

## PROGRAMMING STATE MACHINES ON MICRO CONTROLLERS

Version 1.1

Date: 21 April 2003  
 © 2003 by GenerExe  
[www.generexe.com](http://www.generexe.com)

### 1. What is so hard about State-machines?

State machines, state charts or state diagrams are generally accepted as a superior formalism for modeling dynamic real-time behavior. They are used to manage highly dynamic processes, ranging from communication protocols, your VCR or the reactor of a chemical plant.

However, their implicit parallel nature and the “software-engine” required to execute state machines, have not made them very popular in mainstream software-development. One problem is that software is written mainly in linear text, which makes it hard to keep track of the bigger picture. Programming state-machines requires continuous awareness of what you are supposed to do in a specific state and what certainly NOT! Misunderstanding the overall state-behavior, can turn a seemingly trivial change into a very hard to find bug. I.e. you might use some extra conditions or global variables, accidentally creating ambiguous sub-states or windows in time of undefined behavior.

### 2. State-machines made easy

GenerExe developed XPad. XPad is a cross-compiler suite, allowing you to

visually design and program state/object based software for a growing number of micro controllers.

The state machine diagram divides the “life” of a task in your software in distinct modes of behavior (states). When an event triggers a state-transition, the entry-code of that state is executed.

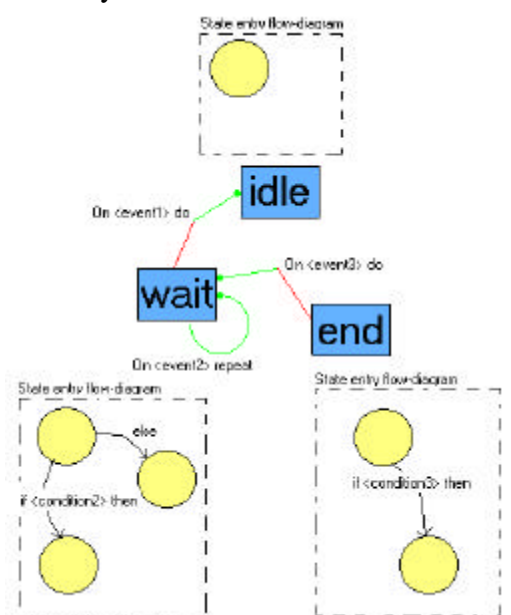


Figure 1. State machine with entry-code in XPad

Entry-code of a state is described by a flow diagram. Flow diagrams may consist of many blocks –lists of functions- connected by decisions, loops, etc.

Furthermore, state machines may include local member functions, which

look just like flow diagrams, but take parameters and can be called from any state in any state diagram. Member functions can access the local resources of a state machine (i.e. local variables).

XPad state machines provide a clear, easy and efficient mechanism to implement multiple parallel tasks, responding to different events at different times, **without the need for a separate RTOS**. When you design the state-machines you also design your executive code. The generated assembly code includes an ultra-thin state machine engine, with the ability to run many state machines in parallel and with integrated support for interrupts, customized to the model you created.

Before compilation, models can be tested and verified in the simulator. The simulator compiles the XPad-model into an intermediate format, suitable for simulation. This intermediate format is also the source for the compiler. XPad compilers only generate the code you actually use. Libraries are included to interface with most industrial interfaces and other libraries can be added easily by including formatted assembly files.

Last but not least, documents can be generated automatically, including all diagrams and comment.

### 3. Micro controller support

XPad focuses mainly on micro controller software development. Smaller micro controllers have not enjoyed the support in tooling many enterprise IT-environments are blessed with:

- Developers often have no choice, but to use the same command-line tools as 20 years ago;
- MCUs are especially scarce in resources and often cannot sustain a full-fledged RTOS;
- MCUs are mostly used to control and monitor processes, making them especially suited for event-driven state machine control.

Development in XPad starts with the choice and configuration of a target. Wizards can help you with these tasks.

