**IterativeRobot**

+StartCompetition(): void
+RobotInit(): void
+DisabledInit(): void
+AutonomousInit(): void
+TeleopInit(): void
+DisabledPeriodic(): void
+AutonomousPeriodic(): void
+TeleopPeriodic(): void
+DisabledContinuous(): void
+AutonomousContinuous(): void
+TeleopContinuous(): void
+SetPeriod(period: double): void
+GetLoopsPerSec(): double
<<destroy>>-IterativeRobot()
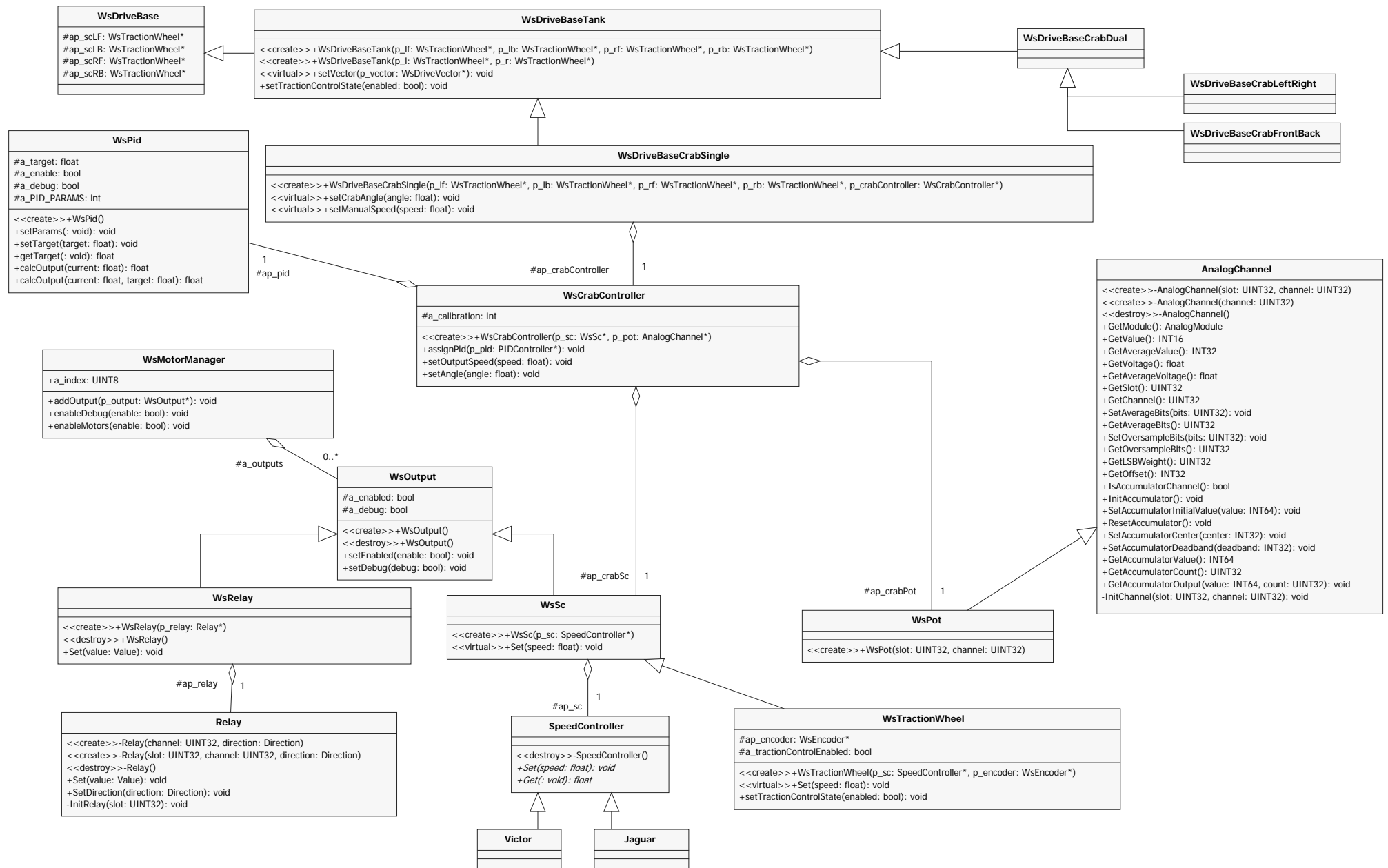<<create>>-IterativeRobot()
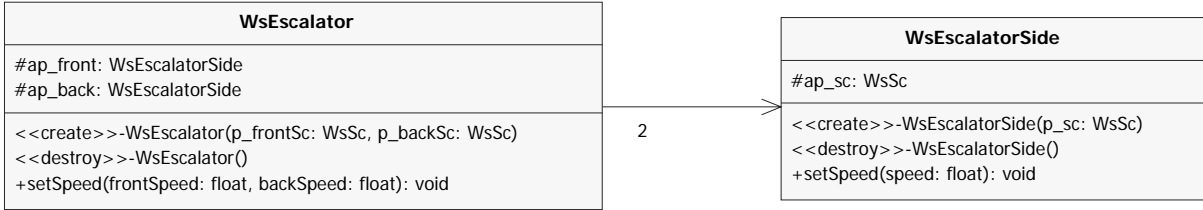-NextPeriodReady(): bool

