**APIRI Intelight/Working Group Proposals**

DRAFT – 05/27/2014

# ATC API Surface Updates

Add Transaction to fio\_fiod\_outputs\_set

The current fio\_fiod\_outputs\_set function creates a race condition for apps that require simultaneous processing of updates to multiple IO modules.

This proposed change addresses this issue by adding two additional functions:

* fio\_fiod\_begin\_outputs\_set(FIO\_APP\_HANDLE app\_handle)
* fio\_fiod\_commit\_outputs\_set FIO\_APP\_HANDLE app\_handle)

These two functions add a transaction like functionality to the fio\_fiod\_outputs\_set mechanism to allow an application to update multiple IO modules before the outputs will be written.

The following calling pattern would be utilized by an application that would like to update multiple IO modules:

* fio\_fiod\_begin\_outputs\_set(my\_app\_handle)
* fio\_fiod\_outputs\_set(my\_app\_handle, device\_1, …)
* fio\_fiod\_outputs\_set(my\_app\_handle, device\_2, …)
* fio\_fiod\_outputs\_set(my\_app\_handle, device\_3, …)
* fio\_fiod\_commit\_outputs\_set(my\_app\_handle)

The FIOMAN will not use any outputs set after the fio\_fiod\_begin\_outputs\_set call until the fio\_fiod\_commit\_outputs\_set is called by the application. If an application does not require a transaction on outputs set, and the fio\_fiod\_begin\_outputs\_set is not called, the fio\_fiod\_outputs\_set will take effect immediately as currently specified.

The following describes the detail of the two new proposed functions:

**NAME**

**fio\_fiod\_begin\_outputs\_set** – Begins a FIOD output points transaction.

**SYNOPSIS**

**#include <fio.h>**

**int fio\_fiod\_begin\_outputs\_set(FIO\_APP\_HANDLE** *app\_handle***)**

**DESCRIPTION**

This function is used to start a FIO output points transaction for a given application. The data in any subsequent fio\_fiod\_outputs\_set call will be buffered by the FIO manager and not sent to any devices until fio\_fiod\_commit\_outputs\_set is called.

*app\_handle* is a **FIO\_APP\_HANDLE** returned by a previously successful **fio\_register**(3fio) call.

**RETURN VALUES**

Upon successful completion, 0 is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

Error codes returned in *errno*:

**EINVAL** The app\_handle argument is invalid or the app already has a pending transaction

**NOTES**

An application is not required to call fio\_fiod\_begin\_outputs\_set to set a FIOD outputs. If this function is never called all calls to fio\_fiod\_outputs\_set will update the outputs without requiring fio\_fiod\_commit\_outputs\_set to be called.

The transaction started by fio\_fiod\_begin\_outputs\_set will not timeout.

**NAME**

**fio\_fiod\_commit\_outputs\_set** – Commits a FIOD output points transaction.

**SYNOPSIS**

**#include <fio.h>**

**int fio\_fiod\_ commit \_outputs\_set(FIO\_APP\_HANDLE** *app\_handle***)**

**DESCRIPTION**

This function is used to commit a FIO output points transaction for a given application.

*app\_handle* is a **FIO\_APP\_HANDLE** returned by a previously successful **fio\_register**(3fio) call.

When an application calls fio\_fiod\_ commit\_outputs\_set any pending FIOD outputs will be processed by the FIO manager and sent to the corresponding devices.

**RETURN VALUES**

Upon successful completion, 0 is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

Error codes returned in *errno*:

**EINVAL** The app\_handle argument is invalid or the app does not have a pending transaction.

**NOTES**

An application is not required to call fio\_fiod\_ commit \_outputs\_set to set a FIOD outputs if fio\_fiod\_ begin \_outputs\_set was never called. If this function is never called all calls to fio\_fiod\_outputs\_set will update the outputs without requiring fio\_fiod\_ commit \_outputs\_set to be called.

## Add Frame Sent Callback

The current API standard provides a mechanism for an application to be notified when a response frame is received, but it does not provide a notification for an application for when an output frame is sent. Without this mechanism it is difficult for an application to synchronize FIOD output updates with the underlying FIO manager frame writes.

This proposed change addresses this issue by adding three additional functions and 1 new signal.

