FUZZYSTUDIO™ 2.0

W.A.R.P.2.0 DEVELOPMENT SYSTEM

ADVANCED DATA

Key Features

- Windows based graphic interface
  - Graphic Variable and Rule Editors
  - Compiler, Debugger, AFM interface
  - MATLAB®, ANSI C software emulators
  - FUZZYSTUDIO™ ADB2.0 manager
- FUZZYSTUDIO™ ADB2.0 development board
  - Up to 8 Input and up to 4 output variables
  - Up to 128 rules
  - W.A.R.P.2.0 programmer on board
  - ZEROPOWER programmer on board
  - RS232-C standard communication
  - Internal 8MHz or external, up to 40MHz, clock working frequency
  - Stand-Alone / Slave working modality
- AFM high level software tool

Introduction

FUZZYSTUDIO™ 2.0 is an easy and useful environment to design a Fuzzy Project for the fuzzy co-processor W.A.R.P.2.0

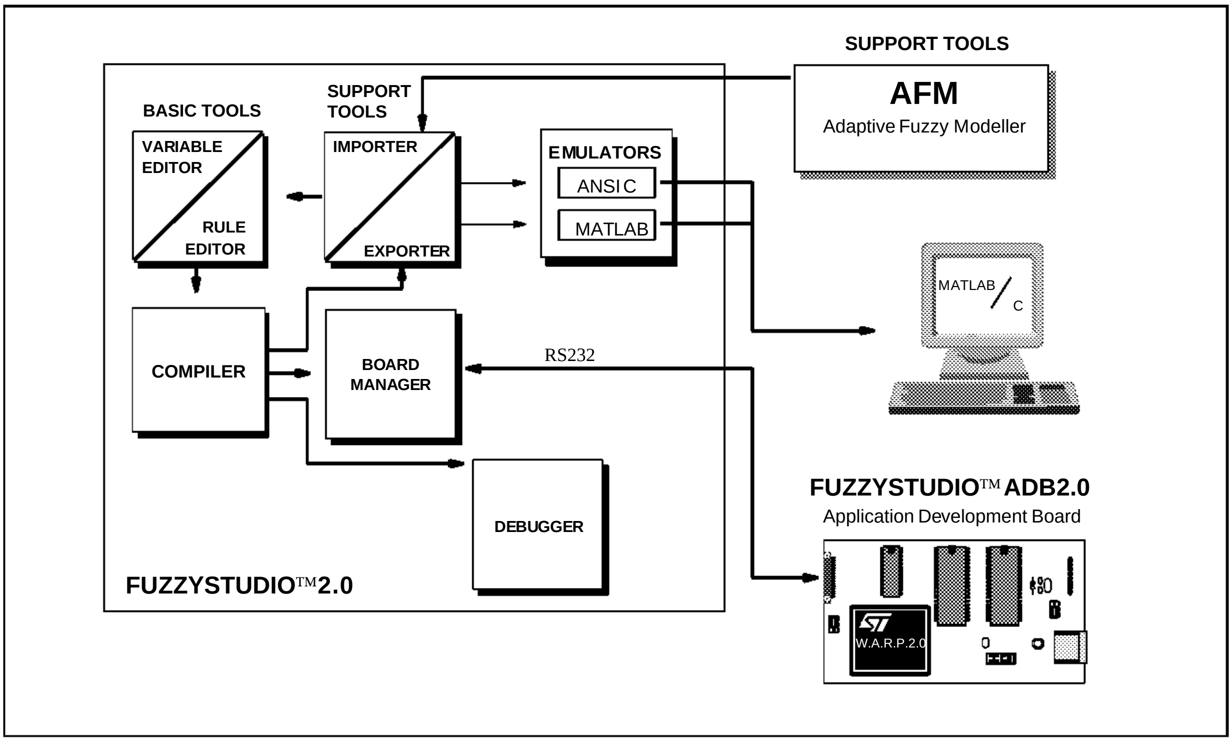
FUZZYSTUDIO™ 2.0 is composed of a complete and usefull suite of software tools and an application development board, FUZZYSTUDIO™ ADB2.0.

It uses the Fuzzy Logic typical terminology and objects, defined by means of a user-friendly graphical interface. This tool allows to program the W.A.R.P.2.0 memories - mounted on the FUZZYSTUDIO™ ADB2.0 - in accordance with the control specs defined by the designer.

Input and output Variables, Membership Functions and Fuzzy Rules can be graphically defined by means of linguistic terms.

FUZZYSTUDIO™ 2.0 guides the designer in his project definition and prevents from mistakes that could be generated by this new way of programming.

Figure 1. FUZZYSTUDIO™ 2.0 Design Flow



This is advance information on a new product now in development or undergoing evaluation. Details are subject to change without notice.

Software Tools

Variable Editor

It allows to define the input and output data Variables and the associated Term Set. It is possible to define and/or modify the Variable symbolic name, the Universe of Discourse boundaries and the associated Membership Functions with their symbolic names.

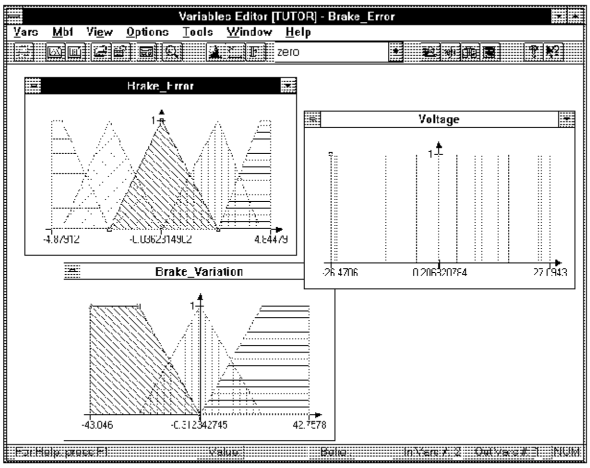
It is possible to define up to 8 input Variables having up to 8 Membership Functions or up to 4 input Variables having up to 16 Membership Functions, and it is possible to define up to 4 output Variables having up to 256 Crisp Value associated.

Only triangular, trapezoidal or crisp Functions can be defined according with W.A.R.P.2.0 device characteristics.

The MDI approach (Multi Document Interface) allows to visualize the Variables with the associated Term Set by means of several windows present on the screen at the same time. In addition, it is possible to graphically define the Membership Functions (by using the mouse), modify and move them or define an entire Term Set with only one command.

The Membership Functions points are shown considering the actual value according to the W.A.R.P.2.0 resolution - 256 levels of discretization for the Universe of Discourse and 16 truth levels.

Figure 2. Variable Editor



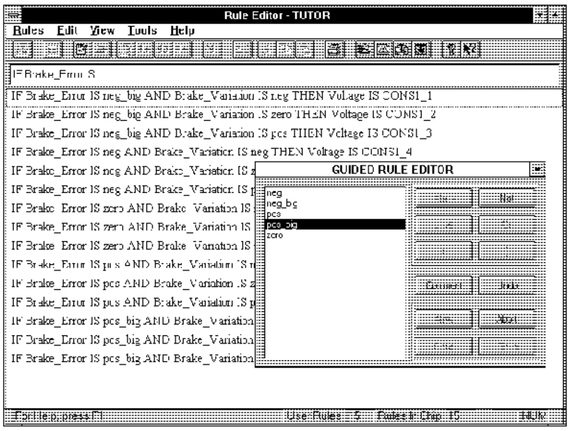
Rule Editor

Rule Editor allows the definition of fuzzy rules, either manually or by using mouse and on-screen keyboard for guided fast definition. In order to avoid syntax errors and to guide user in the rule definition, all not allowed functionalities and keywords are automatically disabled.

Once the rules have been defined, they can be modified manually and submitted automatically to syntactic checking. In addition, it is possible to copy each single rule. The syntactic checking is activated every time a manual editing of the rule is done.

A rule has a valid format when it has no more than 16 antecedent terms and 4 consequent terms and at least one antecedent and one consequent term. The rules are then splitted by Compiler in equivalent simpler rules, having 4 antecedents and one consequent; it is possible to define up to 256 rules of this format.

Figure 3. Rule editor



IF Brake_Error IS

IF Brake_Error IS neg_big AND Brake_Variation IS neg THEN Voltage IS CONST_1
IF Brake_Error IS neg_big AND Brake_Variation IS zero THEN Voltage IS CONST_2
IF Brake_Error IS neg_big AND Brake_Variation IS pos THEN Voltage IS CONST_3
IF Brake_Error IS neg AND Brake_Variation IS neg THEN Voltage IS CONST_4
IF Brake_Error IS neg AND Brake_Variation IS z
IF Brake_Error IS neg AND Brake_Variation IS p
IF Brake_Error IS zero AND Brake_Variation IS
IF Brake_Error IS zero AND Brake_Variation IS
IF Brake_Error IS zero AND Brake_Variation IS
IF Brake_Error IS pos AND Brake_Variation IS n
IF Brake_Error IS pos AND Brake_Variation IS z
IF Brake_Error IS pos AND Brake_Variation IS p
IF Brake_Error IS pos_big AND Brake_Variation
IF Brake_Error IS pos_big AND Brake_Variation
IF Brake_Error IS pos_big AND Brake_Variation

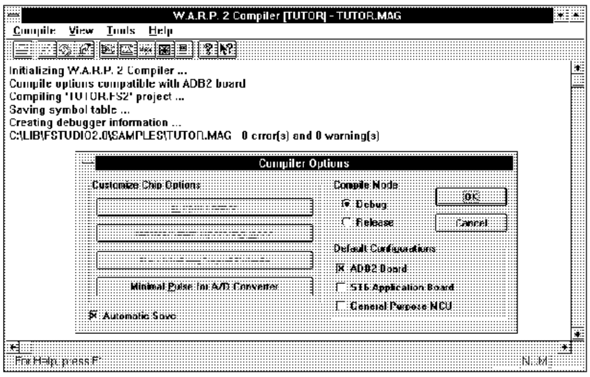
Compiler

Compiler transforms data generated by Editor in the optimized code loadable in W.A.R.P.2.0 memories.

It is possible to generate files having several formats: Binary, Intel Intellec 8/MDS, ASCII and the proprietary format for the Debugger. The first and the second format can be used to work with the development board FUZZYSTUDIO™ ADB2.0 or to program an EPROM by using a standard programmer. The ASCII format allows you to add the fuzzy code at the assembler code of a micro. The latter one is used to perform a low debugging level of fuzzy project by using Debugger.

The provided code machine depends on W.A.R.P.2.0 configuration, determined by the number of Input Variables and the number of associated Membership Functions.

Figure 4. Compiler



```
Initializing W.A.R.P. 2 Compiler ...
Compile options compatible with ADB2 board
Compiling 'TUTOR.FS2' project ...
Saving symbol table ...
Creating debugger information ...
C:\LIB\FSTUDIO2.0\SAMPLES\TUTOR.MAG   0 error(s) and 0 warning(s)
```

Compile Mode
● Debug
○ Release

Default Configurations
☒ ADB2 Board
☐ ST6 Application Board
☐ General Purpose MCU

☒ Automatic Save

Debugger

Debugger allows to verify the correctness of Fuzzy project and to execute the optimization and tuning of control process.

It is possible to monitor the Fuzzy computation obtaining immediately the output values according to the inputs values, or it is possible to analyze the step-by-step process in order to obtain the internal register values, that are Fuzzy computation values. There are two levels for the step-by-step monitoring: at low level using the machine assembler, or at high level using the Fuzzy Rules defined by the user. The two levels can be monitored at the same time.

Breakpoints can be defined to stop the computation in desired points or it is possible to Skip some parts of the fuzzy program to check the importance of some rules compared to others. Two kinds of Watch are possible: textual and graphical. The first Watch lists computational values showing the current value, the second one represents graphically the logical Fuzzy computation flow visualizing the numerical values in the logical point in which they occur in the algorithm.

Input patterns can be defined:

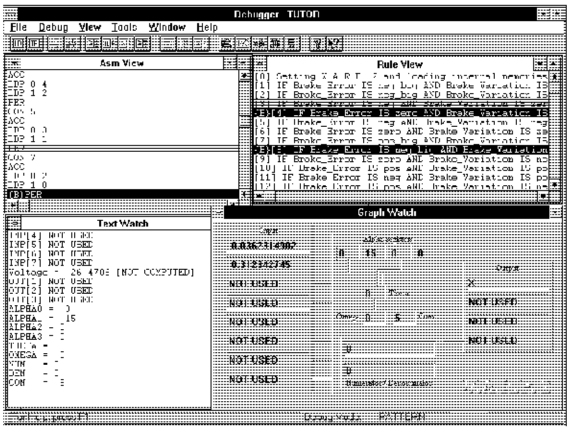
Manually.

In a file defined by the user.

In a file automatically defined by using any tool able to generate data in compatible format.

Computed output values can be stored in an output stream file.

Figure 5. Debugger



```
ACC
IDP 0 4
IDP 1 2
PER
COS 5
ACC
IDP 0 0
IDP 1 1
COS 7
ACC
IDP 0 2
IDP 1 0
(B)PER
```

```
[0] Getting W.A.R.P. 2 and loading internal memories
[1] IF Brake_Error IS neg_big AND Brake_Variation IS
[2] IF Brake_Error IS neg_big AND Brake_Variation IS
[3] IF Brake_Error IS neg AND Brake_Variation IS zer
(B)[4] IF Brake_Error IS zero AND Brake_Variation IS
[5] IF Brake_Error IS neg AND Brake_Variation IS neg
[6] IF Brake_Error IS zero AND Brake_Variation IS ze
[7] IF Brake_Error IS pos_big AND Brake_Variation IS
(B)[8] IF Brake_Error IS neg_lit AND Brake_Variation
[9] IF Brake_Error IS zero AND Brake_Variation IS ne
[10] IF Brake_Error IS pos AND Brake_Variation IS po
[11] IF Brake_Error IS neg AND Brake_Variation IS po
[12] IF Brake_Error IS pos AND Brake_Variation IS ne
```

```
INP[4] NOT USED
INP[5] NOT USED
INP[6] NOT USED
INP[7] NOT USED
Voltage = 26.4703 [NOT COMPUTED]
OUT[1] NOT USED
OUT[2] NOT USED
OUT[3] NOT USED
ALPHA0 = 0
ALPHA1 = 15
ALPHA2 = 0
ALPHA3 = 0
THETA =
OMEGA = 0
NUM  = 0
DEN  = 0
CON  = 8
```

0.0362314902
0.3123442745

Exporter/Importer

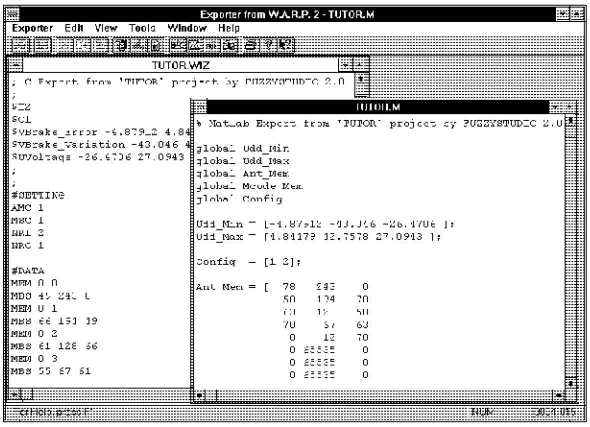
The Exporter provides a codified Fuzzy project, defined with the Editor, which is imported in other environments to perform simulations and validation.

It provides the W.A.R.P.2.0 model in C language, in MATLAB® language, or in the proprietary language Fu.L.L.(Fuzzy Logic Language); the purpose is to have a tool to perform closed-loop simulations of Fuzzy systems using C or MATLAB® softwar emulators or to provide an interface towards the other proprietary S/W tools supporting W.A.R.P.2.0 application development.

User has to describe just the model of the system to be controlled as he likes, whereas the model of Fuzzy controller is automatically generated by the Exporter.

The Importer allows FUZZYSTUDIO™ 2.0 to be interfaced with other proprietary high level software tools supporting W.A.R.P.2.0. For example, the proprietary program for the synthesis of fuzzy controller AFM (Adaptive Fuzzy Modeller) based on Neuro-Fuzzy algorithm, generates projects written in Fu.L.L. language; the Importer allows to load the project so generated into FUZZYSTUDIO™ 2.0 in order to be used by all the tools offered by the environment (code machine generation, debugging, etc. etc.).

Figure 6. Exporter



```
; C Export from 'TUTOR' project by FUZZYSTUDIO 2.0
;
$IZ
$CL
$VBrake_error -4.879_2 4.84
$VBrake_Variation -43.046 4
$UVoltage -26.4706 27.0943
;
;
#SETTING
AMC 1
MSC 1
NRL 2
NRC 1

#DATA
MEM 0 0
MBS 45 240 0
MEM 0 1
MBS 66 151 19
MEM 0 2
MBS 61 128 66
MEM 0 3
MBS 55 67 61
```

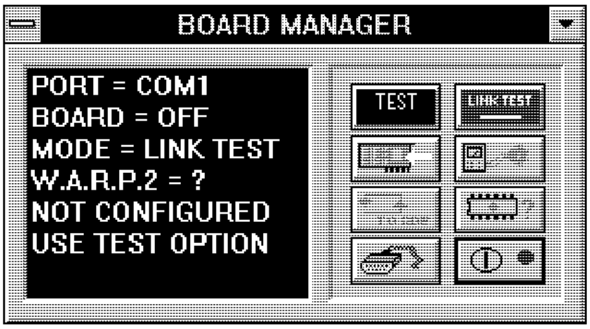
```
% Matlab Export from 'TUTOR' project by FUZZYSTUDIO 2.0

global Udd_Min
global Udd_Max
global Ant_Mem
global Mcode_Mem
global Config

Udd_Min = [-4.87912 -43.046 -26.4706 ];
Udd_Max = [4.84179 12.7578 27.0943 ];

Config = [1 2];

Ant_Mem = [  78   243    0
             50   104   70
             60    15   50
             70    57   63
              0    12   70
              0 65535    0
              0 65535    0
              0 65535    0
```

Board Manager

FUZZYSTUDIO™ 2.0 allows to manage the PC-Board FUZZYSTUDIO™ ADB2.0 (Application Development Board) connected to PC by RS-232 connection cable.

By using the board jointly with the PC it is possible to test the fuzzy project directly on the hardware. It is also possible to test any W.A.R.P.2.0 application while monitoring the quantities considered by the fuzzy processor.

Figure 7. Board Manager



PORT = COM1
BOARD = OFF
MODE = LINK TEST
W.A.R.P.2 = ?
NOT CONFIGURED
USE TEST OPTION

FUZZYSTUDIO™ ADB2.0 DESCRIPTION

The board has been designed to be connected to the RS232 port of an IBM PC 386 (or higher), but it can also work stand alone.

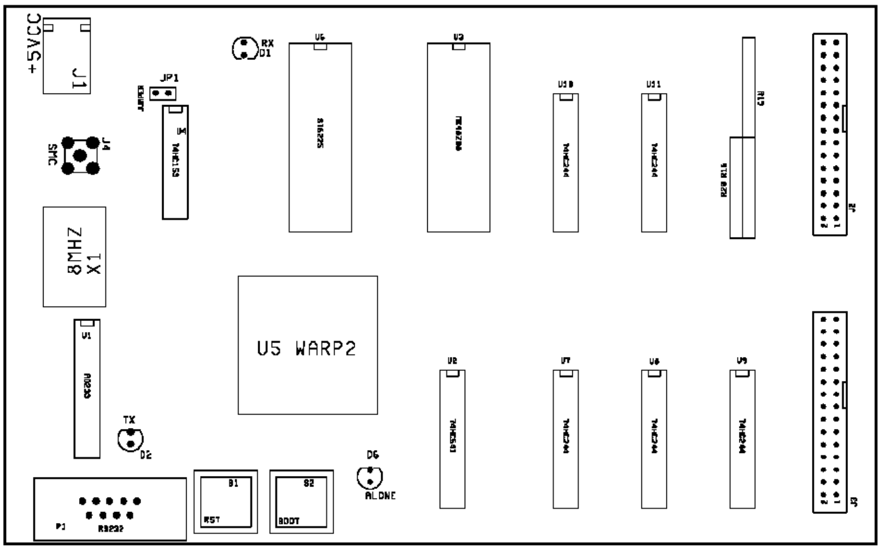
It can manage up to 8 digital inputs and 4 digital outputs.

Inputs and outputs are provided at TTL compatible level. The board allows the user to charge the rules and the membership functions (see FUZZYSTUDIO™ 2.0 User Manual) into the W.A.R.P.2.0 memories.

The clock generator frequency on board is 8 MHz. An automatic trigger is used to synchronize W.A.R.P.2.0 with the external environment (working connected with a PC).

When the board is used deconnected from a PC all the fuzzy data (membership functions and rules) are stored in a ZEROPOWER SRAM.

Figure 8. FUZZYSTUDIO™ ADB2.0 Board Layout



Adaptive Fuzzy Modeller

Adaptive Fuzzy Modeller (AFM) is a tool that easily allows to obtain a model of a system based on Fuzzy Logic data structure, starting from the sampling of a process/function expressed in terms of Input\Output values pairs (patterns).

Its primary capability is the automatic generation of a database containing the inference rules and the parameters describing the membership functions. The generated Fuzzy Logic knowledge base represents an optimized approximation of the process/function provided as input.

The AFM has the capability to translate its project files to FUZZYSTUDIOs™ , MATLAB® and C code, in order to use this environment as a support for simulation and control.

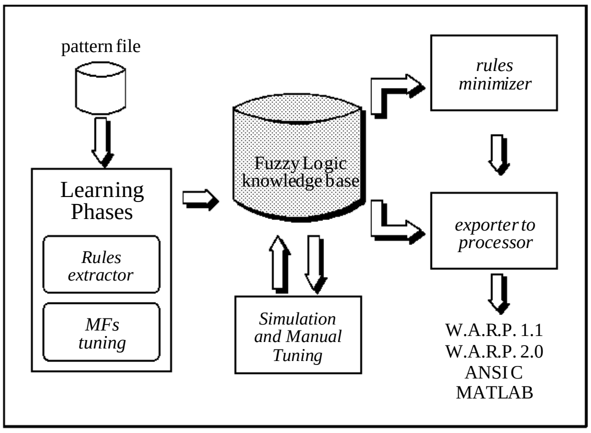
The block diagram illustrates the AFM design flow.

SUPPORTED TARGETS

The supported environment are:

- W.A.R.P. 1.1 using FUZZYSTUDIO™ 1.0
- W.A.R.P.2.0 using FUZZYSTUDIO™ 2.0
- MATLAB®
- C Language
- Fu.L.L. (Fuzzy Logic Language).

Figure 9. AFM Design Flow



FUZZY STUDIO™ 2.0 System Requirements

- MS-DOS version 3.3 or higher
- Microsoft Windows 3.1 or compatible later version
- 486, PENTIUM compatible processor
- 4 MBytes RAM (8 Mbytes recommended)
- Hard Disk with at least 1MBytes free space

Ordering Information

Order Code	Device	Development Tools	
		FUZZYSTUDIO™ ADB2.0	SW Tools
STFLSTUDIO2/KIT	STFLWARP20/PL	W.A.R.P.2.0 W.A.R.P.2.0 programmer ZEROPOWER programmer RS-232 communication handler Internal Clock	Variables and Rules Editor W.A.R.P.2.0 Compiler/Debugger MATLAB® and C SW emulators Importer from AFM

Order Code	Maximum Frequency	Supply Voltage	Temperature Range	Package
STFLWARP20/PL	40 MHz	5±5%	0 °C to 70 °C	PLCC68

Order Code	Description	Supported Target	Functionalities	System Requirement
STFLAFM10/SW	WTA-FAM for Building Rules BACK-FAM for Building MFs	STFLWARP11/PG STFLWARP11/PL STFLWARP20/PL ANSI C MATLAB®	Rules Minimizer Step-by-Step Simulation Simulation from File Local Tuning	MS-DOS 3.1or higher Windows 3.0 or later 486, PENTIUM compatible 8 MB RAM

Information furnished is believed to be accurate and reliable. However, SGS-THOMSON Microelectronics assumes no responsibility for the consequences of use of such information nor for any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent or patent rights of SGS-THOMSON Microelectronics. Specification mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. SGS-THOMSON Microelectronics products are not authorized for use as critical components in life support devices or systems without express written approval of SGS-THOMSON Microelectronics.

FUZZYSTUDIO™ is a trademark of SGS-THOMSON Microelectronics
MS-DOS®, Microsoft® and Microsoft Windows® are registered trademarks of Microsoft Corporation.
MATLAB® is a registered trademark of Mathworks Inc.

SGS-THOMSON Microelectronics GROUP OF COMPANIES
Australia - Brazil - Canada - China - France - Germany - Hong Kong - Italy - Japan - Korea - Malaysia - Malta - Morocco - The Netherlands - Singapore - Spain - Sweden - Switzerland - Taiwan - Thailand - United Kingdom - U.S.A.