# Model Railroad System

2.2.2

Generated by Doxygen 1.9.1

**1 Block Abstract Types (Classes)** 1

1.1 Source Files . . . 1

1.1.1 MRD2U Sensor (USB connected optical sensor) . . . 1

1.1.2 Circuits4Tracks Quad Occupancy Detector with a SR4 (USB connected I/O board) . . . 1

1.1.3 Circuits4Tracks Quad Occupancy Detector connected to a C/MRI SMINI board . . . 2

1.1.4 Circuits4Tracks Quad Occupancy Detector connected to a CTI Train Brain . . . 2

1.2 Common methods and functionality . . . 2

**2 Class Index** 3

2.1 Class List . . . 3

**3 Class Documentation** 5

3.1 C4TSMINI_Block Class Reference . . . 5

3.1.1 Detailed Description . . . 6

3.1.2 Constructor & Destructor Documentation . . . 8

3.1.2.1 C4TSMINI_Block() . . . 9

3.1.3 Member Function Documentation . . . 9

3.1.3.1 _entering() . . . 9

3.1.3.2 _exiting() . . . 10

3.1.3.3 occupiedp() . . . 10

3.1.3.4 propagate() . . . 10

3.1.3.5 validate() . . . 10

3.1.4 Member Data Documentation . . . 10

3.1.4.1 forwardsignal . . . 11

3.1.4.2 isoccupied . . . 11

3.1.4.3 node . . . 11

3.1.4.4 reversesignal . . . 11

3.2 C4TSR4_Block Class Reference . . . 11

3.2.1 Detailed Description . . . 13

3.2.2 Constructor & Destructor Documentation . . . 15

3.2.2.1 C4TSR4_Block() . . . 16

3.2.3 Member Function Documentation . . . 16

3.2.3.1 _entering() . . . 16

3.2.3.2 _exiting() . . . 17

3.2.3.3 occupiedp() . . . 17

3.2.3.4 propagate() . . . 17

3.2.3.5 validate() . . . 17

3.2.4 Member Data Documentation . . . 17

3.2.4.1 forwardsignal . . . 18

3.2.4.2 isoccupied . . . 18

3.2.4.3 reversesignal . . . 18
3.2.4.4 sensemap . . . 18
3.2.4.5 sensor . . . 18
3.3 C4TTB_Block Class Reference . . . 19
3.3.1 Detailed Description . . . 20
3.3.2 Constructor & Destructor Documentation . . . 22
3.3.2.1 C4TTB_Block() . . . 23
3.3.3 Member Function Documentation . . . 23
3.3.3.1 _entering() . . . 23
3.3.3.2 _exiting() . . . 24
3.3.3.3 occupiedp() . . . 24
3.3.3.4 propagate() . . . 24
3.3.3.5 validate() . . . 24
3.3.4 Member Data Documentation . . . 24
3.3.4.1 acela . . . 25
3.3.4.2 forwardsignal . . . 25
3.3.4.3 isoccupied . . . 25
3.3.4.4 reversesignal . . . 25
3.4 MRD2_Block Class Reference . . . 25
3.4.1 Detailed Description . . . 26
3.4.2 Constructor & Destructor Documentation . . . 28
3.4.2.1 MRD2_Block() . . . 29
3.4.3 Member Function Documentation . . . 29
3.4.3.1 _entering() . . . 29
3.4.3.2 _exiting() . . . 30
3.4.3.3 occupiedp() . . . 30
3.4.3.4 propagate() . . . 30
3.4.3.5 validate() . . . 30
3.4.4 Member Data Documentation . . . 30
3.4.4.1 forwardsignal . . . 31
3.4.4.2 reversesignal . . . 31
3.4.4.3 sensor . . . 31

**Index** **33**

# Chapter 1

# Block Abstract Types (Classes)

This folder contains a collection of Tcl code to implement block occupancy detection, using various methods. At the very least, block detection results in signal aspect updates. Code for managing signals is in in the Signals folder.

There are two main ways to detect trains: using optical sensors or using current sensors. Optical sensors generally work via reflection (bouncing a light beam off the bottom of the train), although an across-the-tracks type is possible too. Current sensors work by sensing a current flow when a locomotive, lighted passenger car, or a freight car with resistors installed on its wheelsets passes onto an electrically isolated section of track. In any case, the sensor is connected to the computer somehow, either via USB or via a direct or indirect I/O bit or port.

## 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 *block*. All of these abstract data types include a method named `occupiedp`, which returns a true or false result indicating whether or not the block is occupied. The constructor for these types include options or arguments that define the I/O device(s) that connect to whatever sensors are being used to detect block occupancy.

### 1.1.1 MRD2U Sensor (USB connected optical sensor)

MRD2_Block.tcl contains an abstract data type (MRD2_Block) that implements blocks using one Azatrax MRD2U Sensor for each block.

### 1.1.2 Circuits4Tracks Quad Occupancy Detector with a SR4 (USB connected I/O board)

C4TSR4_Block.tcl contains an abstract data type (C4TSR4_Block) that implements blocks using Circuits4Tracks Quad Occupancy Detectors connected to Azatrax SR4 modules using SSRs. One Circuits4Tracks Quad Occupancy Detector and one SR4 will handle 4 blocks.

### 1.1.3 Circuits4Tracks Quad Occupancy Detector connected to a C/MRI SMINI board

C4TSMINI_Block.tcl contains an abstract data type (C4TSMINI_Block) that implements blocks using Circuits4Tracks Quad Occupancy Detectors connected to input pins of a Bruce Chubb C/MRI Super Mini (SMINI) board. A Bruce Chubb C/MRI Super Mini (SMINI) board has enough inputs to handle a number of Circuits4Tracks Quad Occupancy Detectors.

