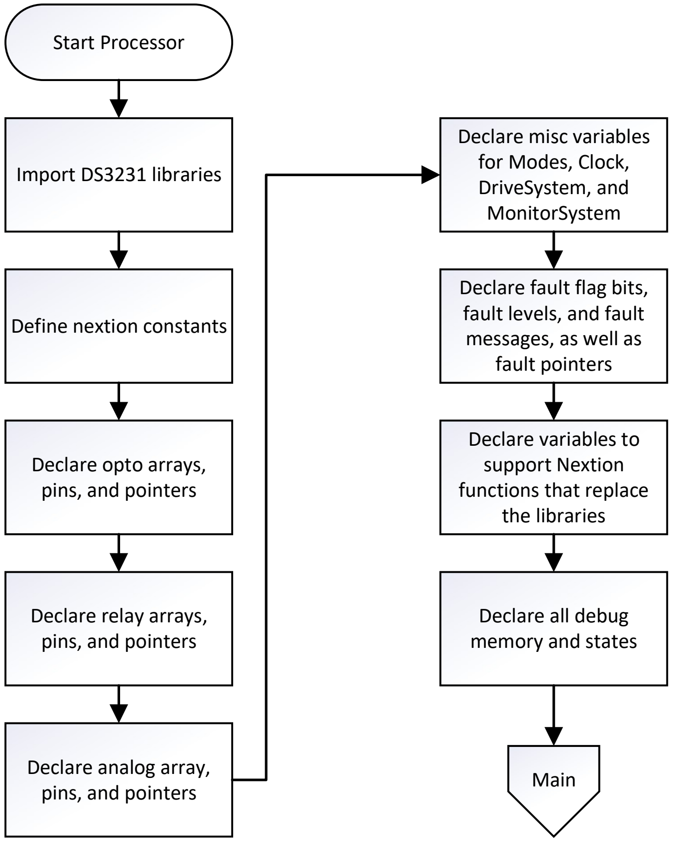
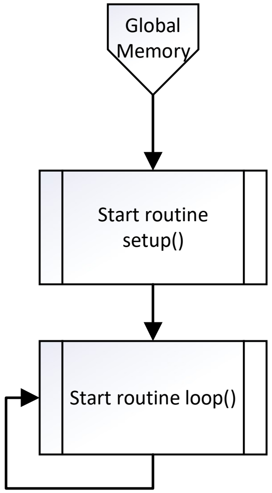

Start Processor
Import DS3231 libraries
Define nextion constants
Declare opto arrays, pins, and pointers
Declare relay arrays, pins, and pointers
Declare analog array, pins, and pointers
Declare misc variables for Modes, Clock, DriveSystem, and MonitorSystem
Declare fault flag bits, fault levels, and fault messages, as well as fault pointers
Declare variables to support Nextion functions that replace the libraries
Declare all debug memory and states
Main

