Notes About Module Manager

August 4, 2010

1 Introduction

Module Manager is an application that is installed when you build the ACM Runtime. This application is responsible for scheduling the partitions inside the module and transferring the inter-partition messages via channels. The module manager is configured with a fixed cyclic schedule with pre-determined hyperperiod. The schedule is computed from the specified partition periods and durations. The module configuration also specifies the hyperperiod value, which is the least common multiple of all partition periods, the partition names, the partition executables, and their scheduling windows. Note that the module manager allows execution of one and only one partition inside a given scheduling window, which is specified with the offset from the start of the hyperperiod and a duration.

Before using the module manager one will have to (1) compute the module schedule and then write the configuration file. Note that this configuration file is generated for you automatically if you use the ACM modeling environment and code generation tools (available as a separate download from the same page).

1.1 Computing Module Schedule:

Let $\mathbb{P}$ be the set of all partitions in a module. Let $\phi(p) \in \mathbb{R}^1 \cap [0, \infty)$ denote the period of partition $p \in \mathbb{P}$. Let $\Delta(p) \in \mathbb{R} \cap [0, \phi(p)]$ denote the duration of time that a partition needs to be executed every $\phi(p)$ time units. Then hyper period H is given as $H = LCM(\phi(\mathbb{P}))^2$, where LCM is the abbreviation for least common multiple.

Then the count of times a partition runs in a hyper period $N(p)$ is given as $N(p) = H/\phi(p)$. Let $\mathbb{O}(p) =< O_i >_{i=1}^{i=N(p)}$ be the sequence of offsets from the start of the hyper period when partition p needs to be started. We will use the notation O_i^p to denote the offset of ith execution for partition p in the hyper period.

The goal is to compute the set of all offset sequences for all partitions, $\mathbb{O}(\mathbb{P})$. A feasible valuation for $\mathbb{O}(\mathbb{P})$ can be found by solving the following system of constraints using a constraint solver library such as gecode[3]:

C1 First start for all partitions must happen before the period ends i.e. $(\forall p \in \mathbb{P})(O_1^p \leq \phi(p))$.

C2 Time between any two executions should be equal to partition period i.e. $(\forall p \in \mathbb{P})(k \in [1, N(p)-1])(O_{k+1}^p = O_k^p + \phi(p))$.

C3 Last start must finish before the hyper period ends i.e. $(\forall p \in \mathbb{P})(O_{N(p)}^p + \Delta(p) \leq H)$

C4 A partition cannot be preempted i.e. $(\forall p \in \mathbb{P})(\forall z \in \mathbb{P})(k \in [1, N(p)])(j \in [1, N(z)])$ $(O_k^p \leq O_j^z \implies O_j^z \geq O_k^p + \Delta(p))$

1.2 Writing The Configuration File

As mentioned earlier, You will not need to write a configuration file by hand if you are using the ACM Modeling Tool Suite to generate code. However, if you wish to only use the ARINC Emulation Layer without the component layer, you will need to write the configuration file by hand. In this section, we will provide the set of steps that can be used to create this file. Follow these steps to write this file.

Note: {Module M. Set P of Partitions. $HyperPeriod(M)$ is the hyperperiod value. Unit is seconds. It can be a floating point number.}
Note: $\{(\forall p \in P)$. $OFFSET(p)$ is the set of Offsets when p should start execution $\}$
Note: $\{(\forall p \in P)$. $EXECUTABLE(p)$ is the relative path to the executable file. $NAME(p)$ is the partition's name.$\}$
Note: $\{(\forall p \in P)$. $DURATION(p)$ is the duration for which p will run during each execution. $\}$

[1] $\mathbb{R}$ is the set of all reals
[2] Here $\phi(\mathbb{P})$ is a used as a succinct representation of set $\{x | x = \phi(p) \wedge p \in \mathbb{P}\}$. We will use this short representation for other sets also.
[3] http://www.gecode.org/

Note: {$(\forall p \in P)$. $SRC\_SP(p)$ is the set of source sampling ports.$(\forall s \in SRC\_SP(p))$ $MAXMESSAGESIZE(s)$ is the maximum message size. $REFRESHPERIOD(s)$ is the refresh period in seconds. It can be a floating point number. $DIRECTION(s) = SOURCE$. $NAME(s)$ is the port's name.}

Note: {$(\forall p \in P)$. $DST\_SP(p)$ is the set of destination sampling ports. $(\forall s \in DST\_SP(p))$ $MAXMESSAGESIZE(s)$ is the maximum message size. $REFRESHPERIOD(s)$ is the refresh period in seconds. It can be a floating point number. $DIRECTION(s) = DESTINATION$. $SRC(s)$ is the name of corresponding source sampling port that this port is connected to. $NAME(s)$ is the port's name.}

