or reverse.

3.2.3.4 _settruedirection()

```
SR4_C4TSR4_Switch::_settruedirection (
             option ,
             value  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.
value	Either forward or reverse.

3.2.3.5 motor()

```
SR4_C4TSR4_Switch::motor (
             route  )
```

The motor method sets the switch motor to align the points for the specificed route.

Parameters

route	The desired route. A value of normal means align the points to the main (straight) route and a value of reverse means align the points to the divergent route.

3.2.3.6 occupiedp()

```
SR4_C4TSR4_Switch::occupiedp ( )
```

The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.

Returns

Yes or no, indicating whether the OS is occupied.

3.2.3.7 pointstate()

```
SR4_C4TSR4_Switch::pointstate ( )
```

The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.

If the state cannot be determined, a value of unknown is returned.

Returns

Normal or reverse, indicating the point state.

3.2.3.8 propagate()

```
SR4_C4TSR4_Switch::propagate (
              aspect ,
              from ,
               ... )
```

Method used to propagate distant signal states back down the line.

Parameters

aspect	The signal aspect that is being propagated.
from	The propagating block.
...	Options: • -direction The direction of the propagation.

3.2.3.9 validate()

```
static SR4_C4TSR4_Switch::validate (
              object ) [static]
```

Type validating code Raises an error if object is not either the empty string or a SR4_C4TSR4_Switch type.

Parameters

object	Some object.

3.2.4 Member Data Documentation

3.2.4.1 _routes