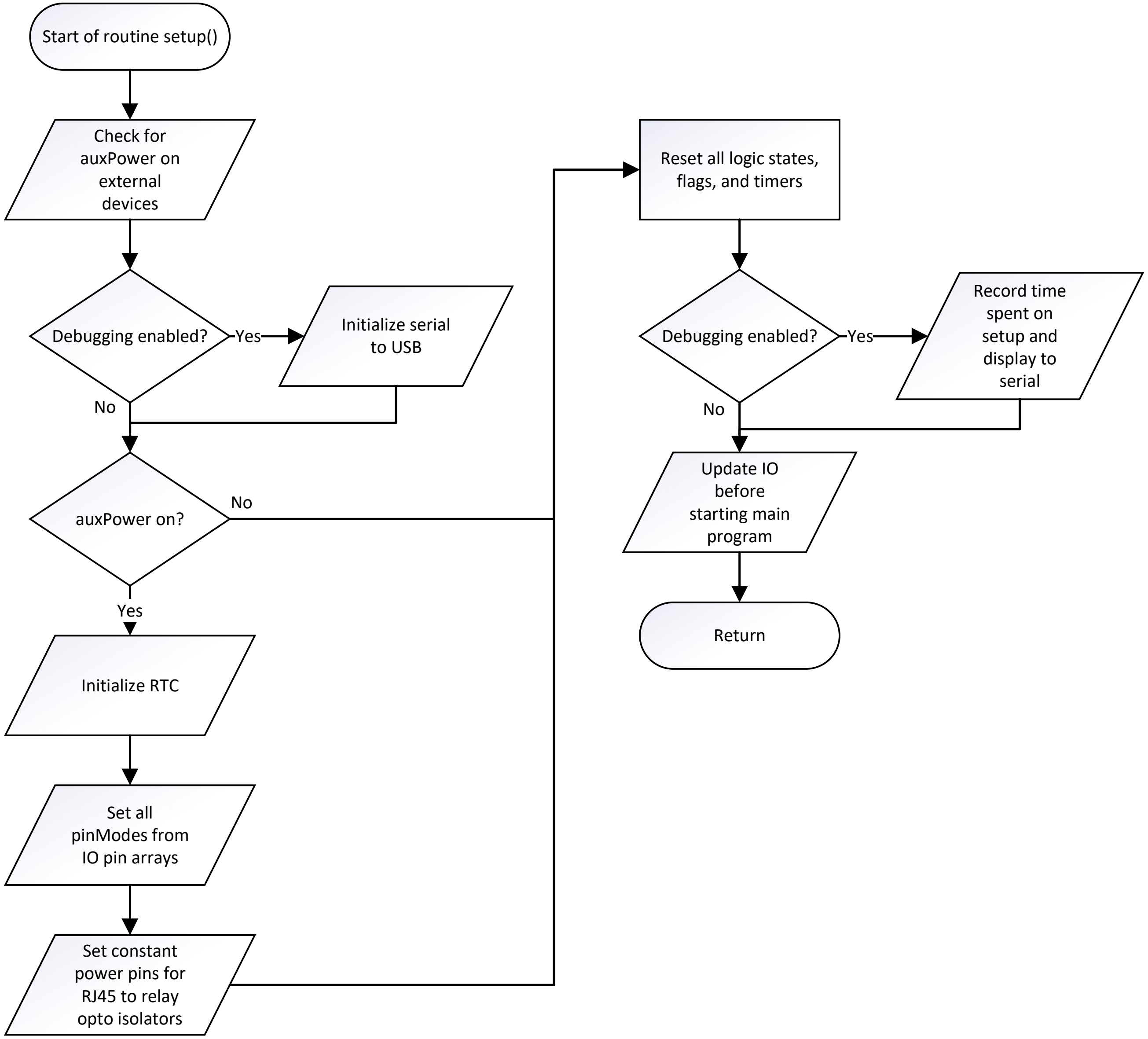
Initialize all variables and memory

TITLE	DRAWN BY	DATE	PAGE
Global Memory	COLE FUERTH	03/20/2020	1 OF 12



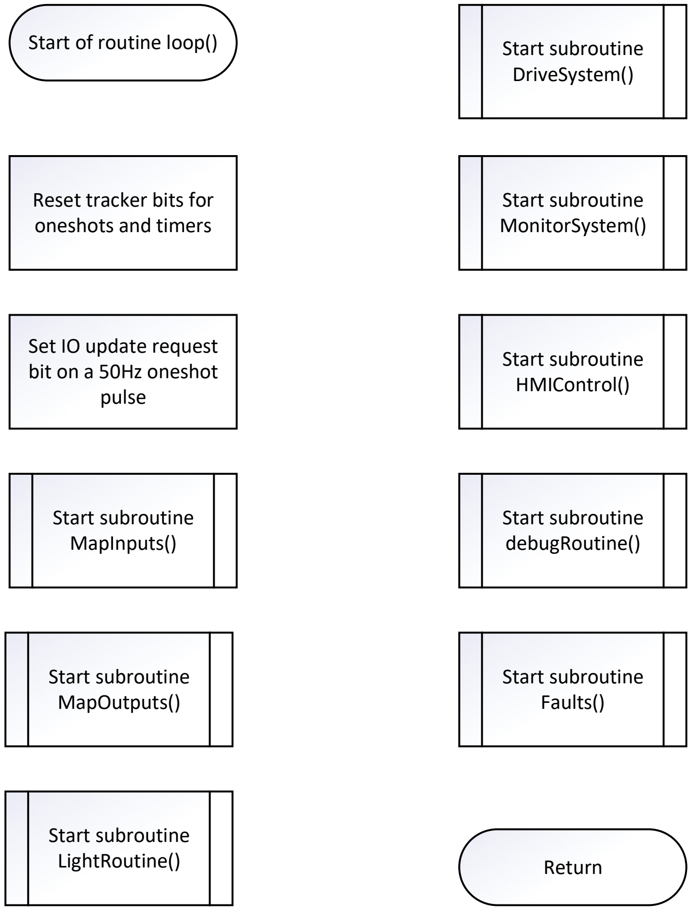
Core Arduino flow

TITLE Main Program Flow	DRAWN BY COLE FUERTH	DATE 03/20/2020	PAGE 2 OF 12


Start of routine setup()
Check for auxPower on external devices
Debugging enabled?
Yes
Initialize serial to USB
No
auxPower on?
No
Yes
Initialize RTC
Set all pinModes from IO pin arrays
Set constant power pins for RJ45 to relay opto isolators
Reset all logic states, flags, and timers
Debugging enabled?
Yes
Record time spent on setup and display to serial
