RTx Controller Firmware

RocketTracks Capstone 2014

Table of Contents

I. Axis Control

A. Axis Control Overview

The vertical and lateral axes are controlled via a PID Feedback loop. The PID loops require a desired and an actual position input for each axis.

B. Axis Control

1. Signal Processing

a) Control Bandwidth and Cutoff Frequency

The control bandwidth frequency f_{CBW} for the closed-loop control system was chosen based on the requirement to track fast-moving objects with video during manual and automated control operation. The cutoff frequency f_C for the ADC input LPF is specified to be one decade higher than f_{CBW}. This convention places f_C far enough above the f_{CBW} to prevent the LPF from limiting performance at the f_{CBW}, while still providing satisfactory noise filtering.

$$f_{CBW} = 2.5Hz$$

$$f_C = 2.5Hz * 10 = 25Hz$$

b) ADC 1-bit Frequency, desired noise level and Filter Order

The frequency of the ADC's LSB f_{1bit} is the minimum frequency at which we will observe a change on the least significant bit. The maximum noise level desired at f_{1bit} is -96dβ. f_{1bit} must be a low enough frequency such that the Nyquist rate is reasonable given our choice of ADC's and microcontroller. With a 3rd Order LPF with our $f_C = 25Hz$, we have:

$$f_{-96d\beta} = 995Hz$$

The Nyquist rate, or minimum rate we must sample the ADC's is then $995Hz * 2 = 1090Hz$, which is a reasonable sample rate for the system.

c) Control Rate and ADC Sampling

A rule of thumb for minimum control rate is $f_{CR} \geq f_{CBW} * 40$, then:

$$f_{CR} \geq 2.5Hz * 40 \geq 100Hz$$

The Nyquist rate must also be satisfied, so the sample rate must be:

$$f_S \geq 1080Hz$$

Choosing $f_{CR} = 100Hz$ and $f_S = 2000Hz$ gives us 20 samples per control loop iteration.

d) Summary of Parameters

$f_{CBW} = 2.5Hz$

$f_C = 25Hz$

$$f_S = 2000Hz$$

$$f_{CR} = 100Hz$$

2. Implementation

The axis control loop is implemented as a hardware timer callback. After reading the preprocessed axis position sensor values, Ethernet messages containing Neutral and Diagnostics data are sent, followed by performance of the Drive Enable procedure, which performs the checks described in section II. The PID loop is then executed and PWM duty cycles are updated for each axis.

II. Drive Enable

A. System Failure Checks

The RTx Controller performs in-range checks and keep-alive checks to ensure that system failures are detected and that the motor drivers are reliably disabled in the event of such a failure. To ensure the axis motor drivers are disabled, the disable signals are de-asserted and toggling of the watchdog is stopped.

1. Manual Remote Keep-Alive

The RTx Controller must receive new Manual Control packets from the RTx Manual Control box periodically. This feature ensures that the system will not remain enabled with the Manual Remote ENABLE switch in the Disabled position, in the event that the Ethernet connection is lost.

2. Analog Reference Voltage

The Analog Reference Voltage, which is part of the axis position sensor circuits, is tested at each control loop iteration to ensure it is within an acceptable range and capable of producing accurate axis position signals.

3. Axis Position Sensor Range

The Axis Position Sensor values are checked for validity at each control loop iteration to ensure there has not been a failure of the position sensor circuits. Failure of the axis position sensors could otherwise cause uncontrolled motion of the axes.

B. Operator Enable Checks

The RTx Controller responds to Enable inputs from the operator, ensuring the system motor drivers are enabled only when the operator chooses.

1. Manual Remote Enable

The last received state of the Manual Remote Enable switch is checked at each control loop iteration. The operator Enable signal cannot override a safety-check that results in the system being disabled.

2. Shell Enable Override

Shell command functions are available on both the RTx Controller and the Manual Remote to enable and disable the motor driver outputs. While this functionality will disable the system regardless of the state of the Manual Remote Enable switch or the results of safety checks, it cannot enable the system by overriding the Manual Remote Enable in the disabled position or failed safety checks.

III. Ethernet API

A. Initialization

Sockets are set up by using the data structures common to the RocketNet code base. IP addresses, MAC addresses and ports are all defined in the PSAS common file net_addrs.c.

1. Creation of Sockets

Ready-made functions are available in the RocketTracks Ethernet API which create the required sockets for interfacing the Manual Remote, the RTx Controller and a Sightline device. These functions are declared, and their particular purpose described in common/rtx/enet_api.h.

B. Communications

1. Data Structures

There are a set of data structures defined in enet_api.h which provide an interface for data communications between the devices in the RTx system.

a) struct ManualData

The ManualData struct contains fields for each of the Manual Remote inputs, and is used for communication Manual Remote commands to the RTx Controller.

i. Enable – Manual Remote Enable switch state. This state may be overridden to DISABLED by the Shell Enable Override functionality
ii. Mode – Manual Remote Mode switch state. Selects Sightline/Automatic or Manual Mode.
iii. Aux – Manual Remote auxiliary switch for future functionality.
iv. latPosition – Lateral Axis Desired Position data. This data should correspond 1-to-1 with RTX axis position sensor values.
v. vertPosition – Vertical Axis Desired Position data. This data should correspond 1-to-1 with RTX axis position sensor values.
vi. Axis3Position – Axis3 Desired Position data. This field is to support future addition of a 3rd axis.
vii. Axis4Position – Axis4 Desired Position data. This field is to support future addition of a 4th axis.

b) struct SLAData

The SLAData struct contains fields for pixel coordinates received from a Sighline device via Ethernet.

i. Column – The target's pixel Column coordinate received from the Sightline device. See the Sightline Communications Protocol for more information.

ii. Row – The target's pixel Row coordinate received from the Sightline device. See the Sightline Communications Protocol for more information.

c) *struct Neutral*

The Neutral struct contains fields for indicating a neutral interlocked state for each axis.

i. latNeutral – Indicates the Neutral status of the Lateral axis.
ii. vertNeutral – Indicates the Neutral status of the Vertical axis.

d) *struct Diagnostics*

The Diagnostics struct contains critical Axis control data which can be sent from the RTx Controller to the Manual Remote for diagnostic and development purposes. The fields in the struct are a subset of fields contained in the CONTROL_AXIS_STRUCT, defined in control.h.

2. Message-passing Functions

A set of send and receive functions are available to transfer data contained in each of the data structures listed above, between the appropriate RTx system devices.

a) *SendManual(ManualData *)*

This function sends an Ethernet packet containing the contents of the ManualData struct from the Manual Remote to the RTx Controller.

b) *SendNeutral(Neutral *)*

This function sends and Ethernet packet containing the contents of the Neutral struct from the RTx Controller to the Manual Remote.

c) *SendDiagnositcs(Diagnostics * lat, Diagnostics * vert)*

This function sends an Ethernet packet containing the contents of the Diagnostics structs from the RTx Controller to the Manual Remote. lat must contain data related to the Lateral axis, while vert must contain Vertical axis data.

d) *ReceiveManual(ManualData *)*

This function attempts to read an Ethernet packet containing Manual Control data from the Manual Remote. If a packet is received, the data is parsed into the ManualData struct.

e) *ReceiveNeutral(Neutral *)*

This function attempts to read an Ethernet packet containing axis neutral interlock status from the RTx Controller. If a packet is received, the data is parsed into the Neutral struct.

f) *ReceiveDiagnostics(Diagnostics * lat, Diagnostics * vert)*

This function attempts to read an Ethernet packet containing Diagnostics data from the RTx Controller.

g) *ReceiveSLA(SLAData *)*

This function attempts to read an Ethernet packet from the Sightline device. If a packet is received, the target pixel coordinates are parsed into the SLAData struct.