Demonstration Model of *fuzzy*TECH® Implementation on M68HC12

by Philip Drake and Jim Sibigtroth of Freescale, Austin; and Constantin von Altrock and Ralph Konigbauer of Inform Software Corp., Chicago (8/96)

INTRODUCTION

This paper presents a demonstration model that illustrates the use of fuzzy logic on a Freescale microcontroller (MCU). The M68HC12 MCU was introduced in mid-1996 by Freescale as an upgrade to the M68HC11 MCU, one of the most widely used microcontrollers in the world. The M68HC12 has been enhanced in a number of ways over the M68HC11, and is the world's first standard MCU that includes a complete fuzzy logic instruction set. The "background debug™ mode" of the M68HC12 supports the real-time remote cross debugging (RTRCD) without interfering with the MCU's operation. To support systems design, Inform Software Corp. and Freescale created the *fuzzy*TECH MCU-68HC12 Edition, which supports both the M68HC12's fuzzy logic instruction set and the M68HC12 background debug mode. To demonstrate the use of both the fuzzy logic instruction set and the background debug mode with the *fuzzy*TECH development system, Inform and Freescale have designed an autonomously guided tank as a demonstration model. This paper discusses the tank's fuzzy logic controller design, as well as the *fuzzy*TECH implementation on the M68HC12 MCU.

Freescale M68HC12 MCU

This new 16-bit microcontroller family includes four fuzzy logic instructions in addition to the memory and on-chip peripheral functions one would expect in a general-purpose microcontroller. The fuzzy logic instructions use existing CPU logic to perform computations including addition, subtraction, multiplication, multiply-and-accumulate, and comparisons, so the speed and efficiency of fuzzy logic programs is greatly improved without increasing the cost of the MCU. A fuzzy inference kernel on the M68HC12 takes one fifth as much code space and executes more than ten times faster than an M68HC11 general-purpose MCU.

The MEM instruction computes one fuzzy input based on a trapezoidal membership function in 625 ns at 8 MHz. This instruction automatically stores the computed result and updates two pointers so several membership functions can be evaluated without executing any other instructions between labels of a system input.

REV performs unweighted min-max rule evaluation on a complete rule list for a fuzzy logic system. REVW optionally allows the user to specify a per-rule weighting factor between zero and one. Since rule lists may include any number of rules, these instructions allow interrupts.

WAV computes sums-of-products and sums-of-weights needed for weighted average defuzzification. These sums are left in the correct CPU registers so that an EDIV instruction immediately after WAV completes the final division to get the weighted average result. Since WAV processes all labels of a system output, it is designed to allow interrupts.

The new background debug interface uses a single dedicated MCU pin for bidirectional communication. Memory can be read or written while the MCU is executing an application program. The CPU can also be stopped to allow access to CPU registers and to allow execution of TRACE and GO commands.

*fuzzy*TECH MCU-68HC12 Edition

*fuzzy*TECH is a family of complete software development systems based on fuzzy logic and NeuroFuzzy technologies [1]. For MCU implementations, *fuzzy*TECH offers assembly code generation to ensure maximum computational performance using as little of the available memory resources as possible.