No
Update IO before starting main program
Return

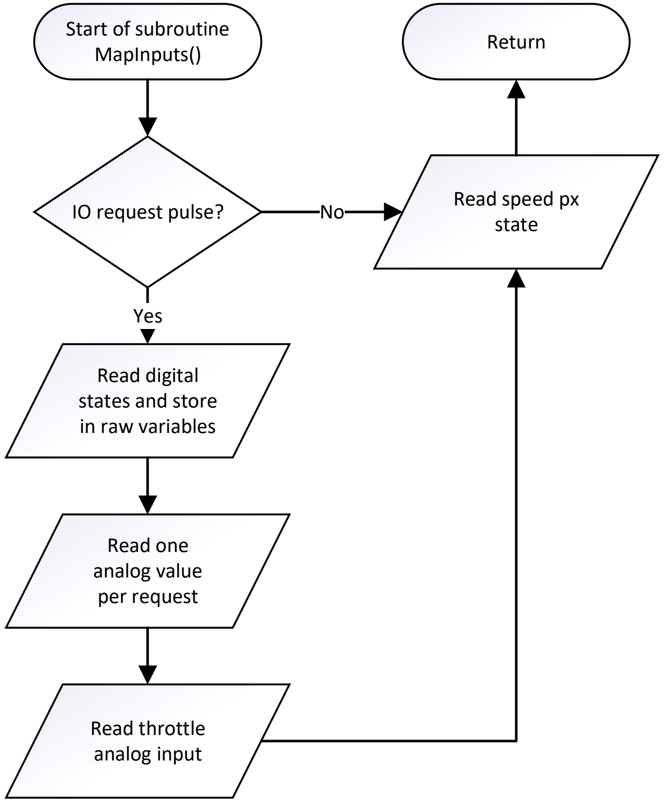
Complete setup of all initial logic states, IO, and serial connections

TITLE	DRAWN BY	DATE	PAGE
Main Program Flow	COLE FUERTH	03/20/2020	3 OF 12


Start of routine loop()
Reset tracker bits for oneshots and timers
Set IO update request bit on a 50Hz oneshot pulse
Start subroutine MapInputs()
Start subroutine MapOutputs()
Start subroutine LightRoutine()
Start subroutine DriveSystem()
Start subroutine MonitorSystem()
Start subroutine HMIControl()
Start subroutine debugRoutine()
Start subroutine Faults()
Return