```
SR4_C4TSR4_Switch::_routes  [static], [private]
```

Route check validation object.

Definition at line 169 of file SR4_C4TSR4_Switch.tcl.

3.2.4.2 isoccupied

`SR4_C4TSR4_Switch::isoccupied [private]`

Saved occupation state.

Definition at line 89 of file SR4_C4TSR4_Switch.tcl.

3.2.4.3 motor

`SR4_C4TSR4_Switch::motor [private]`

Motor device (SR4 outputs)

Definition at line 77 of file SR4_C4TSR4_Switch.tcl.

3.2.4.4 ossensor

`SR4_C4TSR4_Switch::ossensor [private]`

SR4 object.

Definition at line 85 of file SR4_C4TSR4_Switch.tcl.

3.2.4.5 pointsense

`SR4_C4TSR4_Switch::pointsense [private]`

Point sense device (SR4 inputs)

Definition at line 81 of file SR4_C4TSR4_Switch.tcl.

3.2.4.6 sensemap

`SR4_C4TSR4_Switch::sensemap [static], [private]`

Sensor bit mapping to sensor functions.

Definition at line 93 of file SR4_C4TSR4_Switch.tcl.

3.3 SR4_MRD2_Switch Class Reference

Switch (turnout) operation using 1/2 of a SR4.

Public Member Functions

- SR4_MRD2_Switch (name,...)

  Constructor: initialize the switch object.
- occupiedp ()

  The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.
- pointstate ()

  The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.
- motor (route)

  The motor method sets the switch motor to align the points for the specificed route.
- propagate (aspect, from,...)

  Method used to propagate distant signal states back down the line.

Static Public Member Functions

- static validate (object)

  Type validating code Raises an error if object is not either the empty string or a SR4_MRD2_Switch type.

Protected Member Functions

- _entering ()

  Code to run when just entering the OS Sets the signal aspects and propagates signal state.
- _exiting ()

  Code to run when about to exit the OS.

Private Member Functions

- _settruedirection (option, value)

  A method to fake direction for frog facing switches.
- _gettruedirection (option)

  A method to fake direction for frog facing switches.

Private Attributes

- motor

  Motor device (SR4 outputs)
- pointsense

  Point sense device (SR4 inputs)
- ossensor

  Occupency sensor (MRD2)
- forwardsignal

  Signal at the points.
- reversemainsignal

  Signal at the straight frog end.
- reversedivergentsignal

  Signal at the divergent frog end.

Static Private Attributes

- static _routes

  Route check validation object.

3.3.1 Detailed Description

Switch (turnout) operation using 1/2 of a SR4.

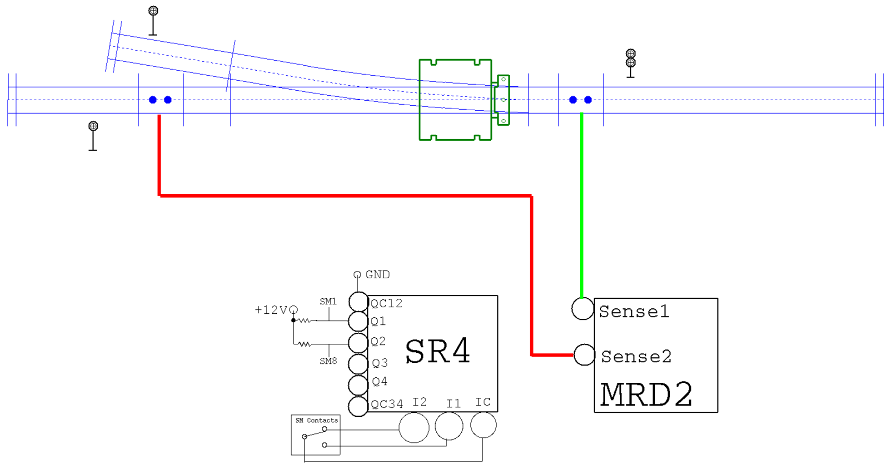

Figure 3.3 Switch controlled by a SR4 with MRD2 OS Detection

Above is a typical switch (turnout) using an Azatrax SR4 to control a Circuitron Tortoise Switch Machine and to sense the point position and an Azatrax MRD2 to sense occupation of the switch. (A high resolution PDF and a Xtrkcad layout file are included.)

Typical usage:
```
SR4 turnoutControl1 \
-this [Azatrax_OpenDevice 0400001234 $::Azatrax_idSR4Product]
# Disable inputs controlling outputs.
turnoutControl1 OutputRelayInputControl 0 0 0 0
# Switch 1 is controlled and sensed by the lower 1/2 of turnoutControl1
SR4_MRD2_Switch switch1 -motorobj turnoutControl1 -motorhalf lower \
-pointsenseobj turnoutControl1 \
-pointsensehalf lower -plate SwitchPlate1 \
-ossensorsn 0200001234
# Switch2 is controlled and sensed by the upper 1/2 of turnoutControl1
SR4_MRD2_Switch switch2 -motorobj turnoutControl1 -motorhalf upper \
-pointsenseobj turnoutControl1 \
-pointsensehalf upper -plate SwitchPlate2 \
-ossensorsn 0200001235
```

For the track work elements use "switchN occupiedp" for the track work elements' occupied script and use "switchN pointstate" for the track work elements' state script. For the switch plate use "switchN motor normal" for the normal script and "switchN motor reverse" for the reverse script.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Switch1
MainWindow ctcpanel invoke Switch2
MainWindow ctcpanel invoke SwitchPlate1
MainWindow ctcpanel invoke SwitchPlate2
update;# Update display
}
```

Author

Robert Heller <heller@deepsoft.com>

Definition at line 53 of file SR4_MRD2_Switch.tcl.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 SR4_MRD2_Switch()

```
SR4_MRD2_Switch::SR4_MRD2_Switch (
              name ,
               ...  )
```

Constructor: initialize the switch object.

Create a low level sensor object and install it as a component. Install the switch's signals, motor, and point sense objects.

Parameters

name	Name of the switch object

Parameters

...	Options: • -motorobj Object (SR4) that controls the motor. • -motorhalf Which half: lower means Q1 and Q2, upper means Q3 and Q4. • -pointsenseobj Object (SR4) that senses the point state. • -pointsensehalf Which half: lower means I1 and I2, upper means I3 and I4. • -ossensorsn Serial number of the MRD2 that is sensing OS. • -diverttimeout Timeout, in seconds to allow for a train to clear the turnout when going on a divergent route. • -direction The current direction of travel. Forward always means entering at the point end. • -forwarddirection The *logial* forward direction. Set this to reverse for a frog facing switch. Default is forward and it is readonly and can only be set during creation. • -forwardsignalobj The signal object protecting the points. Presumed to be a two headed signal, with the upper head relating to the main (straight) route and the lower head relating to the divergent route. The upper head has three colors: red, yellow, and green. The lower head only two: red and green. • -reversemainsignalobj The signal object protecting the straight frog end. Presumed to be single headed (with number plate). • -reversedivergentsignalobj The signal object protecting the divergent frog end. Presumed to be single headed (with number plate). • -previousblock The block connected to the point end. • -nextmainblock The block connected to the straight frog end. • -nextdivergentblock The block connected to the divergent frog end. • -plate The name of the switch plate for this switch.

