struct HUSKYLENSResult

- **Description:** Structure to store the blocks or arrows
- **Member:**
  - `command` Use to determine whether this is arrow or block
    - `COMMAND_RETURN_BLOCK` It is a block
    - `COMMAND_RETURN_ARROW` It is an arrow
  - For a block:
    - `xCenter` X Center of Block
    - `yCenter` Y Center of Block
    - `width` Width of Block
    - `height` Height of Block
    - `ID` ID of Block, see **ID Meaning** below
  - For an arrow:
    - `xOrigin` X Origin of Arrow
    - `yOrigin` Y Origin of Arrow
    - `xTarget` X Target of Arrow
    - `yTarget` Y Target of Arrow
    - `ID` ID of Arrow, see **ID Meaning** below
- **Example:**

```
void printResult(HUSKYLENSResult result){
    if (result.command == COMMAND_RETURN_BLOCK){
   Serial.println(String() + F("Block:xCenter=") + result.xCenter + F(",yCenter=")
+ result.yCenter + F(",width=") + result.width + F(",height=") + result.height +
F(",ID=") + result.ID);
    }
    else if (result.command == COMMAND_RETURN_ARROW){
        Serial.println(String() + F("Arrow:xOrigin=") + result.xOrigin +
F(",yOrigin=") + result.yOrigin + F(",xTarget=") + result.xTarget + F(",yTarget=")
+ result.yTarget + F(",ID=") + result.ID);
    }
    else{
        Serial.println("Object unknown!");
    }
}
```

ID Meaning:

ID	Means
1	The first learned item is detected
2	The second learned item is detected
XXX	The XXXth learned item is detected
0	Item is detected but not learned, like unlearned faces block in grey color.

enum protocolAlgorithm

- **Description:** The enum of the algorithm. Needed when switch to the target algorithm.
- **Member:**
  - `ALGORITHM_FACE_RECOGNITION` FACE RECOGNITION
  - `ALGORITHM_OBJECT_TRACKING`OBJECT TRACKING
  - `ALGORITHM_OBJECT_RECOGNITION` OBJECT RECOGNITION
  - `ALGORITHM_LINE_TRACKING` LINE TRACKING
  - `ALGORITHM_COLOR_RECOGNITION` COLOR RECOGNITION
  - `ALGORITHM_TAG_RECOGNITION` TAG RECOGNITION
  - `ALGORITHM_OBJECT_CLASSIFICATION` OBJECT CLASSIFICATION

bool begin(TwoWire& streamInput)

- **Description:** Setup procedure of HUSKYLENS with Wire(I2C). It will try connect to HUSKYLENS and return whether HUSKYLENS is connected.
- **Arguments:**
  - `streamInput` : It could be Serial, Wire, SoftwareSerial, or other port based on Stream class.
- **Returns:** Whether successful connect and contact with HUSKYLENS.

bool begin(Stream& streamInput)

- **Description:** Setup procedure of HUSKYLENS. It will try connect to HUSKYLENS and return whether HUSKYLENS is connected.
- **Arguments:**
  - `streamInput` : It could be Serial, Wire, SoftwareSerial, or other port based on Stream class.
- **Returns:** Whether successful connect and contact with HUSKYLENS.

void setTimeOutDuration(unsigned long timeOutDurationInput)

- **Description:** Use to set the time out duration on the transmit between request and received from HUSKYLENS to avoid waiting the feedback from HUSKYLENS for a long time. Default value is 100ms.
- **Arguments:**
  - `timeOutDurationInput` : Time out duration on ms.

bool request()

- **Description:** Request all blocks and arrows from HUSKYLENS. This is the place where all the transmit happens.
- **Arguments:** None
- **Returns:** Whether successfully get the result.

bool request(int16_t ID)

- **Description:** Request only blocks and arrows tagged with `ID` from HUSKYLENS.
- **Arguments:**
  - `ID` The target ID of blocks and arrows
- **Returns:** Whether successfully get the result.

bool requestBlocks()

- **Description:** Request all blocks from HUSKYLENS
- **Returns:** Whether successfully get the result.

bool requestBlocks(int16_t ID)

- **Description:** Request only blocks tagged with `ID` from HUSKYLENS
- **Arguments:**
  - `ID` The target ID of blocks
- **Returns:** Whether successfully get the result..

bool requestArrows()

- **Description:** Request all arrows from HUSKYLENS
- **Arguments:**
  - `algorithmType` The algorithm you need. See protocolAlgorithm for details.
- **Returns:** Whether successfully get the result.

bool requestArrows(int16_t ID)

