<< Visual State Machine User Manual>>

<< 可视状态机用户使用手册 >>

1 Introduction of SVM

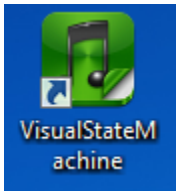
Visual State Machine (briefly in VSM) is a finite state machine, whose state and transition from state to state can be easily represented with visual objects by GUI software. egPLC/RTU products adopt VSM in its user-level software feature customization. Users create their application program using VSM model and download VSM program into egPLC/RTU products. Once program downloaded, binary code is stored in flash memory and automatically run on every power up until overwritten next time by other program. Visual State Machine here especially refers to the PC software provided by Raynix Ltd. With this software installed users can create, edit, compile, link, and download customized programs for egPLC/RTU series products.

egPLC/RTU series products configure their CPU and memory resource with system space and user space. A 32-bit microprocessor running VSM code is embedded in user space for user program execution. This micro-processor support 32-bit floating point data representation and floating point data operations. A state transit instruction is intrinsically facilitated to support VSM program flow switch from state to state.

A state consists of a piece of code execution, and a CPU idle state followed after this code execution. A transition from current state to next state is executed in the end of current state. VSM software is used to create and edit C code for every state, and to check up program grammar, to compile code, to assemble code, and to download binary code.

2 Install VSM software

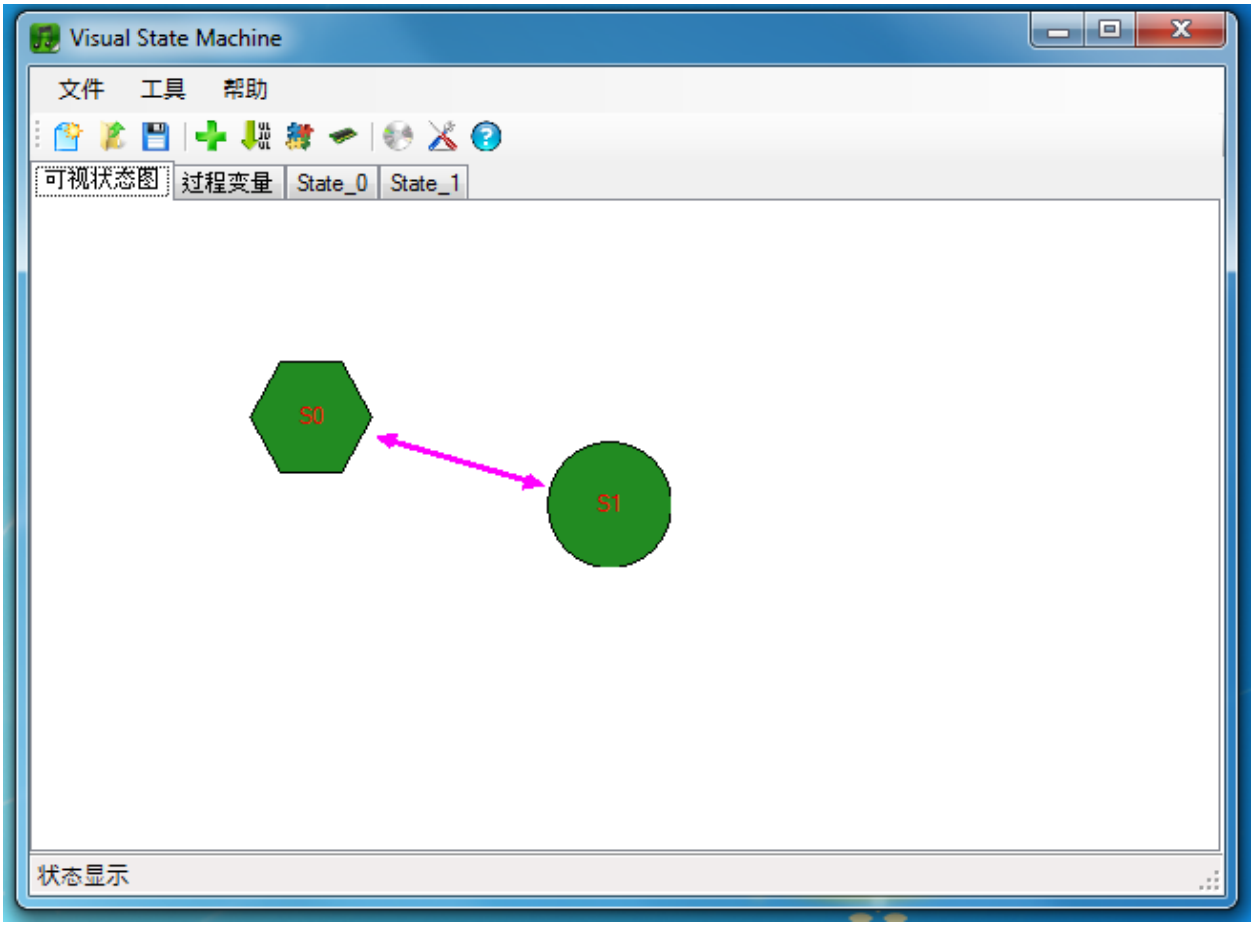
VSM Windows PC software is delivered with a single installation copy “Setup_VSM.msi”. This program can be double clicked from Windows to start and carry out the installation process. However, in case no dotNet components previously resident in host PC machine, users may start to run “setup.exe” program instead, which will direct the installation process by downloading necessary components from dotNet website. The installation of VSM software is quick and simple. Upon installed, an icon is display on PC desktop, indicating the shortcut of VSM.