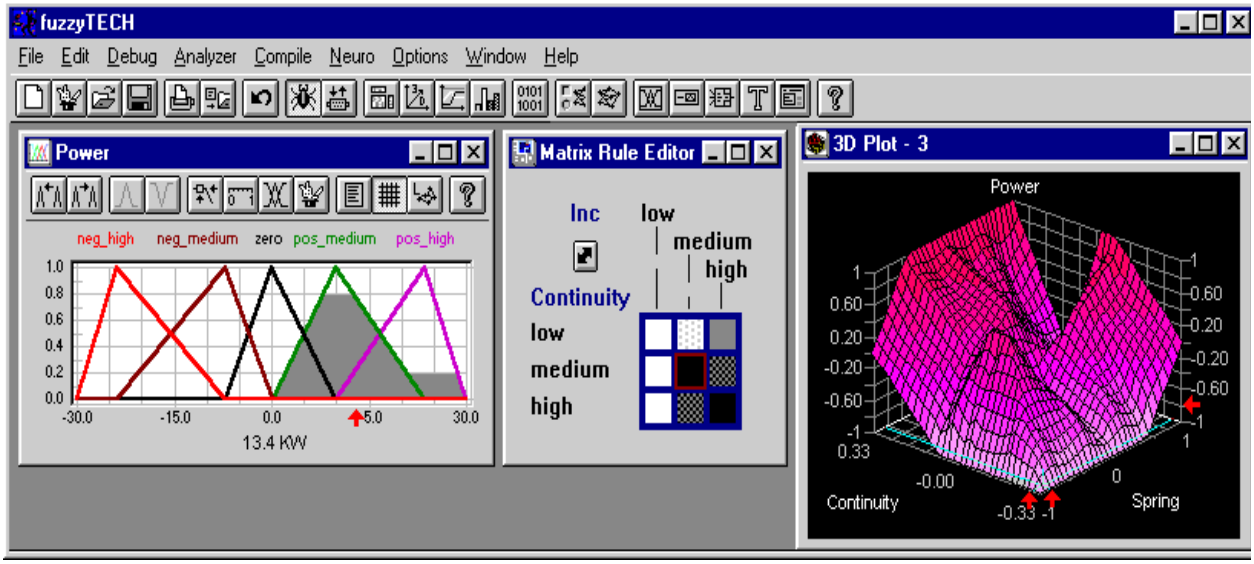

Figure 1 Complete Graphical Development and Debugging Environment

The *fuzzy*TECH MCU-68HC12 Edition is a version of *fuzzy*TECH dedicated to the Freescale M68HC12 family of microcontrollers. In close cooperation with Freescale, Inform's fuzzy logic experts have generated an M68HC12 code generator that optimally utilizes the fuzzy logic functions of the M68HC12 ALU.

In many fuzzy logic applications, an implemented control strategy can best be optimized on the running process. *fuzzy*TECH supports this "on-the-fly" optimization of a running fuzzy logic system by its RTRCD-68HC12 Module, an add-on to the *fuzzy*TECH MCU-68HC12 Edition. The RTRCD Module uses the background debug mode of the M68HC12 MCU to:

1. Visualize the complete fuzzy logic inference in real time using *fuzzy*TECH's analyzers and dynamic editors (See Figure 4).

2. Carry out any modification done in *fuzzy*TECH on the M68HC12 in real time without interfering with the running process. (See Figure 4.)

The Tank Demonstration Model

To demonstrate the implementation and RTRCD optimization of a fuzzy logic system using the M68HC12 and *fuzzy*TECH, Freescale and Inform designed a tank model. This tank model guides itself following a light source using three optical sensors. The tank uses the following periphery:

- light sensor left (input)
- light sensor right (input)
- light sensor back (input)
- motor power left chain (output)
- motor power right chain (output)

Figure 2 shows the scheme of the tank demonstration model. The three optical sensors are used by the tank to detect the direction of the light source. The sensors are simple light dependent resistors (LDR) coupled to a current source as a voltage divider. The resulting voltage is fed into the analog inputs of the M68HC12 MCU.

