Report on robotrain

Generated by MTT using :
(mtt -u -q robotrain rep view)

Mon Sep 13 00:11:44 BST 2004

Contents

List of Figures

1 robotrain_abg.tex

MTT command:

```
mtt robotrain abg tex
```

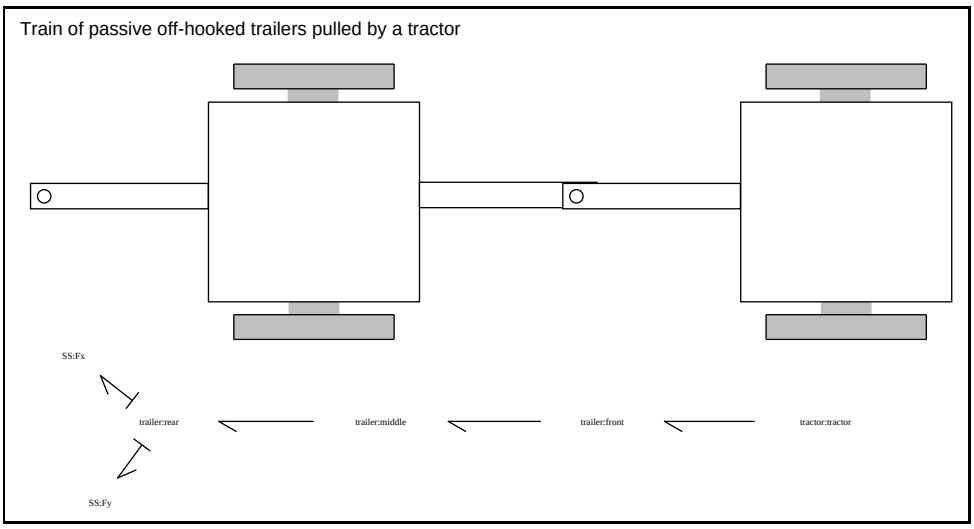

Figure 1: System **robotrain**: acausal bond graph

The acausal bond graph of system **robotrain** is displayed in Figure 1 (on page 3). The model is based on an off-hooked design of a passive multiple trailer system for indoor service robots [1] The equations of motion for each trailer are given by:

$$v_i = v_{i-1}\cos\phi_i + D_{i-1}\dot{\theta}_{i-1}\sin\phi_i \tag{1}$$

$$\dot{\theta}_i = \frac{1}{L_i}\left(v_{i-1}\sin\phi_i + D_{i-1}\dot{\theta}_{i-1}\cos\phi_i\right) \tag{2}$$

$$\phi_i = \theta_{i-1} - \theta_i \tag{3}$$

where v and $\dot{\theta}$ are the translational and rotational velocity of each trailer (in the body axis system), L and D are the lengths of the front and rear moment arms, and the subscripts $_i$ and $_{i-1}$ refer to any trailer and the one in front of it.

The label file is listed in Section 1.2 (on page 5) and the subsystems are listed in Section 1.3 (on page 6).

[1] Jaehyoung Lee, Woojin Chung, Munsang Kim, Chongwon Lee and Jeabok-Song, A passive multiple trailer system for indoor service robots, *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Maui, Hawaii, USA, 2001

1.1 Trajectory

The trajectory of each of the trailer hooks in response to a constant forward and rotational velocity is shown in figure 2 (on page 4). The plot was generated with