### 1.1.4 Circuits4Tracks Quad Occupancy Detector connected to a CTI Train Brain

C4TTB_Block.tcl contains an abstract data type (C4TTB_Block) that implements blocks using Circuits4Tracks Quad Occupancy Detectors connected to the sensor inputs of a CTI Train Brain, Watchman, or Sentry board.Circuits4Tracks Quad Occupancy Detector connected to a

## 1.2 Common methods and functionality

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

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

Eg:

```
MRD2_Block block2 -previousblock block1 -sensorsn 020001234 \
                  -forwardsignalobj signal2
```

There are a pair of common options, `-previousblock` and `-nextblock`, which are the names of the previous block in the forward direction and the name of the previous block in the reverse direction. Another pair of common options, `-forwardsignalobj` and `-reversesignalobj`, 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 four common methods, two public and two private (the private methods should not be used by external code). The public methods are `occupiedp` and `propagate`. The `occupiedp` method returns a true or false (logical) value that indicates whether the block is occupied or not. 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 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:


C4TSMINI_Block
: Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s . . 5

C4TSR4_Block
: Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s . . 11

C4TTB_Block
: Block occupation detection using Circuits4Tracks Quad Occupancy Detector and a CTI Train Brain . 19

MRD2_Block
: Block occupation detection using Azatrax MRD2Us . . . . . 25

# Chapter 3

# Class Documentation

## 3.1 C4TSMINI_Block Class Reference

Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s.

### Public Member Functions

- C4TSMINI_Block (name,...)

  *Constructor: initialize the block object.*
- occupiedp ()

  *The occupiedp method returns yes or no (true or false) indicating block occupation.*
- 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 C4TSMINI_Block type.*

### Private Member Functions

- _entering ()

  *Method for entering a block.*
- _exiting ()

  *Method for exiting a block.*

## Private Attributes

- node

  *SMINI node 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.*

### 3.1.1 Detailed Description

Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s.

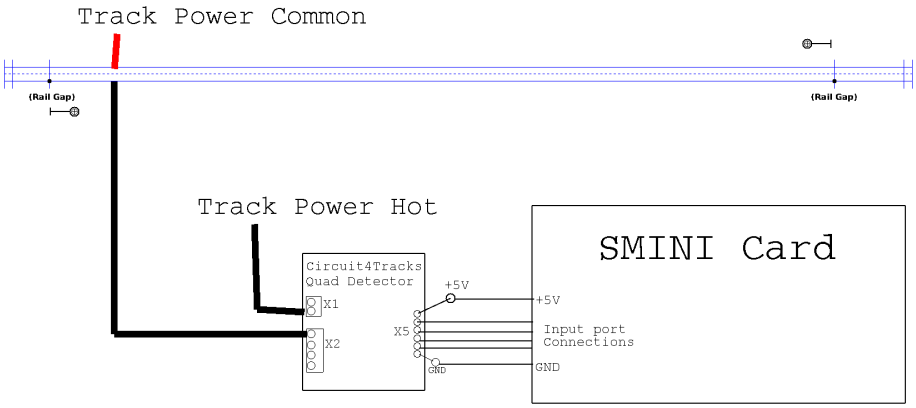


**Figure 3.1 Block detection with a Circuits4Tracks Quad Occupancy Detector and a Chubb SMINI card**

Above is a simple diagram for using Circuits4Tracks Quad Occupancy Detectors for block occupation detection. A Circuits4Tracks Quad Occupancy board has four current sensors. One wires one side of the track power (either DCC or DC) to a common rail and the other side through the Circuits4Tracks Quad Occupancy Detector to rails isolated with gaps (possibly with insulating rail joiners). This code uses a Chubb SMINI board to connect a Circuits4Tracks Quad Occupancy Detectors to the computer via a serial interface.

Typical usage:

Four blocks in a loop:

```
# 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
# The first four bits of the first port are wired to the Circuits4Tracks
# Quad Occupancy Detector
C4TSR4_Block block1 -nodeobj SMINI0 -port 0 -bit 0 -signalobj signal1
C4TSR4_Block block2 -nodeobj SMINI0 -port 0 -bit 1 -signalobj signal2 -previousblock block1
C4TSR4_Block block3 -nodeobj SMINI0 -port 0 -bit 2 -signalobj signal3 -previousblock block2
C4TSR4_Block block4 -nodeobj SMINI0 -port 0 -bit 3 -signalobj signal4 -previousblock block3
block1 configure -previousblock block4
```

A Schematic of the layout would look like this:

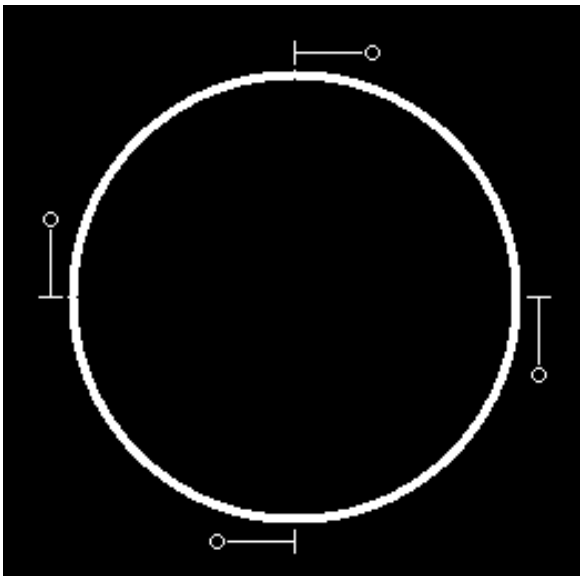

**Figure 3.2 Four block circle**

For the track work elements use "blockN occupiedp" for the track work elements' occupied command: eg Block1 would have 'block1 occupiedp' as its occupied command, that is its edit window would look like:

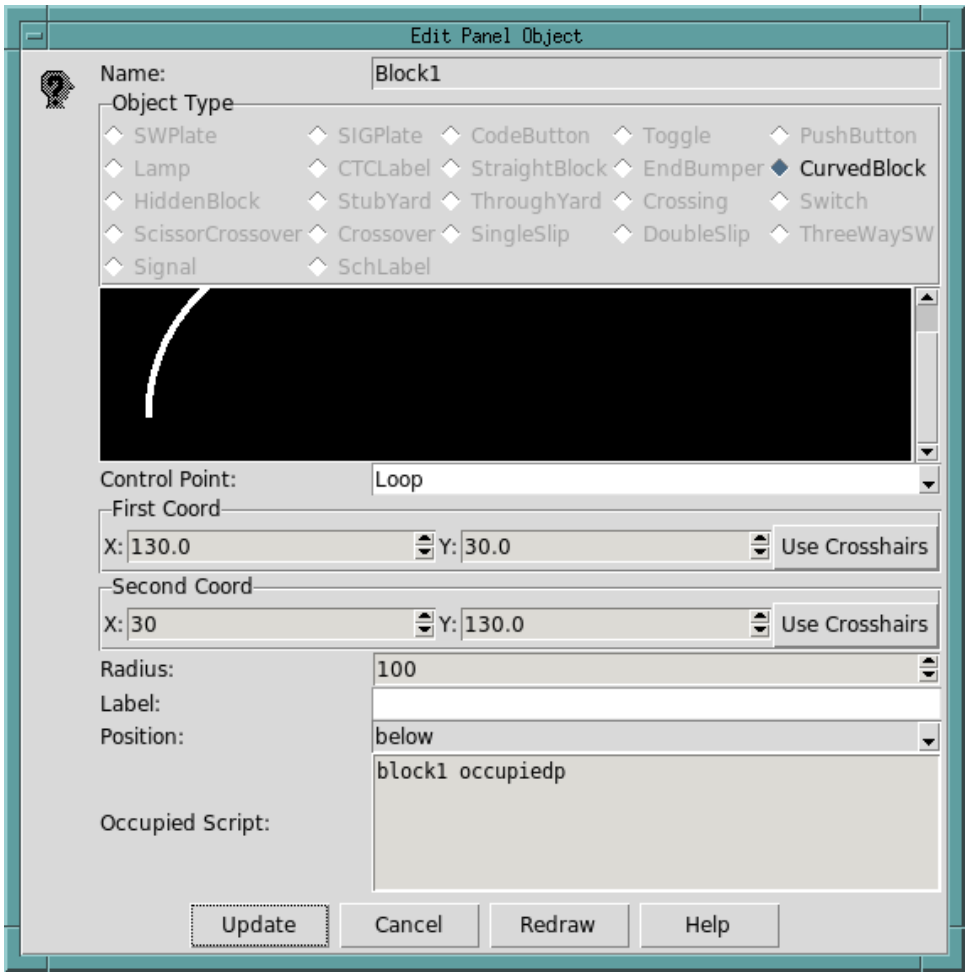


**Figure 3.3 Editing Block1**

The other three blocks would be similar.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Block1
MainWindow ctcpanel invoke Block2
MainWindow ctcpanel invoke Block3
MainWindow ctcpanel invoke Block4
update;# Update display
}
```

**Author**

Robert Heller <heller@deepsoft.com>

Definition at line 62 of file C4TSMINI_Block.tcl.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 C4TSMINI_Block()

```
C4TSMINI_Block::C4TSMINI_Block (
            name ,
             ...  )
```

Constructor: initialize the block object.

Create a lowlevel node object and install it as a component. Install the blocks signal (created elsewhere).

**Parameters**

| | |
|---|---|
| *name* | Name of the block object |
| ... | Options:<br>• -nodeobj This is the Cmri node for this block. This is a CmriNode object, defined in the Control Support package. This option is read-only and must be set at creation time.<br>• -port The input port on the Cmri node. This is an integer greater or equal to 0. This option is read-only and can only be set at creation time. The default is 0.<br>• -bit This defines the input bit on the input port for this block. This is an integer from 0 to 7 inclusive and is read-only and can only be set at creation time. The default is 0.<br>• -forwardsignalobj This block's forward signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -reversesignalobj This block's reverse signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -previousblock Previous block (next block in reverse) – used for 'propagating' signal aspects and must be a C4TSMINI_Block type object. The default is the empty string.<br>• -nextblock Next block (previous block in reverse) – used for 'propagating' signal aspects and must be a C4TSMINI_Block type object. The default is the empty string.<br>• -direction Current running direction, either the word forward or reverse. The default is forward. |

### 3.1.3 Member Function Documentation

#### 3.1.3.1 _entering()

```
C4TSMINI_Block::_entering ( )  [private]
```

Method for entering a block.

#### 3.1.3.2 _exiting()

```
C4TSMINI_Block::_exiting ( )  [private]
```

Method for exiting a block.

#### 3.1.3.3 occupiedp()

```
C4TSMINI_Block::occupiedp ( )
```

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

#### 3.1.3.4 propagate()

```
C4TSMINI_Block::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 (not used). |
| *...* | Options:<br>• -direction The direction of the propagation. |

#### 3.1.3.5 validate()

```
static C4TSMINI_Block::validate (
            object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a C4TSMINI_Block type.

### 3.1.4 Member Data Documentation

#### 3.1.4.1 forwardsignal

`C4TSMINI_Block::forwardsignal [private]`

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

Definition at line 85 of file C4TSMINI_Block.tcl.

#### 3.1.4.2 isoccupied

`C4TSMINI_Block::isoccupied [private]`

Saved occupation state.

Definition at line 93 of file C4TSMINI_Block.tcl.

#### 3.1.4.3 node

`C4TSMINI_Block::node [private]`

SMINI node object.

Definition at line 81 of file C4TSMINI_Block.tcl.

#### 3.1.4.4 reversesignal

`C4TSMINI_Block::reversesignal [private]`

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

Definition at line 89 of file C4TSMINI_Block.tcl.

## 3.2 C4TSR4_Block Class Reference

Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s.

## Public Member Functions

- C4TSR4_Block (name,...)

  *Constructor: initialize the block object.*
- occupiedp ()

  *The occupiedp method returns yes or no (true or false) indicating block occupation.*
- 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 C4TSR4_Block type.*

## Private Member Functions

- _entering ()

  *Method for entering a block.*
- _exiting ()

  *Method for exiting a block.*

## Private Attributes

- sensor

  *SR4 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 sensemap

  *Sensor bit mapping to sensor functions.*

### 3.2.1 Detailed Description

Block occupation detection using Circuits4Tracks Quad Occupancy Detectors and Azatrax SR4s.

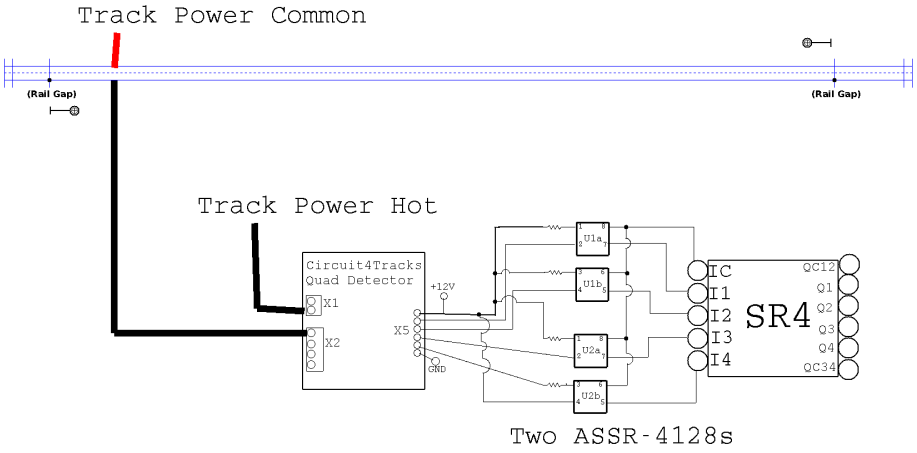


**Figure 3.4 Block detection with a Circuits4Tracks Quad Occupancy Detector and a SR4**

Above is a simple diagram for using Circuits4Tracks Quad Occupancy Detectors for block occupation detection. A Circuits4Tracks Quad Occupancy board has four current sensors. One wires one side of the track power (either DCC or DC) to a common rail and the other side through the Circuits4Tracks Quad Occupancy Detector to rails isolated with gaps (possibly with insulating rail joiners). This code uses Azatrax SR4s to connect a Circuits4Tracks Quad Occupancy Detectors to the computer via USB. A small circuit board with two ASSR-4128s (dual Solid State Relays) and four 1,000 Ohm resistors and some headers connects the Circuits4Tracks board to the SR4.

Typical usage:

Four blocks in a loop:

```
SR4 quadsense1 -this [Azatrax_OpenDevice 0400001234 $::Azatrax_idSR4Product]
C4TSR4_Block block1 -sensorobj quadsense1 -bit 0 -signalobj signal1
C4TSR4_Block block2 -sensorobj quadsense1 -bit 1 -signalobj signal2 -previousblock block1
C4TSR4_Block block3 -sensorobj quadsense1 -bit 2 -signalobj signal3 -previousblock block2
C4TSR4_Block block4 -sensorobj quadsense1 -bit 3 -signalobj signal4 -previousblock block3
block1 configure -previousblock block4
```

A Schematic of the layout would look like this:

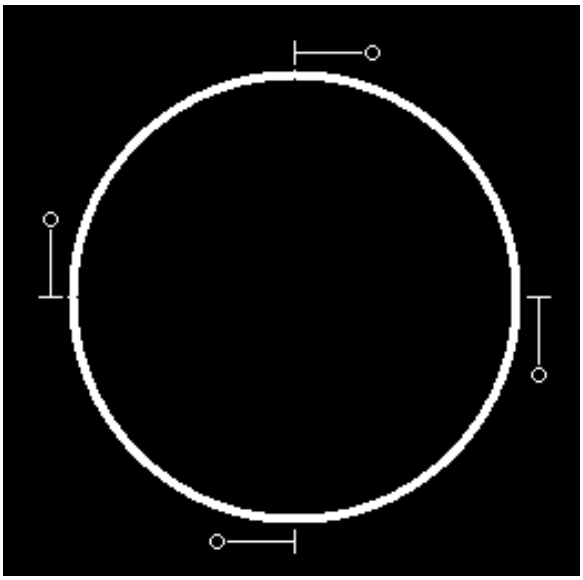

**Figure 3.5 Four block circle**

For the track work elements use "blockN occupiedp" for the track work elements' occupied command: eg Block1 would have 'block1 occupiedp' as its occupied command, that is its edit window would look like:

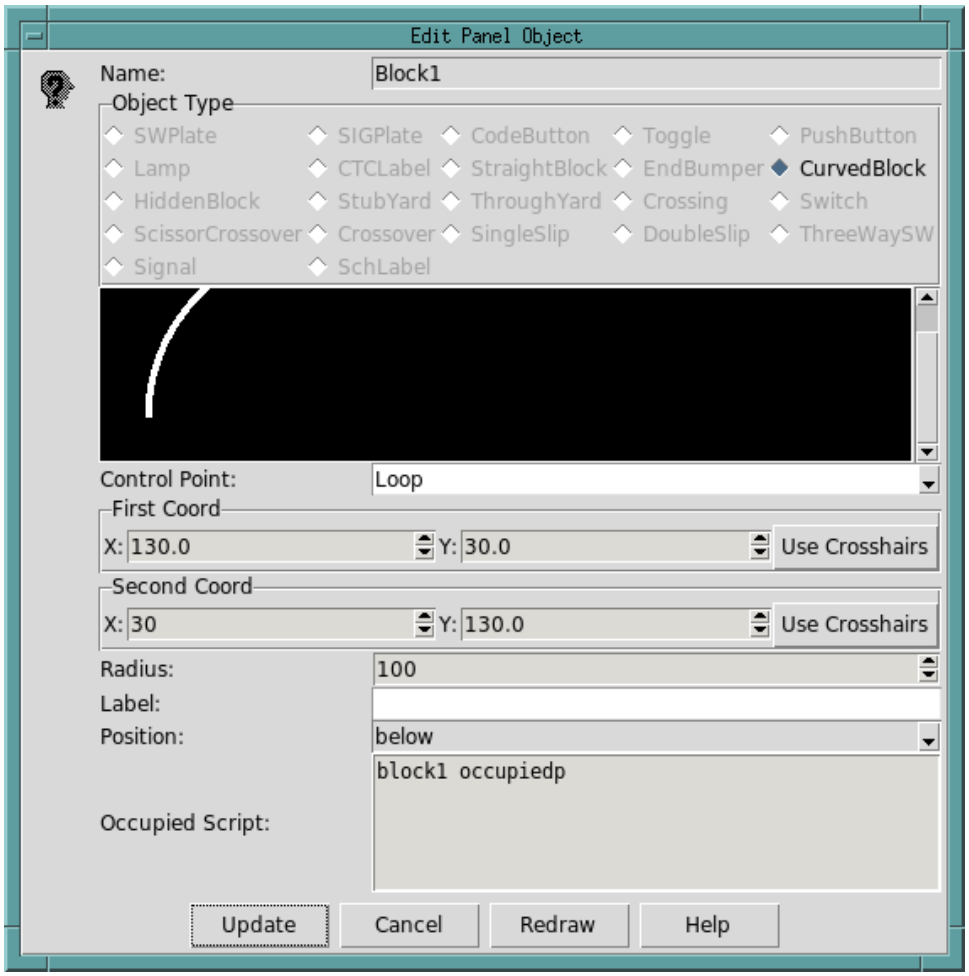


**Figure 3.6 Editing Block1**

The other three blocks would be similar.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Block1
MainWindow ctcpanel invoke Block2
MainWindow ctcpanel invoke Block3
MainWindow ctcpanel invoke Block4
update;# Update display
}
```

**Author**

Robert Heller <heller@deepsoft.com>

Definition at line 59 of file C4TSR4_Block.tcl.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 C4TSR4_Block()

```
C4TSR4_Block::C4TSR4_Block (
            name ,
             ...  )
```

Constructor: initialize the block object.