Under installation directory, the software comes with a “blink” example, which is located in “example” sub-directory. This example only consists of one file, “blink.vsm”. This file can be loaded by Visual State Machine software and users can edit this, modify this and download this into egPLC/RTU series products – to customize input/output close loop control process and to customize other features.

3 Walk Through an Example

Start the Visual State Machine and from File Menu to load an existing project, then load the “blick.vsm” into the GUI software.



We can see from visual state tab “blink.vsm” has 2 states (S0 and S1) and two transitions (from S0 to S1 and the other way round from S1 to S0).

We can also see the S0 state is hexagon shaped object rather than S1 which is round shaped object. Hexagon shaped state is the initial state, where is the entry of the program. Every program has only one initial state, even if only one state is there in a program!

State_0 tab displays the code which is executed in S0 and State_1 tab displays the code which is executed in S1. Here the code is simply OI functions to turn on and turn off the 8-YOUT solid relay switches of egPLC5888 device. And in each state the program will stay idle for 1 second! The unconditional state transition results in a blink effect on the output lamps.

There is no global variables or procedures in this application program.

Once a program is composed ready, we can check grammar, compile, and assemble the code from the “tools” menu and eventually the program is down loaded into the flash memory of the device.

“Grammar checkup” will automatically scan the whole VSM states and code and report the results on the status bar.

"Compilation" will compile the VSM program and report the results on the status bar.

"Assembly" will assemble and link the program and report the results on the status bar.

When all above processes are passed "download" menu item click will send the binary code into the flash memory of the device. Once program downloaded the device will run the user program on powering up the device.

Users now can see the led blinking!