* fio\_fiod\_frame\_send\_notify\_deregister
* fio\_fiod\_frame\_send\_notify\_register
* fio\_query\_frame\_send\_notify\_status
* FIO\_SIGIO\_SENT

The following describes the detail of the two new proposed functions:

**NAME**

**fio\_fiod\_frame\_send\_notify\_deregister** – Deregister a notification request for when a frame is sent.

**SYNOPSIS**

**#include <fio.h>**

**int fio\_fiod\_frame\_send\_notify\_deregister( FIO\_APP\_HANDLE** *app\_handle***,  
FIO\_DEV\_HANDLE** *dev\_handle***,  
unsigned int** *tx\_frame* **)**

**DESCRIPTION**

This function is used to deregister a notification request for when a command frame is sent.

* *app\_handle* is a **FIO\_APP\_HANDLE** returned by a previously successful **fio\_register**(3fio) call. *dev\_handle* is a **FIO\_DEV\_HANDLE** returned by a previously successful **fio\_fiod\_register**(3fio) call. t*x\_frame* is a valid frame number / type for which notification has been registered, using the fio\_query\_frame\_send\_notify\_status(3fio) call. Valid frame numbers/types are in the range 128 - 255.

Notification is generated to the application program utilizing the **FIO\_SIGIO\_SENT** (**SIGRTMIN** + 5) real-time signal. The application program using this service must establish a **FIO\_SIGIO\_SENT** handler, prior to calling this function. The default handling for a **FIO\_SIGIO\_SENT** real-time signal is to terminate the process. When the indicated *rx\_frame* is received or declared in error by the FIO API, the FIO API will generate a **FIO\_SIGIO\_SENT** real-time signal to the waiting application program. The application program must then perform a **fio\_query\_frame\_send\_notify\_status**(3fio) call to discover why a **FIO\_SIGIO\_SENT** real-time signal was generated. See **fio\_query\_frame\_send\_notify\_status**(3fio) for further details.

An application program may register notifications for multiple response frames.

**FIO\_SIGIO\_SENT** is defined as:

**#define FIO\_SIGIO\_SENT (SIGRTMIN + 5)**

**RETURN VALUES**

Upon successful completion 0 is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

Error codes returned in *errno*:

**EINVAL** One or more arguments are invalid.

**ENOMEM** There is not enough memory available for this operation.

**EACCESS** The Frame Notification Service is not currently enabled.

**NAME**

**fio\_fiod\_frame\_send\_notify\_register** – Register a notification request for when a frame is sent.

**SYNOPSIS**

**#include <fio.h>**

**int fio\_fiod\_frame\_send\_notify\_register( FIO\_APP\_HANDLE** *app\_handle***,  
FIO\_DEV\_HANDLE** *dev\_handle***,  
unsigned int** t*x\_frame*,  
**FIO\_NOTIFY** *notify* **)**

**DESCRIPTION**

This function is used to register a notification request for when a command frame is acknowledged (response frame is received by the FIO API) or when an error occurs.

*app\_handle* is a **FIO\_APP\_HANDLE** returned by a previously successful **fio\_register**(3fio) call. *dev\_handle* is a **FIO\_DEV\_HANDLE** returned by a previously successful **fio\_fiod\_register**(3fio) call. *tx\_frame* is a valid frame number/type for which notification is to be registered. Valid frame numbers/types are in the range 128 - 255. *notify* is an indication as to the frequency of notification; **FIO\_NOTIFY\_ONCE** and **FIO\_NOTIFY\_ALWAYS** may be specified to set notification for one occurrence or for all occurrences, respectively.

Notification is generated to the application program utilizing the **FIO\_SIGIO\_SENT** (**SIGRTMIN** + 5) real-time signal. The application program using this service must establish a **FIO\_SIGIO\_SENT** handler, prior to calling this function. The default handling for a **FIO\_SIGIO\_SENT** real-time signal is to terminate the process. When the indicated t*x\_frame* is received or declared in error by the FIO API, the FIO API will generate a **FIO\_SIGIO\_SENT** real-time signal to the waiting application program. The application program must then perform a **fio\_query\_frame\_send\_notify\_status**(3fio) call to discover why a **FIO\_SIGIO** real-time signal was generated. See **fio\_query\_frame\_send\_notify\_status**(3fio) for further details.

An application program may register notifications for multiple response frames.

