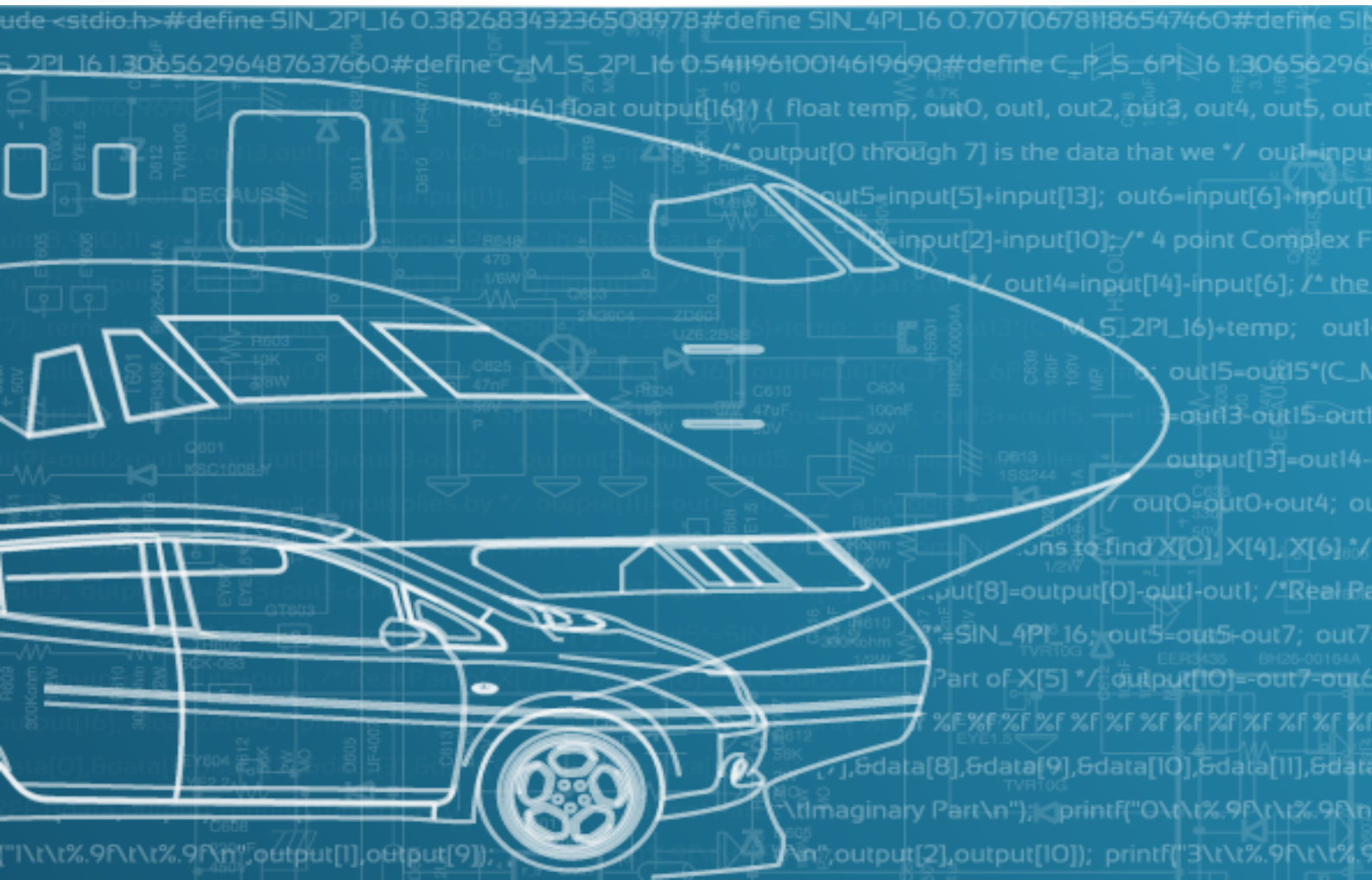
ASTERIOS

KRONO-SAFE

Safe design in real-time

Temporal Variables

Rev K19.5



HOW TO CONTACT US

Corporate headquarters	**KRONO-SAFE** Support 16 avenue Carnot, F - 91300 MASSY
Support contact	Customer Care Portal

VERSION

Document	Temporal Variables
Revision	K19.5
Date	May 16, 2024

COPYRIGHT

Contents

This example introduces Temporal Variables and how they can be used to communicate between PsyC Tasks (Agents and Jobs).

Keywords: `temporal`, `display`, `consult`, `$[<uint>]`

1 Content of the Application

The Application built in this example simulates the acquisition of a message, its decoding and the displaying of the decoded message.

- `input_driver.psy` contains the definition of a Worker `sensor_driver` which Job `j0` is in charge of transferring input data to computation Agents through the Temporal Variable `tv_raw_input_data`.
- `decode.psy` contains the definition of an Agent `decoder` in charge of interpreting the raw input data received from `tv_raw_input_data`. This Agent displays the result of its interpretation in a Temporal Variable `tv_decoded_msg`.
- `filter.psy` contains the definition of an Agent `filter` that consults the decoded data in `tv_decode_msg` and displays in `tv_do_display` whether the decoded message should be printed or not.
- `displayer_driver.psy` contains a Worker `displayer` which Job `print` consults the decoded message in `tv_decoded_msg` and print its content if `tv_do_display` indicates it to do so.

