Floating Point Exceptions in Art and Larsoft

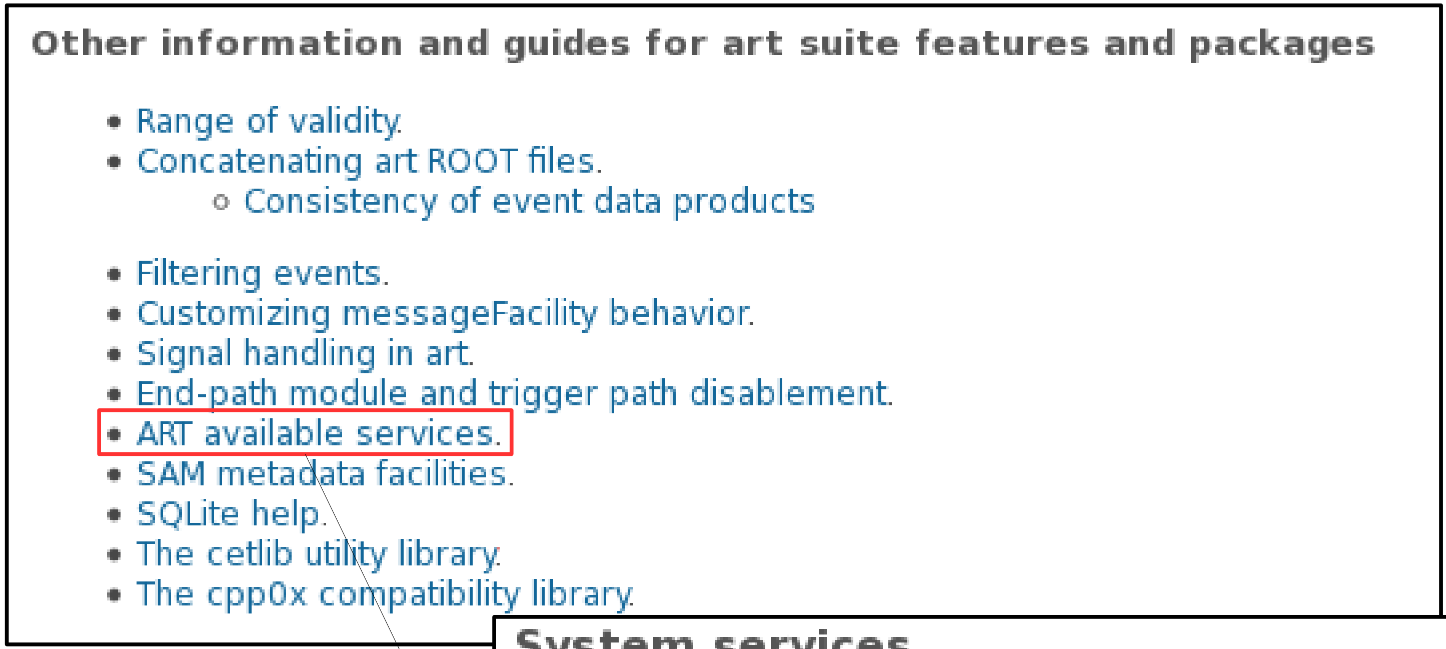
Larsoft Coordiantion Meeting
June 27, 2017