Create a lowlevel sensor object and install it as a component. Install the blocks signal (created elsewhere).

**Parameters**

| | |
|---|---|
| *name* | Name of the block object |
| ... | Options:<br>• -sensorobj This is the SR4 for this (and up to three other blocks). This option is read-only and must be set at creation time.<br>• -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.<br>• -forwardsignalobj This block's forward signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -reversesignalobj This block's reverse signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -previousblock Previous block (next block in reverse) – used for 'propagating' signal aspects and must be a C4TSR4_Block type object. The default is the empty string.<br>• -nextblock Next block (previous block in reverse) – used for 'propagating' signal aspects and must be a C4TSR4_Block type object.<br>The default is the empty string.<br>• -direction Current running direction, either the word forward or reverse. The default is forward. |

### 3.2.3 Member Function Documentation

#### 3.2.3.1 _entering()

```
C4TSR4_Block::_entering ( )  [private]
```

Method for entering a block.

#### 3.2.3.2 _exiting()

```
C4TSR4_Block::_exiting ( )  [private]
```

Method for exiting a block.

#### 3.2.3.3 occupiedp()

```
C4TSR4_Block::occupiedp ( )
```

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

#### 3.2.3.4 propagate()

```
C4TSR4_Block::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 (not used). |
| *...* | Options:<br>• -direction The direction of the propagation. |

#### 3.2.3.5 validate()

```
static C4TSR4_Block::validate (
            object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a C4TSR4_Block type.

### 3.2.4 Member Data Documentation

#### 3.2.4.1 forwardsignal

`C4TSR4_Block::forwardsignal [private]`

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

Definition at line 81 of file C4TSR4_Block.tcl.

#### 3.2.4.2 isoccupied

`C4TSR4_Block::isoccupied [private]`

Saved occupation state.

Definition at line 89 of file C4TSR4_Block.tcl.

#### 3.2.4.3 reversesignal

`C4TSR4_Block::reversesignal [private]`

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

Definition at line 85 of file C4TSR4_Block.tcl.

#### 3.2.4.4 sensemap

`C4TSR4_Block::sensemap [static], [private]`

Sensor bit mapping to sensor functions.

Definition at line 93 of file C4TSR4_Block.tcl.

#### 3.2.4.5 sensor

`C4TSR4_Block::sensor [private]`

SR4 object.

Definition at line 77 of file C4TSR4_Block.tcl.

## 3.3 C4TTB_Block Class Reference

Block occupation detection using Circuits4Tracks Quad Occupancy Detector and a CTI Train Brain.

### Public Member Functions

- C4TTB_Block (name,...)

  *Constructor: initialize the block object.*
- occupiedp ()

  *The occupiedp method returns yes or no (true or false) indicating block occupation.*
- 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 C4TTB_Block type.*

### Private Member Functions

- _entering ()

  *Method for entering a block.*
- _exiting ()

  *Method for exiting a block.*

### 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.*

### 3.3.1 Detailed Description

Block occupation detection using Circuits4Tracks Quad Occupancy Detector and a CTI Train Brain.

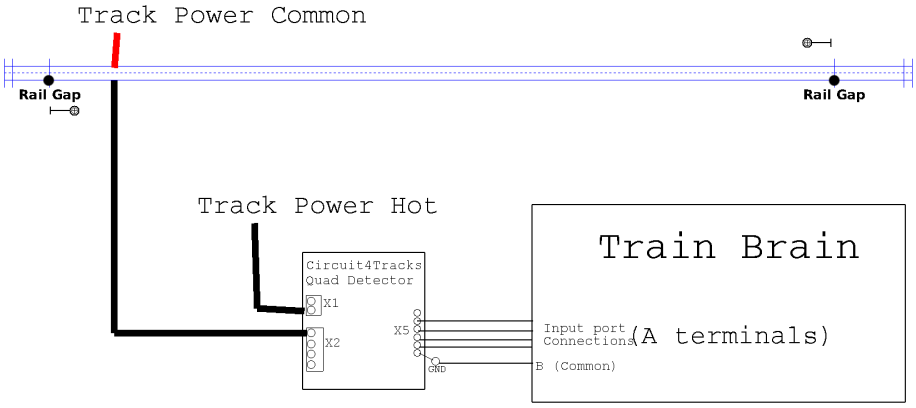


**Figure 3.7 Block detection with a Circuits4Tracks Quad Occupancy Detector and a Train Brain**

Above is a simple diagram for using Circuits4Tracks Quad Occupancy Occupancy board has four current sensors. One wires one side of the track power (either DCC or DC) to a common rail and the other side through the Circuits4Tracks Quad Occupancy Detector to rails isolated with gaps (possibly with insulating rail joiners). This code uses a CTI Train Brain to connect a Circuits4Tracks Quad Occupancy Detectors to the computer via a CTI Acela computer interface.