**FIO\_NOTIFY is defined as:**

**enum fio\_notify**

**{**

**FIO\_NOTIFY\_ONCE,**

**FIO\_NOTIFY\_ALWAYS**

**};**

**typedef enum fio\_notify FIO\_NOTIFY;**

**FIO\_SIGIO\_SENT** is defined as:

**#define FIO\_SIGIO\_SENT (SIGRTMIN + 5)**

**RETURN VALUES**

Upon successful completion 0 is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

Error codes returned in *errno*:

**EINVAL** One or more arguments are invalid.

**ENOMEM** There is not enough memory available for this operation.

**RESTRICTIONS**

None

**NAME**

**fio\_query\_frame\_send\_notify\_status** – Discover why a sent frame notification occurred

**SYNOPSIS**

**#include <fio.h>**

**int fio\_query\_frame\_send\_notify\_status( FIO\_APP\_HANDLE** *app\_handle***,  
FIO\_SEND\_NOTIFY\_INFO \****notify\_info* )

**DESCRIPTION**

This function is used to discover why a notification, via a **FIO\_SIGIO** real-time signal, was sent to the application program by the FIO API.

*app\_handle* is a **FIO\_APP\_HANDLE** returned by a previously successful **fio\_register**(3fio) call. *notify\_info* is a pointer to a **FIO\_SEND\_NOTIFY\_INFO** structure, which will be filled with response frame notification information upon successful completion.

**FIO\_SEND\_NOTIFY\_INFO** is defined as:

**struct fio\_send\_notify\_info**

**{**

**unsigned int t***x\_frame***; /\* Response Frame # \*/**

**unsigned int** *seq\_number***; /\* Sequence Number of frame \*/**

**unsigned int** *count***; /\* # of bytes in frame \*/**

**FIO\_DEV\_HANDLE** *fiod***; /\* FIOD of response frame \*/**

**};**

**typedef struct fio\_send\_notify\_info FIO\_SEND\_NOTIFY\_INFO;**

**FIO\_FRAME\_STATUS** is defined as:

**enum fio\_frame\_status  
{**

**FIO\_FRAME\_ERROR,**

**FIO\_FRAME\_RECEIVED**

**};**

**typedef enum fio\_frame\_status FIO\_FRAME\_STATUS;**

*tx\_frame* will be set to the frame number/type that was received or in error, and is being notified. Valid frame numbers/types are in the range of 128 – 255. *Status* will be set to an indication as to why the notification occurred. Valid values are: **FIO\_FRAME\_ERROR** and **FIO\_FRAME\_RECEIVED**, indicating an error occurred in the transmission of the frame or a valid response frame was received, respectively. *seq\_number* is the sequence number given to the response frame that caused the notification.

**RETURN VALUES**

Upon successful completion, 0 is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

Error codes returned in *errno*:

**EINVAL** One or more arguments are invalid.

**ENOMEM** There is not enough memory available for this operation.

**EACCESS** The Frame Notification Service is not currently enabled.

## Add Timeout to fio\_fiod\_frame\_read

This proposed change allows an application to specify a maximum time to wait for the next received frame of the indicated type, by adding a timeout parameter.

* fio\_fiod\_frame\_read

The timeout parameter is required by all callers of fio\_fiod\_frame\_read. If a new response frame is not received within the timeout an **ETIMEDOUT** error code will be returned.

The following highlights the updated function signature:

**int fio\_fiod\_frame\_read( FIO\_APP\_HANDLE** *app\_handle***,  
FIO\_DEV\_HANDLE** *dev\_handle***,  
unsigned int** *rx\_frame***,  
unsigned int \****seq\_number***,  
unsigned char \****buf*,  **unsigned int** *count,***unsigned int** *timeout* **)**

* The timeout parameter is the timeout used to determine when the read should fail. The value is in milliseconds.

## Add App\_Handle to fpui\_open\_aux\_switch/fpui\_read\_aux\_switch

The current API standard does not require an application handle to be specified when opening or reading the aux switch making it impossible to enforce the exclusive nature of the aux switch.

This proposed change addresses this issue by updating three functions to add an app\_handle parameter.