3.3.3 Member Function Documentation

3.3.3.1 _entering()

```
SR4_MRD2_Switch::_entering ( )  [protected]
```

Code to run when just entering the OS Sets the signal aspects and propagates signal state.

3.3.3.2 _exiting()

```
SR4_MRD2_Switch::_exiting ( )  [protected]
```

Code to run when about to exit the OS.

3.3.3.3 _gettruedirection()

```
SR4_MRD2_Switch::_gettruedirection (
             option  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.

Returns

Either forward or reverse.

3.3.3.4 _settruedirection()

```
SR4_MRD2_Switch::_settruedirection (
             option ,
             value  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.
value	Either forward or reverse.

3.3.3.5 motor()

```
SR4_MRD2_Switch::motor (
             route  )
```

The motor method sets the switch motor to align the points for the specificed route.

Parameters

route	The desired route. A value of normal means align the points to the main (straight) route and a value of reverse means align the points to the divergent route.

3.3.3.6 occupiedp()

```
SR4_MRD2_Switch::occupiedp ( )
```

The occupiedp method returns yes or no (true or false) indicating block (OS) occupation.

Returns

Yes or no, indicating whether the OS is occupied.

3.3.3.7 pointstate()

```
SR4_MRD2_Switch::pointstate ( )
```

The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.

If the state cannot be determined, a value of unknown is returned.

Returns

Normal or reverse, indicating the point state.

3.3.3.8 propagate()

```
SR4_MRD2_Switch::propagate (
             aspect ,
             from ,
              ...  )
```

Method used to propagate distant signal states back down the line.

Parameters

aspect	The signal aspect that is being propagated.
from	The propagating block.
...	Options: • -direction The direction of the propagation.

3.3.3.9 validate()

```
static SR4_MRD2_Switch::validate (
             object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a SR4_MRD2_Switch type.

Parameters

object	Some object.

3.3.4 Member Data Documentation

3.3.4.1 _routes

```
SR4_MRD2_Switch::_routes  [static], [private]
```

Route check validation object.

Definition at line 169 of file SR4_MRD2_Switch.tcl.

3.3.4.2 forwardsignal

```
SR4_MRD2_Switch::forwardsignal  [private]
```

Signal at the points.

Definition at line 85 of file SR4_MRD2_Switch.tcl.

3.3.4.3 motor

```
SR4_MRD2_Switch::motor  [private]
```

Motor device (SR4 outputs)

Definition at line 73 of file SR4_MRD2_Switch.tcl.

3.3.4.4 ossensor

```
SR4_MRD2_Switch::ossensor  [private]
```

Occupency sensor (MRD2)

Definition at line 81 of file SR4_MRD2_Switch.tcl.

3.3.4.5 pointsense

```
SR4_MRD2_Switch::pointsense  [private]
```

Point sense device (SR4 inputs)

Definition at line 77 of file SR4_MRD2_Switch.tcl.

3.3.4.6 reversedivergentsignal

```
SR4_MRD2_Switch::reversedivergentsignal  [private]
```

Signal at the divergent frog end.

Definition at line 93 of file SR4_MRD2_Switch.tcl.

3.3.4.7 reversemainsignal

