MSP432 Pin Configuration

Left Wheel Encoder

- Pin 3.3
- GPIO

Right Wheel Encoder

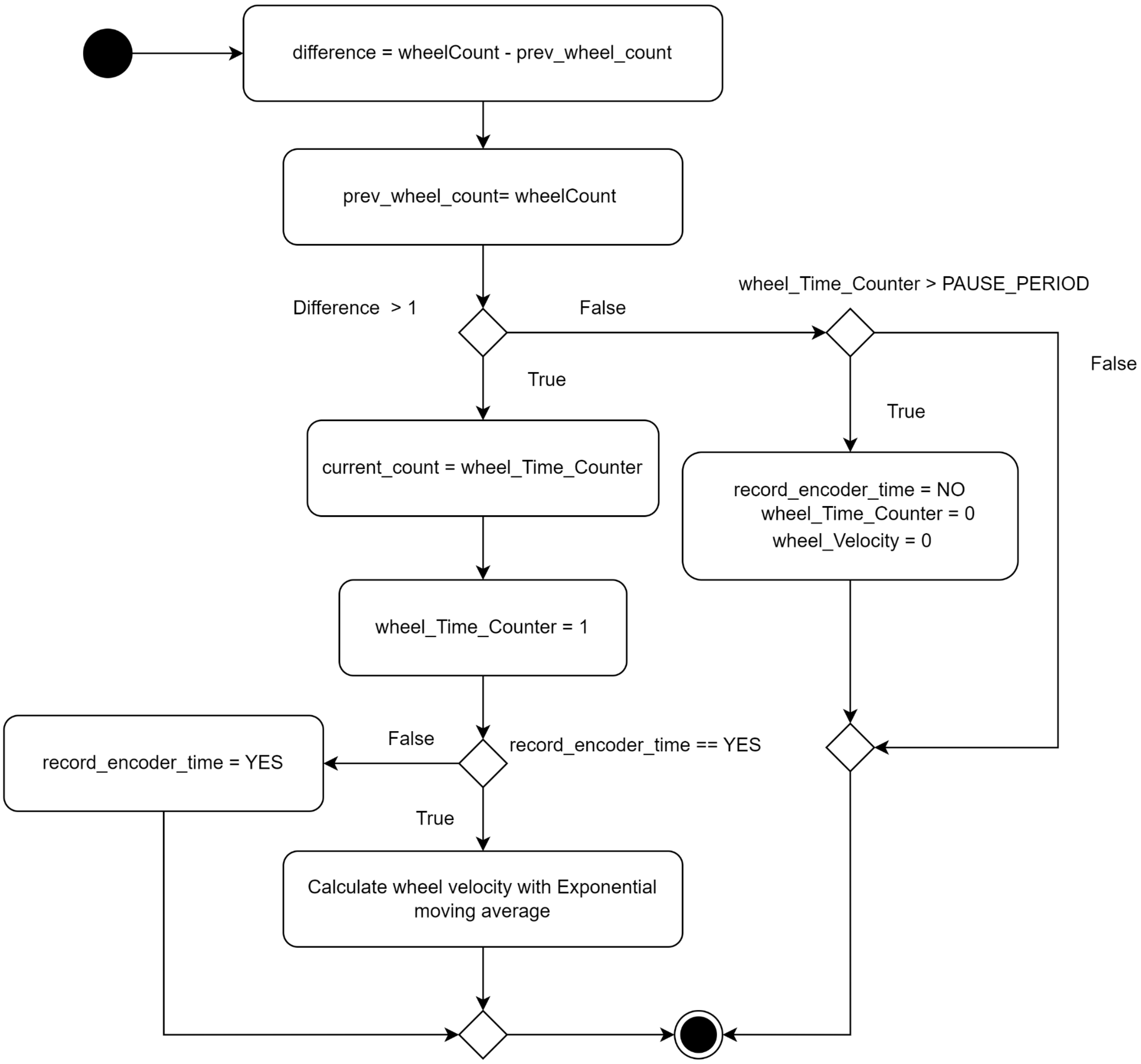
- Pin 3.2
- GPIO

Line sensor ,

- Pin 5.5
- ADC14

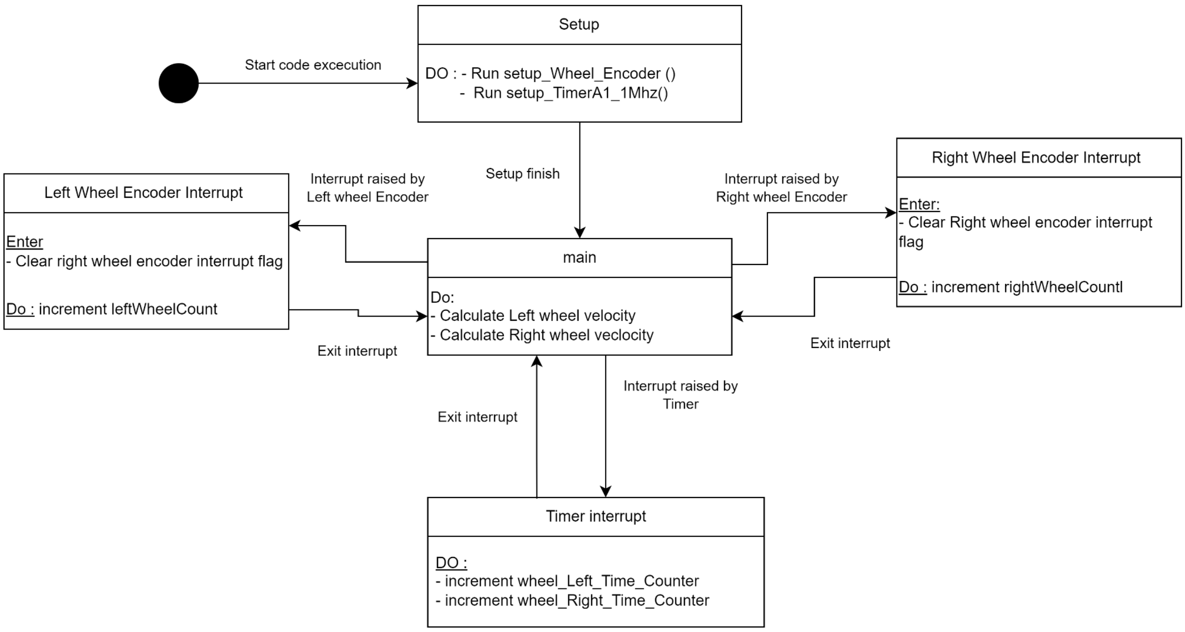
Wheel Velocity algorithm

Wheel velocity flow diagram



difference = wheelCount - prev_wheel_count
prev_wheel_count= wheelCount
Difference > 1
False
True
wheel_Time_Counter > PAUSE_PERIOD
False
True
current_count = wheel_Time_Counter
record_encoder_time = NO
wheel_Time_Counter = 0
wheel_Velocity = 0
wheel_Time_Counter = 1
False
record_encoder_time == YES
record_encoder_time = YES
True
Calculate wheel velocity with Exponential moving average

Wheel velocity of car State Diagram


Setup
DO : - Run setup_Wheel_Encoder ()
- Run setup_TimerA1_1Mhz()
Start code excecution
Setup finish
Left Wheel Encoder Interrupt
Enter
- Clear right wheel encoder interrupt flag
Do : increment leftWheelCount
Interrupt raised by Left wheel Encoder
Exit interrupt
main
Do:
- Calculate Left wheel velocity
- Calculate Right wheel veclocity
Interrupt raised by Right wheel Encoder
Right Wheel Encoder Interrupt
Enter:
- Clear Right wheel encoder interrupt flag
Do : increment rightWheelCountl
Exit interrupt
Exit interrupt
Interrupt raised by Timer
Timer interrupt
DO :
- increment wheel_Left_Time_Counter
- increment wheel_Right_Time_Counter

Barcode scanning algorithm 1

How it works:

When the line sensor starts scanning the very first white surface in the barcode paper, it will start to append 1s and 0s into a variable called “scannedOutput” based on the converted ADC value read from the barcode paper.