Note: {$(\forall p \in P)$. $SRC\_QP(p)$ is the set of source queuing ports. $(\forall q \in SRC\_QP(p))$ $MAXMESSAGESIZE(q)$ is the maximum message size. $MAXNUMBEROFMESSAGES(q)$ is the maximum number of messages this port can have. $DIRECTION(q) = SOURCE$. $NAME(q)$ is the port's name.}

Note: {$(\forall p \in P)$. $DST\_QP(p)$ is the set of destination queuing ports. $(\forall q \in DST\_QP(p))$ $MAXMESSAGESIZE(q)$ is the maximum message size. $MAXNUMBEROFMESSAGES(q)$ is the maximum number of messages this port can have. $DIRECTION(q) = DESTINATION$. $SRC(q)$ is the name of corresponding source queuing port that this port is connected to. $NAME(q)$ is the port's name.}

Note: {$CHANNEL(M)$ are all the inter-partition communication links in the module. A Channel can either support sampling ports or it can support queuing port. They cannot be mixed. A channel can connect 1 source port to multiple destination ports. $(\forall c \in CHANNEL(M))$ $NAME(c)$ is the channel name. $SRC(c)$ is the name of the source port. A channel can have only one source port. $DST(c)$ is the set of names of all destination ports.}

Note: Let I be the number of maximum number of hyper periods that you want this module to run. {This Entry is optional. Default is to run forever until a SIGINT is sent to the Module Manager.}

Note: Let c be the number of cpu core that you want this module to run. A module can run on only one core at a time. {This Entry is optional. Default is to run on cpu core 0.}

Note: You can put a single line C++ kind of comment (//) anywhere in the configuration file. It does not understand multiline comments ($/ * text * /$) right now. So do not use multiline comments.

Pre Condition: All names must be unique. They should not contain any whitespace and any special characters.

Note: Now we start writing the configuration file. In the following statements any string contained within $<$ and $>$ should be replaced with its evaluation. For example for a partition called PART1, the text `<NAME(PARTITION)>` should be be replaced by `PART1`

```
1:  WRITE MAXITERATIONS = $I$ {OPTIONAL}
2:  WRITE CPU = $c$ {OPTIONAL}
3:  WRITE HYPERPERIOD = $HyperPeriod(M)$
4:  for ($\forall p \in P$) do
5:    WRITE PARTITION_NAME=<NAME(P)> {NAME(P) should be replaced with the actual name.}
6:    WRITE <NAME(P)>_EXECUTABLE=<EXECUTABLE(P)>
7:    for $\forall offset \in OFFSET(p)$ do
8:      WRITE <NAME(P)>_SCHEDULE=<offset>,<DURATION(p)>
9:    end for
10:   for $\forall port \in SRC\_SP(p)$ do
11:     WRITE <NAME(P)>_SAMPLINGPORT=<NAME(port)>
12:     WRITE <NAME(port)>_MAXMESSAGESIZE=<MAXMESSAGESIZE(port)>
13:     WRITE <NAME(port)>_REFRESHPERIOD=<REFRESHPERIOD(port)>
14:     WRITE <NAME(port)>_DIRECTION=SOURCE
15:   end for
16:   for $\forall port \in DST\_SP(p)$ do
17:     WRITE <NAME(P)>_SAMPLINGPORT=<NAME(port)>
18:     WRITE <NAME(port)>_MAXMESSAGESIZE=<MAXMESSAGESIZE(port)>
19:     WRITE <NAME(port)>_REFRESHPERIOD=<REFRESHPERIOD(port)>
20:     WRITE <NAME(port)>_DIRECTION=DESTINATION
21:   end for
22:   for $\forall port \in SRC\_QP(p)$ do
23:     WRITE <NAME(P)>_QUEUINGPORT=<NAME(port)>
24:     WRITE <NAME(port)>_MAXMESSAGESIZE=<MAXMESSAGESIZE(port)>
25:     WRITE <NAME(port)>_MAXNUMBEROFMESSAGES=<MAXNUMBEROFMESSAGES(port)>
26:     WRITE <NAME(port)>_DIRECTION=SOURCE
27:   end for
28:   for $\forall port \in DST\_QP(p)$ do
```

```
29:       WRITE <NAME(P)>_QUEUINGPORT=<NAME(port)>
30:       WRITE <NAME(port)>_MAXMESSAGESIZE=<MAXMESSAGESIZE(port)>
31:       WRITE <NAME(port)>_MAXNUMBEROFMESSAGES=<MAXNUMBEROFMESSAGES(port)>
32:       WRITE <NAME(port)>_DIRECTION=DESTINATION
33:     end for
34: end for
35: for $\forall channel \in CHANNEL(M)$ do
36:     WRITE CHANNEL_NAME=<NAME(channel)>
37:     WRITE <NAME(channel)>_SOURCE=<SRC(channel)>
38:     for $\forall portname \in DST(channel)$ do
39:       WRITE <NAME(channel)>_DESTINATION=<portname>
40:     end for
41: end for
```