Details found on the AppendageOutputDetails diagram

Details found on the DriveBaseOutputDetails diagram

Details found on the InputDetails diagram

**WsRobot2009**

#ap_driveJoystick: Joystick*
#ap_crabJoystick: Joystick*
#ap_appendageGamepad: Gamepad*
#ap_driveAxisX: WsControllerAxis*
#ap_driveAxisY: WsControllerAxis*
#ap_crabAxisX: WsControllerAxis*
#ap_crabAxisY: WsControllerAxis*
#ap_elevatorAxisX: WsControllerAxis*
#ap_elevatorAxisY: WsControllerAxis*
#ap_escalatorAxisX: WsControllerAxis*
#ap_escalatorAxisY: WsControllerAxis*
#ap_driveStick: WsControllerPolarStick*
#ap_crabStick: WsControllerPolarStick*
#ap_elevatorStick: WsControllerPolarStick*
#ap_escalatorStick: WsControllerPolarStick*
#ap_turboButton: WsControllerButton*
#ap_tractionButton: WsControllerButton*
#ap_crabDisableButton: WsControllerButton*
#ap_crabManualButton: WsControllerButton*
#ap_shooterOutButton: WsControllerButton*
#ap_shooterOutSlowButton: WsControllerButton*
#ap_shooterInButton: WsControllerButton*
#ap_accumulatorOutButton: WsControllerButton*
#ap_accumulatorInButton: WsControllerButton*
#ap_appendageStickyButton: WsControllerButton*
#ap_appendageStickyStopButton: WsControllerButton*
#ap_accumLoadButton: WsControllerButton*
#ap_appendageControl: WsAppendageControl*
#ap_scDriveFL: SpeedController*
#ap_scDriveBL: SpeedController*
#ap_scDriveFR: SpeedController*
#ap_scDriveBR: SpeedController*
#ap_scCrabL: SpeedController*
#ap_scCrabR: SpeedController*
#ap_scAccumulator: SpeedController*
#ap_scEscalatorF: SpeedController*
#ap_scEscalatorB: SpeedController*
#ap_rlyElevatorL: Relay*
#ap_rlyElevatorR: Relay*
#ap_scShooter: SpeedController*
#ap_wScDriveFL: WsTractionWheel*
#ap_wScDriveFR: WsTractionWheel*
#ap_wScDriveBL: WsTractionWheel*
#ap_wScDriveBR: WsTractionWheel*
#ap_wScCrabL: WsSc*
#ap_wScCrabR: WsSc*
#ap_wScAccumulator: WsSc*
#ap_wScEscalatorF: WsSc*
#ap_wScEscalatorB: WsSc*
#ap_wRlyElevatorL: WsRelay*
#ap_wRlyElevatorR: WsRelay*
#ap_wScShooter: WsSc*
#ap_crabPot: WsPot*
#ap_crabPID: WsPid*
#ap_elevatorL: WsElevator*
#ap_elevatorR: WsElevator*
#ap_dashboard: WsDashboard*
#ap_logger: WsLogger*
#ap_calib: WsCalibration*
#ap_calibrateEnableButton: WsControllerButton*
#ap_calibrateCrabPotLeftButton: WsControllerButton*
#ap_calibrateCrabPotMidButton: WsControllerButton*
#ap_calibrateCrabPotRightButton: WsControllerButton*
#ap_controlMethodCrab: WsControlMethodCrab*
#ap_crabController: WsCrabController*
#ap_driveBase: WsDriveBaseCrabSingle*
#ap_mmgrDriveBase: WsMotorManager*
#ap_mmgrRobot: WsMotorManager*
#ap_accumulator: WsAccumulator*
#ap_escalator: WsEscalator*
#ap_shooter: WsShooter*