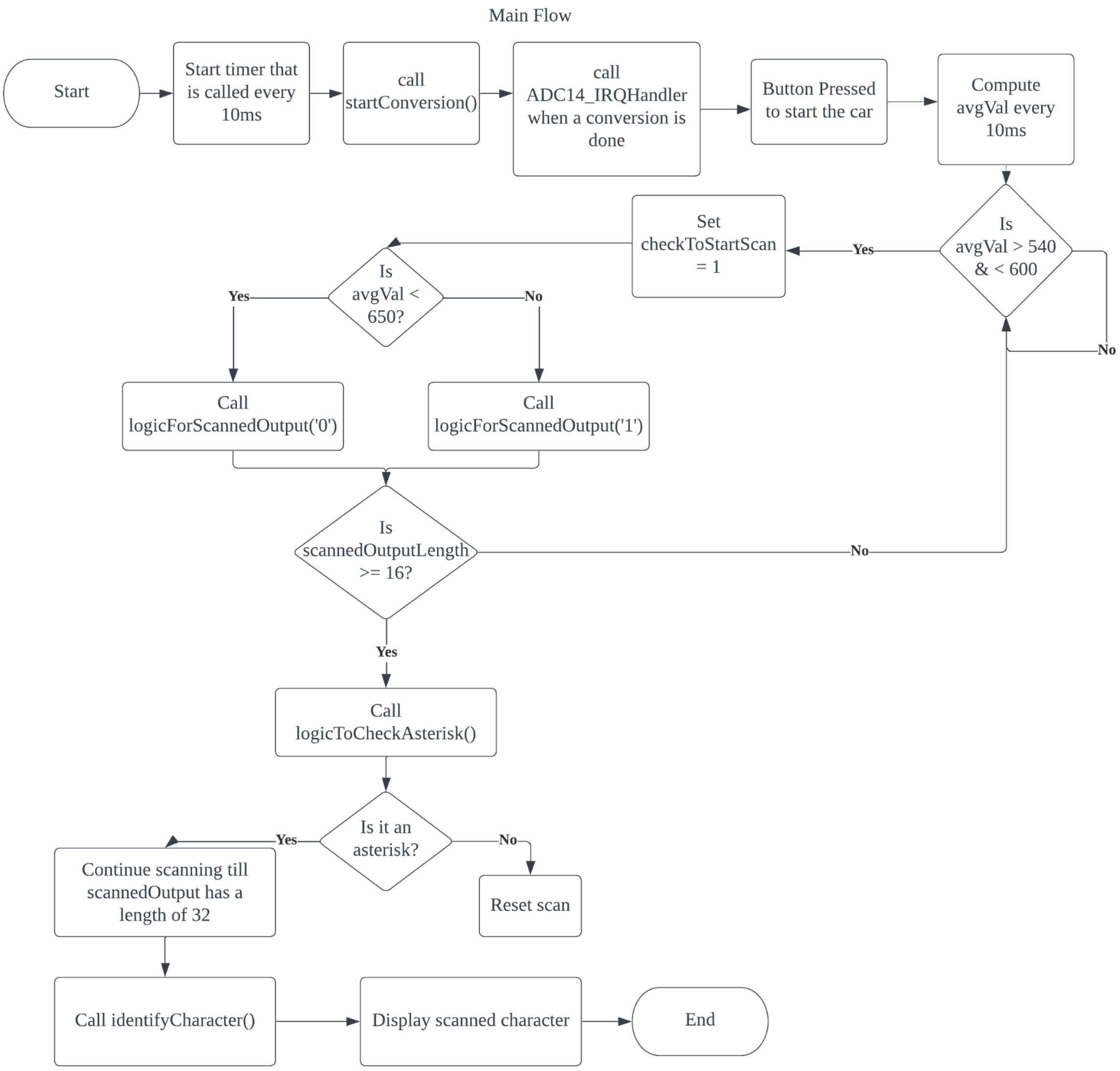
Once “scannedOutput” reaches a length of 16, it will check if it is an asterisk character. If it is an asterisk, it will continue to append 1s and 0s into “scannedOutput”, if not, “scannedOutput” will be reset. The match between “scannedOutput” and the asterisk binary value has to be a match of 75%, which we will treat as a valid barcode.

Once there is a match of 75%, we will continue reading the barcode till “scannedOutput” reaches a length of 32. It will then look for the character which has the highest match in an array called “characters”. Where this “characters” array stores all 44 characters.