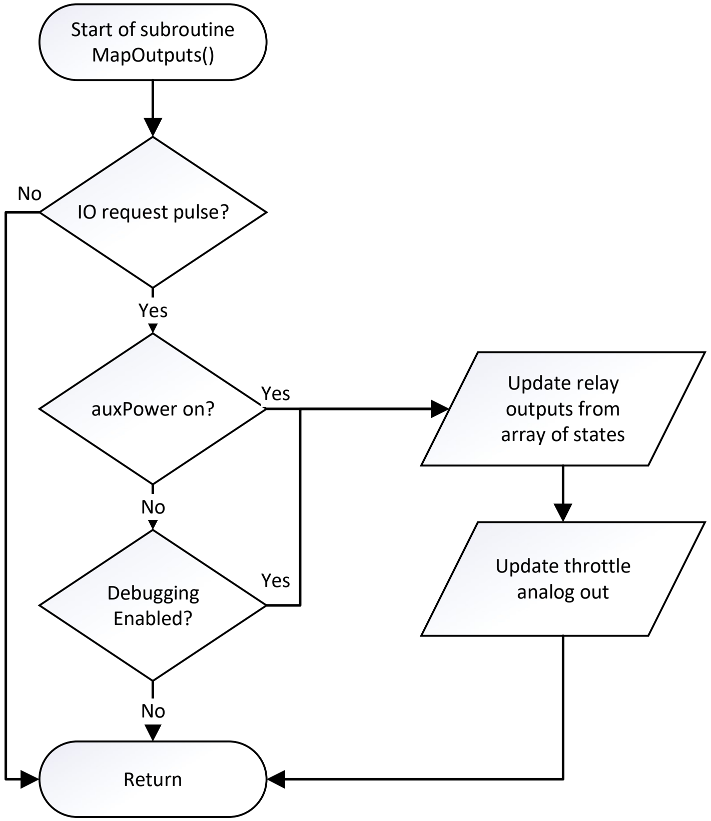
Loop through main program

TITLE	DRAWN BY	DATE	PAGE
Main Program Flow	COLE FUERTH	03/20/2020	4 OF 12


Start of subroutine MapInputs()
IO request pulse?
No
Yes
Read speed px state
Return
Read digital states and store in raw variables
Read one analog value per request
Read throttle analog input