<<destroy>>-WsRobot2009()
+RobotInit(: void): void
+DisabledInit(: void): void
+AutonomousInit(: void): void
+TeleopInit(: void): void
+DisabledPeriodic(: void): void
+AutonomousPeriodic(: void): void
+TeleopPeriodic(: void): void
+AutonomousContinuous(: void): void
+TeleopContinuous(: void): void
+DisabledContinuous(: void): void
#processCalibration(: void): void

# Joystick

```
<<create>>-Joystick(port: UINT32)
<<destroy>>-Joystick()
+GetAxisChannel(axis: AxisType): UINT32
+SetAxisChannel(axis: AxisType, channel: UINT32): void
+GetX(hand: JoystickHand): float
+GetY(hand: JoystickHand): float
+GetZ(: void): float
+GetTwist(: void): float
+GetThrottle(: void): float
+GetAxis(axis: AxisType): float
+GetRawAxis(axis: UINT32): float
+GetTrigger(hand: JoystickHand): bool
+GetTop(hand: JoystickHand): bool
+GetBumper(hand: JoystickHand): bool
+GetButton(button: ButtonType): bool
+GetRawButton(button: UINT32): bool
+GetStickForPort(port: UINT32)
<<create>>-Joystick(port: UINT32, numAxisTypes: UINT32, numButtonTypes: UINT32)
-InitJoystick(numAxisTypes: UINT32, numButtonTypes: UINT32): void
```

# WsRegularJoystick

# WsFlightStickJoystick

# WsAttack3Joystick

# WsGamepadJoystick

# WsPs2Joystick

# WsAppendageControl

```
#ap_elevatorStick: WsControllerPolarStick*
#ap_escalatorStick: WsControllerPolarStick*
#ap_shootOutButton: WsControllerButton*
#ap_shootOutSlowButton: WsControllerButton*
#ap_shootInButton: WsControllerButton*
#ap_accumulatorInButton: WsControllerButton*
#ap_accumulatorOutButton: WsControllerButton*
#ap_stickyEnableButton: WsControllerButton*
#ap_stickyStopButton: WsControllerButton*
#ap_accumLoadButton: WsControllerButton
```

```
<<create>>+WsAppendageControl()
+getAccumulatorCommand(: void): WsAccumulatorStateT
+getEscalatorSpeeds(: void): void
+getLeftElevatorCommand(: void): void
+getRightElevatorCommand(: void): void
+getShooterCommand(: void): WsShooterStateT
+assignElevatorStick(p_stick: WsControllerPolarStick*): void
+assignEscalatorStick(p_stick: WsControllerPolarStick*): void
+assignAccumulatorButtons(p_in: WsControllerButton*, p_out: WsControllerButton*): void
+assignShooterButtons(p_in: WsControllerButton*, p_out: WsControllreButton*, p_outSlow: WsControllerButton*): void
+assignStickyButtons(p_enable: WsControllerButton*, p_stop: WsControllerButton*): void
+assignAccumLoadButton(p_button: WsControllerButton*): void
```

# Gamepad

```
<<create>>-Gamepad(port: UINT32)
<<destroy>>-Gamepad()
+GetLeftX(): float
+GetLeftY(): float
+GetRightX(): float
+GetRightY(): float
+GetAxis(axis: AxisType): float
+GetRawAxis(axis: UINT32): float
+GetNumberedButton(buttonNumber: unsigned): bool
+GetLeftPush(): bool
+GetRightPush(): bool
+GetDPad(): DPadDirection
```

1  #ap_joystick

#ap_gamepad

1

# WsControllerBase

```
#a_axisNum: UINT32
#a_buttonNum: UINT32
#a_channelNum: UINT32
#a_invert: bool
#ap_ds: DriverStation*
```