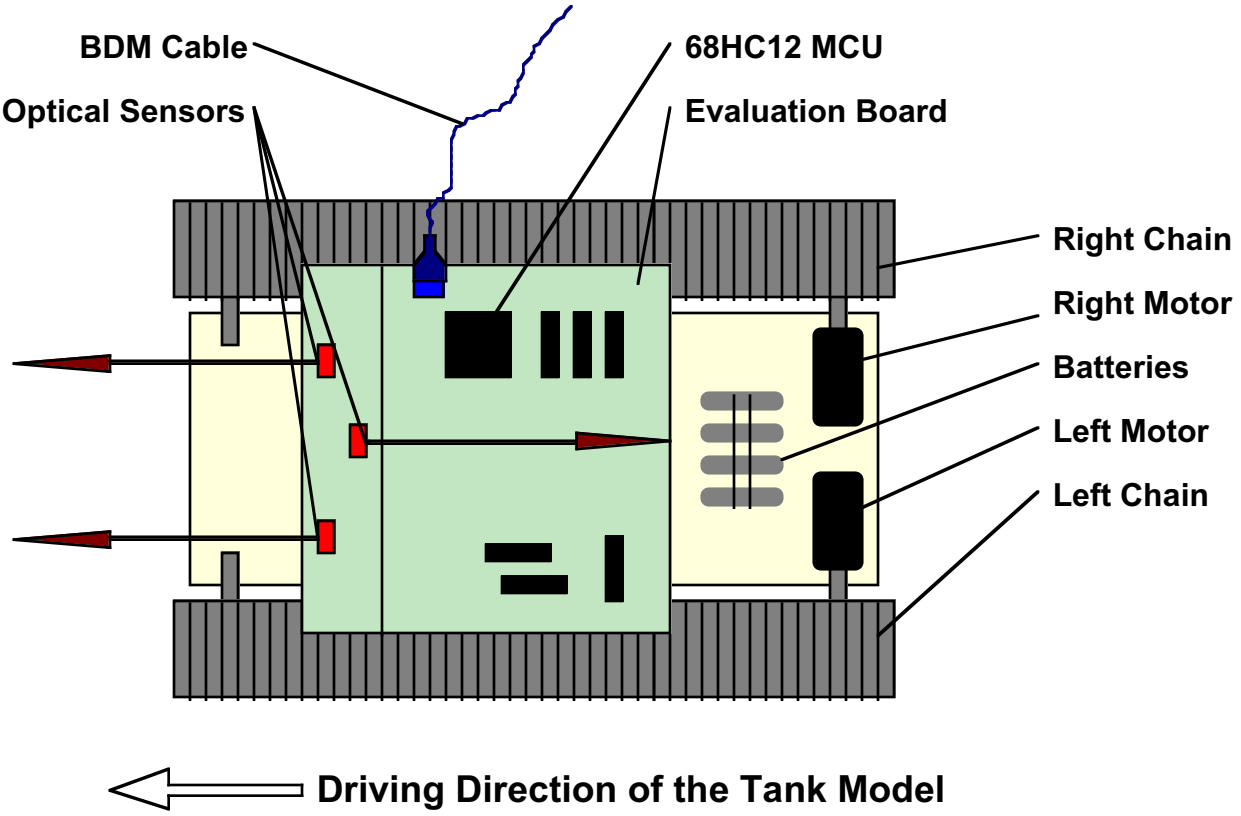

Figure 2 Scheme of the Tank Demonstration Model

To move and turn the tank, two chains are powered by independent motors. The motor's power and direction are controlled by pulse width modulation (PWM) outputs of the M68HC12 MCU via a power stage.

After the mechanical design of the tank was completed and the electrical periphery (sensor signal conditioning and power stage) was completed, the tank's control software was developed. The software of the tank was written in C code using the COSMIC toolkit. The main routine consists of a loop which is called every 10 milliseconds. Within this loop, first the sensor signals are fetched, then the fuzzy logic computation is called, and finally the output of the fuzzy logic computation is fed into the PWM registers.

The Fuzzy Logic Controller

The fuzzy logic computation routine is completely generated by the *fuzzy*TECH MCU-68HC12 Edition. The C header of the generated function is:

```
FLAGS TankCntrl(int SensorLeft, int SensorRight, int
      SensorBack, * int PowerLeft, * int PowerRight);
```

Figure 3 shows the graphical design of the fuzzy logic tank controller in the *fuzzy*TECH development system. The upper left window shows the structure of the fuzzy logic controller. The three inputs of the fuzzy logic system, SensorBack, SensorLeft, and SensorRight, are the light intensities measured by the optical sensors. They feed into the rule block that contains the linguistic control strategy. The two output variables of the fuzzy logic system, PowerLeft and PowerRight, feed directly into the PWM registers of the M68HC12 MCU. A complete listing of the FTL code for this application is attached CARCNTRL.FTL (see page 6).