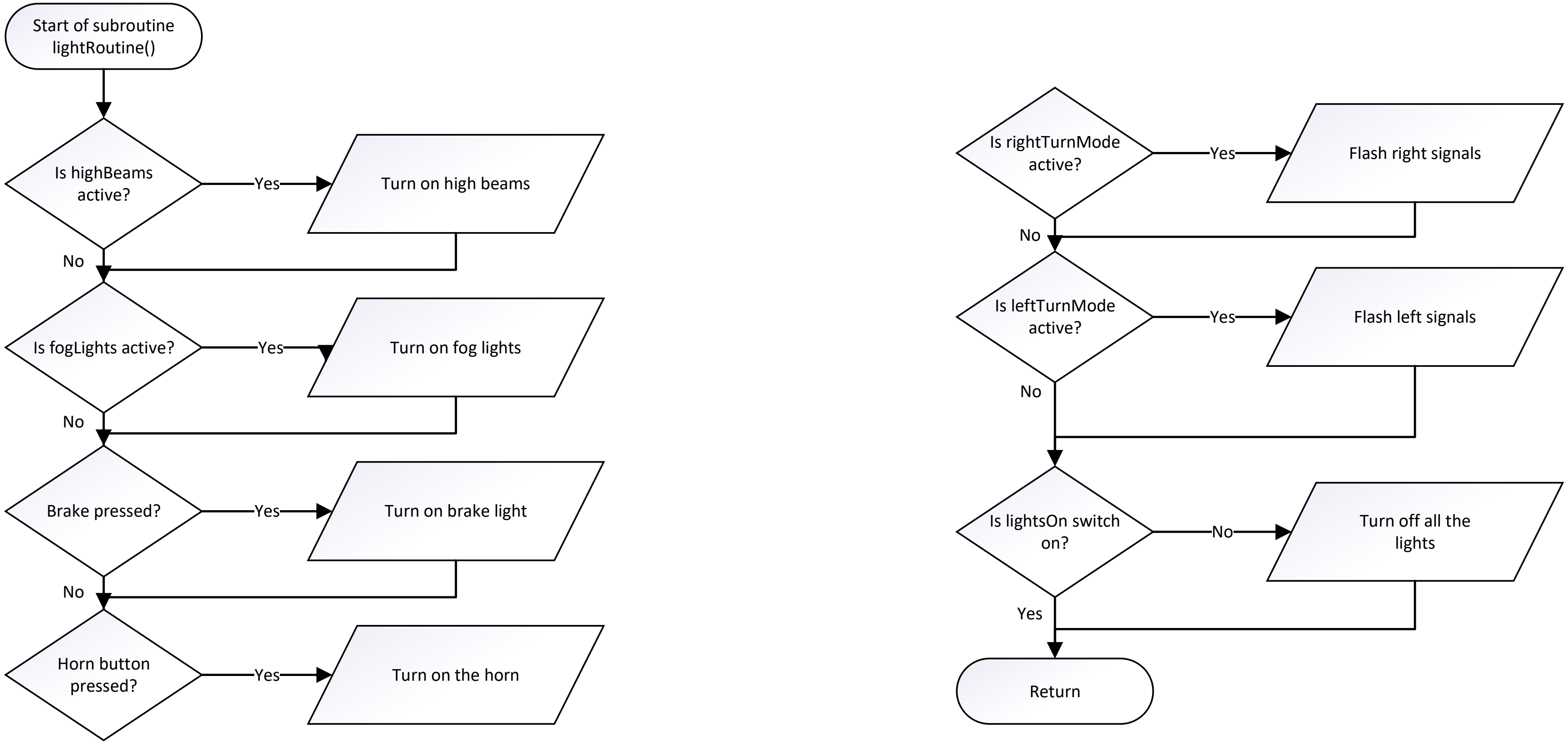
Map & Input signals and states from pins and store inputs in arrays

TITLE	DRAWN BY	DATE	PAGE
Subroutine MapInputs()	COLE FUERTH	03/20/2020	5 OF 12


Start of subroutine MapOutputs()
IO request pulse?
No
Yes
auxPower on?
Yes
No
Update relay outputs from array of states
Debugging Enabled?
Yes
No
Update throttle analog out
Return

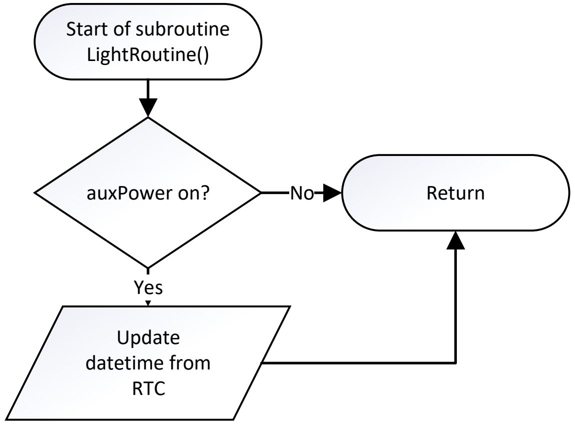
Use arrays for bits and pins to write throttle value and relay states