- **Description:** Request only arrows tagged with `ID` from HUSKYLENS
- **Arguments:**
  - `ID` The target ID of arrows
- **Returns:** Whether successfully get the result.

bool requestLearned()

- **Description:** Request all learned blocks and arrows (ID >=1) from HUSKYLENS.
- **Returns:** Whether successfully get the result.

bool requestBlocksLearned()

- **Description:** Request all learned blocks (ID >=1) from HUSKYLENS.
- **Returns:** Whether successfully get the result.

bool requestArrowsLearned()

- **Description:** Request all learned arrows (ID >=1) from HUSKYLENS.
- **Returns:** Whether successfully get the result.

int available()

- **Description:** Return the count of blocks and arrows available to read. (Works like Serial or Wire)
- **Arguments:** None
- **Returns:** The count of blocks and arrows left in the buffer.

HUSKYLENSResult read()

- **Description:** Read blocks or arrows.(Works like Serial or Wire)
- **Returns:** blocks or arrows in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

bool isLearned()

- **Description:** Get whether HUSKYLENS have learn something.
- **Returns:** Whether HUSKYLENS have learn something.

bool isLearned(int ID)

- **Description:** Get whether HUSKYLENS have learn something tagged with `ID`.
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
- **Returns:** Whether HUSKYLENS have learn something tagged with `ID`.

int16_t frameNumber()

- **Description:** Get the number of frame HUSKYLENS have processed. Once HUSKYLENS process one frame, this number will increase by one.
- **Returns:** The number of frame HUSKYLENS have processed.

int16_t countLearnedIDs()

- **Description:** Get the count of (faces, colors, objects or lines) you have learned on HUSKYLENS. This value will depend on how many times you learn something on HUSKYLENS.
- **Returns:** The count of (faces, colors, objects or lines) you have learned on HUSKYLENS.

int16_t count()

- **Description:** Get count of all blocks and arrows.
- **Returns:** The count of all blocks and arrows.

int16_t count(int16_t ID)

- **Description:** Get count of all blocks and arrows tagged with `ID`.
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
- **Returns:** The count of all blocks and arrows tagged with `ID`.

int16_t countBlocks()

- **Description:** Get count of all blocks.
- **Returns:** The count of all blocks.

int16_t countBlocks(int16_t ID)

- **Description:** Get count of all blocks tagged with `ID`.
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
- **Returns:** The count of all blocks tagged with `ID`.

int16_t countArrows()

- **Description:** Get count of all arrows.
- **Returns:** The count of all blocks and arrows.

int16_t countArrows(int16_t ID)

- **Description:** Get count of all arrows tagged with `ID`.
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
- **Returns:** The count of all arrows tagged with `ID`.

int16_t countLearned()

- **Description:** Get count of all learned blocks and arrows (ID >=1)
- **Returns:** The count of all learned blocks and arrows (ID >=1)

int16_t countBlocksLearned()

- **Description:** Get count of all learned blocks (ID >=1)
- **Returns:** The count of all learned blocks (ID >=1)

int16_t countArrowsLearned()

- **Description:** Get count of all learned arrows (ID >=1)
- **Returns:** The count of all learned arrows (ID >=1)

HUSKYLENSResult get(int16_t index)

- **Description:** Get one of the blocks and arrows.
- **Arguments:**
    - `index` The index of blocks and arrows, which is ordered by the received sequence. It should less than `count()`
- **Returns:** block or arrow in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult get(int16_t ID, int16_t index)

- **Description:** Get one of the blocks and arrows tagged with `ID`
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
    - `index` The index of blocks and arrows, which is ordered by the received sequence. It should less than `count(ID)`
- **Returns:** block or arrow tagged with `ID` in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getBlock(int16_t index)

- **Description:** Get one of the blocks.
- **Arguments:**
    - `index` The index of blocks, which is ordered by the received sequence. It should less than `countBlocks()`
- **Returns:** block in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getBlock(int16_t ID, int16_t index)

- **Description:** Get one of the blocks tagged with `ID`
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
    - `index` The index of blocks, which is ordered by the received sequence. It should less than `countBlocks(ID)`
- **Returns:** block tagged with `ID` in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getArrow(int16_t index)

- **Description:** Get one of the arrows.
- **Arguments:**
    - `index` The index of arrows, which is ordered by the received sequence. It should less than `countArrows()`
- **Returns:** arrow in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getArrow(int16_t ID, int16_t index)

- **Description:** Get one of the arrows tagged with `ID`
- **Arguments:**
    - `ID` The target ID. See ID Meaning above for details.
    - `index` The index of arrow, which is ordered by the received sequence. It should less than `countArrows(ID)`
- **Returns:** arrow tagged with `ID` in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getLearned(int16_t index)

