Safe Virtual Machine for C

The ultra small target independent Virtual Machine

Dynamic code in C for deeply embedded and distributed systems

---

Altreonic is breaking new grounds by complementing its OpenComRTOS with the capability to run processor independent code on any node in a networked system. The Safe Virtual Machine (SVM) was generated from a formal description and requires less than 3 Kibytes per task for code and about 10-15 Kibytes for data. As the native OpenComRTOS also only requires 5-15 Kibytes, this capability opens new possibilities for memory and power constrained embedded systems.

Capabilities:

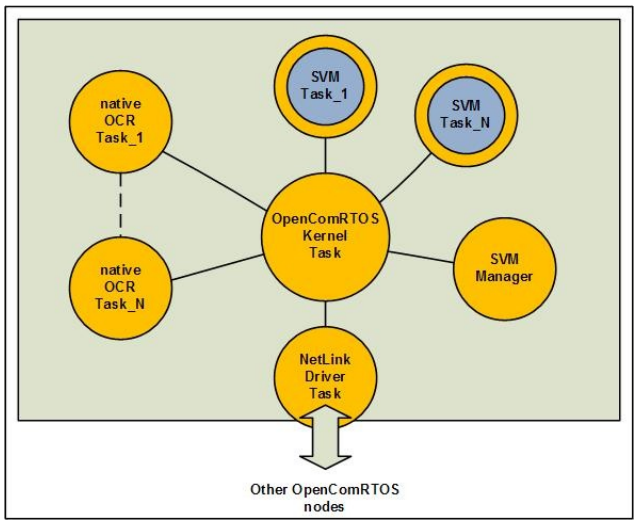
- Loading and running code on a remote node without halting it
- Dynamic code loading at runtime
- Dynamic task migration at runtime
- Transparent communication with native OpenComRTOS tasks on the same or on remote nodes
- Safe execution: SafeVM tasks cannot violate memory boundaries.
- The mechanism can be applied to other binary targets.

Future releases will also support calling native functions for maximum performance and direct execution of the binary images on the target processor.

Applications:

- Remote diagnostics.
- Fail safe and fault tolerant control.
- Processor independent programming.

Thanks to the use of OpenComRTOS, SafeVM tasks can operate system wide across all nodes in the network. The user can also put several SafeVM tasks on the same node. The natively running OpenComRTOS itself acts as a virtual machine for the SVM tasks, isolating them from the underlying hardware details while providing full access.



How it works:

The SafeVM capability is provided by a SafeVM task manager that can start and stop the SVM task, monitor their status and interact with the SafeVM host task. The latter is composed of a safe VM execution module and the actual SafeVM task. The SafeVM task it- self interacts with the rest of the node and the net- work through pre-compiled services, including the standard OpenComRTOS services.

Altreonic NV - Gemeentestraat 61A b1, B-3210 Linden, Belgium.
info.request@altreonic.com, www.altreonic.com, tel. +32 16 20 20 59

Altreonic

Safe Virtual Machine for C
From Deep Space to Deep Sea

Available SafeVM task services:

SVM_LoadTask
// loads the SVM task in the SVM workspace

SVM_StartTask
// starts the SVM task

SVM_StopTask
// Stops the SVM task

SVM_ClearTask
// Clears the workspace memory of the SVM // Equivalent to unloading the SVM task

SVM_GetErrorInfo
// Returns the last error code

SVM_GetState
// Returns the current state of the SVM

OpenComRTOS services

OpenComRTOS services, callable by the SVM task, operate transparently across a networked or multicore systems.
Services include: preemptive priority based task scheduling, events, counting semaphores, fifos, resource locks with support for priority inheritance, packet pools, memory pools, DataEvent, BlackBoard, MBQ. See the OpenComRTOS datasheets and manuals for more details.
The SVM task can be written and compiled using all features of ANSI-C, including libraries.

Relative performance

Measured on ARM Cortex M3:

Program code for SVM
(manager + execution module): < 3 KiBytes

Relative RTOS performance:
Example semaphore loop with one task being a SVM_Task: about 7x slower.

Computational performance:
depends on application code.

Example program:

This example first loads a first SVM Task on the node, waits until it stops, and then up- loads a new SVM Task to the node and starts it.

```
#include <SVM_HostService/
          SVM_HostClient.h>
void Task1_entry(L1_TaskArguments
Args)
{
FILE *fo;
SVM_STATE state;
// TASK1 - binary file with image
fo = fopen(SVM_TASK1,"r");
assert(fo!=NULL);
// buffer for storing program
L1_UINT32 buffer[128];
L1_UINT32 n = fread(buffer,
              sizeof(L1_UINT32),
              128,
              fo);
assert(n!=0);
// send task to SVM task
SVM_LoadTask(SVM_Manager,
                buffer,
                n*4,
                5000);
// start task on the node
SVM_StartTask(SVM_Manager);
// wait until first task stops
do {

     SVM_GetState(SVM,&state);
     if(state==VSP_VM_ERROR)
     exit(1);
    }
while(state!=SVM_VM_STOPPED);
// Load second task
// SVM_TASK2 - binary file with image
   fo = fopen(SVM_TASK2,"r");
assert(fo!=NULL);
n = fread(buffer,
           sizeof(L1_UINT32),
           128,
           fo);
assert(n!=0);
SVM_LoadTask(SVM_Manager,
           buffer,
           n*4,
           5000);
```

Altreonic