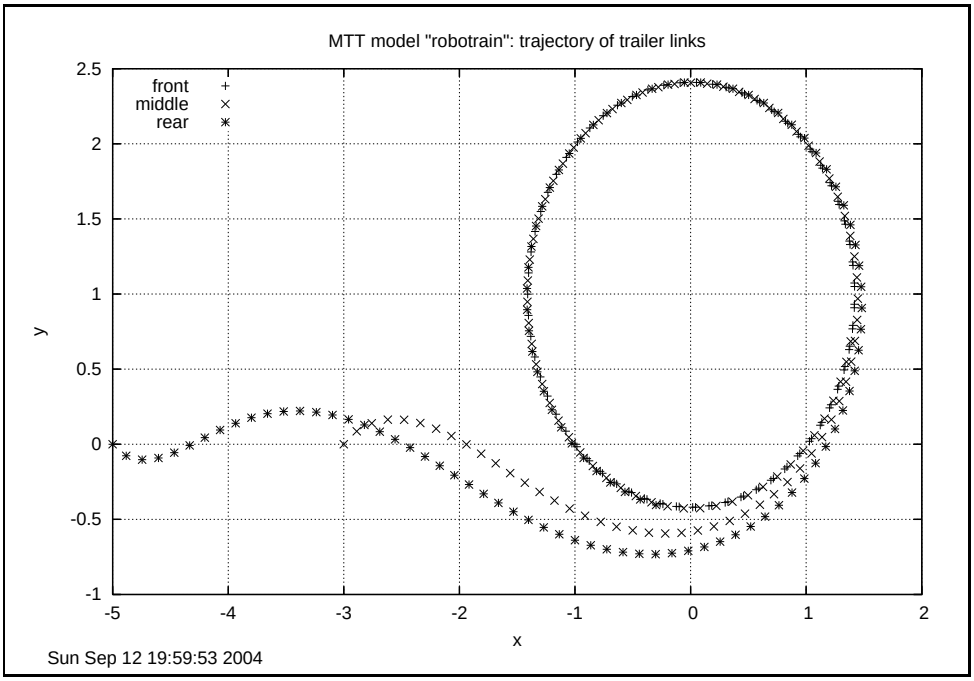

Figure 2: Trajectory of trailer links

the following script:

```
#! /bin/sh
mtt -cc -no-reduce robotrain odeso ps
cd MTT_work
#set data style lines
cat <<EOF | gnuplot
set timestamp
set key left
set xlabel 'x'
set ylabel 'y'
set grid
set title 'MTT model "robotrain": trajectory of trailer links'
plot   'robotrain_odes.dat2' using 4:5   title 'front'
replot 'robotrain_odes.dat2' using 8:9   title 'middle'
replot 'robotrain_odes.dat2' using 12:13 title 'rear'
set terminal postscript eps
set output "trajectory.ps"
replot
```

```
EOF
cd ..
cp MTT_work/trajectory.ps .
gv trajectory.ps
```

1.2 Summary information

System robotrain:

Interface information:

Parameter $1 represents actual parameter **D**

Parameter $2 represents actual parameter **L**

Variable declarations:

This component has no PAR declarations

Units declarations:

This component has no UNITs declarations

The label file: robotrain_lbl.txt

```
#SUMMARY robotrain
#DESCRIPTION Detailed description here

## System robotrain, representation lbl, language txt
## File robotrain_lbl.txt
## Generated by MTT on Sun Sep 12 19:26:13 BST 2004

  ########################################
  ##### Model Transformation Tools #####
  ########################################

## Port aliases

## Argument aliases
#ALIAS $1 D
#ALIAS $2 L
```

```
## Each line should be of one of the following forms:
##      a comment (ie starting with #)
##      component-name cr_name arg1,arg2,..argn
##      blank

## ---- Component labels ----

## Component type SS
Fx SS external,external
Fy SS external,external

## Component type tractor
tractor lin D

## Component type trailer
front lin D;L
middle lin D;L
rear lin D;L
```

1.3 Subsystems

1. tractor (1)
   (a) INTF: flow integrator (0)
   (b) Sf Simple flow source (0)
   (c) rotation (0)
2. trailer (0)
   (a) De Simple effort detector (1)
   (b) Df Simple flow detector (0)
   (c) INTF: flow integrator (0)
   (d) rotation (0)

1.4 rotation

The acausal bond graph of system **rotation** is displayed in Figure 3 (on page 7) . The label file is listed in Section 1.4.2 (on page 11) and the subsystems are listed in Section 1.4.3 (on page 13).