H. Greenlee

Overview

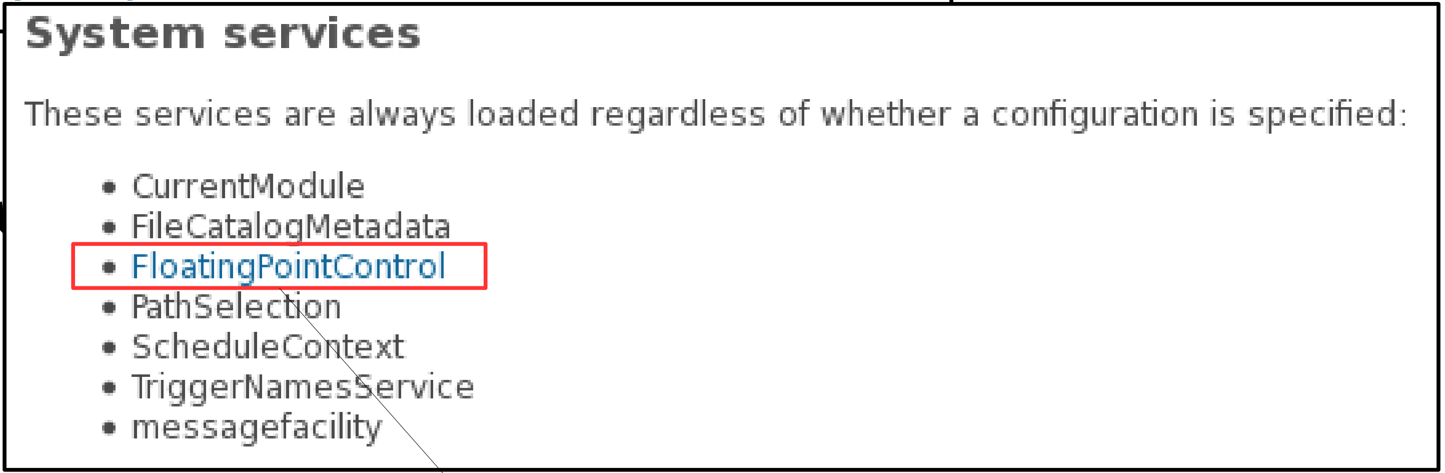
- I became aware (or was reminded) of the FPE issue after some recently observed crashes due to larpandora code doing validity checks on floating point values while importing data from art (recob::Hit) to pandora framework.
  - Exposed existence of nan's in recob::Hit objects.
- How art handles FPE's.
  - floating_point_control art built in service.
  - Default vs. ideal behavior.

Floating_point_control Service on Art Wiki



Other information and guides for art suite features and packages

- Range of validity.
- Concatenating art ROOT files.
  - Consistency of event data products
- Filtering events.
- Customizing messageFacility behavior.
- Signal handling in art.
- End-path module and trigger path disablement.
- ART available services.
- SAM metadata facilities.
- SQLite help.
- The cetlib utility library.
- The cpp0x compatibility library.