* fpui\_open\_aux\_switch
* fpui\_read\_aux\_switch
* fpui\_close\_aux\_switch

The following describes the updated function signatures:

**NAME**

**fpui\_open\_aux\_switch** – Read from the Aux Switch interface

**SYNOPSIS**

**#include <fpui.h>**

**fpui\_aux\_handle fpui\_open\_aux\_switch()**

**DESCRIPTION**

The **fpui\_open\_aux\_switch(**3fpui**)** library call is used to reserve exclusive access to the Aux Switch. One, and only one, process may hold the reservation at a time.

**RETURN VALUE**

**fpui\_open\_aux\_switch(**3fpui**)** returns the new descriptor, or -1 if an error occurred with *errno* set appropriately.

**ERRORS**

**ENOMEM** Insufficient memory was available.

**EACCES** The request access to the underlying device or object is not allowed. This could occur if the Aux Switch is currently opened by another process.

**EFAULT** A reference to an inaccessible memory area was attempted.

**EINVAL** Request not valid.

**NAME**

**fpui\_read\_aux\_switch** – Read from the Aux Switch interface

**SYNOPSIS**

**#include <fpui.h>**

**int fpui\_read\_aux\_switch( fpui\_aux\_handle** *aux\_handle* **)**

**DESCRIPTION**

The **fpui\_read\_aux\_switch(**3fpui**)** library call will return TRUE if the Aux Switch is on and FALSE if the SWITCH is off.

*aux\_handle* is a **fpui\_aux\_handle** returned by a previously successful **fpui\_open\_aux\_switch**(3fpui) call.

**RETURN VALUE**

**fpui\_read\_aux\_switch(**3fpui**)** will return 1 if the Aux Switch is on, 0 if the Aux Switch is off, or -1 if an error occurred with *errno* set appropriately.

**ERRORS**

**EBADF** The underlying object or device used to access the Aux Switch is invalid.

**EINTR** The **fpui\_close\_aux\_switch**(3fpui) call was interrupted by a signal.

**EIO** An I/O error occurred.

**NOTES**

None

**RESTRICTIONS**

None

**SEE ALSO**

**fpui\_open\_aux\_switch**(3fpui), **fpui\_close\_aux\_switch**(3fpui)

**NAME**

**fpui\_close\_aux\_switch** – Close the Aux Switch interface

**SYNOPSIS**

**#include <fpui.h>**

**int fpui\_close\_aux\_switch( fpui\_aux\_handle** *aux\_handle* **)**

**DESCRIPTION**

The **fpui\_close\_aux\_switch**(3fpui) releases exclusive access and closes the Aux Switch interface. Any resources allocated when opened are returned to the system for reuse.

*aux\_handle* is a **fpui\_aux\_handle** returned by a previously successful **fpu\_open\_aux\_switch**(3fpui) call.

**RETURN VALUE**

On success, 0 (zero) is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

**EBADF** The underlying object or device used to access the Aux Swith is invalid.

**EINTR** The **fpui\_close\_aux\_switch**(3fpui) call was interrupted by a signal.

**EIO** An I/O error occurred.

**NOTES**

None

**RESTRICTIONS**

None

**SEE ALSO**

**fpui\_open\_aux\_switch**(3fpui), **fpui\_read\_aux\_switch**(3fpui)

## Add File Handle Parameter to tod\_request/cancel\_tick/onchange\_signal()

In order to maintain a persistent file descriptor to the ATC time-of-day driver in support of the signaling functions, necessary for signal delivery to the calling process, it is recommended that the tod\_request\_tick\_signal() and tod\_request\_onchange\_signal() return a context variable which may be passed by the tod\_cancel\_tick\_signal() and tod\_cancel\_onchange\_signal() functions.

The following describes the updated function signatures:

**NAME**

**tod\_request\_onchange\_signal** – Request local time changed signals

**SYNOPSIS**

**#include <tod.h>**

**int tod\_request\_onchange\_signal(int** *signalnum***)**

**DESCRIPTION**

The **tod\_request\_onchange\_signal**(3tod) library call requests that the signal *signalnum* be sent to the calling process whenever the local time is changed by any source other than the time tick source. This includes a signal being sent when local time changes due to a daylight saving time adjustment.

**RETURN VALUES**

On success, a file descriptor to the underlying time-of-day device is returned. On error, -1 is returned with *errno* set appropriately.

**ERRORS**

**EINVAL** *signalnum* is not a valid signal.

Any other errors shall be due to a system call error in the library implementation, in which case the values of *errno* shall correspond to the standard Linux system call error codes

**NOTES**

If a signal has already been requested by the calling process when this library function is called, the previous signal being sent will be replaced with the signal specified by *signalnum*.