# WsControllerButton

```
<<create>>+WsControllerButton(p_js: Joystick*, button: UINT32)
<<create>>+WsControllerButton(p_pad, Gamepad*, button: UINT32)
<<create>>+WsControllerButton(channel: UINT32)
<<virtual>>+getState(: void): bool
```

# WsDsAnalog

```
<<create>>+WsDsAnalog(channel: UINT32)
+getValue(): UINT32
```

# WsControllerAxis

```
<<create>>+WsControllerAxis(p_js: Joystick*, axis: UINT32)
<<create>>+WsControllerAxis(p_pad: Gamepad*, axis: UINT32)
<<virtual>>+getValue(: void): float
```

2

# WsControlMethodBase

# WsTankControlMethod

# WsControllerStickyButton

```
#a_statePrev: bool
#a_stickyState: bool
```

```
<<create>>+WsControllerStickyButton(p_js: Joystick*, button: UINT32)
<<virtual>>+getState(: void): bool
```

# WsControllerMultiStickyButton

```
#a_stickyState: bool
#a_offButtonNum: UINT32
```

```
<<create>>+WsControllerMultiStickyButton(p_js: Joystick*, onButton: UINT32, offButton: UINT32)
<<virtual>>+getState(: void): bool
```

# WsControllerAnalogButton

```
#a_threshold: float
#a_aboveOn: bool
```

```
<<create>>+WsControllerAnalogButton(p_js: Joystick*, axis: UINT32, thresh: float, aboveOn: bool)
<<virtual>>+getState(: void): bool
```

# WsControllerTransitionButton

# WsControllerPolarStick

```
#ap_xAxis: WsControllerAxis*
#ap_yAxis: WsControllerAxis*
```

```
<<create>>+WsControllerPolarStick(xAxis: WsControllerAxis*, yAxis: WsControllerAxis*)
+getRadius(: void): float
+getAngle(: void): float
+getX(: void): float
+getY(: void): float
#calcAngle(x: float, y: float): float
#calcRadius(x: float, y: float): float
```

+ap_driveStick

# WsControlMethodArcade

```
#ap_turboButton: WsControllerButton*
#a_turboScaleFactor: float
```

```
<<create>>+WsControlMethodArcade()
+assignDriveStick(p_stick: WsControllerPolarStick*)
+assignTurboButton(p_turbo: WsControllerButton*)
<<virtual>>+getDriveVector(: void): WsDriveVector
```

1

# WsDriveVector

```
+a_speed: float
+a_direction: float
```

```
<<create>>+WsDriveVector(speed: float, direction: float)
```

1  #ap_crabStick

# WsControlMethodCrab

```
#ap_crabDisableButton: WsControllerButton*
#ap_crabManualButton: WsControllerButton*
#a_crab_stick_radius_thresh: float
#a_is_crabbed_thresh: float
#a_crab_manu_scale: float
```

```
<<create>>+WsControlMethodCrab()
+assignCrabStick(p_stick: WsControllerPolarStick*)
+assignCrabDisableButton(p_disable: WsControllerButton*)
+assignCrabManualButton(p_manual: WsControllerButton*)
<<virtual>>+getDriveVector(: void): WsDriveVector
+getCrabEnableState(: void): bool
+getCrabWheelAngle(: void): float
+getCrabManualSpeed(: void): float
```

**WsDriveBase**

#ap_scLF: WsTractionWheel*
#ap_scLB: WsTractionWheel*
#ap_scRF: WsTractionWheel*
#ap_scRB: WsTractionWheel*

---

**WsDriveBaseTank**

<<create>>+WsDriveBaseTank(p_lf: WsTractionWheel*, p_lb: WsTractionWheel*, p_rf: WsTractionWheel*, p_rb: WsTractionWheel*)
<<create>>+WsDriveBaseTank(p_l: WsTractionWheel*, p_r: WsTractionWheel*)
<<virtual>>+setVector(p_vector: WsDriveVector*): void
+setTractionControlState(enabled: bool): void

---

**WsDriveBaseCrabDual**

---

**WsDriveBaseCrabLeftRight**

---

**WsDriveBaseCrabFrontBack**

---

**WsPid**

#a_target: float
#a_enable: bool
#a_debug: bool
#a_PID_PARAMS: int

<<create>>+WsPid()
+setParams(: void): void
+setTarget(target: float): void
+getTarget(: void): float
+calcOutput(current: float): float
+calcOutput(current: float, target: float): float

