TiCOS Code Generator README

Introduction

The goal of the code generator is to ease the process of configuring an application running on TiCOS. To this end, the code generator automatically generates all files required to compile a TiCOS example. It takes as input an XML file containing information about the example you want to build and generates a complete "generated-code" directory example.

Installation and usage

The Code Generator use the GNOME XML Library (Libxml2). You can download and install it from http://www.xmlsoft.org/

Download the Code-Generator directory. In this directory you can find:

- A directory (xml) containing some XML file examples and the .xsd file used by the code generator to validate the XML files;
- The Makefile used to build the code generator;
- The src directory;

To compile the code generator program just type:

make

To generate the "generated-code" directory (and all files required to run an example) just type:

./code-gen xml/[file.xml]

The file.xml should contain the description of the example you want to build: in this XML document you can specify some parameters such as the number of partitions, the ports, etc. (for a complete description of the XML elements see the next section). This XML document should be valid with respect to the XML schema contained in the XSD file.

The code generator performs some checks on some value specified in the XML files (e.g., it checks if the user stack value is a multiple of 4 or if the every source port has a destination port, etc.); if no error occurs it generates the "generated-code" directory.

You can copy the "generated-code" directory in your TiCOS/examples/<example-name> directory, compile and running it on the PROARTIS_sim.

XML file description

You should use the xml file to configure your example. You can specify 4 types of settings:

- Hardware settings (optional)
- Kernel settings (optional)
- Partitions settings
- Ports settings (optional)

In case optional settings are not specified, the default values specified in the src/include/arinc653-xml-conf.h are used.

Hardware settings

You can specify the bus frequency, the frequency div and the frequency shift. If omitted the default value (74,1,0) are used .

Kernel settings

You can specify the idle stack, the user stack and the priority level. If omitted the default value 4096, 8192, and 10 are used.

It is not possible to specify:

- debug directives (i.e. the cod-gen program does not generate any DEBUG-related macro definition);
- which simulator (PROARTIS sim or QEMU) the application will be run on (i.e. the cod-gen program generate an application example running on PROARTIS sim. If you want to run it on QEMU you have to add the macro POK_NEEDS_QEMU_SETUP in the file *generated-code/cpu/kernel/deployment.h*).

Partitions settings

For every partition you should specify:

- The scheduler type (optional, if not specified the O1 scheduler is used);
- The partition's threads (see later); you should not specify the partition main thread;
- The partition size;
- The partition base address and load address (optional if you specify one, two or three partitions);
- The partition slot. Only one slot can be specified for partition (not taking into account the sub-slot needed for the postwrite operations). You should also specify the slot duration, the slot position (inside the major frame) and the duration of postwrite sub-slot if any; you can not specify which ports can be flushed or preloaded at the end or beginning of every partition (i.e. the "pok_outputports_to_flush" and the "pok_inputports_to_preload" arrays will be automatically built taking into account all the ports of the partition.
- The number of asynchronous events (optional, the default value is 0);
- The max size that blackboards messages can have (this value is referred to all the blackboards of the partition);
- The max size that buffer messages can have and the max number of messages that a buffer can contain (these values are referred to all the buffers of the partition);

It is not possible to specify:

- Aperiodic threads (aperiodic threads used to split the threads containing blocking functions are automatically generated);
- debug directives (i.e. the cod-gen program does not generate any DEBUG-related macro definition).

For each thread you can specify:

- the priority level (required);
- a port/blackboard/buffer/event and the action the thread should perform on the associated object (set/wait for events, display/read for blackboards, send/receive for buffers).
    - In case the O(1) scheduler is specified, the generator will split a thread that executes an action on event/blackboard/buffer in two threads, one periodic (having the priority you specified) and one aperiodic (having the immediately lower priority value). For instance, if you specified a thread with priority 9 that WAITs for an event, the generator will create two threads, one (periodic) having priority 9, and one (aperiodic) having priority 8. As a consequence, in this case, when you specify the priority you have to remember that a thread executing an action on event/blackboard/buffer will be splitted in two threads and that in fact will use two priority levels, the one you specify and the one immediately lower.
    - In case the O(1)-split scheduler is specified, it is possible to associate to threads only events. In this case only threads with a potentially blocking action (i.e. WAIT) will be split in two threads, one periodic and one aperiodic (with no priority level associate to it).

Restrictions:

- The priority you assigned to the threads should be greater or equal than 2 (being 0 and 1 priority levels assigned to partition main threads and the idle thread).
- Although it should be possible to specify more than one communication object (port/blackboard/buffer/event) for each thread, in practice this option has not been tested yet.
- Currently, the O(1)-split scheduler cannot be specified if the threads of the partition executed action on ports/blackboards/buffers.

Ports settings

For every port you should specify:

- The port name (optional);
- The port id;
- The port kind (“sampling” or “queueing”);
- The port direction (“source” or “destination”);
- For every source port, the ids of the destination ports (a source sampling port can have only one destination port);

The partition to which the port belongs to is deduced from the thread information (the ids of one or more ports can be specified for every thread);

Other information

Only examples with one, two and three partitions have been generated and tested. If you want to generate code for more than three partitions just specified the base virtual address and the load address of the partitions.

The *code-gen* program settings along with the default values that are assigned to the some macro can be found in the file *src/include/arinc653-xml-conf.h*.