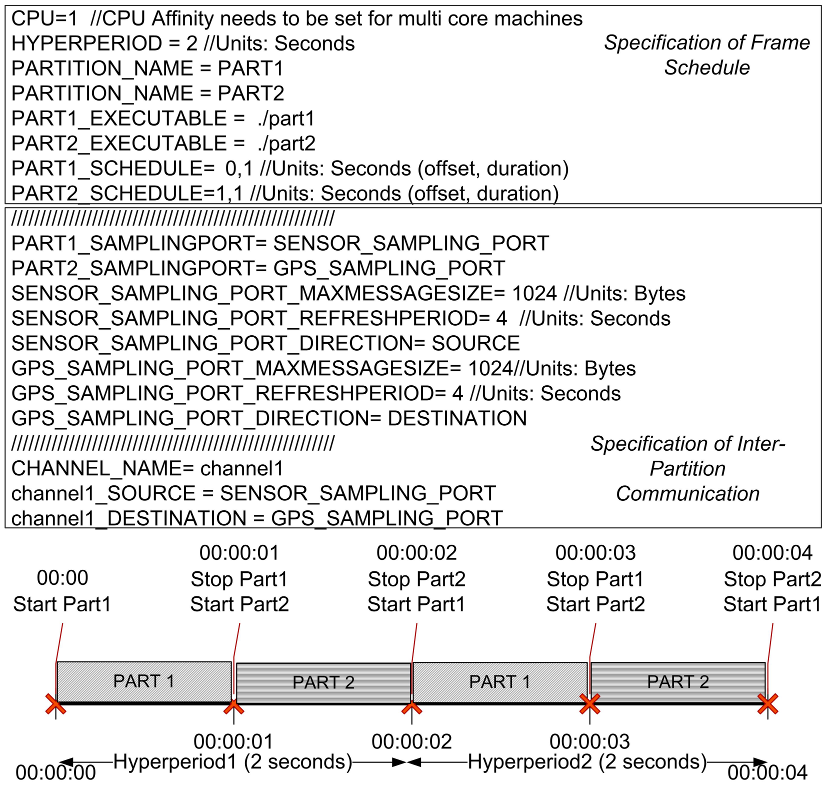
1.3 Invoking Module Manager

Table 1: Algorithm for Module Manager

```
Given: P {Set of all partitions.}
Given: $(\forall p \in P)\ PERIOD(p)$ {period of all partitions}
Given: $(\forall p \in P).\ DURATION(p)$ {duration for which p will run during each execution.}
Given: $H = LCM(period(P))$ {Hyper period value}
Given: $(\forall p \in P).\ OFFSET(p)$ {set of Offsets within a hyper period when p should start execution }
Given: $(\forall p \in P).\ EXECUTABLE(p)$ {relative path to the executable file.}
Given: $(\forall p \in P).\ SRC\_SP(p)$ {set of source sampling ports.}.
Given: $(\forall p \in P).\ SRC\_QP(p)$ {set of source queuing ports}
Given: $CHANNELS$ {inter-partition communication links. $SRC(c)$ is the source port. $DST(c)$ is the set of all destination ports.}
Pre Condition: $\sum_{p \in P} DURATION(p)/PERIOD(p) \le 1$
Begin Module Manager
 1: Set scheduling policy to $SCHED\_FIFO$
 2: Set CPU affinity to a single core.
 3: for $channel \in CHANNELS$ do
 4:     for $p \in P$ do
 5:       if $SRC(channel) \in SRC\_SP(p)$ or $SRC(channel) \in SRC\_QP(p)$ then
 6:         $p.SRCCHANNEL.append(channel)$
 7:       end if
 8:     end for
 9: end for
10: Initialize $OFFSETS$ {$OFFSETS$ is a sequence of tuple <time {value starting from 0}, Partition>}
11: for $p \in P$ do
12:     $OFFSETS.add(OFFSET(p), p)$
13: end for
14: Sort $OFFSETS$ in ascending order based on the time value.
15: for $p \in P$ do
16:     childprocess =fork($EXECUTABLE(p)$)
17:     Wait on handshake from childprocess subject to a timeout
18:     if timeout then
19:       Shutdown Module
20:       Exit
21:     end if
22: end for
23: for entry in $OFFSETS$ do
24:     $T1 \leftarrow currenttime$
25:     $T2 \leftarrow T1 + DURATION(entry.Partition)$
26:     Send $SIGCONT$ to $EXECUTABLE(entry.Partition)$ {SIGCONT is a POSIX signal}
27:     $clock\_nanosleep(T2)$ {Use a high-resolution clock such as clock realtime in LINUX.}
28:     Send $SIGSTOP$ to $EXECUTABLE(entry.Partition)$ {SIGSTOP is a POSIX signal}
29:     for $c \in entry.Partition.SRCCHANNEL()$ do
30:       $c.fire()$ {Move message from source port of the channel to all the destination ports}
31:     end for
32: end for
End Module Manager
```