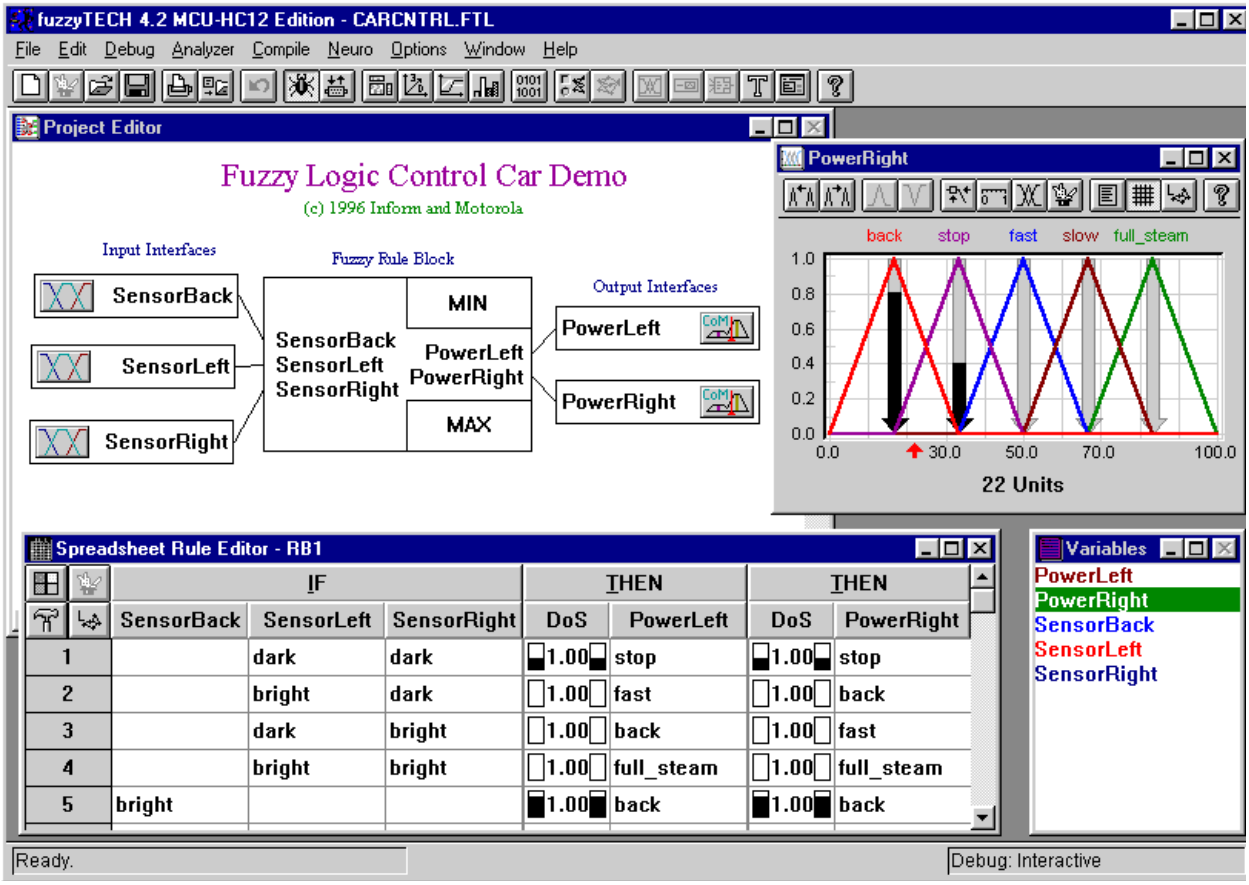

Figure 3 Fuzzy Logic Controller Development Using *fuzzy*TECH

The upper right window shows the membership functions and the defuzzification process of the output variable “PowerRight”. The lower window shows the five fuzzy control rules of the system in a spreadsheet-type representation. *fuzzy*TECH produces the entire system as ready-to-use code for the M68HC12 by selecting the “Compile” menu. For a complete development guide for fuzzy logic systems and sample software, refer to [4].