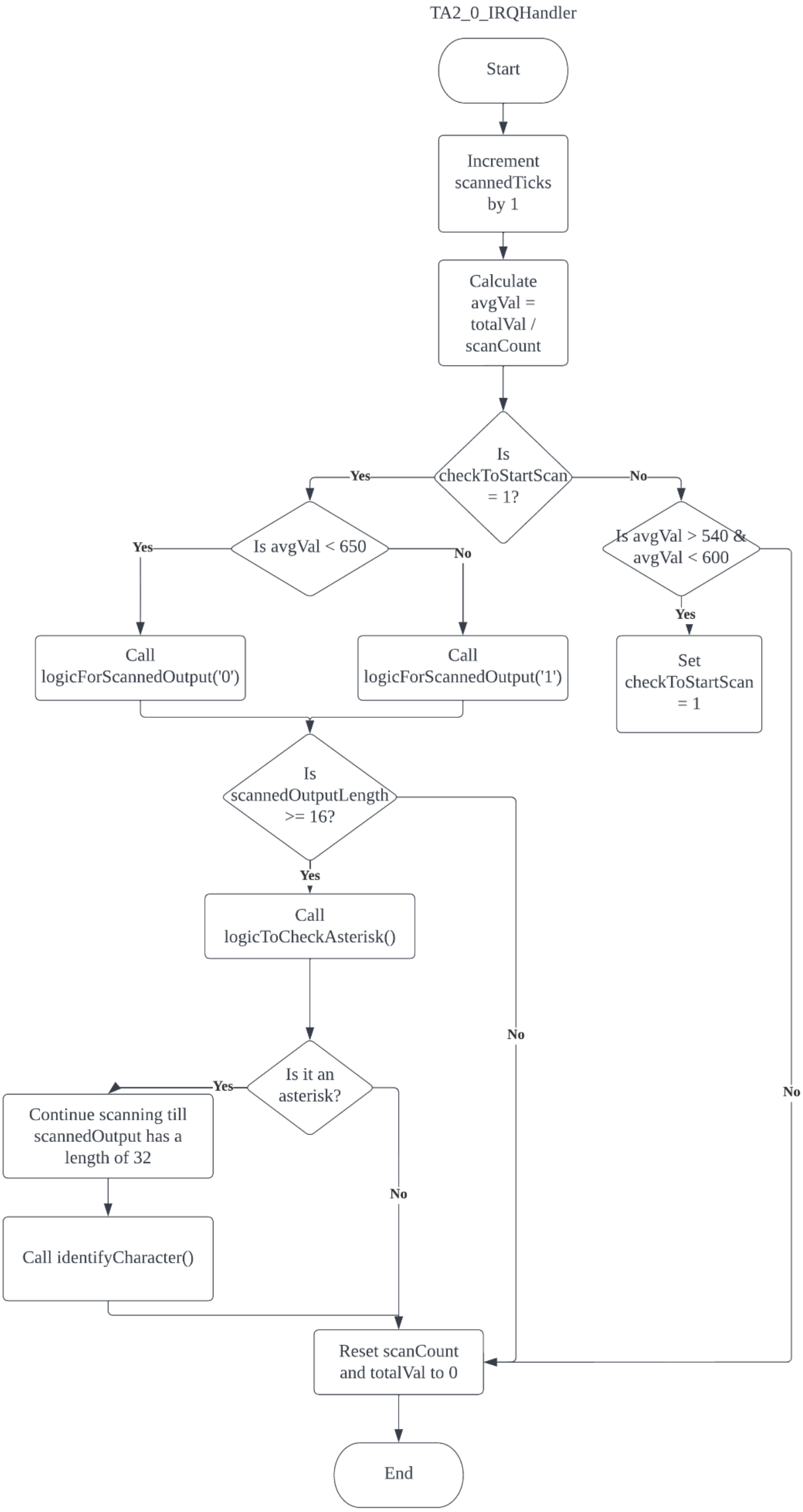
After displaying the character, we will reset “scannedOutput” and wait for another barcode to be scanned.

Flowchart:

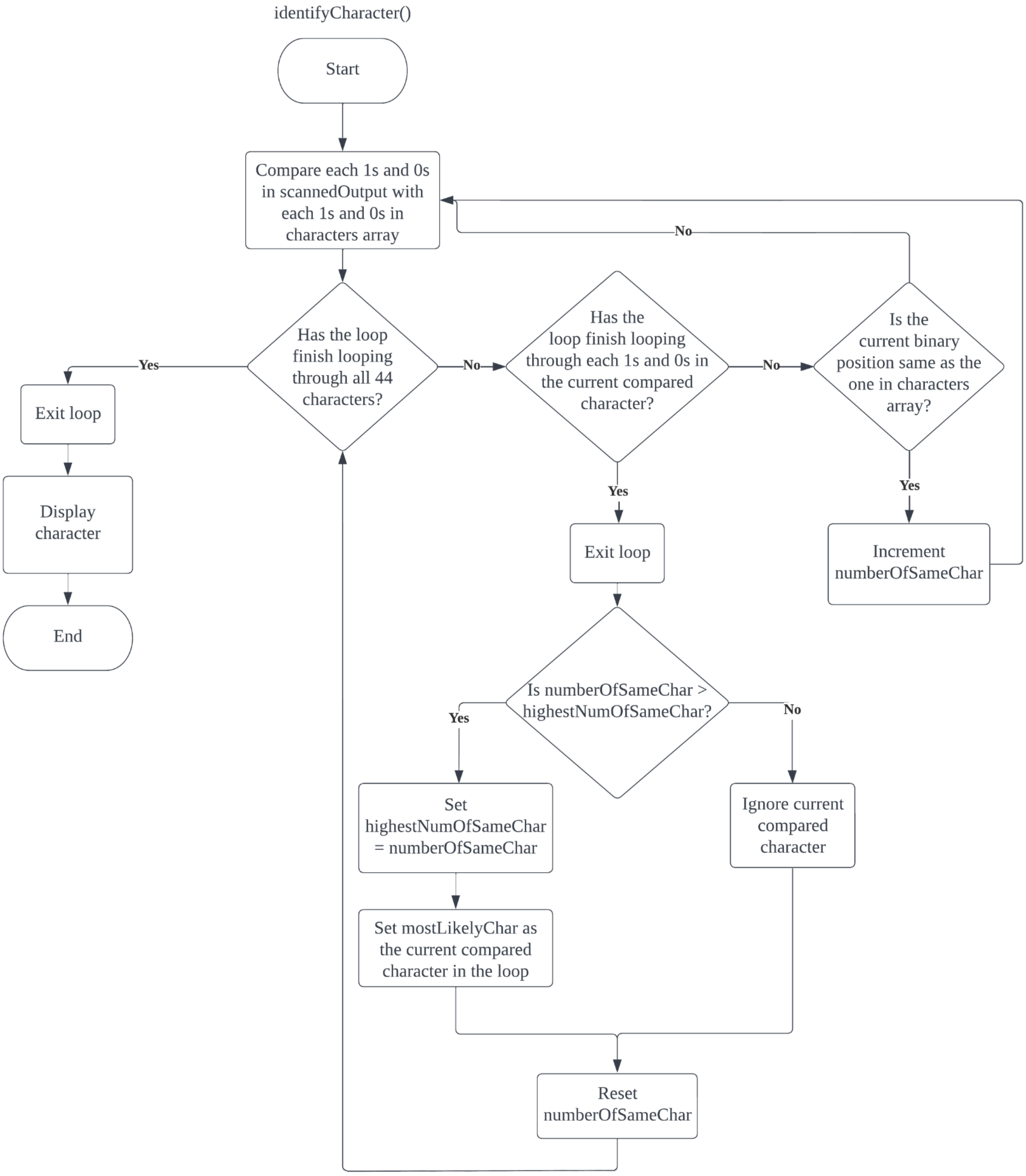
Main Flow


Main Flow
Start
Start timer that is called every 10ms
call startConversion()
call ADC14_IRQHandler when a conversion is done
Button Pressed to start the car
Compute avgVal every 10ms
Is avgVal > 540 & < 600
Yes
No
Set checkToStartScan = 1
Is avgVal < 650?
Yes
No
Call logicForScannedOutput('0')
Call logicForScannedOutput('1')
Is scannedOutputLength >= 16?
No
Yes
Call logicToCheckAsterisk()
Is it an asterisk?
Yes
No
Continue scanning till scannedOutput has a length of 32
Reset scan
Call identifyCharacter()
Display scanned character
End

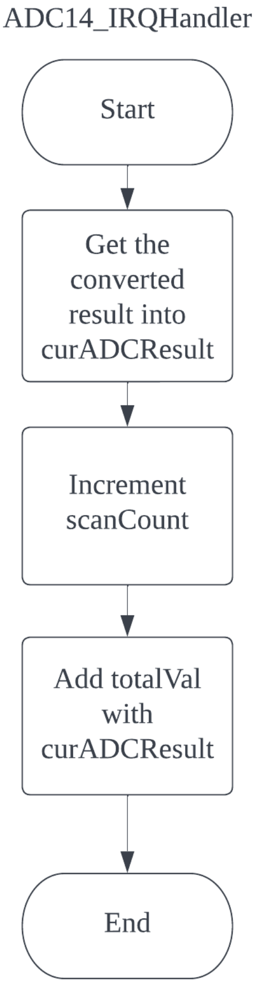
TA2_0_IRQHandler (Timer function)


TA2_0_IRQHandler
Start
Increment scannedTicks by 1
Calculate avgVal = totalVal / scanCount
Is checkToStartScan = 1?
Yes
No
Is avgVal < 650
Yes
No
Is avgVal > 540 & avgVal < 600
Yes
No
Call logicForScannedOutput('0')
Call logicForScannedOutput('1')
Set checkToStartScan = 1
Is scannedOutputLength >= 16?
Yes
No
Call logicToCheckAsterisk()
Is it an asterisk?
Yes
No
Continue scanning till scannedOutput has a length of 32
Call identifyCharacter()
Reset scanCount and totalVal to 0
End