Follow the directions in the ACM runtime readme to setup your environment. Then you can invoke the module manager `$ModuleManager␣Configuration.cfg`. Once invoked,the module manager checks that the schedule is valid before the system can be initialized i.e. all scheduling windows within a hyperperiod can be executed without overlap. i.e., whether the schedule confirms to constraints C1 to C4 described above. Then, it implements the schedule using the `SCHED_FIFO` policy of the Linux kernel. It is responsible for checking that the schedule is valid before the system can be initialized i.e. all scheduling windows within a hyper period can be executed without overlap. Algorithm 1 describes the functioning of a module manager in pseudo code. Note that module manager is also responsible for transferring the inter-partition messages across the configured channels.

1.3.1 Example



```
CPU=1  //CPU Affinity needs to be set for multi core machines
HYPERPERIOD = 2 //Units: Seconds
PARTITION_NAME = PART1
PARTITION_NAME = PART2
PART1_EXECUTABLE =  ./part1
PART2_EXECUTABLE =  ./part2
PART1_SCHEDULE=  0,1 //Units: Seconds (offset, duration)
PART2_SCHEDULE=1,1 //Units: Seconds (offset, duration)
/////////////////////////////////////////////////////////
PART1_SAMPLINGPORT= SENSOR_SAMPLING_PORT
PART2_SAMPLINGPORT= GPS_SAMPLING_PORT
SENSOR_SAMPLING_PORT_MAXMESSAGESIZE= 1024 //Units: Bytes
SENSOR_SAMPLING_PORT_REFRESHPERIOD= 4  //Units: Seconds
SENSOR_SAMPLING_PORT_DIRECTION= SOURCE
GPS_SAMPLING_PORT_MAXMESSAGESIZE= 1024//Units: Bytes
GPS_SAMPLING_PORT_REFRESHPERIOD= 4 //Units: Seconds
GPS_SAMPLING_PORT_DIRECTION= DESTINATION
/////////////////////////////////////////////////////////
CHANNEL_NAME= channel1
channel1_SOURCE = SENSOR_SAMPLING_PORT
channel1_DESTINATION = GPS_SAMPLING_PORT
```

Specification of Frame Schedule
Specification of Inter-Partition Communication
00:00 Start Part1
00:00:01 Stop Part1 Start Part2
00:00:02 Stop Part2 Start Part1
00:00:03 Stop Part1 Start Part2
00:00:04 Stop Part2 Start Part1
PART 1
PART 2
PART 1
PART 2
00:00:00
00:00:01
00:00:02
00:00:03
00:00:04
Hyperperiod1 (2 seconds)
Hyperperiod2 (2 seconds)

Figure 1: An example module configuration and the time line of events as they occur.

Figure 1 shows the configuration file for a module with two partitions, $PART1$ and $PART2$. Their executables are `./part1` and `./part2` respectively. The hyper period value is 2 $seconds$. $PART1$ has an offset and duration of 0 and 1 second respectively. $PART2$ has an offset and duration of 1 and 1 second respectively. Note that the effective period of both partitions is 2 seconds in this example. $PART1$ has a source sampling port called $SENSOR\_SAMPLING\_PORT$. Its max message size is 1024 bytes, refresh period is 4 seconds. $PART2$ has a destination port, $GPS\_SAMPLING\_PORT$. Its max message size is 1024 bytes, refresh period is 4 seconds. We have one channel called $channel1$. It connects $SENSOR\_SAMPLING\_PORT$ to $GPS\_SAMPLING\_PORT$. Figure 1 also shows the example execution time line of this module. Note that in this example, the channel will be fired at the end of each cyclic execution of partition $part1$. You can find more examples of configurations files in various demos included with the ACM Runtime Environment.