2 A Short Introduction to Temporal Variables

A Temporal Variable is a sampled C variable displayed by a Task to other Tasks. The use of this communication mean is detailed in the *PsyC Language Description* in the section *Communication Means*.

Let us remind several key principles that are necessary to understand the exchanges performed in this example.

Temporal Variables are based on a Clock. A sample of the variable can be accessed for each tick of this base Clock.

The consulting Agents can access to the variable's value according to the following *visibility principle*: an Agent can access a sample if and only if it is timestamped at a date preceding its current Earliest Start Date. This date from which samples can be consulted is named the *visibility horizon* of the Agent.

When the displaying Agent modifies a Temporal Variable, the first sample that contains the effect of this modification is timestamped at the first tick of the base Clock of the Temporal Variable following the current Deadline of the displayer. This date at which the modification of the variable is made visible is named the *publication horizon* of the displayer.

Since Jobs are executed “on” a tick, their *visibility horizon* or their *publication horizon* is not defined the same way:

- The *publication horizon* of the `output` Jobs is the date of the tick “on” which they are executed. This is illustrated by the samples produced by the Job `sensor_driver_j0` (see the figure in the next section). The compiler (`psyko`) and the Asterios RTK ensures that all the tasks that may access these samples will be able to see them if their Earliest Start Date is after the tick “on” which the Job is executed. In this example, it is the case for the Agent `decoder`.
- The *visibility horizon* of the `input` Jobs is the date of the tick “on” which they are executed. This is illustrated by the samples consulted by the Job `displayer_print` (see the figure in the next section). *psyko* and the *ASTERIOS Real-Time Kernel* ensure that all the Agents that may make visible these samples will be executed before the `input` Job. In this example, it is the case for the Agents `decoder` and `filter`.

Warning: The execution “on” a tick of a Job only means that this Job is seen executed “on” this tick by the Agents with which it communicate. This does not state when the Job is physically executed; it only means that:

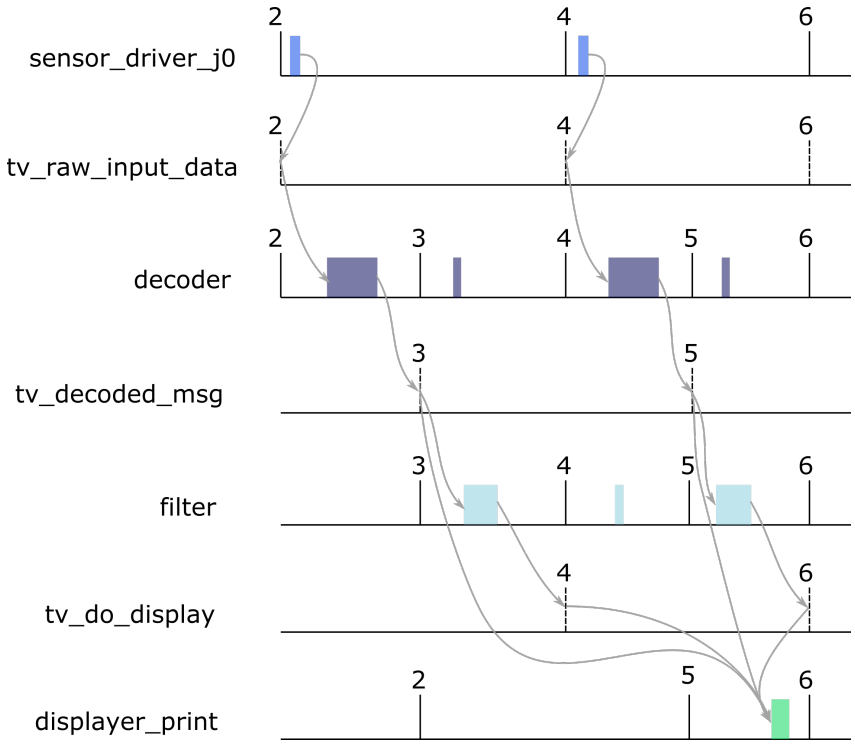
- an “input” Job is allowed to consult all the data visible at the tick “on” which the Job is executed;
- an “ouput” Job is allowed to produce data that will be visible at the tick on which the Job is executed.

3 Communication Pattern

The Application in this example illustrates:

- the publication from a Job and from an Agent: the Job `sensor_driver_j0` displays `tv_raw_input_data` and the Agent `decoder` displays `tv_decode_msg` for instance;
- the consultation from Agents and from Jobs: the Agent `decoder` consults `tv_raw_input_data` and the Job `displayer_print` consults `tv_do_display` for example;
- the consulting of one Temporal Variable by two Tasks: `decoder` and `displayer_print` consult `tv_decoded_msg`;
- the consulting of several Temporal Variables by a Task: `displayer_print` consults `tv_do_display` and `tv_decoded_msg`;
- the consulting of several samples of the same Temporal Variable in a given Elementary Action: `displayer_print` consults two items of each of the Temporal Variables to which it accesses.

The communication pattern of the Application is the following:


sensor_driver_j0
tv_raw_input_data
decoder
tv_decoded_msg
filter
tv_do_display
displayer_print
2
3
4
5
6