identifyCharacter

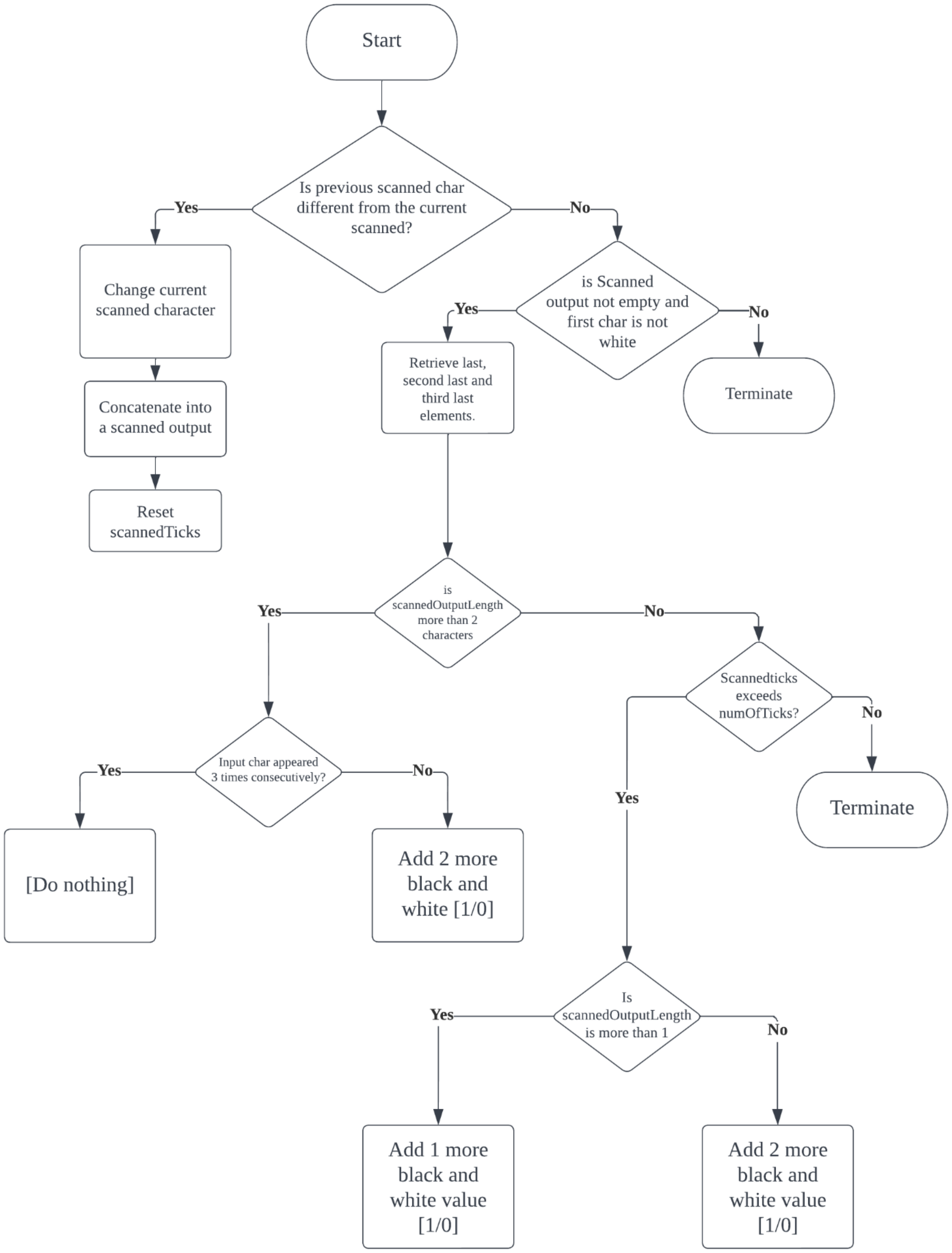

identifyCharacter()
Start
Compare each 1s and 0s in scannedOutput with each 1s and 0s in characters array
Has the loop finish looping through all 44 characters?
Yes
Exit loop
Display character
End
No
Has the loop finish looping through each 1s and 0s in the current compared character?
No
Is the current binary position same as the one in characters array?
No
Yes
Increment numberOfSameChar
Yes
Exit loop
Is numberOfSameChar > highestNumOfSameChar?
Yes
Set highestNumOfSameChar = numberOfSameChar
Set mostLikelyChar as the current compared character in the loop
No
Ignore current compared character
Reset numberOfSameChar