**RESTRICTIONS**

None

**SEE ALSO**

**tod\_cancel\_onchange\_signal**(3tod)

**NAME**

**tod\_cancel\_onchange\_signal** – cancels local time changed signals

**SYNOPSIS**

**#include <tod.h>**

**int tod\_cancel\_onchange\_signal(int** *fd***)**

**DESCRIPTION**

The **tod\_cancel\_onchange\_signal**(3tod) library call cancels any local time change signal from being sent to the calling process.

*fd* is a file descriptor returned by a previous **tod\_request\_onchange\_signal**(3tod) call.

**RETURN VALUES**

On success, 0 is returned. On error, -1 is returned with *errno* is set appropriately.

**ERRORS**

**EINVAL** if the *fd* parameter is invalid.

Any other errors shall be due to a system call error in the library implementation, in which case the values of *errno* shall correspond to the standard Linux system call error codes.

**NOTES**

If the process that has requested this signal dies, the equivalent to this call will be performed automatically to release any necessary resources.

**RESTRICTIONS**

None

**SEE ALSO**

**tod\_request\_onchange\_signal**(3tod)

**NAME**

**tod\_request\_tick\_signal** – Request a signal on each TOD tick

**SYNOPSIS**

**#include <tod.h>**

**int tod\_request\_tick\_signal(int** *signalnum***)**

**DESCRIPTION**

The **tod\_request\_tick\_signal**(3tod) library call requests that the signal *signalnum* be sent to the calling process at each tick of the time of day clock. The frequency of the time of day clock can be determined by calling **tod\_get\_timesrc\_freq**(3tod).

**RETURN VALUES**

On success, a file descriptor to the underlying time-of-day device is returned. On error, -1 is returned with *errno* is set appropriately.

**ERRORS**

**EINVAL** *signalnum* is not a valid signal.

Any other errors shall be due to a system call error in the library implementation, in which case the values of *errno* shall correspond to the standard Linux system call error codes

**NOTES**

If a signal has already been requested by the calling process when this library function is called, the previous signal being sent will be replaced with the signal specified by *signalnum*.

**RESTRICTIONS**

None

**SEE ALSO**

**tod\_cancel\_tick\_signal**(3tod)**, tod\_get\_timesrc\_freq**(3tod)

**NAME**

**tod\_cancel\_tick\_signal** – Cancel signal request for TOD ticks

**SYNOPSIS**

**#include <tod.h>**

**int tod\_cancel\_tick\_signal(int** *fd***)**

**DESCRIPTION**

The **tod\_cancel\_tick\_signal**(3tod) library call cancels any time of day clock tick signal from being sent to the calling process.

*fd* is a file descriptor returned by a previous **tod\_request\_tick\_signal**(3tod) call.

**RETURN VALUES**

On success, 0 is returned. On error, -1 is returned with *errno* is set appropriately.

**ERRORS**

**EINVAL** if the *fd* parameter is invalid.

Any errors shall be due to a system call error in the library implementation, in which case the values of *errno* shall correspond to the standard Linux system call error codes.

**NOTES**

If the process that has requested this signal dies, the equivalent to this call will be performed automatically to release any necessary resources.

**RESTRICTIONS**

None

**SEE ALSO**

**tod\_request\_tick\_signal**(3tod)

## Add fpui\_panel\_present function to detect physical front panel presence

This proposal is for a new function to allow an application program to determine if a physical front panel device is present. In conjunction with this change, the fpui\_get\_window\_size function shall always return the dimensions of the virtual display, even when a physical front panel device is not present. The following describes the new fpui\_panel\_present function signature:

**NAME**

**fpui\_panel\_present** – Return if front panel display is present

**SYNOPSIS**

**#include <fpui.h>**

**int fpui\_panel\_present( fpui\_handle** *hdl* **)**

**DESCRIPTION**

The **fpui\_panel\_present(**3fpui**)** library call is used to detect the presence or absence of a physical Front Panel Display.

**RETURN VALUE**

**fpui\_panel\_present(**3fpui**)** returns 1 if a front panel display is present, 0 if no front panel display is connected, or -1 if an error occurred with *errno* set appropriately.