```
SR4_MRD2_Switch::reversemainsignal  [private]
```

Signal at the straight frog end.

Definition at line 89 of file SR4_MRD2_Switch.tcl.

3.4 TB_Switch Class Reference

Switch (turnout) operation using a CTI Train Brain and Yardmaster.

Public Member Functions

- TB_Switch (name,...)

  Constructor: initialize the block object.
- occupiedp ()

  The occupiedp method returns yes or no (true or false) indicating block occupation.
- pointstate ()

  The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.
- motor (route)

  The motor method sets the switch motor to align the points for the specificed route.
- propagate (aspect, from,...)

  Method used to propagate distant signal states back down the line.

Static Public Member Functions

- static validate (object)

  Type validating code Raises an error if object is not either the empty string or a TB_Switch type.

Protected Member Functions

- _entering ()

  Code to run when just entering the OS Sets the signal aspects and propagates signal state.
- _exiting ()

  Code to run when about to exit the OS.

Private Member Functions

- _settruedirection (option, value)

  A method to fake direction for frog facing switches.
- _gettruedirection (option)

  A method to fake direction for frog facing switches.

Private Attributes

- acela

  Acela object.
- forwardsignal

  Signal object (typically a three color, one head block signal.
- reversesignal

  Signal object (typically a three color, one head block signal.
- isoccupied

  Saved occupation state.

Static Private Attributes

- static _pointsense

  Point sense bit values.
- static _routes

  Route check validation object.

3.4.1 Detailed Description

Switch (turnout) operation using a CTI Train Brain and Yardmaster.

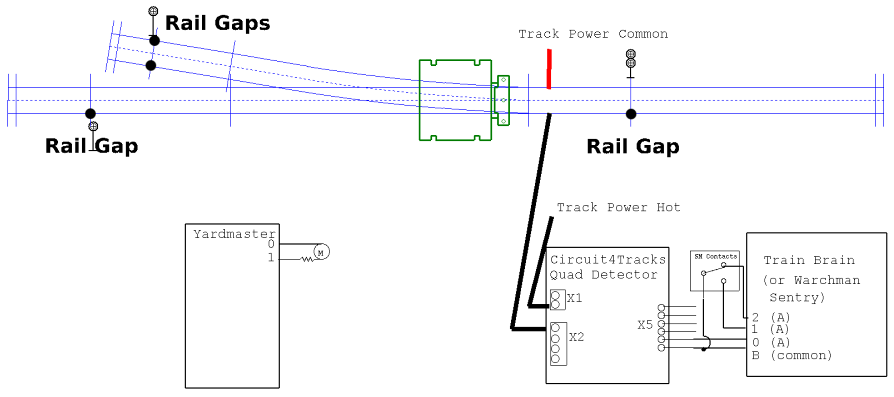

Figure 3.4 Switch controlled by CTI's Yardmaster and Train Brain

Above is a typical switch (turnout) using a CTI Yardmaster to control a Circuitron Tortoise Switch Machine and a CTI Train Brain to sense the point position and a Circuits4Track quad occupancy detector to sense occupation of the switch.

Typical usage:
```
# Connect to the CTI network via a CTI Acela at /dev/ttyACM0
ctiacela::CTIAcela acela /dev/ttyACM0
# Switch 1 is controled by bits 0 and 1 of the Yardmaster, and sensed
# with bits 0 (occupation), 1 and 2 (point position).
TB_Switch switch1 -acelaobj acela -motoraddress 0 -osaddress 0 \
-pointsense 1 -plate SwitchPlate1
# Switch 2 is controled by bits 2 and 3 of the Yardmaster, and sensed
# with bits 3 (occupation), 4 and 5 (point position).
TB_Switch switch2 -acelaobj acela -motoraddress 2 -osaddress 3 \
-pointsense 4 -plate SwitchPlate2
```

For the track work elements use "switchN occupiedp" for the track work elements' occupied script and use "switchN pointstate" for the track work elements' state script. For the switch plate use "switchN motor normal" for the normal script and "switchN motor reverse" for the reverse script.

Then in the Main Loop, you would have:
```
while {true} {
MainWindow ctcpanel invoke Switch1
MainWindow ctcpanel invoke Switch2
MainWindow ctcpanel invoke SwitchPlate1
MainWindow ctcpanel invoke SwitchPlate2
update;# Update display
}
```

Author

Robert Heller <heller@deepsoft.com>

Definition at line 49 of file TB_Switch.tcl.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 TB_Switch()

