Fridley Test Cell Interface with Horiba

Version 6

February 19, 2024

Developed by Transportation Laboratories

Version History

Version	Date	Revision Description
1	1/25/2016	Initial publication
2	8/23/2018	Format with SGS brand
3	5/4/2020	Retrofit to new template, revise tense to current where applicable
4	12/6/2021	Removed *Section 3 Using Processes and Utilities* that contained inline usage for `serFrdly` and `iQueueMgr` and added hyperlinked cross-references to their usage help on cyflex.com in *Section 2.1 serFrdly* on page 5 and *Section 2.2 iQueueMgr* on page 5.
5	6/7/2022	Updated all hypertext linked cross-references to cyflex.com usage help descriptions
6	2/19/2024	Rebrand to TRP Laboratories

Document Conventions

This document uses the following typographic and syntax conventions.

- Commands, command options, file names or any user-entered input appear in Courier type. Variables appear in Courier italic type.

  Example: Select the `cmdapp-`*`relVersion-buildVersion`*`.zip` file….

- User interface elements, such as field names, button names, menus, menu commands, and items in clickable dropdown lists, appear in Arial bold type.

  Example: **Type**: Click **Select Type** to display drop-down menu options.

- Cross-references are designated in Arial italics.

  Example: Refer to *Figure 1*…

- Click intra-document cross-references and page references to display the stated destination.

  Example: Refer to *Section 1 Overview* on page 1.

  The clickable cross-references in the preceding example are *1*, *Overview*, and on page 1.

CyFlex Documentation

CyFlex documentation is available at https://cyflex.com/. View **Help & Docs** topics or use the **Search** facility to find topics of interest.

Table of Contents

List of Figures

LIST OF TABLES

1 Overview

The testing process of gensets at CPGF/Fridley includes emissions measurement. Multiple test cells need access to the single emissions bench as part of their testing process. Processes translate the current emissions requests, which follow the AK protocol in such a way that the required information can be obtained from the Horiba bench. Since multiple test cells need to have access to the emission bench, a queuing process manages test cell access to the Horiba bench. All of these processes reside on a system that is referred to as the multiplexer node. This document describes this translation process and the queuing process that are part of the multiplexer node.

2 Interface Approach

A subset of the AK protocol messages are translated. In addition, there are several requests unique to the queuing process are also be processed. The list of supported commands is described later in this section.

The AK commands from the test cell are transferred to the multiplexer node over a serial port via the process ASC_Frdly. The interface between the Emissions bench and the multiplexer node is a TCP IP interface. The commands for the Emissions bench are transferred using the MsgProtocol standard service.

All commands for the test cell are processed by the task serFrdly. This task determines what kind of request is being made and passes it on to the appropriate task.

- If the command is associated with obtaining access to the Horiba bench, it is sent to the iQueueMgr queue manager.
- If the test cell has been granted access to the Horiba bench and the command is associated with an Emissions bench operation, the command is transferred to gp_test.

Figure 1 on page 3 depicts the interface between the test cell, the system performing the translation (multiplexer node), and the Emissions bench.

Figure 2 on page 4 shows a more detailed diagram of the interface.