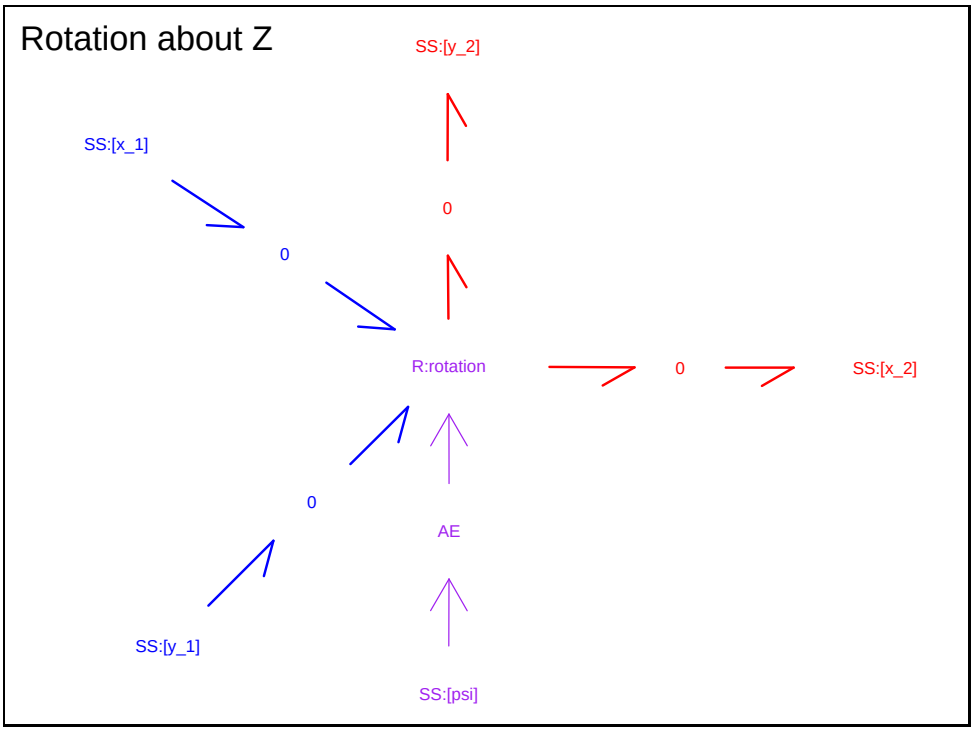

Figure 3: System **rotation**: acausal bond graph

1.4.1 rotate_z constitutive relationship

The rotation between body and Earth axes is achieved using the "rotate_z" constitutive relationship which is implemented as a Reduce file (.cr) and a Perl module (.pm) to permit the model to be built with or without MTT's -no-reduce option.

rotate_z.cr

```
operator rotate_z;

let rotate_z (R,~out_causality,1,
   ~x1,~causality1,1,
   ~y1,~causality2,2,
   ~x2,~causality3,3,
   ~y2,~causality4,4,
   ~psi,~causality5,5) =>% x1
      x2*cos(psi)-y2*sin(psi);

let rotate_z (R,~out_causality,2,
   ~x1,~causality1,1,
   ~y1,~causality2,2,
   ~x2,~causality3,3,
   ~y2,~causality4,4,
   ~psi,~causality5,5) =>% y1
      x2*sin(psi)+y2*sin(psi);

let rotate_z (R,~out_causality,3,
   ~x1,~causality1,1,
   ~y1,~causality2,2,
   ~x2,~causality3,3,
   ~y2,~causality4,4,
   ~psi,~causality5,5) =>% x2
      x1*cos(psi)+y1*sin(psi);

let rotate_z (R,~out_causality,4,
   ~x1,~causality1,1,
   ~y1,~causality2,2,
   ~x2,~causality3,3,
   ~y2,~causality4,4,
   ~psi,~causality5,5) =>% y2
      -x1*sin(psi)+y1*cos(psi);

let rotate_z (R,~out_causality,5,
   ~x1,~causality1,1,
   ~y1,~causality2,2,
   ~x2,~causality3,3,
   ~y2,~causality4,4,
   ~psi,~causality5,5) =>% x1
      0;
```

```
;end;
```