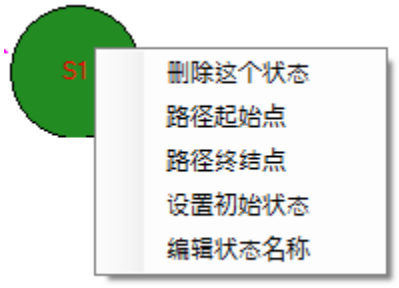
4 Compose a Program

Start "Create a new project" from the "File", then users can add new states, add new transitions, and edit the code. The code is C language and VSM software package provide a set of predefined system library interface where "#include" keyword is not needed! The predefined system library is defined as follows.

```
/*
* make a transition from one piece of code to other piece of code
* addr is the label of the code, or name of function/sub routine.
* This is used for VSM switch from state to state. This performs a go to function
* without return
*/
void switch_to(long addr);


/*
* delay amount of time in milliseconds, 1 tick takes 10ms, so actual delay time is round
* down to the nearest 10-time integer in millisecond.
*/
void br_delay(int ms);


/*
* basic input a long data from IO port by given address
*/
long br_inport(long addr);


/*
* basic output a long data to IO port by given address
*/
void br_outport(long addr, long dat);


/*
* set RS485 master port baud rate, 9600 by default
*/
void br_set_baudrate(long bd);


/*
* receive or send a character from or to RS485 master port
* for receive, it is a synchronous blocked function, waiting for the character until got
*/
```

```
char br_recv_char(void);
void br_send_char(char c);


/*
* send a block data to RS485 master port, the string length and block data length
* should be no more than 32 bytes, otherwise long message should configured in short
* by user. Users may calculate the sending schedule according to baud rate
*/
void br_send_string(char *s);
void br_send_block(char *s, int num);


/*
* read a line of char from RS485 master port and store to string until a '\r' or '\n'
* is met.
* A null-terminated string is stored in st[]
*/
char *br_gets(char *st);


/*
* print a formatted number to RS485 master port
* the supported format is
* "%i" for int and output in hex
* "%l" for long and output in hex
* "%c" for character
* "%s" for string
*/
void br_printf(char *fmt, ...);


/*
* read a hex number from human readable string literal with a fixed length required
*/
long br_get_hex(char *s, int n);
```

To add a new state
On the visual state graphics tab user can click down the right mouse button on the background area to bring up a context menu, which is the "add new state" command. Once a new state is added to the project, its attributes can be modified through the context menu of the object. What it display as follows.



From the context menu, one can delete this state object, one can change the name of this state object, and one can mark this state as the initial sate of the program.

Accordingly, users can create transitions from state to state. A transition is defined by a source state and a target state. Transitions are thus created by state pairs and even more source state and target state can be the same state!

A state transition code is automatically generated in state tab page and users can refine the transition conditions to properly control their programs!

In any case, a transition state, switch_to() function, must be encountered in the end of every state code execution.

After grammar checkup and compilation, the project can be saved on the disk for future use and for exchange purpose.

5 Build and Download a Program

When user start to build a program, the software will install all states in a single C file internally and then the C file is undergone a series of processing step by step. During grammar checkup system will report display every single error message if any. When no error is there the status bar display “grammar checkup pass” information. This message suggest the next process can be performed following along the whole processing chain.

Program building process steps will include grammar checkup, compilation, assembling, and downloading. Before downloading egPLC/RTU device should be connected to the USB port or through TCP/IP network!

6 VSM for USB and Net Device

USB port device is logically connected to the PC host as soon as the USB cable is connected from host PC to egPLC/RTU. Accordingly the program downloading is carried out to access USB port by default in this case. For downloading TCP/IP port devices, a target IP address should be provided beforehand in dialogue setting box. This IP address is not used for target device setting update, rather provide the Visual State Machine software the target address of the device. When a Net port device is attached to the network and this IP address is specified in the connection field and no USB port egPLC/RTU device is present, then the downloading process is forwarded to the TCP/IP network.

As a summary, when USB device is connected the downloading direction goes to USB device, otherwise, the downloading direction searches the network by the specified IP address.

7 Language Specifications

Visual State Machine uses a subset of ANSI/C grammar, which does not support enumeration data structure. It support 4 basic signed data types, which are char (8-bit signed integer), int (16-bit signed integer), long (32-bit signed integer), and float (32-bit IEEE-754 floating point number). VSM supports combined data structures, shuch as structure, union, array, etc.

The variables in expression should be performed based on the same data type, VSM do not automatically perform data type conversion. Any data type conversion should use explicit convention.

VSM support macro definition and header file including preprocessing, however, the directory and file name convention should comply with Unix naming convention rather than Windows naming convention. No space or “-“ is allowed in a file name or path name. The

strict requirement of VSM in source level code helps to remove any ambiguous language code and generate and produce stable and definite binaries!