Figure 1: interface between the Emissions bench and Multiplexer Node

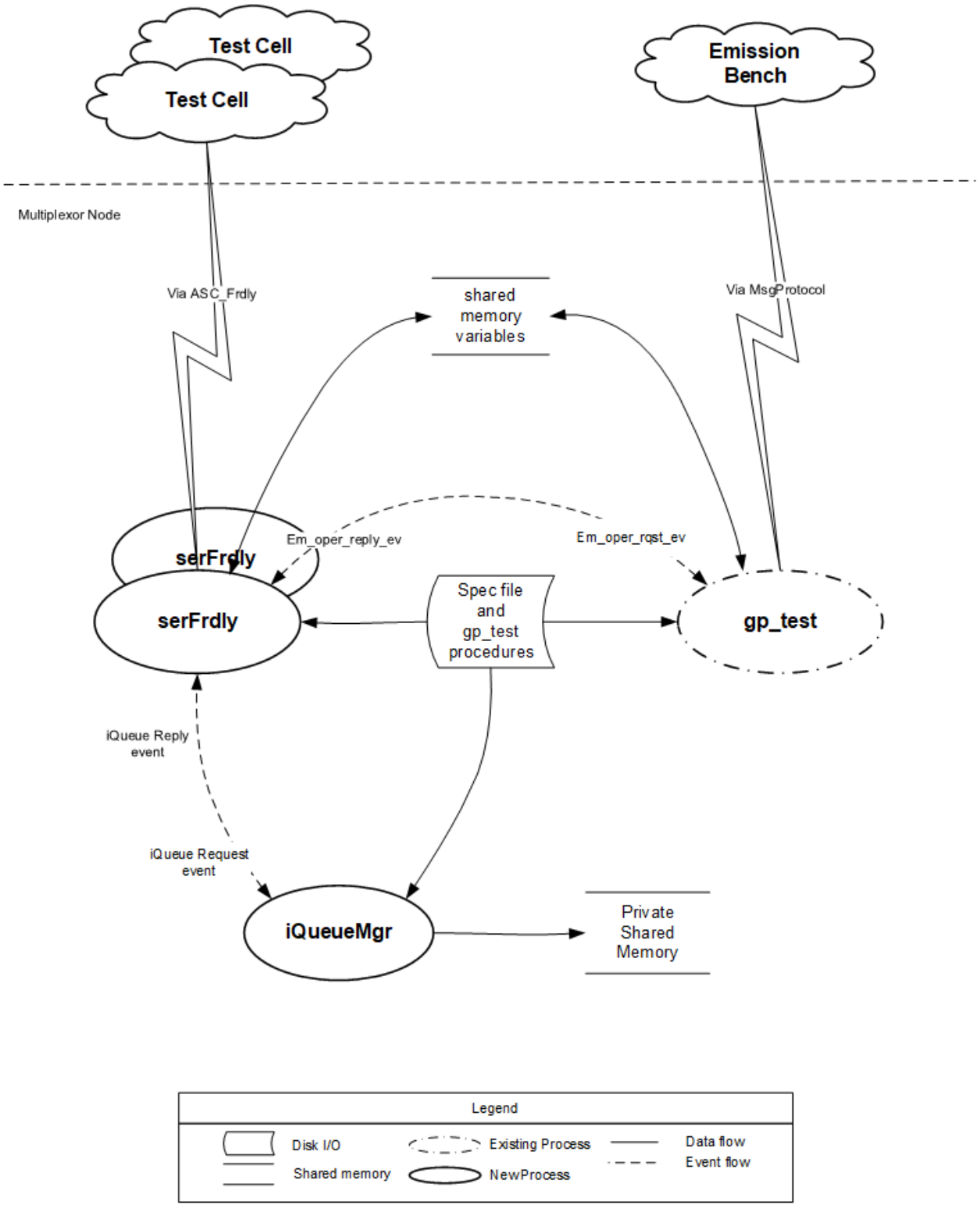
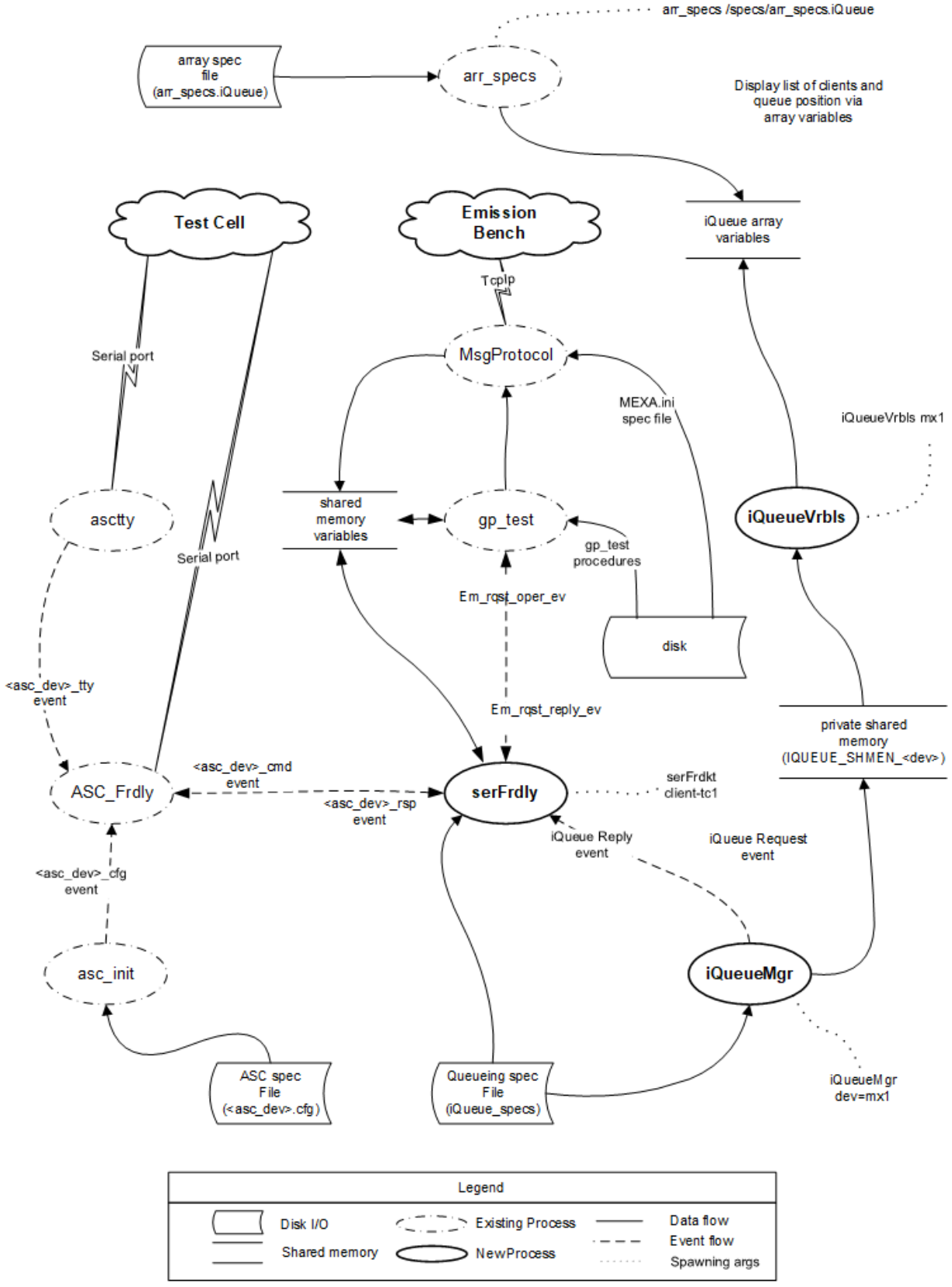