rotate_z.pm

```
#-------------------------------------------------------------------------------
# Model Transformation Tools
#-------------------------------------------------------------------------------


package rotate_z;

#-------------------------------------------------------------------------------
# rotation of x-y plane about z
#-------------------------------------------------------------------------------

use strict;
use warnings;

#-------------------------------------------------------------------------------
# standard module header (see perlmod for explanation)
#-------------------------------------------------------------------------------
BEGIN {
    use Exporter   ();
    our ($VERSION, @ISA, @EXPORT, @EXPORT_OK, %EXPORT_TAGS);

    $VERSION     = 1.00;

    @ISA         = qw(Exporter);
    @EXPORT      = qw(&rotate_z); # CR name
    %EXPORT_TAGS = ( );
}

#-------------------------------------------------------------------------------
# declaration of specific component implementations
#-------------------------------------------------------------------------------
sub rotate_z_R(@); # R

#-------------------------------------------------------------------------------
# main function: selects which subfunction to call
#-------------------------------------------------------------------------------
sub rotate_z (@) {

    my $retval;

    $_ = $_[0];

    s/\((.*)\)/$1/; # strip brackets
    my @args = split (/,/); # split arguments
```

```
    $_ = $args[0]; # get component type

    # select rule to use
    if (/^R|r$/) { $retval = rotate_z_R (@args); }

    # if a substitution has been made ($retval)
    if ($retval)
    {
return $retval; # return substituted expression
    }
    else # return nothing
    {
return;
    }
}

#-------------------------------------------------------------------------------
# R
#-------------------------------------------------------------------------------
sub rotate_z_R (@) {

    my @args = @_;
    my $retval = '';

    if ($#args == 18-1)
    {
my ($component,
    $out_causality,
    $out_port,
    $x1,
    $causality1,
    $port1,
    $y1,
    $causality2,
    $port2,
    $x2,
    $causality3,
    $port3,
    $y2,
    $causality4,
    $port4,
    $psi,
    $causality5,
    $port5) = @args;

# [ x2 ]   [ +cos(psi) +sin(psi) 0 ] [ x1 ]
# [ y2 ] = [ -sin(psi) +cos(psi) 0 ] [ y1 ]
# [ z2 ]   [    0          0     1 ] [ z1 ]
```

```
# for reverse transformation (x2->x1) use psi=-psi
# note that cos(-psi)=cos(psi) and sin(-psi)=-sin(psi)

if ($out_port == 1) # x1
{
    $retval = "(($x2)*(+cos($psi))+($y2)*(-sin($psi)))";
}
elsif ($out_port == 2) # y1
{
    $retval = "(($x2)*(+sin($psi))+($y2)*(+cos($psi)))";
}
elsif ($out_port == 3) # x2
{
    $retval = "(($x1)*(+cos($psi))+($y1)*(+sin($psi)))";
}
elsif ($out_port == 4) # y2
{
    $retval = "(($x1)*(-sin($psi))+($y1)*(+cos($psi)))";
}
elsif ($out_port == 5)
{
    $retval = "(0)";
}
    }

    if ($retval)
    {
return $retval;
    }
    else
    {
return;
    }
}

#--------------------------------------------------------------------------------
1; # return true
```

1.4.2 Summary information

System rotation:

Interface information:

Port in represents actual port **x_earth,y_earth**

Port out represents actual port **x_body,y_body**

Variable declarations:

This component has no PAR declarations

Units declarations:

This component has no UNITs declarations

The label file: rotation_lbl.txt

```
#SUMMARY rotation
#DESCRIPTION Detailed description here

## System rotation, representation lbl, language txt
## File rotation_lbl.txt
## Generated by MTT on Tue Sep  7 16:59:01 BST 2004

  ########################################
  ##### Model Transformation Tools #####
  ########################################

## Port aliases
#ALIAS in x_earth,y_earth
#ALIAS out x_body,y_body

## Argument aliases

## Each line should be of one of the following forms:
##      a comment (ie starting with #)
##      component-name cr_name arg1,arg2,..argn
##      blank

## ---- Component labels ----

## Component type 0 (anonymous => default parameters)
# 0
# 0
# 0
# 0
```

```
# 0

## Component type AF (anonymous => default parameters)
# AF

## Component type R
rotation rotate_z

## Component type SS
[psi] SS external,external
[x_body] SS external,external
[x_earth] SS external,external
[y_body] SS external,external
[y_earth] SS external,external
```

1.4.3 Subsystems

1.5 tractor

The acausal bond graph of system **tractor** is displayed in Figure 4 (on page 14) . The label file is listed in Section 1.5.1 (on page 13) and the subsystems are listed in Section 1.5.2 (on page 16).