---

**WsDriveBaseCrabSingle**

<<create>>+WsDriveBaseCrabSingle(p_lf: WsTractionWheel*, p_lb: WsTractionWheel*, p_rf: WsTractionWheel*, p_rb: WsTractionWheel*, p_crabController: WsCrabController*)
<<virtual>>+setCrabAngle(angle: float): void
<<virtual>>+setManualSpeed(speed: float): void

1   #ap_pid

#ap_crabController   1

---

**WsCrabController**

#a_calibration: int

<<create>>+WsCrabController(p_sc: WsSc*, p_pot: AnalogChannel*)
+assignPid(p_pid: PIDController*): void
+setOutputSpeed(speed: float): void
+setAngle(angle: float): void

---

**AnalogChannel**

<<create>>-AnalogChannel(slot: UINT32, channel: UINT32)
<<create>>-AnalogChannel(channel: UINT32)
<<destroy>>-AnalogChannel()
+GetModule(): AnalogModule
+GetValue(): INT16
+GetAverageValue(): INT32
+GetVoltage(): float
+GetAverageVoltage(): float
+GetSlot(): UINT32
+GetChannel(): UINT32
+SetAverageBits(bits: UINT32): void
+GetAverageBits(): UINT32
+SetOversampleBits(bits: UINT32): void
+GetOversampleBits(): UINT32
+GetLSBWeight(): UINT32
+GetOffset(): INT32
+IsAccumulatorChannel(): bool
+InitAccumulator(): void
+SetAccumulatorInitialValue(value: INT64): void
+ResetAccumulator(): void
+SetAccumulatorCenter(center: INT32): void
+SetAccumulatorDeadband(deadband: INT32): void
+GetAccumulatorValue(): INT64
+GetAccumulatorCount(): UINT32
+GetAccumulatorOutput(value: INT64, count: UINT32): void
-InitChannel(slot: UINT32, channel: UINT32): void

---

**WsMotorManager**

+a_index: UINT8

+addOutput(p_output: WsOutput*): void
+enableDebug(enable: bool): void
+enableMotors(enable: bool): void

#a_outputs   0..*

---

**WsOutput**

#a_enabled: bool
#a_debug: bool

<<create>>+WsOutput()
<<destroy>>+WsOutput()
+setEnabled(enable: bool): void
+setDebug(debug: bool): void

#ap_crabSc   1

#ap_crabPot   1

---

**WsRelay**

<<create>>+WsRelay(p_relay: Relay*)
<<destroy>>+WsRelay()
+Set(value: Value): void

#ap_relay   1

---

**WsSc**

<<create>>+WsSc(p_sc: SpeedController*)
<<virtual>>+Set(speed: float): void

#ap_sc   1

---

**WsPot**

<<create>>+WsPot(slot: UINT32, channel: UINT32)

---

**Relay**

<<create>>-Relay(channel: UINT32, direction: Direction)
<<create>>-Relay(slot: UINT32, channel: UINT32, direction: Direction)
<<destroy>>-Relay()
+Set(value: Value): void
+SetDirection(direction: Direction): void
-InitRelay(slot: UINT32): void

---

**SpeedController**

<<destroy>>-SpeedController()
+Set(speed: float): void
+Get(: void): float

---

**WsTractionWheel**

#ap_encoder: WsEncoder*
#a_tractionControlEnabled: bool

<<create>>+WsTractionWheel(p_sc: SpeedController*, p_encoder: WsEncoder*)
<<virtual>>+Set(speed: float): void
+setTractionControlState(enabled: bool): void

---

**Victor**

---

**Jaguar**

## WsAccumulator

#ap_sc: WsSc
#a_accumSpeedIn: float
#a_accumSpeedOut: float

<<create>>-WsAccumulator(p_sc: WsSc)
<<destroy>>-WsAccumulator()
+setState(state: WsAccumulatorStateT): void

## WsEscalator

#ap_front: WsEscalatorSide
#ap_back: WsEscalatorSide

<<create>>-WsEscalator(p_frontSc: WsSc, p_backSc: WsSc)
<<destroy>>-WsEscalator()
+setSpeed(frontSpeed: float, backSpeed: float): void