Typical usage:

Four blocks in a loop:

```
# Connect to the CTI Acela via USB the serial interface at /dev/ttyACM0
ctiacela::CTIAcela acela /dev/ttyACM0
# The first four bits of the first Train Brain are wired to the Circuits4Tracks
# Quad Occupancy Detector
C4TTB_Block block1 -acelaobj acela -address 0 -signalobj signal1
C4TTB_Block block2 -acelaobj acela -address 1 -signalobj signal2 -previousblock block1
C4TTB_Block block3 -acelaobj acela -address 2 -signalobj signal3 -previousblock block2
C4TTB_Block block4 -acelaobj acela -address 3 -signalobj signal4 -previousblock block3
block1 configure -previousblock block4
```

A Schematic of the layout would look like this:

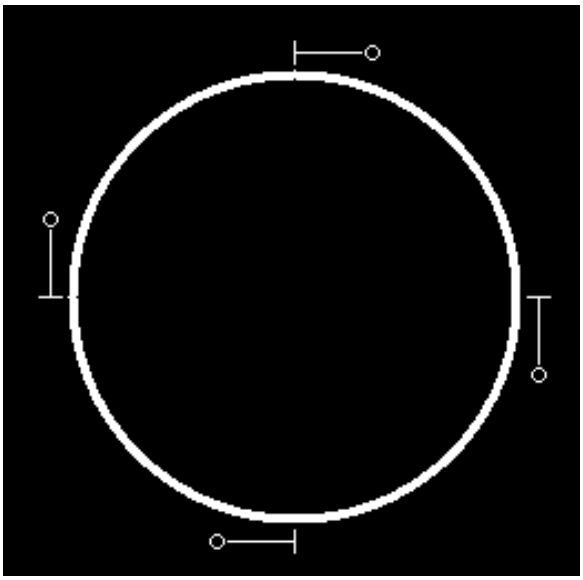

**Figure 3.8 Four block circle**

For the track work elements use "blockN occupiedp" for the track work elements' occupied command: eg Block1 would have 'block1 occupiedp' as its occupied command, that is its edit window would look like:

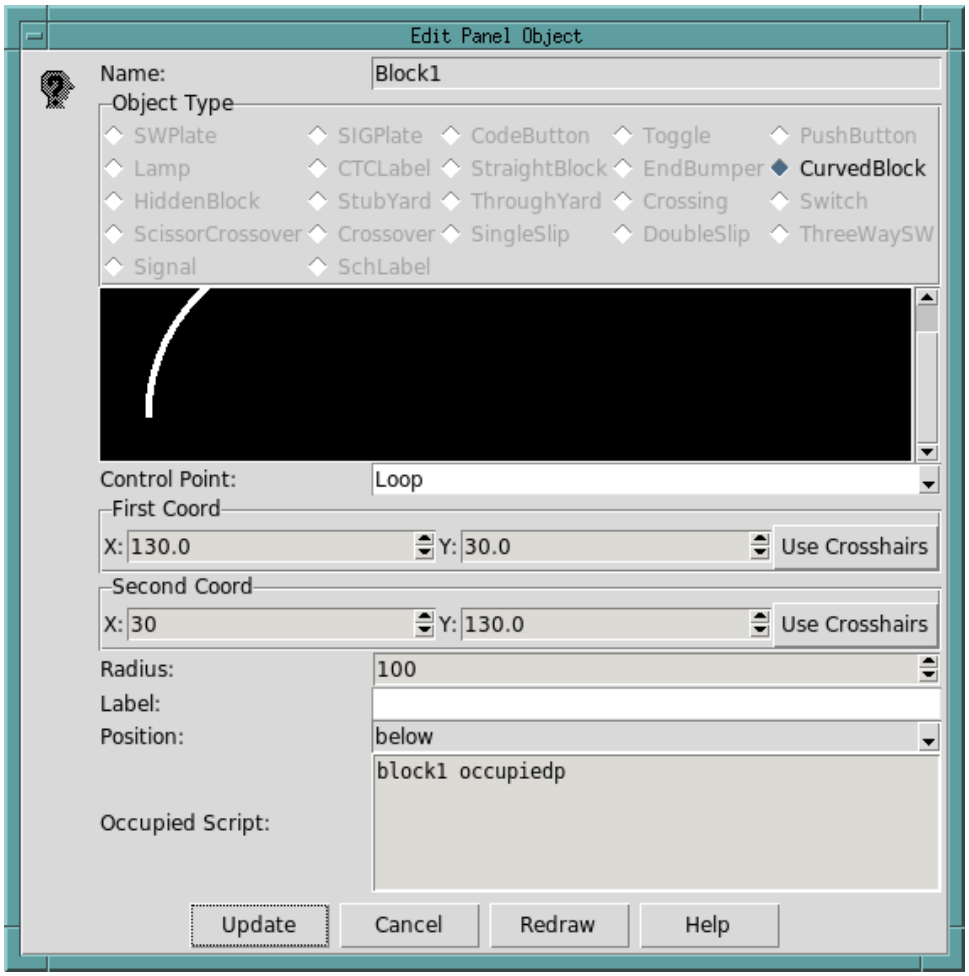


**Figure 3.9 Editing Block1**

The other three blocks would be similar.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Block1
MainWindow ctcpanel invoke Block2
MainWindow ctcpanel invoke Block3
MainWindow ctcpanel invoke Block4
update;# Update display
}
```

**Author**

Robert Heller <heller@deepsoft.com>

Definition at line 59 of file C4TTB_Block.tcl.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 C4TTB_Block()

```
C4TTB_Block::C4TTB_Block (
            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:<br>• -acelaobj This is the CTIAcela object. This option is read-only and must be set at creation time.<br>• -address 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.<br>• -forwardsignalobj This block's forward signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -reversesignalobj This block's reverse signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -previousblock Previous block (next block in reverse) – used for 'propagating' signal aspects and must be a C4TTB_Block type object. The default is the empty string.<br>• -nextblock Next block (previous block in reverse) – used for 'propagating' signal aspects and must be a C4TTB_Block type object.<br>The default is the empty string.<br>• -direction Current running direction, either the word forward or reverse. The default is forward. |

### 3.3.3 Member Function Documentation

#### 3.3.3.1 _entering()

```
C4TTB_Block::_entering ( )  [private]
```

Method for entering a block.

#### 3.3.3.2 _exiting()

```
C4TTB_Block::_exiting ( )  [private]
```

Method for exiting a block.

#### 3.3.3.3 occupiedp()

```
C4TTB_Block::occupiedp ( )
```

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

#### 3.3.3.4 propagate()

```
C4TTB_Block::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 (not used). |
| *...* | Options:<br>• -direction The direction of the propagation. |

#### 3.3.3.5 validate()

```
static C4TTB_Block::validate (
            object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a C4TTB_Block type.