Real-Time Remote Cross Debugging

Fuzzy logic provides designers with a technique that can directly affect the outcome of a design by adjusting the inputs and rules in real time.

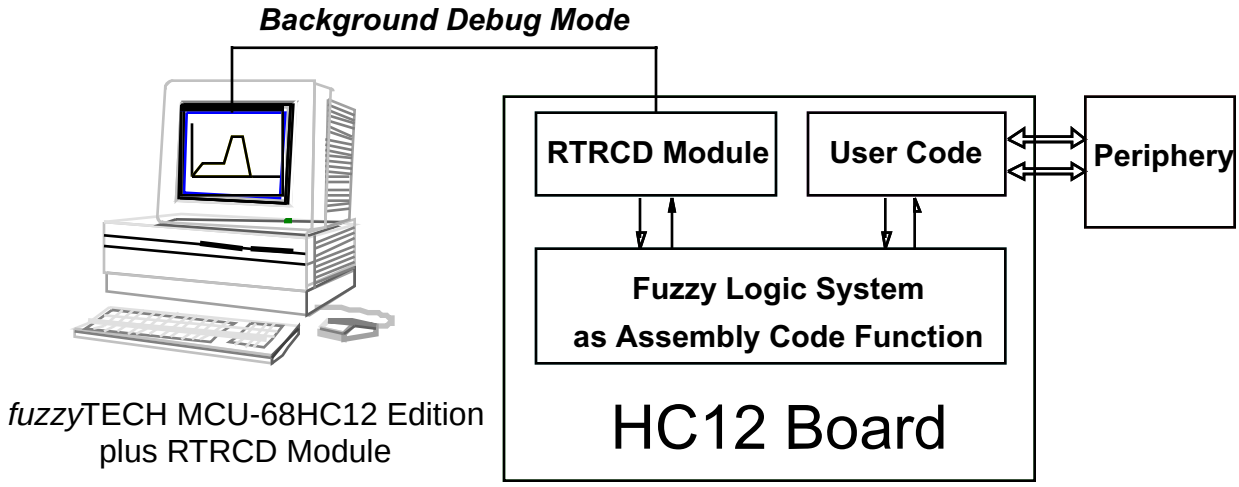

Figure 4 Real-Time Remote Cross Debugging

*fuzzy*TECH supports this development “on-the-fly” with its RTRCD add-on module. The development steps are:

- Design the fuzzy logic controller “off-line”, that is, using the graphical design tools and simulation environment of *fuzzy*TECH.
- Compile the fuzzy logic system to M68HC12 code using *fuzzy*TECH.
- Link the RTRCD driver provided with *fuzzy*TECH and implement the code on the M68HC12 board.
- Run the tank. At any time, the M68HC12 board may be connected to the PC running *fuzzy*TECH using the background debug interface cable.
- In *fuzzy*TECH’s Monitor Mode, you can open any editor and any analyzer of *fuzzy*TECH to view and analyze the fuzzy logic controller while running.
- In *fuzzy*TECH’s Online Mode, you may also modify the fuzzy logic controller while running. Any change takes effect instantly on the running tank controller.

Conclusion

While the control objective implemented in the tank is not overly complex and could have been solved using different techniques, its fuzzy logic solution, involving only five fuzzy rules, is strikingly simple and transparent. Even more, adding more sensors and drivers in the fuzzy logic controller for more complex operations can be done in a very short time.

The new Freescale M68HC12 MCU is the world’s first standard MCU with an embedded fuzzy logic instruction set. Using this instruction set, the *fuzzy*TECH MCU-68HC12 Edition can implement highly complex fuzzy logic systems using only a few hundred bytes of ROM that compute in less than a millisecond. Such high performance renders the use of dedicated fuzzy coprocessors unnecessary in most applications. Because the fuzzy logic instruction set of the M68HC12 does not come with any price tag attached, the M68HC12 will enable the use of fuzzy logic in mass-market high-speed applications, such as car engine control, anti-lock brakes, traction control, inter-vehicle dynamics control, hard disk drive control, servo motor control, and cellular phones.

