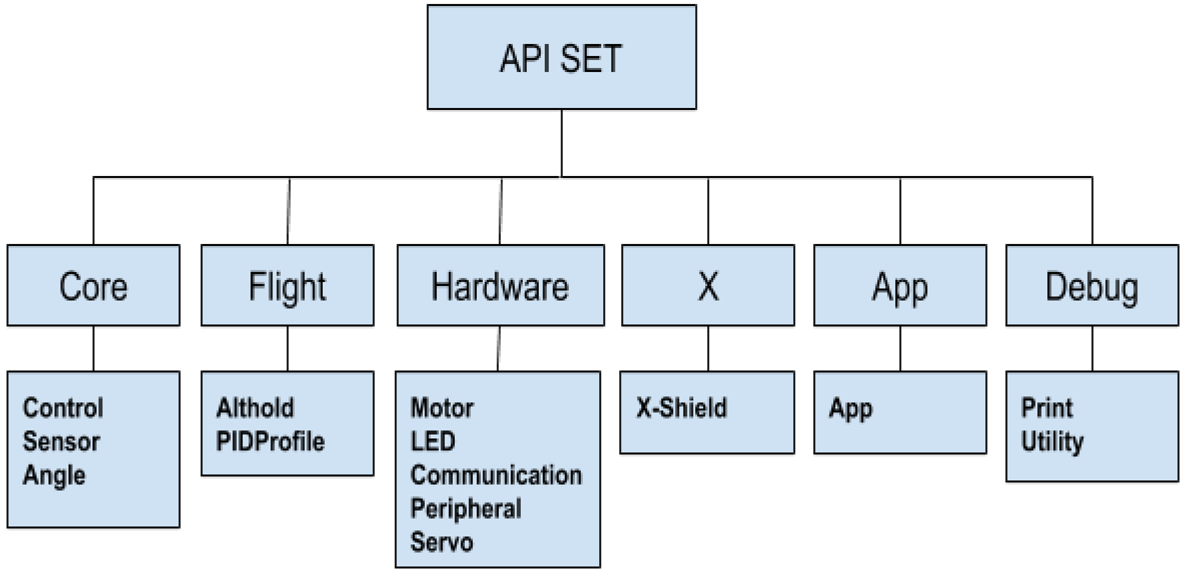
Introduction

This document contains information about available APIs for PlutoX on Cygnus IDE.

API set overview



API Set In Details:

1: Control APIs (Core)

(i) bool isOkToArm(void)

- Returns true when Pluto has finished its initial calibrations and now ready to arm i.e. to start motors.

(ii) bool isArmed(void)

- Returns true when Pluto has been armed.

(iii) bool arm(void)

` - Starts the motors and return true on success.

(iv) bool disArm(void)

- Stops all the motors and return true on success.

(v) void setRC(rc_channels_e channel, int16_t value)

- Sets RC channel value.
- Available channels with min and max value respectively as follows:

1: RC_THROTTLE (1000, 2000) – in Althold mode 1500 is neutral value, below 1500 to 1000 Pluto will come downwards and above 1500 to 2000 Pluto will go upwards, in Throttle mode Pluto will go upwards at any throttle value.

2: RC_ROLL (1000, 2000) - In both Althold and Throttle mode 1500 is neutral value, from 1500 to 1000 Pluto will roll to left and from 1500 to 2000 Pluto will roll to right.

3: RC_PITCH (1000,2000)- In both Althold and Throttle mode 1500 is neutral value, from 1500 to 1000 Pluto will pitch to backward and from 1500 to 2000 Pluto will pitch to forward.

4: RC_YAW (1000, 2000) - In both Althold and Throttle mode 1500 is neutral value, from 1500 to 1000 Pluto will yaw to left and from 1500 to 2000 Pluto will yaw to right.

Note: Pluto has two flying modes
1. Althold mode- in which Pluto maintains the height where you leave the throttle control i.e set to 1500
2. Throttle mode- in doesn’t maintain any height when you leave throttle, you need to continuously provide throttle input in keep it upward.

(vi) int16_t   getRC(rc_channels_e channel)

-Returns the current value RC channel

(vii) void disableExternalRC(rc_channels_e channel)

-Disables the external RC input for provided channel.

(viii) void disableAllExternalRC(void)

-Disables the external RC input for all channels.

(ix) void enableExternalRC(rc_channels_e channel)

-Enables external RC input for provided channel.

(x) void enableAllExternalRC(void)

-Enables external RC input for all channels.

Note- External RC inputs are enable by default.

(xi) void disableFlightStatus(bool disable)

- Disables default LEDs functions used for indication of different flight status if set true.

(xii) bool checkFlightStatus(f_status_e status)

- Check if status is currently active or not.
- Available flight status are as follow:

1: FS_Accel_Calibration- becomes active when Pluto requires accelerometer calibration
2: FS_Mag_Calibration- becomes active when Pluto requires magnetometer calibration
3: FS_Low_battery- becomes active when Pluto has battery below 3.4 Volt
4: FS_LowBattery_inFlight- becomes active when Pluto reaches battery below 3.1 V while flying
5: FS_Crash- becomes active when Pluto crashes.
6: FS_Signal_loss- becomes active when there is communication loss between Pluto and mobile App.
7: FS_Not_ok_to_arm- becomes active when Pluto is not ready to arm (because of things like it needs calibration or need to keep on plane surface).
8: FS_Ok_to_arm- becomes active when Pluto is ready to arm.

9: FS_Armed- becomes active when Pluto is armed.

(xiii) f_status_e getFlightStatus(void)

- Get currently active flight status.

(xiv) void setCommand(flight_command_e command)

-Sets the flight command.
-Available commands are: COMMAND_TAKE_OFF, COMMAND_LAND

(xv) void setFailsafe(failsafe_e failsafe, bool activate)

- Enable or disable different failsafes (LOW_BATTERY, INFLIGHT_LOW_BATTERY, CRASH, ALL).

- All failsafes are enabled by default.

(xvi) void setHeading(int16_t heading)

-Sets the Pluto's heading.

(xvii) void setUserLoopFrequency(uint16_t frequency)

- Sets plutoPilot() function frequency in milliseconds

- Allowed range is 0ms-300ms: default is 100ms

Usage: To access Control APIs you have to use predefined object 'Control'.

Examples: 1- Control.setRC(RC_THROTTLE, 1700) ; setting throttle to 1700
2- Control.setFailsafe(CRASH, false); disabling the crash failsafe check
3- Control.arm(); arme the Pluto;

2: Sensor APIs (Core)

(i) Accelerometer:

(a) int16_t getX(void)
- Returns X component of acceleration.

(b) int16_t getY(void)
- Returns Y component of acceleration.

(c) int16_t getZ(void)
- Returns Z component of acceleration.

(d) int32_t getNetAcc(void)
- Returns net acceleration.

Note- Unit: cm/sec^2.

(ii) Gyroscope:

(a) int16_t getX(void)

- Returns angular rate about Pluto’s X-axis.

(b) int16_t getY(void)

- Returns angular rate about Pluto’s Y-axis.

(c) int16_t getZ(void)

- Returns angular rate about Pluto’s Z-axis.

Note- Unit: degrees/sec

(iii) Magnetometer:

(a) int16_t getX(void)

- Returns X component of magnetic field.

(b) int16_t getY(void)

- Returns Y component of magnetic field.

(c) int16_t getZ(void)

- Returns Z component of magnetic field.

Note- Unit: microTesla

(iv) Barometer:

(a) int32_t getPressure(void)

- Returns barometer’s pressure (mbar) reading.

(b) int16_t getTemperature(void)

- Returns barometer’s temperature reading.

Note- Returns temperature in multiple of 0.01degC
i.e. 30degC = 3000.

Usage: To access Sensor APIs you have to use predefined respective sensors object ‘Accelerometer’, ‘Gyroscope,’ Magnetometer,’ Barometer.

Examples: 1- Accelerometer. getNetAcc (); //get net acceleration
2- Barometer. getPressure(); //get pressure

3: Angle APIs (Core)

(i) int16_t get(inclination_t angle)

- Returns current angle (AG_ROLL, AG_PITCH, AG_YAW) value.

Note- Returns angle (AG_ROLL, AG_PITCH) in multiple of 0.1 degree i.e. 120 degree=1200 and angle(AG_YAW) in degree(0-360).

Usage: To access Angle APIs you have to use predefined object ‘Angle’.

Examples: 1- Angle.get(AG_ROLL) //get Pluto’s current roll angle

4: AltHold APIs (Flight)

(i) int32_t getEstimatedAltitude(void)

- Returns height (cm) estimated for althold.

(ii) int32_t getVelocityZ(void)

- Returns Z-axis velocity (cm/sec) in used by althold.

(iii) void activateAlthold(bool activate)

- Activates althold if set true: if set false disables althold and starts throttle mode.

Note: althold is default mode.

(iv) bool isAltholdModeActivate(void)

- Returns true if Pluto is in althold mode else false on throttle mode.

(v) void setAltholdHeight (int32_t height)

- **Set's height (cm) for althold.**

(vi) void setRelativeAltholdHeight (int32_t height)

- Set's height (cm) for althold relative to Pluto's current height.

(vii) int32_t getAltholdHeight (void)

- Get current althold height.

Usage: To access Althold APIs you have to use predefined object ‘Alhold’.

Examples: 1- Althold.activateAlthold(); //activates althold mode

2- Althold. getVelocityZ(); // get Z axis velocity

5: PIDProfile APIs (Flight)

(i) uint8_t getRoll_P()

- Get Kp value of Roll PID.

(ii) uint8_t getRoll_I()

- Get Ki value Roll PID.

(iii) uint8_t getRoll_D()

- Get Kd value of Roll PID.

(iv) uint8_t getPitch_P()

- Get Kp of Pitch PID.

(v) uint8_t getPitch_I()

- Get Ki value of Pitch PID.

(vi) uint8_t getPitch_D()

- Get Kd value of Pitch PID.

(vii) uint8_t getYaw_P()

- Get Kp value of Yaw PID.

(viii) uint8_t getYaw_I()

- Get Ki value of Yaw PID.

(ix) uint8_t getYaw_D()

- Get Kd value of Yaw PID.

(x) uint8_t getAlt_P()

- Get Kp value of Altitude Hold PID.

(xi) uint8_t getAlt_I()

- Get Ki value of Altitude Hold PID.

(xii) uint8_t getAlt_D()

- Get Kd value of Altitude Hold PID.

(xiii) void setRoll_P()

- Set Kp value of Roll PID.

(xiv) void setRoll_I()

- Set Ki value Roll PID.

(xv) void setRoll_D()

- Set Kd value of Roll PID.

(xvi) void setPitch_P()

- Set Kp of Pitch PID.

(xvii) void setPitch_I()

- Set Ki value of Pitch PID.

(xviii) void setPitch_D()

- Set Kd value of Pitch PID.

(xix) void setYaw_P()

- Set Kp value of Yaw PID.

(xx) void setYaw_I()

- Set Ki value of Yaw PID.

(xxi) void setYaw_D()

- Set Kd value of Yaw PID.

(xxii) void setAlt_P()

- Set Kp value of Altitude Hold PID.

(xxiii) void setAlt_I()

- Set Ki value of Altitude Hold PID.

(xxiv) void setAlt_D()

- Set Kd value of Altitude Hold PID.

(xxv) void setDefault()

- Reset all PIDs to default values.

Note: It is recommended to use PID set functions in plutoInit().

Usage: To access PIDProfiel APIs you have to use predefined object PID.

Examples: 1- PID.getYaw_D(); //Get Kp value of Yaw PID

2- PID.setAlt_P(40); // Set Kp value of Altitude Hold PID to 40.

3- PID.setDefault(); // Reset all PIDs to default values.

6: Motor APIs (Hardware)

(i) void set(motor_e motor,int16_t value)

-Sets value for motor PWM, allowed values from 1000-2000.

-At 1000 motor will stop.

(ii) void setDirection(motor_e motor,motor_direction_e direction)

- Sets motors rotation direction.
- FORWARD for normal configured rotation.
- BACKWARD for opposite of configured rotation.

Note: Pluto-X has total 8 motors(M1-M8); by default outer 4 motors(M5-M8) are configured for Quad and remaning 4(M1-M4) are available as spare for extra usage; In reverse Mode inner 4(M1-M4) gets configured for Quad but remaining outer 4(M5-M8) are not available as spare for extra usage; only inner 4 motors(M1-M4) can change direction.

Usage: To access Motor APIs you have to use predefined object 'Motor'.

Examples: 1- Motor.set(M6, 1300) //set M6 to 1300

2- Motor.setDirection(M3, BACKWARD) //set M6 rotation to backwards

7: LED APIs (Hardware)

(i) void ledOp(led_e name, led_state_e state)

-sets LED (L_LEFT,L_MID,L_RIGHT, right) to a particular state (ON, OFF, TOGGLE).

Usage: Use Function directly.
Example: ledOp(L_LEFT, ON); // ON left LED

8: UART APIs (Hardware-Communication)

(i) void init(uint32_t baud_rate, UARTportMode_t mode,UARTportOptions_t options);

-Initialize the UART with specified baud rate , mode and options.
- Available baud_rate: 0, 9600, 19200, 38400, 57600, 115200, 230400, 250000

- Mode can be set to RX, TX, RXTX
  - Available options are:
    1: NOT_INVERTED - 0 V represents 0 and 3.3 V represents 1.
    2: INVERTED - 0 V represents 1 and 3.3 V represents 0
    3: STOPBITS_1 - Stop bits sent at the end of every character allow the receiving
    4: STOPBITS_2 - signal hardware to detect the end of a character and to resynchronize with the character stream.
    5: PARITY_NO - no parity
    6: PARITY_EVEN - even parity
    7: UNIDIR - Unidirectional
    8: BIDIR - Bidirectional

- UART Tx available on unibus 2 and Rx available on unibus pin 3

(ii) uint8_t read8()

- Reads one byte UART data.

(iii) uint16_t read16()

- Reads two bytes of UART data.

(iv) uint32_t read32()

- Reads four bytes of UART data

(v) void write(uint8_t data)

- Writes one byte of data

(vi) void write(const char *str)

- Writes stream of characters.

(vii) void write(uint8_t *data, uint16_t length)

- Writes data of specified length.

(viii) bool rxBytesWaiting(void)

- Returns true if bytes are available in Rx buffer, false otherwise.

(ix) bool txBytesFree(void)

- Returns true if there are no bytes in Tx Buffer, false otherwise

Usage: To access UART APIs you have to use predefined object ‘UART’.

Examples: 1- UART.init(115200, RXTX, NOT_INVERTED); //Initialize UART with 115200 baud rate and both receiving and transmitting mode enabled

2- UART.write(0x52) // Writing 0x52 hex number on UART
3- UART.read8() // Read one byte from Rx Buffer.

9: GPIO APIs (Hardware-Peripheral)

(i) void init(unibus_e pin_number, GPIO_Mode_e mode, GPIO_Speed_e speed)

-Setups pin to GPIO with specified mode and speed.
-Available pins are (Pin1, Pin2, Pin3, Pin4, Pin5).
-Available modes are:
1: AIN - analog Input
2: IN_FLOATING - input floating
3: IPD - input pulled down
4: IPU - Input with pull-up resistor
5: Out_OD - output open drain
6:Out_PP - output push-pull
7:AF_OD - alternate function open drain

8:AF_PP - alternate function push pull
9:AF_PP_PD - alternate function push pull with pull down register
10: AF_PP_PU - <alternate function push pull with pull up register
- Available speeds: SP_10MHz , SP_2MHz, SP_50MHz

(ii) void AFConfig(unibus_e pin_number, GPIO_AF_e AF)

-Seta alternate function.

(iii) void setHigh(unibus_e pin_number)

- Sets pin High.

(iv) void setLow(unibus_e pin_number)

- Sets pin Low.

(v) uint16_t read(unibus_e pin_number)

- Reads pin.

Usage: To access GPIO APIs you have to use predefined object 'GPIO'.

Examples: 1- GPIO.init(Pin1, GPIO_Out_PP, SP_2MHz) //sets pin 1 to GPIO with output mode and 2 MHz speed
2- GPIO.setHigh(Pin1) // set GPIO pin 1 to High

10: ADC APIs (Hardware-Peripheral)

(i) void init(unibus_e pin_number)

- Configures ADC at specified pin.
-Available pins are (Pin2, Pin3, Pin4, Pin5).

(ii) void read(unibus_e pin_number)

- Reads ADC value at specified pin.

Usage: To access ADC APIs you have to use predefined object 'ADC'.

Examples: 1- ADC.init(Pin4) // set pin 4 for ADC
2- ADC.read(Pin4) // read ADC value at pin 4

11: Timer APIs (Hardware-Peripheral)

(i) void init(unibus_e pin_number, uint16_t hz)

-Configures Timer at pin.
-Available pins are (Pin1, Pin4, Pin5).

(ii) void setPWM(uint16_t value)

- Sets PWM value ranged from 1000-2000 for Timer.

Usage: To access Timer APIs create your own Timer object and access functions.

Examples:
```
1- Timer_P myTimer              // creating Timer object
2- myTimer.init(Pin5)           // configuring Timer at pin 5
3- myTimer.setPWM(1200) //giving 1200 PWM to myTimer
```

12: Servo APIs (Hardware)

(i) void set(servo_e servo,int16_t value)

- Sets value for servo PWM, allowed values from 1000-2000.
- 1500 is mid value.

Note: PlutoX currently has one servo(S1) configured on pin 2 of Unibus.

Usage: To access Servo APIs you have to use predefined object ‘Servo’.

Examples: 1- Servo.set(S1,1300) //set S1 to 1300

13: Xshield APIs (X)

(i) void init()

- Initialises Xshield.

(ii) void startRanging()

- Starts laser ranging.

(iii) void startRanging()void activateAlthold(bool activate)

- Stops laser ranging.

(iv) uint16_t getRange(laser_e laser)

- Returns true if Pluto is in althold mode else false on throttle mod.

Usage: To access Xshield APIs you have to use predefined object ‘Xshield.

Examples: 1- Xshield.getRange(LEFT) //get range from left sensor

15: App APIs (App)

(i) int16_t getAppHeading(void)

- Get the Pluto Controller apps current heading (available on Android only).

(ii) bool isArmSwitchOn(void)

- Returns true if Pluto Controller apps arm switch is on, false otherwise(available on Android only).

Usage: To access App APIs you have to use predefined object ‘App’.

Examples: 1- App.getAppHeading() //get the apps current heading
2- App.isArmSwitchOn() // check if arm switch is ON on the app

15: Print APIs (Debug)

(i) void monitor(String tag, int32_t number)

- Prints integer number with tag on Pluto Monitor on Cygnus IDE with 100 ms frequency.

(ii) void monitor (String tag, double number, uint8_t precision)

- Prints float number with tag on Pluto Monitor on Cygnus IDE with 100 ms frequency.

iii) void monitor (String message)

- Prints string message on Pluto Monitor on Cygnus IDE with 100 ms frequency.

(iv) void redGraph(double value, uint8_t precision)

- Plots value on Red Graph of Pluto Graphs on Cygnus IDE.

(v) void blueGraph(double value, uint8_t precision)

- Plots value on Blue Graph of Pluto Graphs on Cygnus IDE.

(vi) void greenGraph(double value, uint8_t precision)

- Plots value on Green Graph of Pluto Graphs on Cygnus IDE.

Usage: To access Print APIs you have to use predefined object ‘Print’.

Example: 1- Print.monitor(“TAG”, 34);   // print 34 with associated TAG message
2- Print.redGraph(myVariable, 5); // print myVariable at RED GRAPH

16: Utility APIs (Debug)

(i) uint32_t microsT(void)

- Returns current system time in microseconds.

(ii) bool Timer::set(uint32_t time)

- Sets the timer and returns true after every provided time(in milliseconds)

(iii) bool Timer::reset(uint32_t time)

- Resets the timer.

Note- Allowed time range for Timer is 20ms to 5000ms, Timer is continues one.

Usage: Use microsT(void) directly, to use Timer create Timer object and access the function.

```
Examples:  1- Timer printTimer           // creating Timer object
           2- printTimer.set(200)        // starting printTimer with 200 milliseconds bound
```