```
TB_Switch::TB_Switch (
            name ,
             ...  )
```

Constructor: initialize the block object.

Install an CTIAcela object as a component created elsewhere). Install the blocks signal (created elsewhere).

Parameters

name	Name of the block object
...	Options: • -acelaobj This is the CTIAcela object. This option is read-only and must be set at creation time. • -motoraddress The address of the motor control bits (two successive bits). This is an integer from 0 to 65535 inclusive. This option is read-only and can only be set at creation time. The default is 0. • -osaddress The address of the sensor bit for this block. This is an integer from 0 to 65535 inclusive. This option is read-only and can only be set at creation time. The default is 0. • -pointsense The address of the sensor bits for the point state sense (two successive bits). This is an integer from 0 to 65535 inclusive. This option is read-only and can only be set at creation time. The default is 0. • -direction The current direction of travel. Forward always means entering at the point end. • -forwarddirection The *logial* forward direction. Set this to reverse for a frog facing switch. Default is forward and it is readonly and can only be set during creation. • -forwardsignalobj The signal object protecting the points. Presumed to be a two headed signal, with the upper head relating to the main (straight) route and the lower head relating to the divergent route. The upper head has three colors: red, yellow, and green. The lower head only two: red and green. • -reversemainsignalobj The signal object protecting the straight frog end. Presumed to be single headed (with number plate). • -reversedivergentsignalobj The signal object protecting the divergent frog end. Presumed to be single headed (with number plate). • -previousblock The block connected to the point end. • -nextmainblock The block connected to the straight frog end. • -nextdivergentblock The block connected to the divergent frog end. • -plate The name of the switch plate for this switch.

3.4.3 Member Function Documentation

3.4.3.1 _entering()

```
TB_Switch::_entering ( )  [protected]
```

Code to run when just entering the OS Sets the signal aspects and propagates signal state.

3.4.3.2 _exiting()

```
TB_Switch::_exiting ( )  [protected]
```

Code to run when about to exit the OS.

3.4.3.3 _gettruedirection()

```
TB_Switch::_gettruedirection (
             option  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.

Returns

Either forward or reverse.

3.4.3.4 _settruedirection()

```
TB_Switch::_settruedirection (
             option ,
             value  )  [private]
```

A method to fake direction for frog facing switches.

Parameters

option	This is always -direction.
value	Either forward or reverse.

3.4.3.5 motor()

```
TB_Switch::motor (
             route  )
```

The motor method sets the switch motor to align the points for the specificed route.

Parameters

route	The desired route. A value of normal means align the points to the main (straight) route and a value of reverse means align the points to the divergent route.

3.4.3.6 occupiedp()

```
TB_Switch::occupiedp ( )
```

The occupiedp method returns yes or no (true or false) indicating block occupation.

3.4.3.7 pointstate()

```
TB_Switch::pointstate ( )
```

The pointstate method returns normal if the points are aligned to the main route and reverse if the points are aligned to the divergent route.

If the state cannot be determined, a value of unknown is returned.

Returns

Normal or reverse, indicating the point state.

3.4.3.8 propagate()

```
TB_Switch::propagate (
            aspect ,
            from ,
             ...  )
```

Method used to propagate distant signal states back down the line.

Parameters

aspect	The signal aspect that is being propagated.
from	The propagating block.
...	Options: • -direction The direction of the propagation.

3.4.3.9 validate()

```
static TB_Switch::validate (
            object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a TB_Switch type.

3.4.4 Member Data Documentation

3.4.4.1 _pointsense

```
TB_Switch::_pointsense  [static], [private]
```

Point sense bit values.

Definition at line 149 of file TB_Switch.tcl.

3.4.4.2 _routes

```
TB_Switch::_routes  [static], [private]
```

Route check validation object.

Definition at line 161 of file TB_Switch.tcl.

3.4.4.3 acela

`TB_Switch::acela [private]`

Acela object.

Definition at line 73 of file TB_Switch.tcl.

3.4.4.4 forwardsignal

`TB_Switch::forwardsignal [private]`

Signal object (typically a three color, one head block signal.

Definition at line 77 of file TB_Switch.tcl.

3.4.4.5 isoccupied

`TB_Switch::isoccupied [private]`

Saved occupation state.

Definition at line 85 of file TB_Switch.tcl.

3.4.4.6 reversesignal

`TB_Switch::reversesignal [private]`

Signal object (typically a three color, one head block signal.

Definition at line 81 of file TB_Switch.tcl.

Index