Figure 2: Fridley Test Cell interface with Horiba Emissions Bench



The following sub-sections describe the processes and utilities shown in *Figure 1* on page 3.

2.1 serFrdly

There are three possible arguments to `serFrdly` when it is spawned:

1. The name of the client being serviced
2. The name of the specification file
3. The root string assigned to the serial port connection

The only required argument is the name of the client being serviced For details of the spawning arguments, refer to cyflex.com usage help for `serFrdly`.

The name of the client must agree with a list of clients specified in the specification file. Refer to *Section 3 Specification File* on page 8 for an example specification file. There is an instance of `serFrdly` on the multiplexer node for each client (test cell) needing access to the Emissions bench.

The interface between the client and `serFrdly` is controlled by the `ASC_Frdly` serial port interface. CyFlex events are sent/received between `serFrdly` and `ASC_Frdly`. `ASC_Frdly` is then responsible for the transfers between the client and the multiplexer node.

The communication between the `iQueueMgr` queue manager and `serFrdly` is via CyFlex events. The event `IQUEUE_REQUEST_EVENT` is received by the manager and the event `IQUEUE_REPLY_EVENT` is received by `serFrdly`.

As mentioned above, the interface between the Emissions bench and the multiplexer node is a TCP IP interface that is managed by the `MsgProtocol.` standard service. The communication between `serFrdly` and `MsgProtoco` is a `gp_test` procedure developed for this purpose. As a result, `serFrdly` must communicate with `gp_test`, which is accomplished via CyFlex events and shared memory variables. The events are used to indicate that an Emission bench operation is being requested and shared memory variables contain the particular operation requested. Specify the names of these events and variables in the specification file used by `serFrdly` under the keyword `@EMISSION_REQUEST_GP_TEST_INFO`.

2.2 iQueueMgr

The spawning arguments for the `iQueueMgr` queue manager are the name of the device being managed, the name of the specification file, and a flag that indicates if shared memory should be initialized. If memory is not initialized, then any existing request for the bench will remain active. The only argument that is required is the name of the device being managed. The name of the device being managed must be one of the devices listed in the specification file. Refer to cyflex.com usage help for detail on spawning `iQueueMgr`.

When `iQueueMgr` receives a request for access to the device, it checks to see if the device is available and if it is, it gives the requestor access to the device. If it is not available, the clients name is placed in the queue for the device and it returns the queue position to the requestor. When the device is released by the current user, the next client in line automatically receives access to the device. In addition, a private shared memory area is updated that contains the list of clients for the device being managed.

2.3 iQueueVrbls

The `iQueueVrbls` utility allows the display of the queue for a given device. The utility accesses the private shared memory area, updated by `iQueueMgr` and sets variables with the information about the device queue. These variables then can be displayed on a display screen of the multiplexer node.

The variable names updated by `iQueueVrbls` are hard-coded. As a result, if different variables are desired, the utility application will need to be rebuilt. The hard-coded variables are created via two additional spec files, `arr_specs.iQueue` and `gen_labels.iQueue`.