2

## WsEscalatorSide

#ap_sc: WsSc

<<create>>-WsEscalatorSide(p_sc: WsSc)
<<destroy>>-WsEscalatorSide()
+setSpeed(speed: float): void

## WsElevator

#ap_relay: WsRelay
#ap_topSwitch: DigitalInput
#ap_bottomSwitch: DigitalInput
#a_elevatorOffValue: Value
#a_elevatorUpValue: Value
#a_elevatorDownValue: Value

<<create>>-WsElevator(p_relay: WsRelay)
+assignLimitSwitches(p_topLimit: DigitalInput, p_bottomLimit: DigitalInput): void
+moveUp(: void): void
+moveDown(: void): void
+turnOff(: void): void
+moveToPosition(position: WsElevatorPosT): bool

## WsShooter

#ap_sc: WsSc
#a_shooterSpeedIn: float
#a_shooterSpeedOut: float
#a_shooterSpeedOutSlow: float

<<create>>-WsShooter(p_sc: WsSc)
+setState(state: WsShooterStateT): void

**WsAutonomousMgr**

#ap_progNumSelector: WsDsAnalog*
#ap_locationSelector: WsDsAnalog*
#ap_lockInSwitch: WsControllerButton*
#ap_driveBase: WsDriveBase*
#ap_accumulator: WsAccumulator*
#ap_escalator: WsEscalator*
#ap_elevatorLeft: WsElevator*
#ap_elevatorRight: WsElevator*
#ap_shooter: WsShooter*
#a_progNum: UINT8
#a_location: StartingLocationT

<<create>>+WsAutonomousMgr(p_progNumSelector: WsDsAnalog*, p_locationSelector: WsDsAnalog*, p_lockInSwitch: WsControllerButton*)
<<destroy>>+WsAutonomousMgr()
+assignDriveBase(p_driveBase: WsDriveBase*): void
+assignAccumulator(p_accum: WsAccumulator*): void
+assignEscalator(p_esc: WsEscalator*): void
+assignElevators(p_elevLeft: WsElevator*, p_elevRight: WsElevator*): void
+assignShooter(p_shooter: WsShooter*): void
+processLockin(: void): void
+beginAutonomous(: void): void
+run(: void): bool

#ap_program        1

**WsAutoProgramBase**

#ap_driveBase: WsDriveBase*
#ap_accumulator: WsAccumulator*
#ap_escalator: WsEscalator*
#ap_elevatorLeft: WsElevator*
#ap_elevatorRight: WsElevator*
#ap_shooter: WsShooter*

<<create>>+WsAutoProgramBase()
<<virtual>>+run(: void): bool
+assignDriveBase(p_driveBase: WsDriveBase*): void
+assignAccumulator(p_accum: WsAccumulator*): void
+assignEscalator(p_esc: WsEscalator*): void
+assignElevators(p_elevLeft: WsElevator*, p_elevRight: WsElevator*): void
+assignShooter(p_shooter: WsShooter*): void
+turnOffDriveBase(: void): void
+turnOffAppendages(: void): void

**WsAutoOutpostHomeDsBase**

**WsAutoOutpostHomeDsStraight**

**WsAutoSleeper**

<<create>>+WsAutoSleeper()
<<virtual>>+run(: void): bool

**WsAutoOutpostOutpostBase**

#ap_timer: Timer*
#a_progState: WsAutoOutpostOutpostStateT
#a_encoderCount: UINT32
#a_timerCount: double
#a_initDelay: double
#a_initAccumState: WsAccumulatorStateT
#a_initEscFrontSpeed: float
#a_initEscBackSpeed: float
#a_initShooterState: WsShooterStateT
#a_initElevLeftState: WsElevatorStateT
#a_initElevRightState: WsElevatorStateT
#a_drive1CrabAngle: float
#a_drive1Vector: WsDriveVector
#a_drive1Timeout: double
#a_drive2Pause: double
#a_drive2CrabAngle: float
#a_drive2Vector: WsDriveVector
#a_drive2Timeout: double

<<create>>+WsAutoOutpostOutpostBase()
<<virtual>>+run(: void): bool

**WsAutoOutpostOppDsBase**

**WsAutoOutpostOppDsStraight**

**WsAutoOutpostOutpostLeft**

#a_initDelay: double
#a_drive1CrabAngle: float
#a_drive1Vector: WsDriveVector
#a_drive1Timeout: double
#a_drive2Pause: double
#a_drive2CrabAngle: float
#a_drive2Vector: WsDriveVector
#a_drive2Timeout: double