**ERRORS**

**EBADF** *hdl* is not a valid descriptor.

**EFAULT** A reference to an inaccessible memory area was attempted.

**EINVAL** Request not valid.

**NAME**

**fpui\_get\_window\_size** – Get the current window size

**SYNOPSIS**

**#include <fpui.h>**

**int fpui\_get\_window\_size( fpui\_handle** *hdl*, **int \*** *row*, **int \*** *column* **)**

**DESCRIPTION**

The **fpui\_get\_window\_size**(3fpui) library call will return the current size of the associated virtual display to the application. Parameters: *hdl* is the descriptor returned by **fpui\_open**(3fpui). *row* a reference of an integer where the number of rows will be stored. *column* a reference of an integer where the number of columns will be stored.

**RETURN VALUE**

**fpui\_get\_window\_size**(3fpui) will return 0 (zero) upon success and -1 upon error with *errno* set appropriately.

**ERRORS**

**EBADF** *hdl* is not a valid descriptor.

**EFAULT** A reference to an inaccessible memory area was attempted.

**EINVAL** Request not valid.

**NOTES**

This function always returns the dimensions of the associated virtual display regardless of the presence or otherwise of the actual Front Panel Display. The Front Panel hardware does not support this operation directly. The inquiry is serviced by the window which maintains this information.

**RESTRICTIONS**

None

**SEE ALSO**

None

## Remove fpui\_get/set\_led functions

It is proposed to remove the fpui\_get\_led and fpui\_set\_led functions as they restrict access to the front panel LED to the application whose virtual window currently has focus, and require any application accessing the front panel LED to register a FPUI virtual window.

# System Settings Screen Updates

## Proposed Rearrangement of System Configuration Menu Items (ATC5401 Section 3.2.1)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**| ATC CONFIGURATION INFORMATION |**

**| SELECT ITEM [0-F] |**

**|0 System Time 1 Ethernet Port 1 |**

**|2 Ethernet port 2 3 System Services |**

**|4 Linux Info 5 API Info |**

**|6 Host EEPROM 7** *configitem7* **|**

**|8** *configitem8* **9** *configitem9* **|**

**|[UP/DN ARROW] [FRONT PANEL- NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The proposed split of the Ethernet Configuration into port 1 and port2 separate configuration screens also requires separate menu item entries.

The proposed split of the Linux/API Information into separate screens also requires separate menu item entries.

## Proposed Separate and Enhanced Ethernet Configuration Screens (ATC5401 Section 3.2.3)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| ETHERNET CONFIGURATION |**

**| ETHERNET PORT 1 (***##***:***##***:***##***:***##***:***##***:***##***) |**

**|Port Mode:** *disabled/static/dhcp* **|**

**|IP Address:** *###***.***###***.***###***.***###* **|**

**|Subnet Mask:** *###***.***###***.***###***.***###* **|**

**|Default Gateway:** *###***.***###***.***###***.***###* **|**

**|DNS Server:** *###***.***###***.***###***.***###* **|**

**|[UP/DN ARROW] [APPLY-ENT] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| ETHERNET CONFIGURATION |**

**| ETHERNET PORT 1 (***##***:***##***:***##***:***##***:***##***:***##***) |**

**|Host Name:** *hostname* **|**

**|Packets Sent GD:***numpackets* **BD:***numpackets***|**

**|Packets Rcvd GD:***numpackets* **BD:***numpackets***|**

**| |**

**| |**

**|[UP/DN ARROW] [APPLY-ENT] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| ETHERNET CONFIGURATION |**

**| ETHERNET PORT 2 (***##***:***##***:***##***:***##***:***##***:***##***) |**

**|Port Mode:** *disabled/static/dhcp* **|**

**|IP Address:** *###***.***###***.***###***.***###* **|**

**|Subnet Mask:** *###***.***###***.***###***.***###* **|**

**|Default Gateway:** *###***.***###***.***###***.***###* **|**

**|DNS Server:** *###***.***###***.***###***.***###* **|**

**|[UP/DN ARROW] [APPLY-ENT] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| ETHERNET CONFIGURATION |**

**| ETHERNET PORT 2 (***##***:***##***:***##***:***##***:***##***:***##***) |**

**|Host Name:** *hostname* **|**

**|Packets Sent GD:***numpackets* **BD:***numpackets***|**

**|Packets Rcvd GD:***numpackets* **BD:***numpackets***|**

**| |**

**| |**

**|[UP/DN ARROW] [APPLY-ENT] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The proposal is to separate the Ethernet port 1 and port2 configuration into separate screens to avoid excessive scrolling to access Ethernet port 2 configuration.