TITLE	DRAWN BY	DATE	PAGE
Subroutine MapOutputs()	COLE FUERTH	03/20/2020	6 OF 12



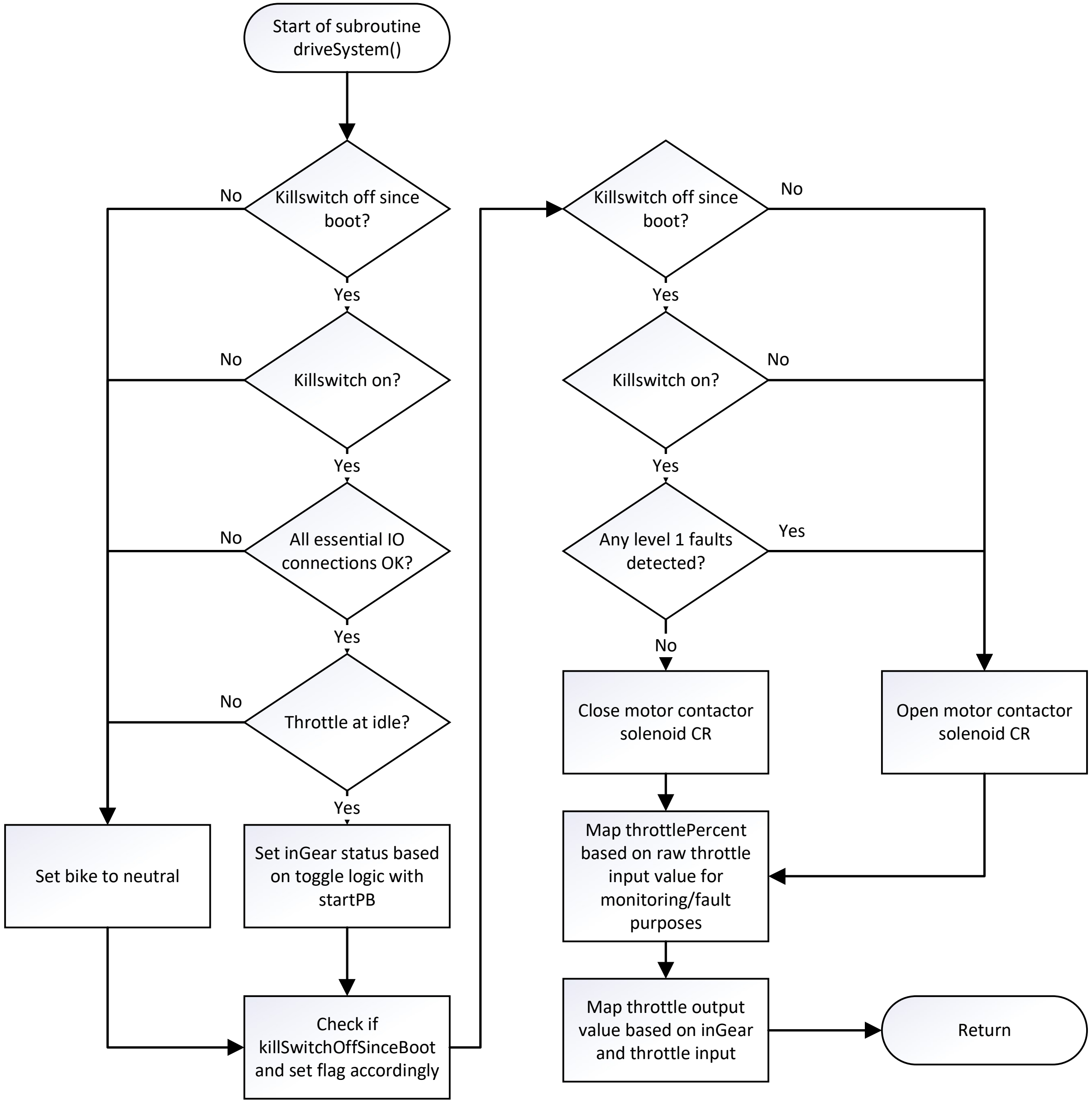
All bits used to control individual light relays are conditioned here

TITLE	DRAWN BY	DATE	PAGE
Subroutine LightRoutine()	COLE FUERTH	03/20/2020	7 OF 12