ADC14_IRQHandler

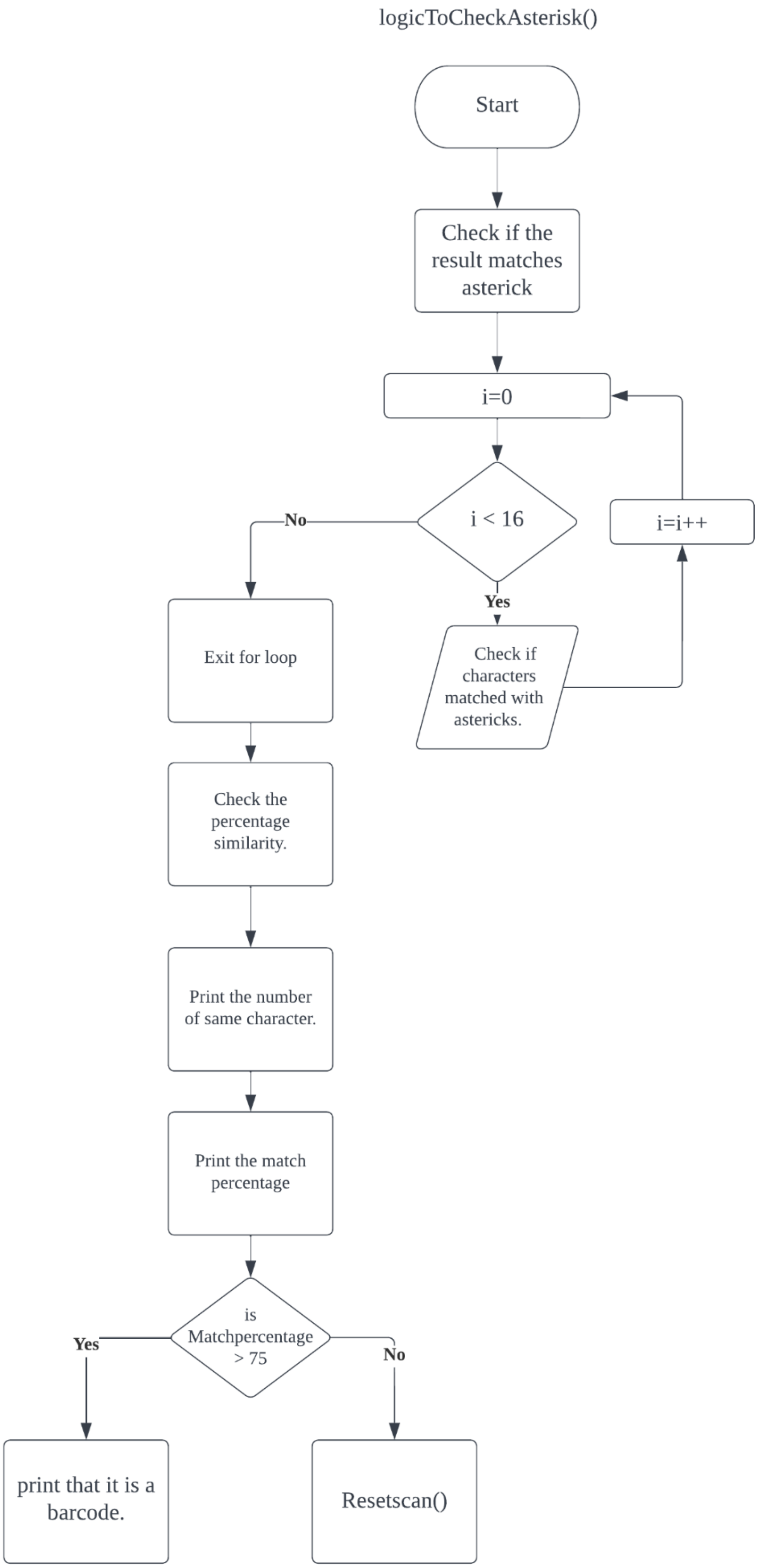

ADC14_IRQHandler
Start
Get the converted result into curADCResult
Increment scanCount
Add totalVal with curADCResult
End

logicForScannedOutput

logicForScannedOutput(*char* inputChar, *int* scannedOutputLength)


Start
Is previous scanned char different from the current scanned?
Yes
No
Change current scanned character
Concatenate into a scanned output
Reset scannedTicks
is Scanned output not empty and first char is not white
Yes
No
Retrieve last, second last and third last elements.
Terminate
is scannedOutputLength more than 2 characters
Yes
No
Scannedticks exceeds numOfTicks?
No
Terminate
Yes
Input char appeared 3 times consecutively?
Yes
No
[Do nothing]
Add 2 more black and white [1/0]
Is scannedOutputLength is more than 1
Yes
No
Add 1 more black and white value [1/0]
Add 2 more black and white value [1/0]

logicToCheckAsterisk


logicToCheckAsterisk()
Start
Check if the result matches asterick
i=0
i < 16
i=i++
No
Yes
Exit for loop
Check if characters matched with astericks.
Check the percentage similarity.
Print the number of same character.
Print the match percentage
is Matchpercentage > 75
Yes
No
print that it is a barcode.
Resetscan()

Main issues:

- The reading of the character from the barcode is inconsistent.
- It assumes that the car will be moving at a constant speed.
- Works best if the car PWM duty cycle is 2500.