2.4 gp_test

As previously mentioned, the `gp_test` process controls the Emission Bench. This is accomplished via various test procedures; however, `gp_test` does not send commands to the Bench directly That function is handled by the `MsgProtocol` standard service.

2.5 Accessing the Bench

The Bench access process follows:

1. The test cell issues a request, .e.g. SMGA (sample)
2. The request is received by the process `serFrdly` and it:
    a. Sets a variable with the new request
    b. Signals gp_test that a new request has been made
3. `gp_test` executes the test procedure associated with the new request.
4. The test procedure sends the request to one of the processes that make up `MsgProtocol`. Ultimately, the request is received and processed by the Bench.

Note:

Several processes make up the complete `MsgProtocol` process.

These actions are repeated for each request that accesses the Bench. Refer to *Section 2.6.2 AK Requests* on page 7 for the test procedure that is associated with each of the supported requests.

2.6 Supported Requests

2.6.1 Non-AK Requests

Table 1: Non-AK Requests

Request	Description
SREQ	Request control of Bench
SRQP	Priority request control of Bench
AQUE	Request position in queue
SABT	Release control of Bench

2.6.2 AK Requests

Table 2: AK Requests

Request	Description	Associated `gp_test` Procedure Located in `/specs/gp/mexamain/`
SDRN	Trap Drain	`gp_PurgeAll`
SENT	Select Sample Point	`gp_SelectSampleLine`
SMGA	Start Sampling	`gp_MeasureAll`
SARE	Auto-Range ON	`gp_SetRanges`
SARA	Auto-Range OFF	`gp_FixRanges`
STBY	Standby	`gp_StandbyAll`
SATK	Auto Zero/Span Calibration	`gp_CalAll`
AKON	Request Concentration*	
SSPL	Purge	`gp_PurgeAll`
AEMB	Request Analyzer Range	`Gp_GetRange`

***** The request for concentration does not directly access the Bench. The Bench is configured to stream the concentration values directly to variables on the multiplexer node. When a request for concentration is received, the current value of the variable on the multiplexer node is returned to the test cell.

3 Specification File

```
@DEVICES
# num devices
   2
# device names
   mx1
   mx2

@CLIENTS_SAMPLE_PTS
         # if a client has more than one sample point, there is a line
         # entered for each sample point.
# client         sample  sample pt     valid devices
# name           point     vrbl        for sample point

   tc1            7      MxLineSelect         mx1
   tc2            2      MxLineSelect         mx1
   tc3            3      MxLineSelect         mx1


                  # the variable name is constructed
                  # from the following root and the device
                  # names that were specified above.
                  # eg. 'select_dev_mx1'

@INDEX_VARIABLE_ROOT
# root name of variable
  select_dev

@EMISSION_VARIABLES
# request code    variable name
#    number
   1              LCO_ppm
   2              HCO_ppm
   3              CO2_ppm
   5              O2_ppm
   6              NOX_ppm
   8              HHC_ppm
$

@RANGE_VARIABLES
#  request code            calculation for range value
#     value
      1              “MxActive_rng:Line1:HCO * MxAna_sel:Line1:HCO”
      2              “MxActive_rng:Line1:CO2 * MxAna_sel:Line1:CO2”
      3              “MxActive_rng:Line1:LCO * MxAna_sel:Line1:LCO”
      4              “MxActive_rng:Line1:O2  * MxAna_sel:Line1:O2”
      5              “MxActive_rng:Line1:NO  * MxAna_sel:Line1:NO”
      6              “MxActive_rng:Line1:THC * MxAna_sel:Line1:THC”
      7              “MxActive_rng:Line1:O2  * MxAna_sel:Line1:O2”
```

```
     8              “MxActive_rng:Line1:NOx * MxAna_sel:Line1:NOx”

@EMISSION_REQUEST_GP_TEST_INFO
#  vrbl name      -  This is the string variable that the serFrdly
#                    process will set with the current request from
#                    the Test cell.
#  rqst event     -  The name of the event that is set by serFrdly and
#                    received by gp_test when an emission request
#                    command is received.
#  reply varbl    -  The name of the variable that will contain the
#                    status of gp_test processing a command
#  reply event    -  The name of the event that is set by gp_test when
#                    a requested command is complete.
#  rqst_oper_args -  Contains the arguments that were received for the
#                    particular request being processed.

#
#vrbl name    rqst event       reply varbl  reply event      rqst_oper_args
 Em_rqst_oper Em_rqst_oper_ev    Status_GP_MX  Em_rqst_reply_ev    Em_rqst_args
```