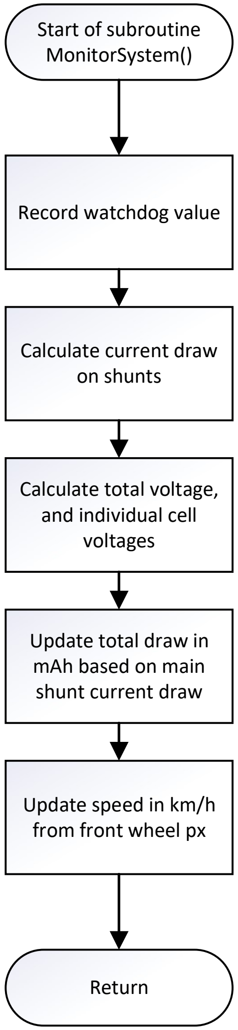
Update clock value based on RTC

TITLE	DRAWN BY	DATE	PAGE
Subroutine LightRoutine()	COLE FUERTH	03/20/2020	8 OF 12


Start of subroutine driveSystem()
Killswitch off since boot?
No
Yes
Killswitch on?
No
Yes
All essential IO connections OK?
No
Yes
Throttle at idle?
No
Yes
Set bike to neutral
Set inGear status based on toggle logic with startPB
Check if killSwitchOffSinceBoot and set flag accordingly
Killswitch off since boot?
No
Yes
Killswitch on?
No
Yes
Any level 1 faults detected?
Yes
No
Close motor contactor solenoid CR
Open motor contactor solenoid CR
Map throttlePercent based on raw throttle input value for monitoring/fault purposes
Map throttle output value based on inGear and throttle input
Return

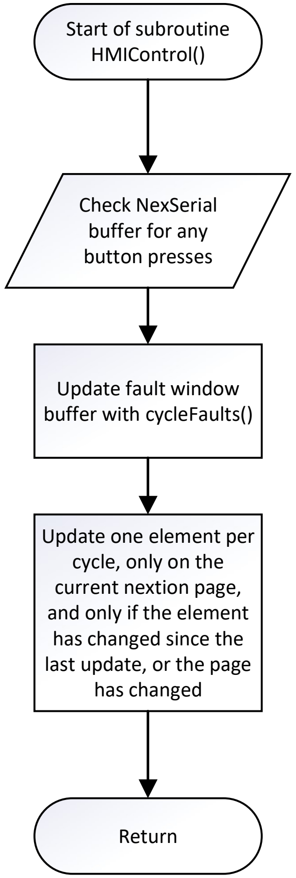
User throttle input and states from Modes() are used to generate a PWM signal for the ESC

TITLE	DRAWN BY	DATE	PAGE
Subroutine DriveSystem()	COLE FUERTH	03/20/2020	9 OF 12


Start of subroutine MonitorSystem()
Record watchdog value
Calculate current draw on shunts
Calculate total voltage, and individual cell voltages
Update total draw in mAh based on main shunt current draw
Update speed in km/h from front wheel px
Return