### 3.3.4 Member Data Documentation

#### 3.3.4.1 acela

`C4TTB_Block::acela [private]`

Acela object.

Definition at line 77 of file C4TTB_Block.tcl.

#### 3.3.4.2 forwardsignal

`C4TTB_Block::forwardsignal [private]`

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

Definition at line 81 of file C4TTB_Block.tcl.

#### 3.3.4.3 isoccupied

`C4TTB_Block::isoccupied [private]`

Saved occupation state.

Definition at line 89 of file C4TTB_Block.tcl.

#### 3.3.4.4 reversesignal

`C4TTB_Block::reversesignal [private]`

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

Definition at line 85 of file C4TTB_Block.tcl.

## 3.4 MRD2_Block Class Reference

Block occupation detection using Azatrax MRD2Us.

## Public Member Functions

- MRD2_Block (name,...)

  *Constructor: initialize the block object.*
- occupiedp ()

  *The occupiedp method returns yes or no (true or false) indicating block occupation.*
- 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 MRD2_Block type.*

## Private Member Functions

- _entering ()

  *Method for entering a block.*
- _exiting ()

  *Method for exiting a block.*

## Private Attributes

- sensor

  *MRD2 object.*
- forwardsignal

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

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

### 3.4.1 Detailed Description

Block occupation detection using Azatrax MRD2Us.

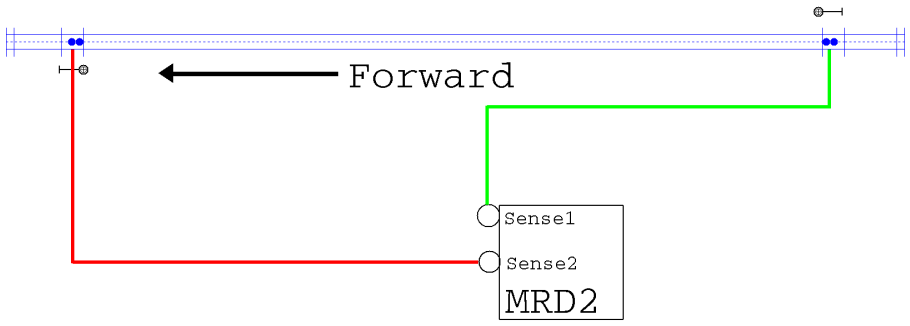


**Figure 3.10 Block detection using a MRD2**

Above is a simple diagram for using Azatrax MRD2Us for block occupation detection. The Azatrax MRD2U has two IR sensors and one can be use to test for entering a block and one for leaving a block.

Typical usage:

Four blocks in a loop:

```
MRD2_Block block1 -sensorsn 0200001234 -forwardsignalobj signal1
MRD2_Block block2 -sensorsn 0200001235 -forwardsignalobj signal2 -previousblock block1
MRD2_Block block3 -sensorsn 0200001236 -forwardsignalobj signal3 -previousblock block2
MRD2_Block block4 -sensorsn 0200001237 -forwardsignalobj signal4 -previousblock block3
block1 configure -previousblock block4
```

A Schematic of the layout would look like this:

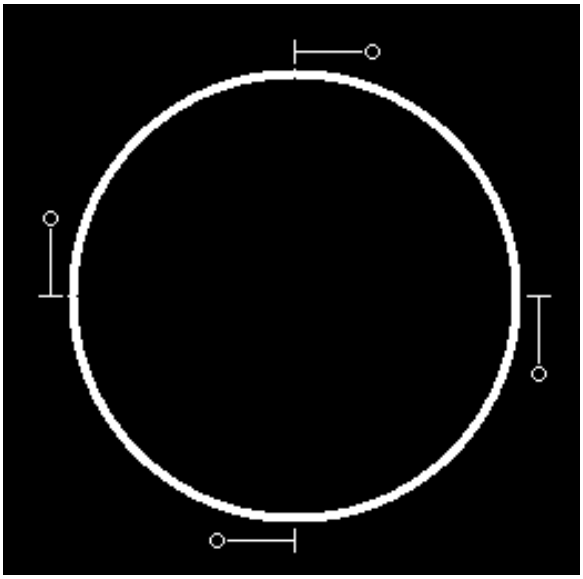

**Figure 3.11 Four block circle**

For the track work elements use "blockN occupiedp" for the track work elements' occupied command: eg Block1 would have 'block1 occupiedp' as its occupied command, that is its edit window would look like:

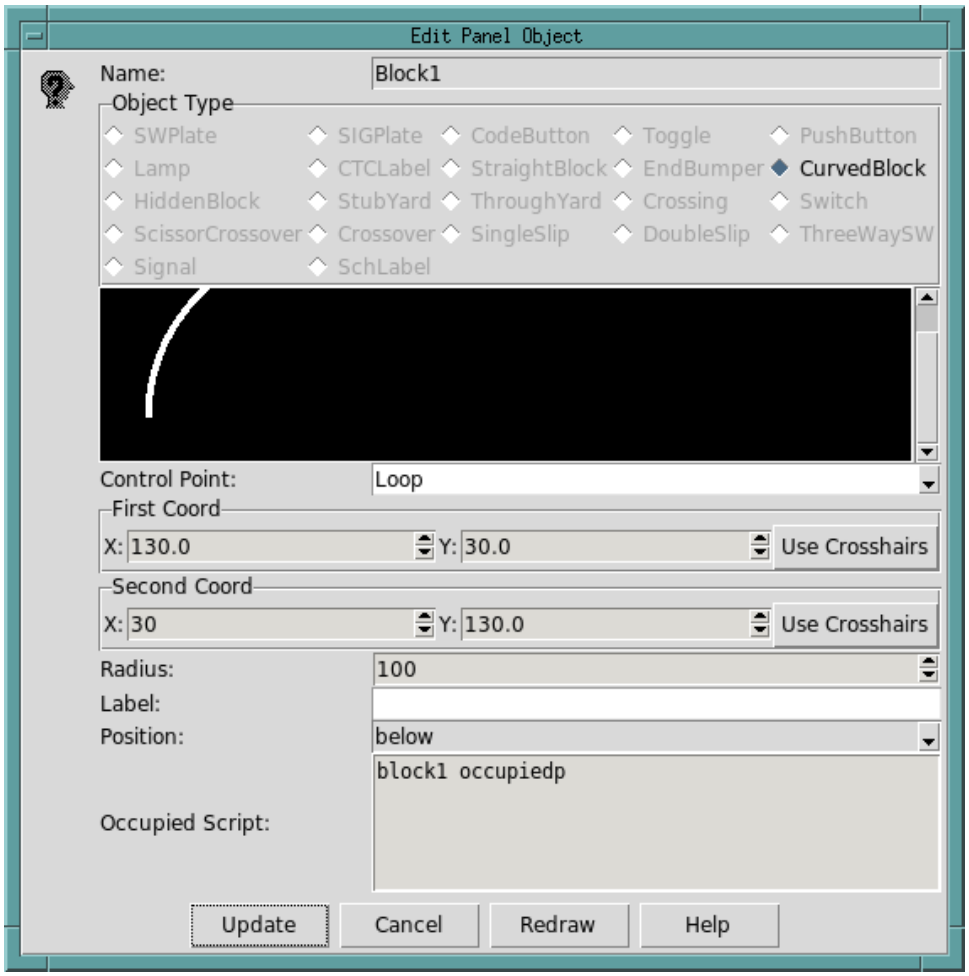


**Figure 3.12 Editing Block1**

The other three blocks would be similar.

Then in the Main Loop, you would have:

```
while {true} {
MainWindow ctcpanel invoke Block1
MainWindow ctcpanel invoke Block2
MainWindow ctcpanel invoke Block3
MainWindow ctcpanel invoke Block4
update;# Update display
}
```

**Author**

Robert Heller <heller@deepsoft.com>

Definition at line 51 of file MRD2_Block.tcl.

### 3.4.2 Constructor & Destructor Documentation

#### 3.4.2.1 MRD2_Block()

```
MRD2_Block::MRD2_Block (
            name ,
             ...  )
```

Constructor: initialize the block object.

Create a low level sensor object and install it as a component. Install the blocks signal (created elsewhere).

**Parameters**

| | |
|---|---|
| *name* | Name of the block object |
| ... | Options:<br>• -sensorsn Serial number of the MRD2U for this block. This option is read-only and must be set at creation time.<br>• -forwardsignalobj This block's forward signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -reversesignalobj This block's reverse signal. This option is read-only and can only be set at creation time. The default is the empty string.<br>• -previousblock Previous block (next block in reverse) – used for 'propagating' signal aspects and must be a MRD2_Block type object. The default is the empty string.<br>• -nextblock Next block (previous block in reverse) – used for 'propagating' signal aspects and must be a MRD2_Block type object.<br>The default is the empty string.<br>• -direction Current running direction, either the word forward or reverse. The default is forward. |

### 3.4.3 Member Function Documentation

#### 3.4.3.1 _entering()

```
MRD2_Block::_entering ( )  [private]
```

Method for entering a block.

#### 3.4.3.2 _exiting()

```
MRD2_Block::_exiting ( )  [private]
```

Method for exiting a block.

#### 3.4.3.3 occupiedp()

```
MRD2_Block::occupiedp ( )
```

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

#### 3.4.3.4 propagate()

```
MRD2_Block::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 (not used). |
| *...* | Options:<br>• -direction The direction of the propagation. |

#### 3.4.3.5 validate()

```
static MRD2_Block::validate (
            object  )  [static]
```

Type validating code Raises an error if object is not either the empty string or a MRD2_Block type.

### 3.4.4 Member Data Documentation

#### 3.4.4.1 forwardsignal

`MRD2_Block::forwardsignal [private]`

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

Definition at line 72 of file MRD2_Block.tcl.

#### 3.4.4.2 reversesignal

`MRD2_Block::reversesignal [private]`

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

Definition at line 76 of file MRD2_Block.tcl.

#### 3.4.4.3 sensor

`MRD2_Block::sensor [private]`

MRD2 object.

Definition at line 68 of file MRD2_Block.tcl.

# Index


_entering
    C4TSMINI_Block, 9
    C4TSR4_Block, 16
    C4TTB_Block, 23
    MRD2_Block, 29
_exiting
    C4TSMINI_Block, 9
    C4TSR4_Block, 16
    C4TTB_Block, 23
    MRD2_Block, 29

acela
    C4TTB_Block, 24

C4TSMINI_Block, 5
    _entering, 9
    _exiting, 9
    C4TSMINI_Block, 8
    forwardsignal, 10
    isoccupied, 11
    node, 11
    occupiedp, 10
    propagate, 10
    reversesignal, 11
    validate, 10
C4TSR4_Block, 11
    _entering, 16
    _exiting, 16
    C4TSR4_Block, 15
    forwardsignal, 17
    isoccupied, 18
    occupiedp, 17
    propagate, 17
    reversesignal, 18
    sensemap, 18
    sensor, 18
    validate, 17
C4TTB_Block, 19
    _entering, 23
    _exiting, 23
    acela, 24
    C4TTB_Block, 22
    forwardsignal, 25
    isoccupied, 25
    occupiedp, 24
    propagate, 24
    reversesignal, 25
    validate, 24

forwardsignal
    C4TSMINI_Block, 10
    C4TSR4_Block, 17
    C4TTB_Block, 25
    MRD2_Block, 30

isoccupied
    C4TSMINI_Block, 11
    C4TSR4_Block, 18
    C4TTB_Block, 25

MRD2_Block, 25
    _entering, 29
    _exiting, 29
    forwardsignal, 30
    MRD2_Block, 28
    occupiedp, 30
    propagate, 30
    reversesignal, 31
    sensor, 31
    validate, 30

node
    C4TSMINI_Block, 11

occupiedp
    C4TSMINI_Block, 10
    C4TSR4_Block, 17
    C4TTB_Block, 24
    MRD2_Block, 30

propagate
    C4TSMINI_Block, 10
    C4TSR4_Block, 17
    C4TTB_Block, 24
    MRD2_Block, 30

reversesignal
    C4TSMINI_Block, 11
    C4TSR4_Block, 18
    C4TTB_Block, 25
    MRD2_Block, 31

sensemap

C4TSR4_Block, 18
sensor
C4TSR4_Block, 18
MRD2_Block, 31

validate
C4TSMINI_Block, 10
C4TSR4_Block, 17
C4TTB_Block, 24
MRD2_Block, 30