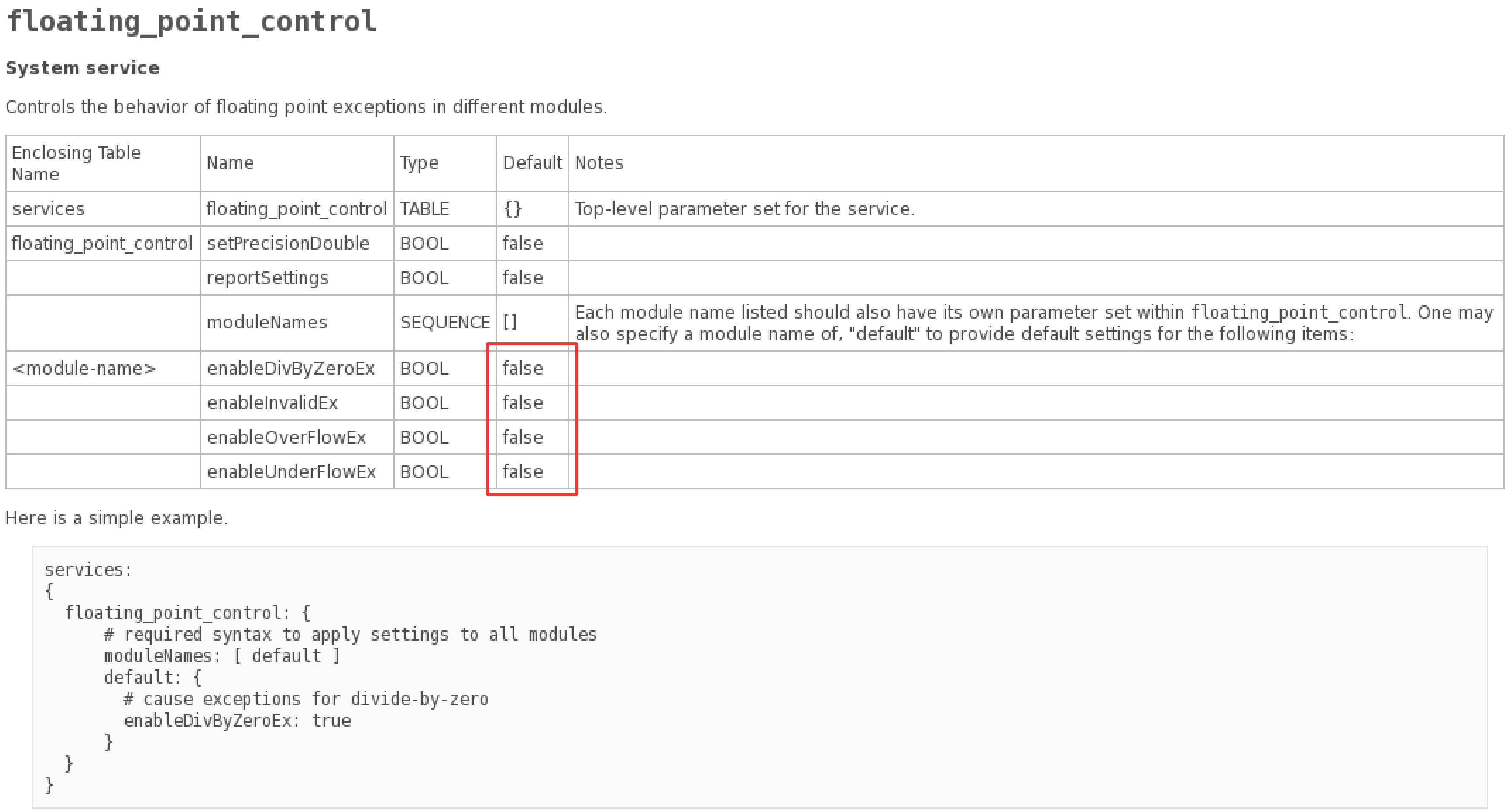
System services

These services are always loaded regardless of whether a configuration is specified:

- CurrentModule
- FileCatalogMetadata
- FloatingPointControl
- PathSelection
- ScheduleContext
- TriggerNamesService
- messagefacility

Next page

Floating_point_control on Art Wiki



floating_point_control

System service

Controls the behavior of floating point exceptions in different modules.

Enclosing Table Name	Name	Type	Default	Notes
services	floating_point_control	TABLE	{}	Top-level parameter set for the service.
floating_point_control	setPrecisionDouble	BOOL	false	
	reportSettings	BOOL	false	
	moduleNames	SEQUENCE	[]	Each module name listed should also have its own parameter set within floating_point_control. One may also specify a module name of, "default" to provide default settings for the following items:
<module-name>	enableDivByZeroEx	BOOL	false	
	enableInvalidEx	BOOL	false	
	enableOverFlowEx	BOOL	false	
	enableUnderFlowEx	BOOL	false	

Here is a simple example.

```
services:
{
  floating_point_control: {
      # required syntax to apply settings to all modules
      moduleNames: [ default ]
      default: {
        # cause exceptions for divide-by-zero
        enableDivByZeroEx: true
      }
  }
}
```

Default and Desired FPE Handling

- By default, art programs ignore FPE's.
  - Is that the configuration we want? Probably not.
- If enabled, FPEs geneate a SIGFPE signal.
- Here is what I would consider an ideal configuration.

```
services.floating_point_control: {
  reportSettings: true
  moduleNames: [ default ]
  default: {
    enableDivByZeroEx: true
    enableInvalidEx: true
    enableOverFlowEx: true
    EnableUnderFlowEx: false
  }
}
```

Some Concluding Thoughts

- How are other experiments handling FPEs?
- If we (MicroBooNE) turn on FPEs by default now, we will certainly find that our code is shot through with unsuspected FPEs.
  - Verified experimentally.
- Getting to the point where running with FPEs enabled for all modules will require a campaign to eradicate FPEs.
- Should such a campaign be mounted?
  - Yes, in my opinion.
- Who will participate? Can larsoft team help?
- It would be nice if there was a SIGFPE handler that could print a traceback (feature request?).