<<create>>+WsAutoOutpostOutpostLeft()

**WsAutoOutpostOutpostStraight**

#a_initDelay: double
#a_drive1CrabAngle: float
#a_drive1Vector: WsDriveVector
#a_drive1Timeout: double
#a_drive2Pause: double
#a_drive2CrabAngle: float
#a_drive2Vector: WsDriveVector
#a_drive2Timeout: double

<<create>>+WsAutoOutpostOutpostStraight()

**WsAutoOutpostOutpostRight**

#a_initDelay: double
#a_drive1CrabAngle: float
#a_drive1Vector: WsDriveVector
#a_drive1Timeout: double
#a_drive2Pause: double
#a_drive2CrabAngle: float
#a_drive2Vector: WsDriveVector
#a_drive2Timeout: double

<<create>>+WsAutoOutpostOutpostRight()

## WsEncoder

#a_addr: UINT8
#a_validEncoder: bool

<<create>>+WsEncoder(addr: UINT8)
<<destroy>>+WsEncoder()
+isValid(: void): bool
+getAddress(: void): UINT8
<<virtual>>+getDrivenWheelSpeed(: void): INT16
<<virtual>>+getIdlerWheelSpeed(: void): INT16
<<virtual>>+getSpeedDiff(: void): INT16
<<virtual>>+getSlipRatio(: void): INT16

## WsEncoderManager

#a_encoderIds: UINT8

<<create>>+WsEncoderManager()
<<destroy>>+WsEncoderManager()
+init(: void): bool
+getEncoder(encoder: WsWheelPosT): WsEncoder*
+discoverEncoder(wheel: WsWheelPosT, direction: WsDirectionT): bool
+discoverDone(: void): void

## WsI2cEncoder

#kManufacturerBaseRegister: UINT8
#kManufacturerSize: UINT8
#kSensorTypeBaseRegister: UINT8
#kSensorTypeSize: UINT8
#kIdlerWheelSpeedRegister: UINT8
#kDrivenWheelSpeedRegister: UINT8
#kSpeedDiffRegister: UINT8
#kSlipRatioRegister: UINT8
#ap_i2c: I2C*

<<create>>+WsI2cEncoder(slot: UINT32, addr: UINT8)
<<destroy>>+WsI2cEndocer()
<<virtual>>+getDrivenWheelSpeed(: void): INT16
<<virtual>>+getIdlerWheelSpeed(: void): INT16
<<virtual>>+getSpeedDiff(: void): INT16
<<virtual>>+getSlipRatio(: void): INT16

## WsShmEncoder

#a_invert: bool

<<create>>+WsShmEncoder(addr: UINT8, invert: bool)
<<destroy>>+WsShmEncoder()
<<virtual>>+getDrivenWheelSpeed(: void): INT16
<<virtual>>+getIdlerWheelSpeed(: void): INT16
<<virtual>>+getSpeedDiff(: void): INT16
<<virtual>>+getSlipRatio(: void): INT16

Sequence diagram with the following lifelines:

- : WsControlMethodCrab
- ap_driveAxisY : WsControllerAxis
- ap_crabAxisX : WsControllerAxis
- ap_driveStick : WsAttack3Joystick
- ap_crabStick : WsAttack3Joystick
- ap_driveBase : WsDriveBaseCrabSingle
- ap_crabController : WsCrabController
- ap_crabPID : WsPid
- ap_crabPot : AnalogChannel
- ap_scDriveFL : Jaguar
- ap_scCrab : Jaguar

Messages:

1 : getDriveVector()
2 : getValue()

*will happen for x and y*

3 : GetRawAxis()
4 : axis_value
5 : value
6 : driveVector
7 : getCrabEnableState()
8 : crabEnableState
9 : getCrabWheelAngle()
10 : getValue()
11 : GetRawAxis()

*Calculate wheel angle from axis value*

12 : axis_value
13 : value
14 : wheel_angle

*Drive motor speeds calculated from the vector*

15 : setVector()
16 : Set()
17 : setCrabAngle()
18 : setAngle()
19 : GetValue()
20 : potValue
21 : calcOutput()
22 : outputSpeed
23 : setOutputSpeed()
24 : Set()