The complete FTL source code of the tank controller and a demo version of *fuzzy*TECH to view, test and analyze the fuzzy logic system can be downloaded from the Inform web server [1]. On this web server you also find additional application notes on fuzzy logic in automotive engineering and other application areas. For a complete introduction to fuzzy logic design, refer to [4].

Literature

[1] *fuzzy*TECH Home Page at: “http://www.inform-ac.com”

Freescale M68HC12 Home Page at: “http://www.mot.com/m68hc12”

[2] Freescale, *CPU12 Reference Manual*, CPU12RM/AD available from www.

[3] Freescale, *MC68HC812A4 Technical Summary*, MC68HC812A4TS/D available from www.

[4] von Altrock, C., *Fuzzy Logic and NeuroFuzzy Applications Explained*, Prentice Hall,

ISBN 0-13-368456-2 (1995).

[5] *fuzzy*TECH 4.0 Manual

For a complete listing of Advanced MCU Literature, order document BR1116/D available from www.

CARCNTRL.FTL

```
PROJECT {
  NAME = CARCNTRL.FTL;
  TITLE =CARCNTRL.FTL;
  DATEFORMAT = DD.MM.YYYY;
  LASTCHANGE = 29.08.1996;
  CREATED    = 21.08.1996;
  SHELL = MCU_HC12;
  SHELLOPTIONS {
    ONLINE_REFRESHTIME = 55;
    ONLINE_TIMEOUTCOUNT = 1100;
    ONLINE_PORT = SERIAL;
    ONLINE_CODE = ON;
    COMMENTS    = ON;
    TRACE_BUFFER = (OFF, PAR(255));
    FTL_BUFFER = (OFF, PAR(1));
    PASSWORD   = OFF;
    PUBLIC_IO = ON;
    FAST_CMBF = OFF;
    FAST_COA  = OFF;
    FILE_CODE = OFF;
    BTYPE = 8_BIT;
    C_TYPE = ANSI;
  } /* SHELLOPTIONS */
  MODEL {
    VARIABLE_SECTION {
      LVAR {
        NAME    = PowerLeft;
        BASEVAR = Units;
        LVRANGE = MIN(0.000000), MAX(100.000000),
                  MINDEF(0), MAXDEF(255),
                  DEFAULT_OUTPUT(50.000000);
        RESOLUTION = XGRID(0.500000), YGRID(1.000000),
                     SHOWGRID (ON), SNAPTOGRID(ON);
        COLOR = RED (128), GREEN (0), BLUE (0);
        TERM {
          TERMNAME = back;
          POINTS = (0.000000, 0.000000),
                   (16.500000, 1.000000),
                   (33.500000, 0.000000),
                   (100.000000, 0.000000);
          SHAPE = LINEAR;
          COLOR = RED (255), GREEN (0), BLUE (0);
        }
        TERM {
          TERMNAME = stop;
```

```
      POINTS = (0.000000, 0.000000),
               (16.500000, 0.000000),
               (33.500000, 1.000000),
               (50.000000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (255), BLUE (0);
    }
    TERM {
      TERMNAME = slow;
      POINTS = (0.000000, 0.000000),
               (50.000000, 0.000000),
               (66.500000, 1.000000),
               (83.500000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (128), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = fast;
      POINTS = (0.000000, 0.000000),
               (33.500000, 0.000000),
               (50.000000, 1.000000),
               (66.500000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (255);
    }
    TERM {
      TERMNAME = full_steam;
      POINTS = (0.000000, 0.000000),
               (66.500000, 0.000000),
               (83.500000, 1.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (128), BLUE (0);
    }
  }  /* LVAR */
  LVAR {
    NAME    = PowerRight;
    BASEVAR = Units;
    LVRANGE = MIN(0.000000), MAX(100.000000),
              MINDEF(0), MAXDEF(255),
              DEFAULT_OUTPUT(50.000000);
    RESOLUTION = XGRID(0.500000), YGRID(1.000000),
                 SHOWGRID (ON), SNAPTOGRID(ON);
    COLOR = RED (0), GREEN (128), BLUE (0);
    TERM {
      TERMNAME = back;
      POINTS = (0.000000, 0.000000),
               (16.500000, 1.000000),