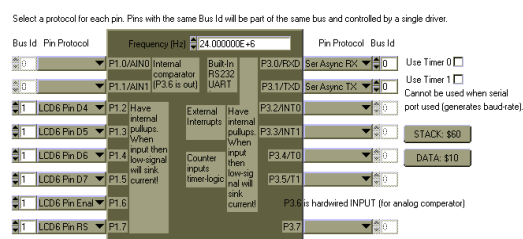


Figure 2. Hardware configuration panel in XPad

On the configuration panel you see the pin-layout of the micro controller. Selection-lists on each pin allow you to assign functions, supported by the peripherals behind that pin, to multiple interface buses. Each bus results in a bus-variable, which when passed to built-in libraries functions, informs XPad and its compilers about how to manipulate which output ports.

Currently XPad supports the Motorola 68HC11 Family, most Microchip PIC 14core midrange devices and a large number of 8051 devices. While XPad comes with compilers for specific families of devices, you can easily add support for specific devices yourself, by creating your own hardware configuration panels, wizards and include-libraries.

## 4. Some examples

The diagram in figure 3 shows a simple XPad state-machine, which waits for a serial character and echoes it back.

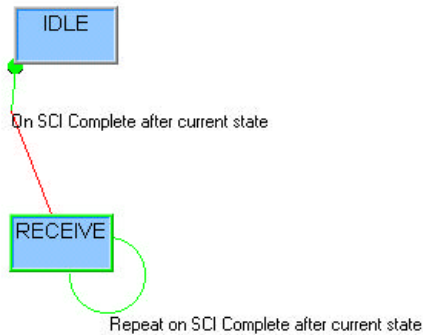


Figure 3. State-machine to echo back characters received on a serial port.

The IDLE-state initializes the serial port for 9600 baud. (Figure 4).



Figure 4. A flow diagram with a single block initializing the serial port.

When the SCI-complete event occurs, (the serial port interrupt on the 68HC11 MCU), the RECEIVE-state is executed, which reads the character from the serial port and transmits it back. (Figure 5).

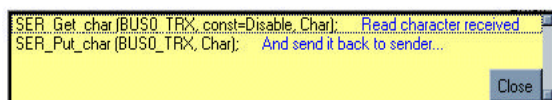


Figure 5. A flow diagram with a single block reading and writing the serial port.

Finally the state-machine waits for the SCI-complete event again to execute the RECEIVE-state again.

The diagram in figure 6 shows an XPad example for the Atmel AT89C2051.

In the IDLE-state the serial port and Timer0 are setup. The timer is configured to continuously timeout, generating Timer0-events.

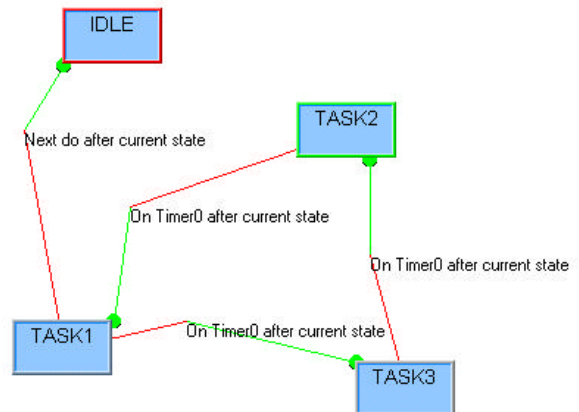


Figure 6. State diagram running 3 states in a cyclical fashion, driven by the timer.

Next the state-machine immediately proceeds to the TASK1-state (via the NEXT-event), which transmits the letter "A" over the serial port. Similarly Timer0 events, put the state-machine into the TASK2-state, transmitting the letter "B" and the TASK3-state, transmitting the letter "C". The following Timer0 event brings the state-machine back into the TASK1-state, starting the whole cycle over again.

**Note** that the state machines in figure 3 and figure 6 could be part of the same model. (For the same target MCU). The compiler just as easily generates code to run the 2 state machines in parallel.

You can download a copy of XPad from <http://www.generexe.com>. Low cost compiler modules (USD40 or less) can be downloaded separately from the same site. Limited compiler modules can be tried for free.