Barcode scanning algorithm 2

This algorithm uses the relative width pattern of the barcode to match CODE39 alphabets

1. The relative bar width of each black or white bar is derived and recorded by measuring the number of sequential sampled values is above or below a certain value threshold (e.g. Black > 750 , White <=750).
   1.1. These values are stored in an array of 9 integer values.
   1.2. Integers at odd indexes are

2. From the collected bar width , the threshold value to differentiate wide and slim bars is calculated by
   2.1. From an array of 9 bar width value , get the minimum and maximum value
   2.2. Formula to calculate bar width threshold, threshold = (MAX + MIN)/ 2 + MIN

3. Using the calculated threshold value , we can get the pattern of the array bar code width values as a binary character array (e.g. “100010001”) of wide and short bars.
4. We then match the binary character pattern array to bar code 39 patterns of “A” to “Z” characters

Issues faced with barcode reading implementation

1. Sampled values from line sensors are very inconsistent, values for white paper can randomly change between 600 to 1000 and black bars between 900 to 1400 . This makes testing very difficult and time consuming due to constant reconfiguration of colour value threshold.
2. Random acceleration and jerking motion of robot car causes misreading of the line sensors sampled values. Implementing a bar code reader requires consistent motion

to be able to obtain a quality reading, the inherent inconsistency of the robot car movements hinders it.

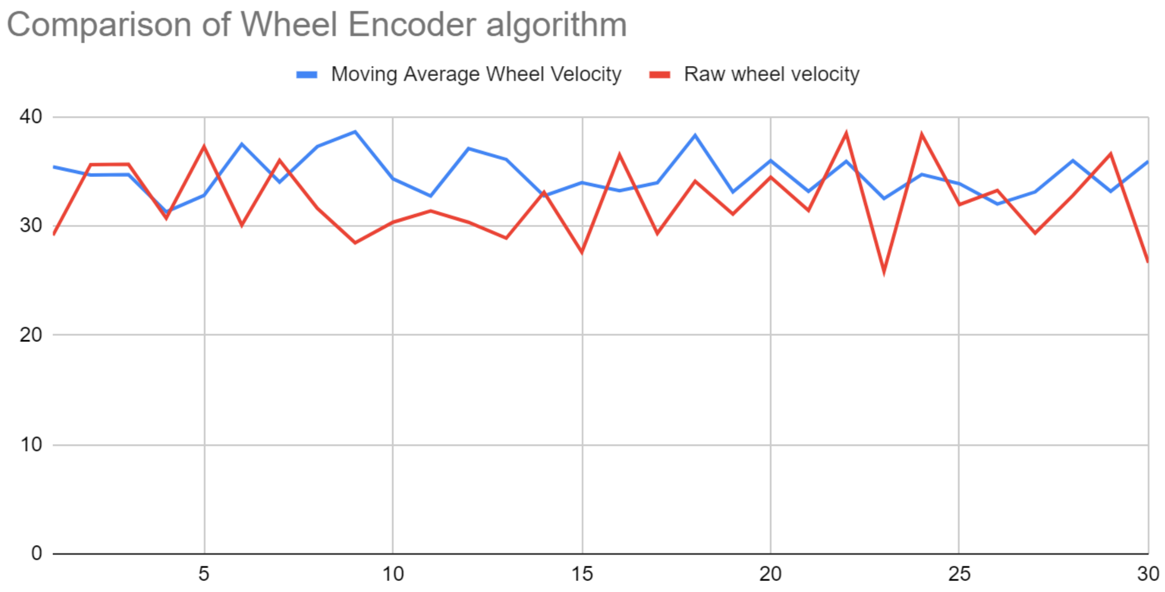
Sensor Algorithms

Calculating wheel velocity with Wheel Encoders

Real time Wheel velocity is derived from the time taken for the disc to rotate one complete notch.

- Wheel velocity is calculated on every interrupt raised by the wheel encoder
- Wheel velocity = (wheel circumference of one notch) / (time taken for one notch)

Moving average was applied to wheel velocity to smoothen spikes in value.



Sampling from Line sensor

Between 10 millisecond intervals, the sampled line sensor values are summed up and the number of times it has been sampled during the interval is counted. Every 10 milliseconds, a timer interrupt is raised to get the average value of line sensor values within the 10 milliseconds interval .

Pros:

- Ability to scan Big, Medium, Small barcode sizes

Cons:

- May not be accurate all the time.
- Assumes that the car is moving at a constant speed