1.5.1 Summary information

System tractor:

Interface information:

Parameter $1 represents actual parameter **D**

Port out represents actual port **x2,y2**

Variable declarations:

This component has no PAR declarations

Units declarations:

This component has no UNITs declarations

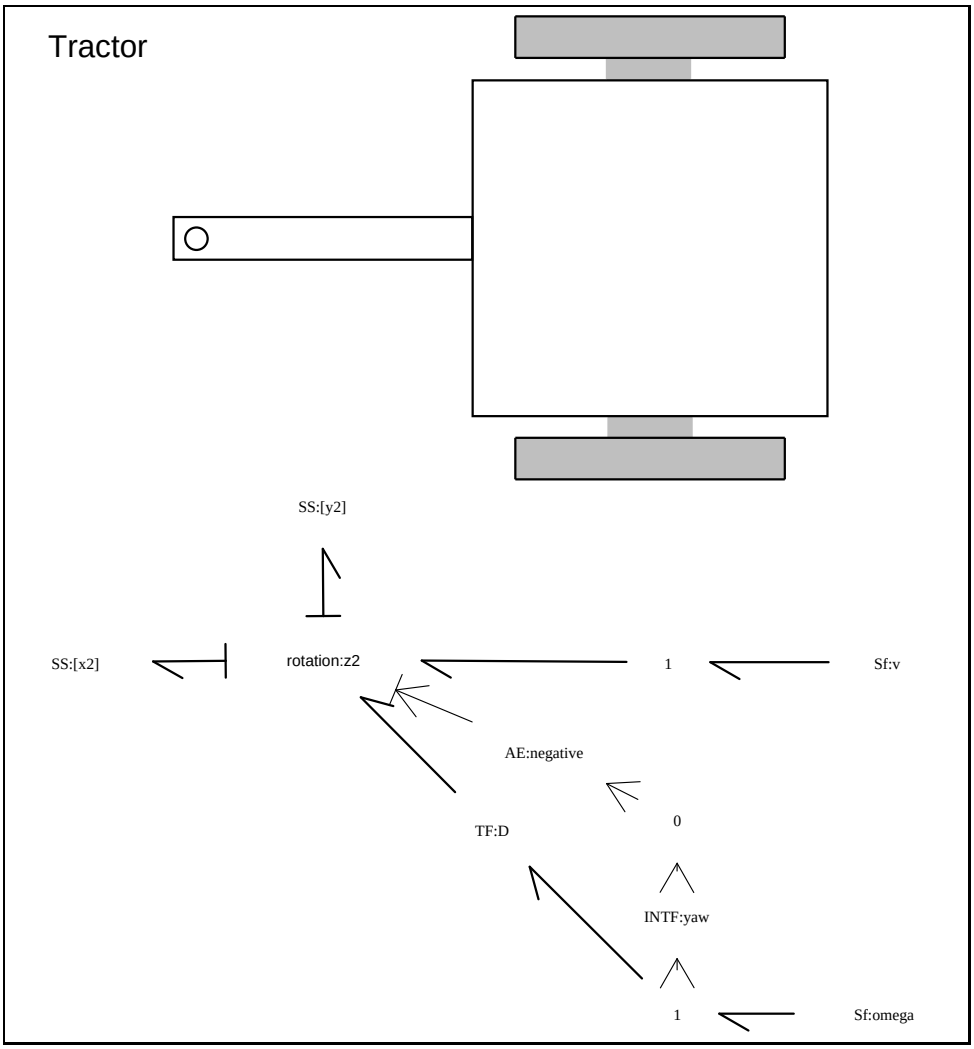

Figure 4: System **tractor**: acausal bond graph

The label file: tractor_lbl.txt

```
#SUMMARY tractor
#DESCRIPTION Detailed description here

## System tractor, representation lbl, language txt
## File tractor_lbl.txt
## Generated by MTT on Wed Sep  8 00:42:45 BST 2004

  ########################################
  ##### Model Transformation Tools #####
  ########################################

## Port aliases
#ALIAS out x2,y2

## Argument aliases
#ALIAS $1 D

## Each line should be of one of the following forms:
##      a comment (ie starting with #)
##      component-name cr_name arg1,arg2,..argn
##      blank

## ---- Component labels ----

## Component type 1 (anonymous => default parameters)
# 1
# 1
# 1

## Component type AE
negative  lin effort,-1

## Component type INTF
yaw   none

## Component type SS
[x2]   SS external,external
[y2]   SS external,external
```

```
## Component type Sf
omega   SS external
v   SS external

## Component type TF
D   lin flow,-D

## Component type rotation
z2   rotate_z
```