               (33.500000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = stop;
      POINTS = (0.000000, 0.000000),
               (16.500000, 0.000000),
               (33.500000, 1.000000),
               (50.000000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (115), GREEN (0), BLUE (110);
    }
    TERM {
      TERMNAME = slow;
```

```
      POINTS = (0.000000, 0.000000),
               (50.000000, 0.000000),
               (66.500000, 1.000000),
               (83.500000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (128), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = fast;
      POINTS = (0.000000, 0.000000),
               (33.500000, 0.000000),
               (50.000000, 1.000000),
               (66.500000, 0.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (255);
    }
    TERM {
      TERMNAME = full_steam;
      POINTS = (0.000000, 0.000000),
               (66.500000, 0.000000),
               (83.500000, 1.000000),
               (100.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (128), BLUE (0);
    }
  }  /* LVAR */
  LVAR {
    NAME    = SensorBack;
    BASEVAR = Units;
    LVRANGE = MIN(0.000000), MAX(1.000000),
              MINDEF(0), MAXDEF(255),
              DEFAULT_OUTPUT(0.500000);
    RESOLUTION = XGRID(0.005000), YGRID(1.000000),
                 SHOWGRID (ON), SNAPTOGRID(ON);
    COLOR = RED (0), GREEN (0), BLUE (255);
    TERM {
      TERMNAME = dark;
      POINTS = (0.000000, 1.000000),
               (0.250000, 1.000000),
               (0.500000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = medium;
      POINTS = (0.000000, 0.000000),
               (0.250000, 0.000000),
               (0.500000, 1.000000),
               (0.750000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (255), BLUE (0);
    }
    TERM {
      TERMNAME = bright;
      POINTS = (0.000000, 0.000000),
               (0.500000, 0.000000),
               (0.750000, 1.000000),
               (1.000000, 1.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (255);
    }
  }  /* LVAR */
  LVAR {
    NAME    = SensorLeft;
```

```
    BASEVAR = Units;
    LVRANGE = MIN(0.000000), MAX(1.000000),
              MINDEF(0), MAXDEF(255),
              DEFAULT_OUTPUT(0.500000);
    RESOLUTION = XGRID(0.005000), YGRID(1.000000),
                 SHOWGRID (ON), SNAPTOGRID(ON);
    COLOR = RED (255), GREEN (0), BLUE (0);
    TERM {
      TERMNAME = dark;
      POINTS = (0.000000, 1.000000),
               (0.250000, 1.000000),
               (0.500000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = medium;
      POINTS = (0.000000, 0.000000),
               (0.250000, 0.000000),
               (0.500000, 1.000000),
               (0.750000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (255), BLUE (0);
    }
    TERM {
      TERMNAME = bright;
      POINTS = (0.000000, 0.000000),
               (0.500000, 0.000000),
               (0.750000, 1.000000),
               (1.000000, 1.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (0), BLUE (255);
    }
  }  /* LVAR */
  LVAR {
    NAME    = SensorRight;
    BASEVAR = Units;
    LVRANGE = MIN(0.000000), MAX(1.000000),
              MINDEF(0), MAXDEF(255),
              DEFAULT_OUTPUT(0.500000);
    RESOLUTION = XGRID(0.005000), YGRID(1.000000),
                 SHOWGRID (ON), SNAPTOGRID(ON);
    COLOR = RED (0), GREEN (0), BLUE (116);
    TERM {
      TERMNAME = dark;
      POINTS = (0.000000, 1.000000),
               (0.250000, 1.000000),
               (0.500000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (255), GREEN (0), BLUE (0);