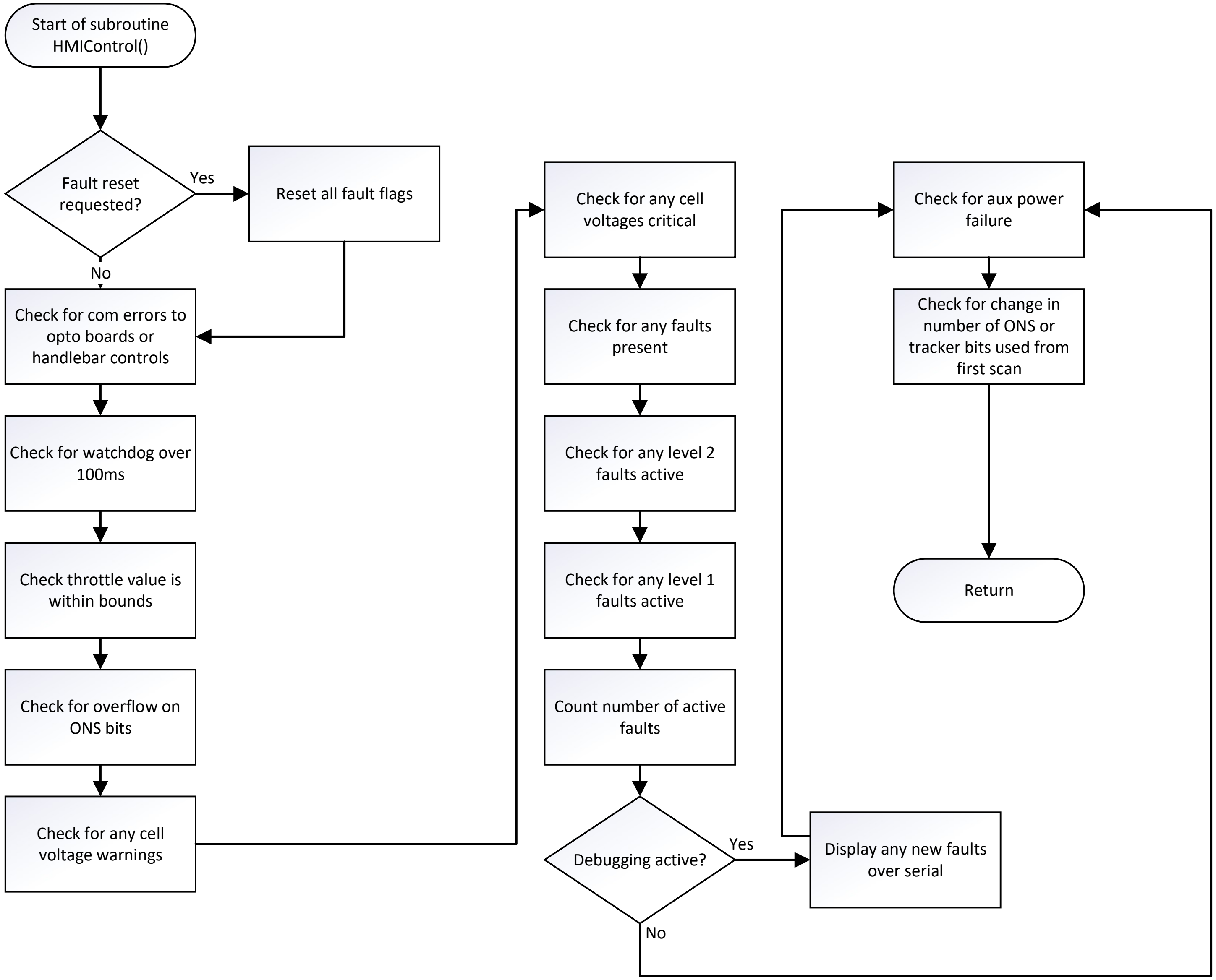
Use inputs from analog sensors to read and record telemetry from the system, including Vbat, current draw, and efficiency.

TITLE	DRAWN BY	DATE	PAGE
Subroutine MonitorSystem()	COLE FUERTH	03/20/2020	10 OF 12


Start of subroutine HMIControl()
Check NexSerial buffer for any button presses
Update fault window buffer with cycleFaults()
Update one element per cycle, only on the current nextion page, and only if the element has changed since the last update, or the page has changed
Return

Update Nextion elements based on telemetry gathered

TITLE	DRAWN BY	DATE	PAGE
Subroutine HMIControl()	COLE FUERTH	03/20/2020	11 OF 12


Start of subroutine HMIControl()
Fault reset requested?
Yes
Reset all fault flags
No
Check for com errors to opto boards or handlebar controls
Check for watchdog over 100ms
Check throttle value is within bounds
Check for overflow on ONS bits
Check for any cell voltage warnings
Check for any cell voltages critical
Check for any faults present
Check for any level 2 faults active
Check for any level 1 faults active
Count number of active faults
Debugging active?
Yes
Display any new faults over serial
No
Check for aux power failure
Check for change in number of ONS or tracker bits used from first scan
Return

Set fault flags based on current machine status

TITLE	DRAWN BY	DATE	PAGE
Subroutine Faults()	COLE FUERTH	03/20/2020	12 OF 12