1.5.2 Subsystems

1. INTF: flow integrator (0)
2. Sf Simple flow source (0)
3. rotation (0)

1.6 trailer

The acausal bond graph of system **trailer** is displayed in Figure 5 (on page 17) . The label file is listed in Section 1.6.1 (on page 16) and the subsystems are listed in Section 1.6.2 (on page 19).

1.6.1 Summary information

System trailer:

Interface information:

Parameter $1 represents actual parameter **D**

Parameter $2 represents actual parameter **L**

Port in represents actual port **x1,y1**

Port out represents actual port **x2,y2**

Variable declarations:

This component has no PAR declarations

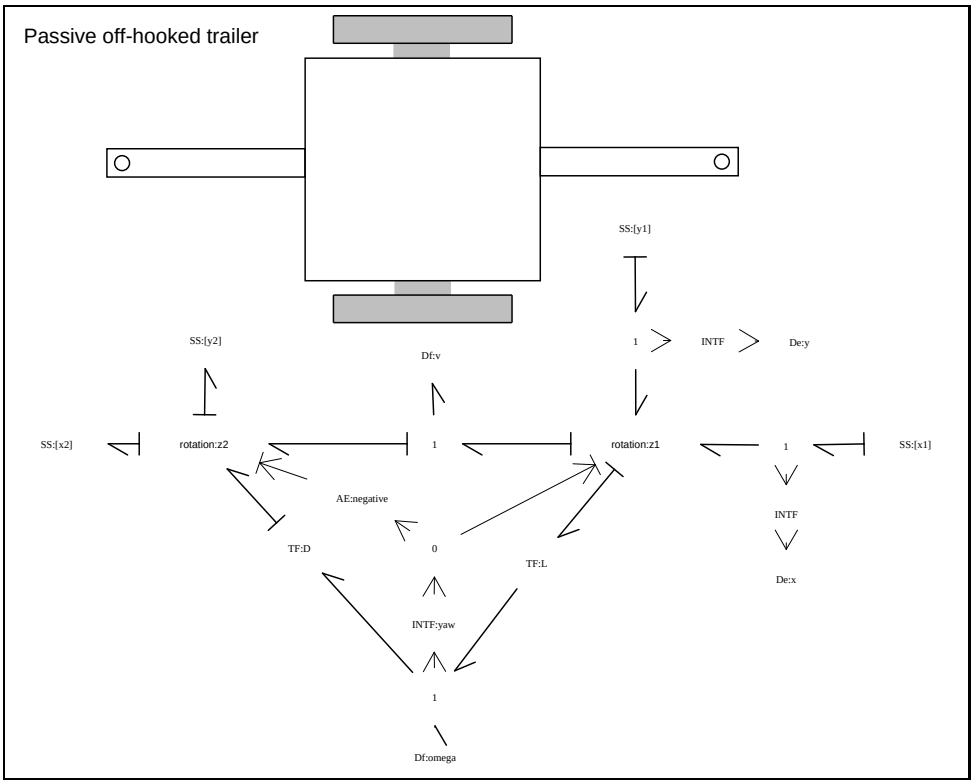

Figure 5: System **trailer**: acausal bond graph

Units declarations:

This component has no UNITs declarations

The label file: trailer_lbl.txt

```
#SUMMARY trailer
#DESCRIPTION Detailed description here

## System trailer, representation lbl, language txt
## File trailer_lbl.txt
## Generated by MTT on Wed Sep  8 00:41:53 BST 2004

  ########################################
  ##### Model Transformation Tools #####
  ########################################

## Port aliases
#ALIAS in x1,y1
#ALIAS out x2,y2

## Argument aliases
#ALIAS $1 D
#ALIAS $2 L

## Each line should be of one of the following forms:
##      a comment (ie starting with #)
##      component-name cr_name arg1,arg2,..argn
##      blank

## ---- Component labels ----

## Component type 1 (anonymous => default parameters)
# 1
# 1
# 1

## Component type AE
negative  lin effort,-1

## Component type De
```

```
x   SS external
y   SS external

## Component type Df
omega   SS external
v   SS external

## Component type INTF
yaw   none

## Component type SS
[x1]   SS external,external
[x2]   SS external,external
[y1]   SS external,external
[y2]   SS external,external

## Component type TF
D   lin flow,-D
L   lin effort,L

## Component type rotation
z1   rotate_z
z2   rotate_z
```

1.6.2 Subsystems

1. De Simple effort detector (1)
2. Df Simple flow detector (0)
3. INTF: flow integrator (0)
4. rotation (0)

2 robotrain_struc.tex

MTT command:

```
mtt robotrain struc tex
```

List of inputs for system robotrain			
	Component	System	Repetition
1	Fx	robotrain__Fx	1
2	Fy	robotrain__Fy	1
3	u	robotrain__tractor__omega__u	1
4	u	robotrain__tractor__v__u	1

List of outputs for system robotrain			
	Component	System	Repetition
1	Fx	robotrain__Fx	1
2	Fy	robotrain__Fy	1
3	y	robotrain__front__x__y	1
4	y	robotrain__front__y__y	1
5	y	robotrain__front__omega__y	1
6	y	robotrain__front__v__y	1
7	y	robotrain__middle__x__y	1
8	y	robotrain__middle__y__y	1
9	y	robotrain__middle__omega__y	1
10	y	robotrain__middle__v__y	1
11	y	robotrain__rear__x__y	1
12	y	robotrain__rear__y__y	1
13	y	robotrain__rear__omega__y	1
14	y	robotrain__rear__v__y	1

List of states for system robotrain			
	Component	System	Repetition
1	mttC	robotrain__tractor__yaw__mttC	1
2	mttC	robotrain__front__mttINTF__mttC	1
3	mttC	robotrain__front__mttINTF_2__mttC	1
4	mttC	robotrain__front__yaw__mttC	1
5	mttC	robotrain__middle__mttINTF__mttC	1
6	mttC	robotrain__middle__mttINTF_2__mttC	1
7	mttC	robotrain__middle__yaw__mttC	1
8	mttC	robotrain__rear__mttINTF__mttC	1
9	mttC	robotrain__rear__mttINTF_2__mttC	1
10	mttC	robotrain__rear__yaw__mttC	1

3 robotrain_sympar.tex

MTT command:

```
mtt robotrain sympar tex
```

Parameter	System
D	robotrain
L	robotrain

Table 1: Parameters

4 robotrain_state.txt

MTT command:

```
mtt robotrain state txt
```

```
## -*-octave-*- Put Emacs into octave-mode ##

##
## System robotrain, representation state, language txt;
## File robotrain_state.txt;
## Generated by MTT on Sun Sep 12 18:46:40 BST 2004;

robotrain__tractor__yaw__mttC = 0.0;
robotrain__front__mttINTF__mttC = -D;
robotrain__front__mttINTF_2__mttC = 0.0;
robotrain__front__yaw__mttC = 0.0;
robotrain__middle__mttINTF__mttC = -(D+L+D);
robotrain__middle__mttINTF_2__mttC = 0.0;
robotrain__middle__yaw__mttC = 0.0;
robotrain__rear__mttINTF__mttC = -(D+L+D+L+D);
robotrain__rear__mttINTF_2__mttC = 0.0;
robotrain__rear__yaw__mttC = 0.0;
```

5 robotrain_input.txt

MTT command:

```
mtt robotrain input txt
```

```
## -*-octave-*- Put Emacs into octave-mode ##

##
## System robotrain, representation input, language txt;
## File robotrain_input.txt;
## Generated by MTT on Fri Sep 10 00:39:34 BST 2004;



robotrain__Fx = 0.0;
robotrain__Fy = 0.0;
robotrain__tractor__omega__u = 1.0;
robotrain__tractor__v__u = 1.0;
```