- **Description:** Get one of the learned blocks and arrows (ID >=1)
- **Arguments:**
    - `index` The index of blocks and arrows, which is ordered by the received sequence. It should less than `countLearned()`
- **Returns:** block or arrow in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getBlockLearned(int16_t index)

- **Description:** Get one of the learned blocks (ID >=1)
- **Arguments:**
    - `index` The index of blocks, which is ordered by the received sequence. It should less than `countBlocksLearned()`
- **Returns:** block in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

HUSKYLENSResult getArrowLearned(int16_t index)

- **Description:** Get one of the learned arrows (ID >=1)
- **Arguments:**
    - `index` The index of arrows, which is ordered by the received sequence. It should less than `countArrowsLearned()`
- **Returns:** arrow in struct HUSKYLENSResult. See HUSKYLENSResult above for details.

bool writeAlgorithm(protocolAlgorithm algorithmType)

- **Description:** Let HUSKYLENS switch to the target algorithm you need.
- **Arguments:**
  - `algorithmType` The target algorithm. See protocolAlgorithm for details.
- **Returns:** Whether success.

bool writeLearn(int ID)

- **Description:** Let HUSKYLENS learn with `ID`. (It only works in Object Classification)
- **Arguments:**
  - `ID` The target ID. See ID Meaning above for details.
- **Returns:** Whether success.

bool writeForget()

- **Description:** Let HUSKYLENS forget all. (It only works in Object Classification)
- **Returns:** Whether success.

bool setCustomName(String name,uint8_t id)

- **Description:** Set a custom name for a learned object with a specified ID. For example, if you have learned your face with an ID of 1, you can use setCustomName("Robert",1) to rename the learned face to "Robert".
- **Arguments:**
  - `name` The specified custom name
  - `id` The ID of the object you want to set the custom name for
- **Returns:** Whether success.

bool savePictureToSDCard()

- **Description:** Save a photo from the HuskyLens camera onto the SD Card.
- **Returns:** Whether success. If there is no SD Card inserted or an SD Card Error, there will be a UI popup on the HuskyLens outlining the issue.

bool saveScreenshotToSDCard()

- **Description:** Save a screenshot of the HuskyLens UI onto the SD Card.
- **Returns:** Whether success.

bool saveModelToSDCard(int fileNum)

- **Description:** Save the current algorithms model file (its learned object data) to the SD Card. The file will be the in the format "AlgorithimName_Backup_*FileNum*.conf"
- **Arguments:**
  - `fileNum` The specified file number to be used in the name for the file
- **Returns:** Whether success. If there is no SD Card inserted or an SD Card Error, there will be a UI popup on the HuskyLens outlining the issue.

bool loadModelFromSDCard(int fileNum)

- **Description:** Load a model file from the SD Card to the current algorithm and refresh the algorithm. The loaded file will be the following format "AlgorithimName_Backup_*FileNum*.conf"
- **Arguments:**
  - `fileNum` The specified file number to be used in the name for the file
- **Returns:** Whether success. If there is no SD Card inserted or an SD Card Error, there will be a UI popup on the HuskyLens outlining the issue.

bool customText(String text,uint16_t x,uint8_t y)

- **Description:** Place a string of text (less than 20 characters) on top of the HuskyLens UI. The position of the texts (X,Y) coordinate is the top left of the text box.
    - You can have at most 10 custom texts on the UI at once, and if you continue adding texts you will replace previous texts in a circular fashion. For example, if you enter 10 texts you will fill the text buffer. If you then insert a new text object, you will overwrite the first text position (textBuffer[0]). Inserting another new text object will overwrite the second text position (textBuffer[1]).
    - Each text is uniquely identified by its (X,Y) coordinate, so you can replace the text string at a (X,Y) coordinate instead of adding a new text object. For example, if you insert "TEST_1" at (120,120) and then later submit "TEST_2" at (120,120), you will replace the string "TEST_1" with "TEST_2" and maintain an overall text count of 1.
- **Arguments:**
    - `text` The specified text you wish to enter on the screen
    - `x` The X coordinate for the UI Object (0-320)
    - `y` The Y coordinate for the UI Object (0-240)
- **Returns:** Whether success.

bool clearCustomText()

- **Description:** Clear and delete all custom UI texts from the screen.
- **Returns:** Whether success.

bool isPro()

- **Description:** Detect whether the HuskyLens is a Pro or Standard Model
- **Returns:** True is Pro Model, False if Standard

bool checkFirmwareVersion()

- **Description:** Check if the onboard firmware is out of date. If it is an old firmware, there will be a UI message that pops up on the screen
- **Returns:** Whether success.