Additionally it is proposed to add the display of the Ethernet port MAC address alongside the port number heading line 2.

It is also proposed that the “Port Enabled” field be changed to “Port Mode” and the option changed to a 3-way option comprising the choices “disabled”, “static” and “dhcp”.

The subsequent fields are rearranged in order from that of the ATC5401 Standard layout, to provide a more conventional order of related fields with the basic settings visible with less or no scrolling.

## Proposed Enhanced Linux Information Screen (ATC5401 Section 3.2.5)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| LINUX INFORMATION |**

**|Linux Release:** *release* **|**

**|Linux Version:** *version* **|**

**|Machine Hardware Type:** *type* **|**

**|Memory Total:** *####***MB Free: ####MB |**

**|Filesystem Total: ####MB Free: ####MB |**

**|Load Average:** *loadavg* **|**

**|[UP/DN ARROW] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| LINUX INFORMATION |**

**|Uptime:** *####* **days** *##* **hours** *##* **mins |**

**|**  **|**

**|**  **|**

**|**  **|**

**|**  **|**

**|**  **|**

**|[UP/DN ARROW] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

It is proposed to separate the Linux Information screen from the combined Linux/API Information screen of the ATC5401 Standard in order to access related information in a more concise manner and with less or no scrolling necessary.

In addition it is proposed to display more relevant fields from the options returned by the “uname” utility function: the Linux release (actually kernel version number, e.g. “2.6.32”); the Linux version (actually the kernel build number and date of build); the machine hardware type (e.g. “ppc”).

The “Network Node Hostname” field is dropped due to being a duplicate of the “Host Name” field of the Ethernet Configuration Screens.

The “Processor Type” and “Hardware Platform” fields are dropped due to only being available from the “uname” utility function as “unknown” on non-x86 architecture machines.

The “Operating System” field is dropped due to obviousness, i.e. always returns “GNU/Linux” on Linux platforms.[

It is proposed to add the following Linux information fields, available from the “sysinfo” and “df” utility functions, in order to provide additional diagnostic data: “Memory Total and Free”, the total and free usable main memory size; “Filesystem Total and Free”, the total and free space on the root filesystem; “Load Average”, the 1, 5 and 15 minute processor load averages; “Uptime”, the time since last boot. These fields shall be refreshed with current data once every minute while the screen is displayed.

## Proposed Enhanced API Information Screen (ATC5401 Section 3.2.5)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| API INFORMATION |**

**|FIO API Version:** *text* **|**

**|FIO API LKM Version:** *text* **|**

**|FPUI API Version:** *text* **|**

**|FPUI API LKM Version:** *text* **|**

**|TOD API Version:** *text* **|**

**| |**

**|[UP/DN ARROW] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

It is proposed to separate the API Information screen from the combined Linux/API Information screen of the ATC5401 Standard in order to access related information in a more concise manner and with less or no scrolling necessary.

The fields are renamed to be more relevant to the provisions of the API Reference Implementation.

## App Management Screen Addition

To be completed

## API Configuration Manager Addition (ATC5401 Section 3.2.4)

Suggest that the FIO API, FPUI API and TOD API appear as line items on the System Services screen.

## Time Source Configuration Screen Addition

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| TIME SOURCE CONFIGURATION |**

**|ATC Time Source:** *timesrc* **|**

**|GPS Serial Port:** *SP1/SP2/SP3/SP8/none* **|**

**|NTP Peer Address:** *###.###.###.###* **|**

**| |**

**| |**

**| |**

**|[UP/DN ARROW] [APPLY-ENT] [QUIT-\*\*NEXT]|**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Suggest a new system configuration menu item and screen to allow configuration of the time source including additional settings relating to TOD\_TIMESRC\_EXTERNAL1 (GPS) for serial port connection, and TOD\_TIMESRC\_EXTERNAL2 (NTP) for NTP network peer address. The “timesrc” field is configurable from the enumeration “LINESYNC/RTCSQWR/CRYSTAL/GPS/NTP”.