    }
    TERM {
      TERMNAME = medium;
      POINTS = (0.000000, 0.000000),
               (0.250000, 0.000000),
               (0.500000, 1.000000),
               (0.750000, 0.000000),
               (1.000000, 0.000000);
      SHAPE = LINEAR;
      COLOR = RED (0), GREEN (255), BLUE (0);
    }
    TERM {
      TERMNAME = bright;
      POINTS = (0.000000, 0.000000),
               (0.500000, 0.000000),
```

```
                (0.750000, 1.000000),
                (1.000000, 1.000000);
        SHAPE = LINEAR;
        COLOR = RED (0), GREEN (0), BLUE (255);
      }
    }  /* LVAR */
  }  /* VARIABLE_SECTION */

  OBJECT_SECTION {
    INTERFACE {
      INPUT = (SensorLeft, CMBF);
      POS = -222, -64;
    }
    INTERFACE {
      INPUT = (SensorRight, CMBF);
      POS = -223, -15;
    }
    INTERFACE {
      INPUT = (SensorBack, CMBF);
      POS = -220, -110;
    }
    INTERFACE {
      OUTPUT = (PowerLeft, COM);
      POS = 119, -89;
    }
    RULEBLOCK {
      NAME = RB1;
      INPUT = SensorBack, SensorLeft, SensorRight;
      OUTPUT = PowerLeft, PowerRight;
      AGGREGATION = (MIN_MAX, PAR (0.000000));
      RESULT_AGGR = MAX;
      POS = -71, -108;
      RULES {
        IF    SensorLeft = dark
          AND SensorRight = dark
        THEN  PowerLeft = stop   WITH 1.000;
        IF    SensorLeft = dark
          AND SensorRight = dark
        THEN  PowerRight = stop   WITH 1.000;
        IF    SensorLeft = bright
          AND SensorRight = dark
        THEN  PowerLeft = fast   WITH 1.000;
        IF    SensorLeft = bright
          AND SensorRight = dark
        THEN  PowerRight = back   WITH 1.000;
        IF    SensorLeft = dark
          AND SensorRight = bright
        THEN  PowerLeft = back   WITH 1.000;
        IF    SensorLeft = dark
          AND SensorRight = bright
        THEN  PowerRight = fast   WITH 1.000;
        IF    SensorLeft = bright
          AND SensorRight = bright
        THEN  PowerLeft = full_steam   WITH 1.000;
        IF    SensorLeft = bright
          AND SensorRight = bright
        THEN  PowerRight = full_steam   WITH 1.000;
        IF    SensorBack = bright
        THEN  PowerLeft = back   WITH 1.000;
        IF    SensorBack = bright
        THEN  PowerRight = back   WITH 1.000;
      }  /* RULES */
    }
    INTERFACE {
      OUTPUT = (PowerRight, COM);
      POS = 119, -41;
    }
    REMARK {
```

```
        TEXT = Fuzzy Logic Control Car Demo;
        POS = -102, -189;
        FONTSIZE = 24;
        COLOR = RED (160), GREEN (1), BLUE (175);
      }
      REMARK {
        TEXT = (c) 1996 Inform and Freescale;
        POS = -48, -164;
        FONTSIZE = 14;
        COLOR = RED (0), GREEN (145), BLUE (0);
      }
      REMARK {
        TEXT = Input Interfaces;
        POS = -179, -137;
        FONTSIZE = 14;
        COLOR = RED (0), GREEN (0), BLUE (97);
      }
      REMARK {
        TEXT = Fuzzy Rule Block;
        POS = -30, -131;
        FONTSIZE = 14;
        COLOR = RED (0), GREEN (0), BLUE (74);
      }
      REMARK {
        TEXT = Output Interfaces;
        POS = 140, -113;
        FONTSIZE = 14;
        COLOR = RED (0), GREEN (0), BLUE (85);
      }
    }  /* OBJECT_SECTION */
  }  /* MODEL */
}  /* PROJECT */

TERMINAL {
    BAUDRATE     = 9600;
    DATABITS     = 8;
    PARITY       = 78;
    STOPBITS     = 1;
    PROTOCOL     = NO;
    CONNECTION   = PORT2;
    INPUTBUFFER  = 1024;
    OUTPUTBUFFER = 1024;
}  /* TERMINAL */

ONLINE {
  TIMESTAMP = 1996082915574700;
}  /* ONLINE */
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor