**Main Program Flow Chart**

X: Along the guitar neck

Y: From chord to chord

Z: Commit or release

: Function call

FILES INCLUDED:

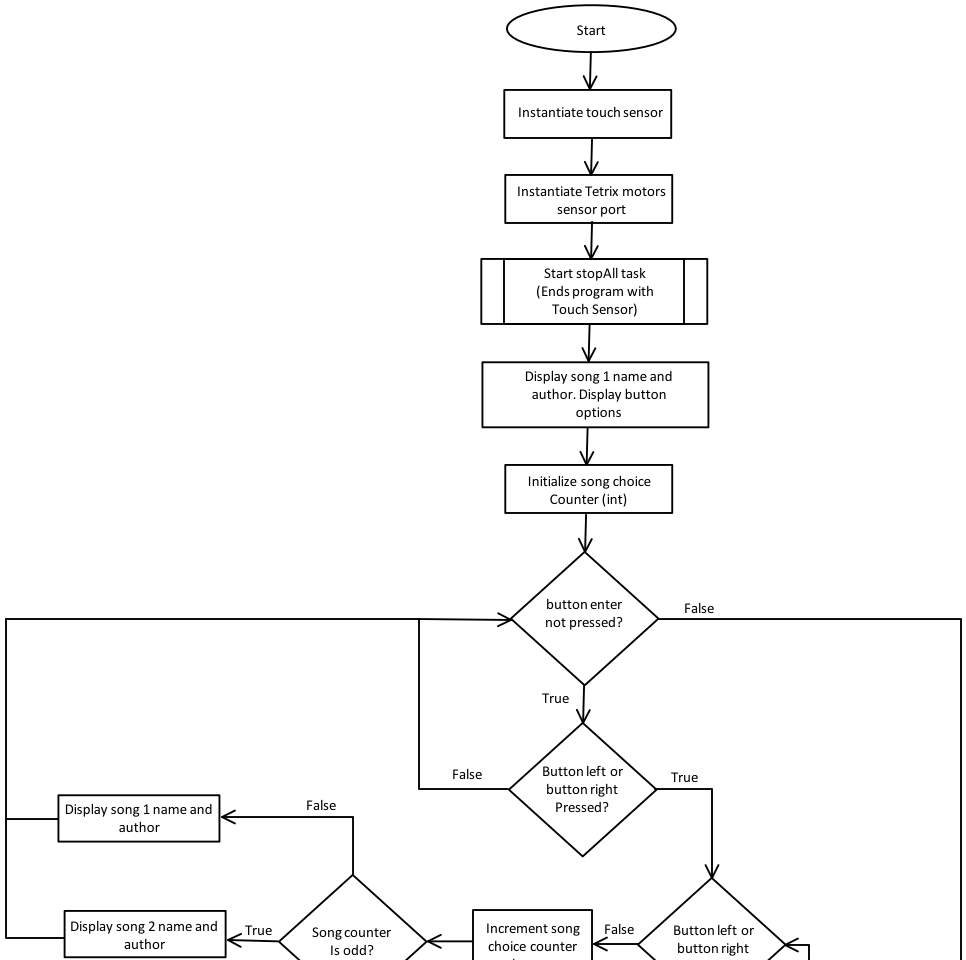
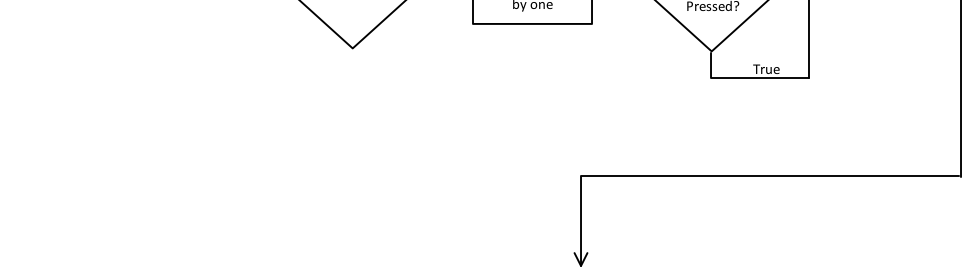
* EV3\_FileIO.c
* EV3Servo-lib-UW.c

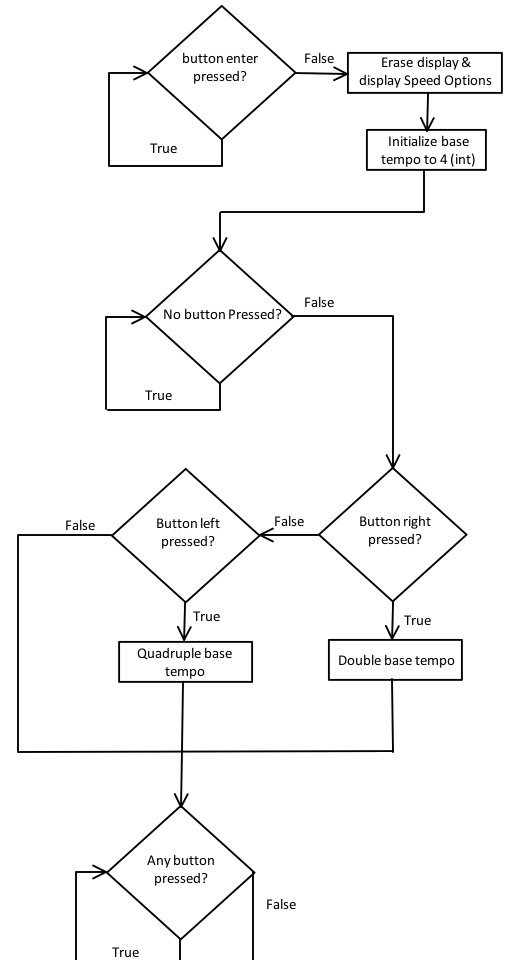
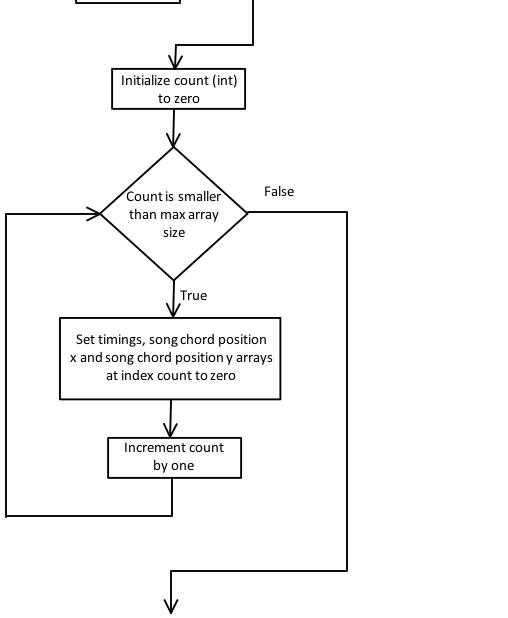
GLOBAL CONSTANTS:

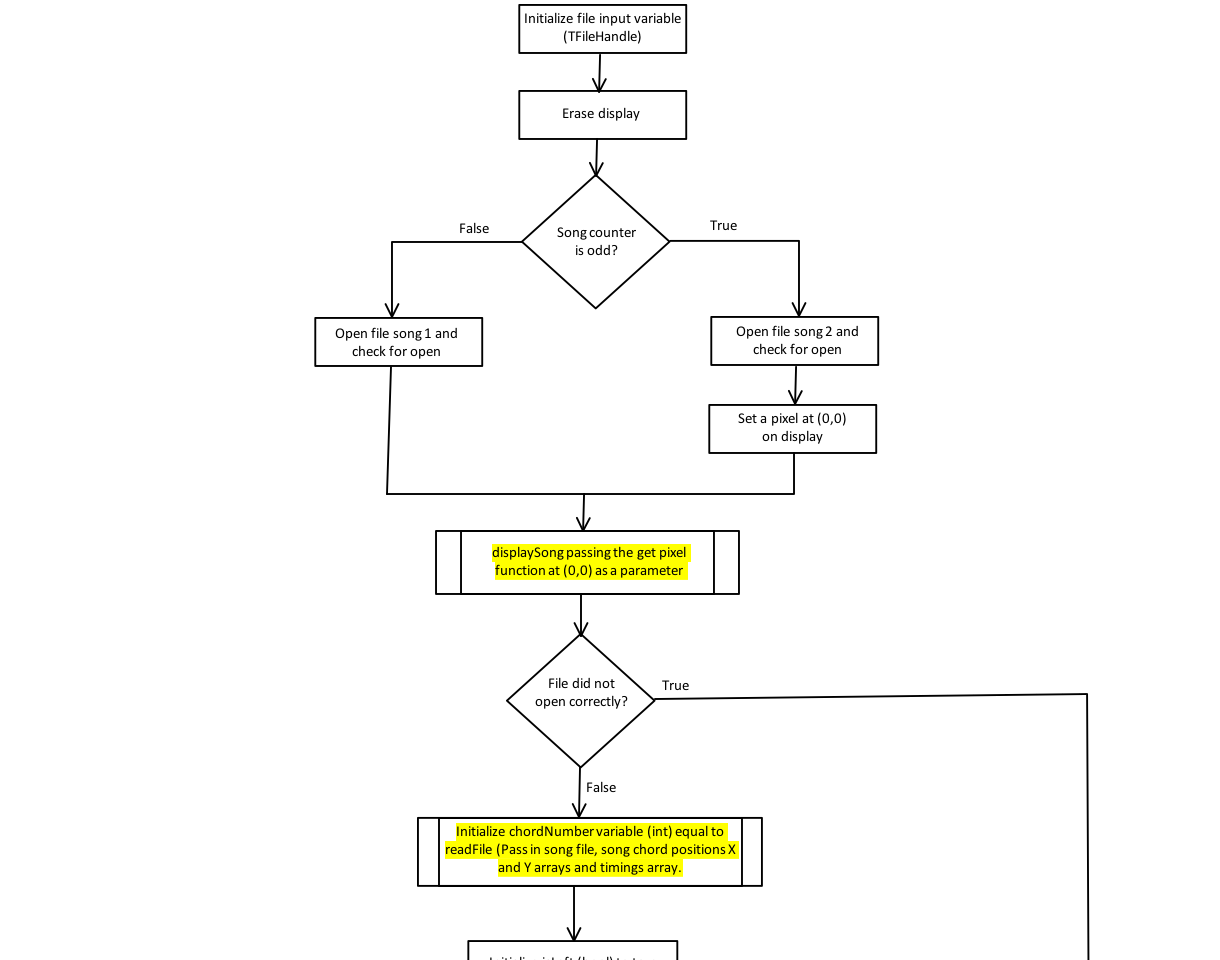
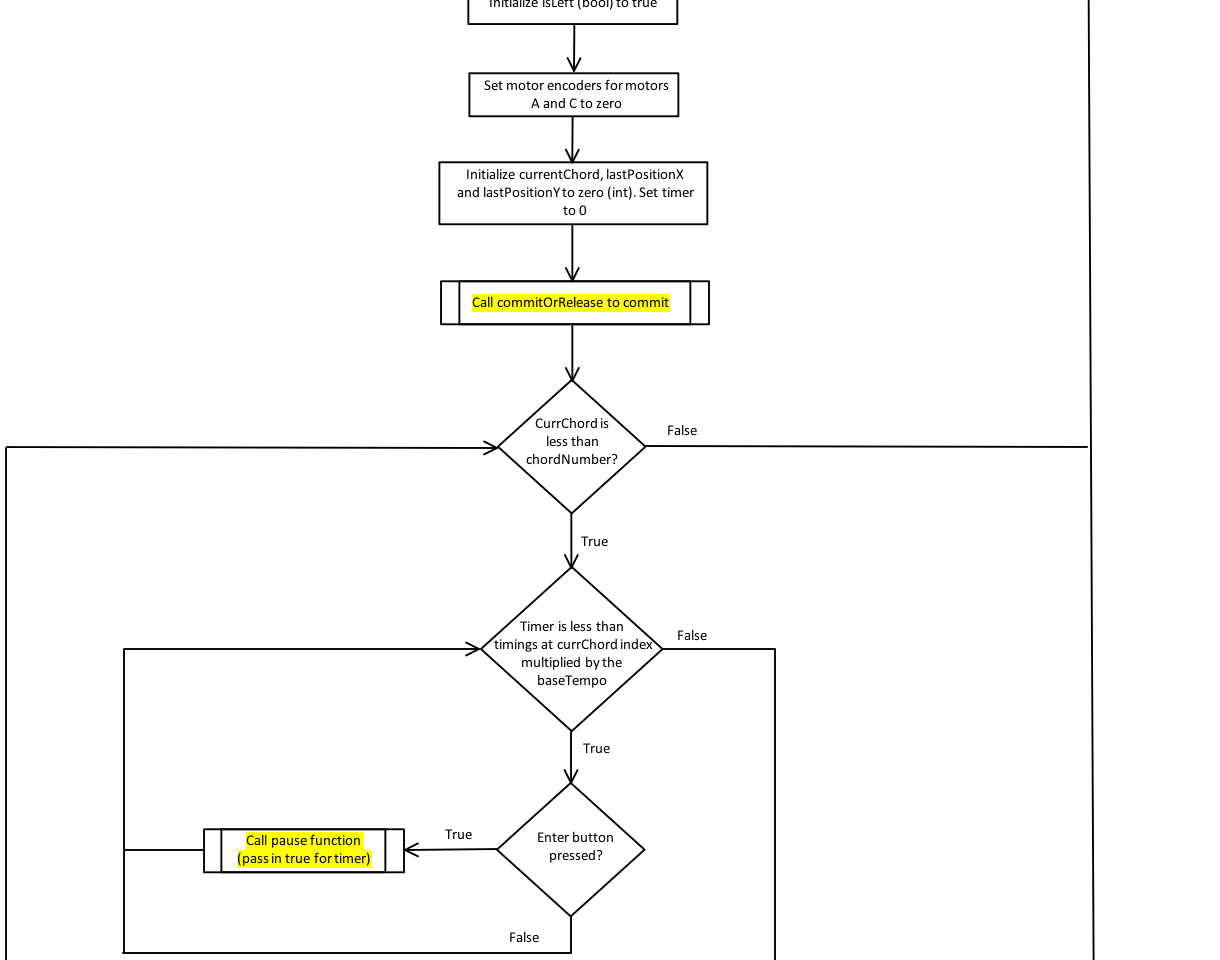
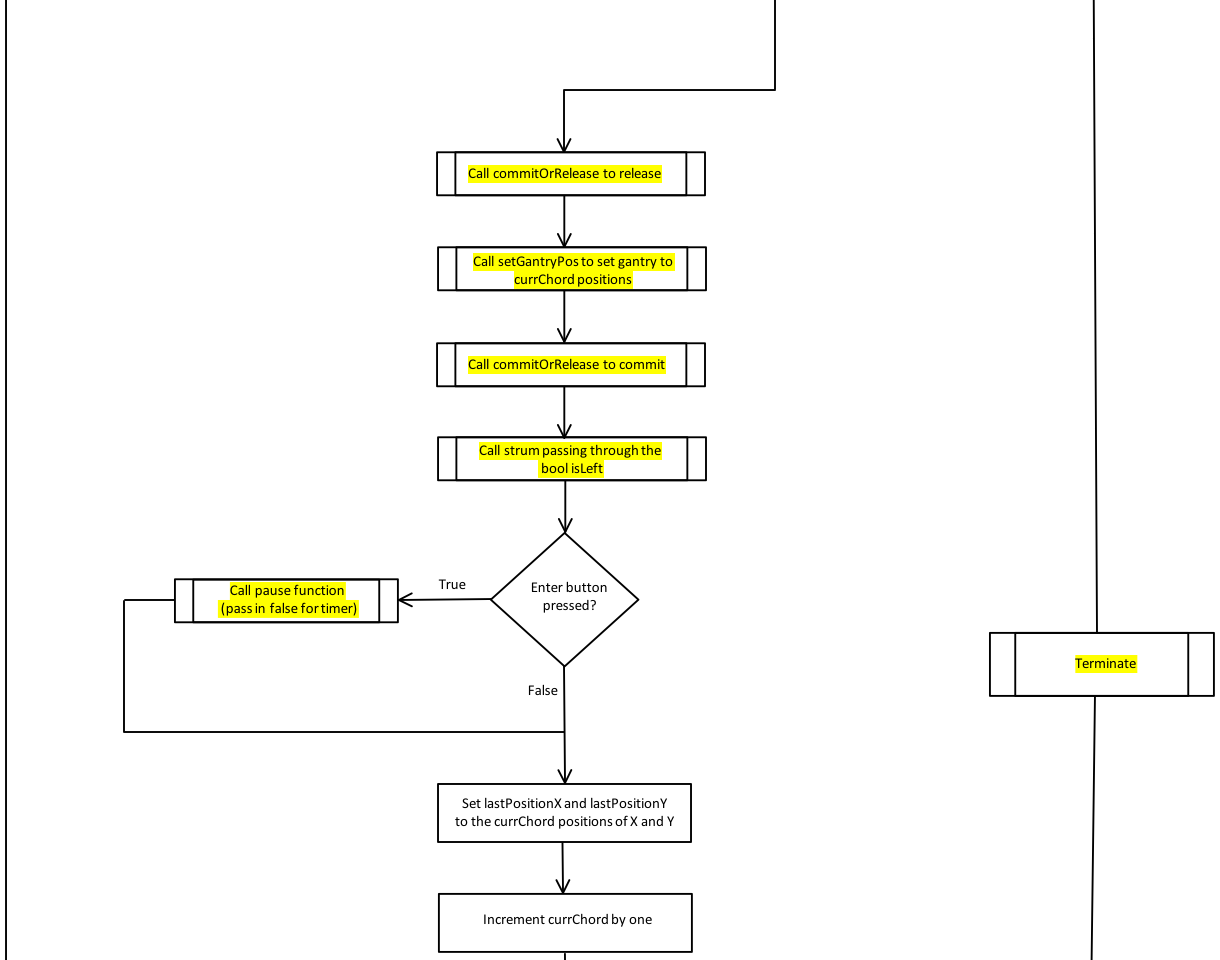
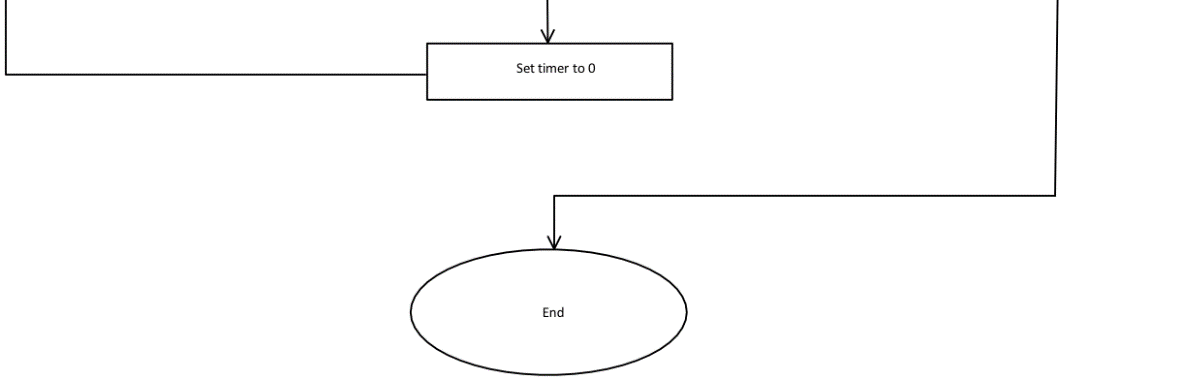
* Max song array size
* X encoder value array
* Y position value array
* Power for commit and release
* Power for X displacement motors
* First strum position
* Second strum position
* Commit encoder limit

GLOBAL VARIABLE ARRAYS (MADE GLOBAL BECAUSE OF MEMORY ISSUES):

* Song chord X encoder values
* Song chord Y positions
* Timings (time per chord)

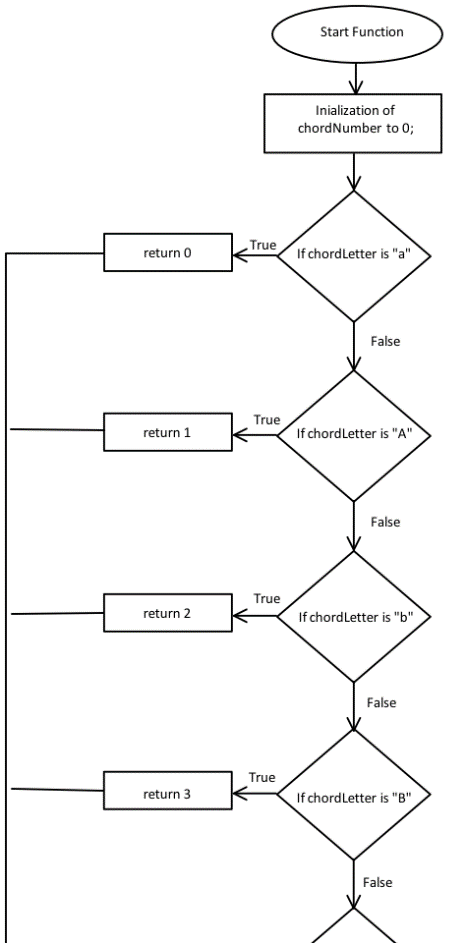
**chordNum Flow Chart**

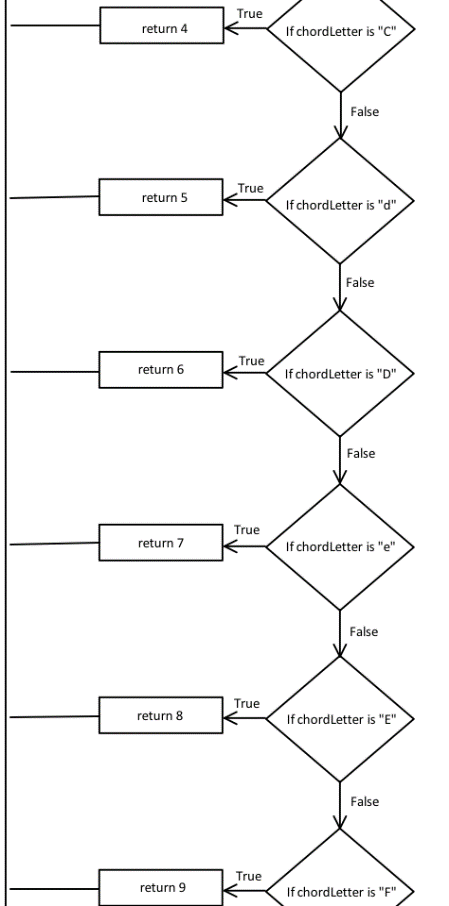
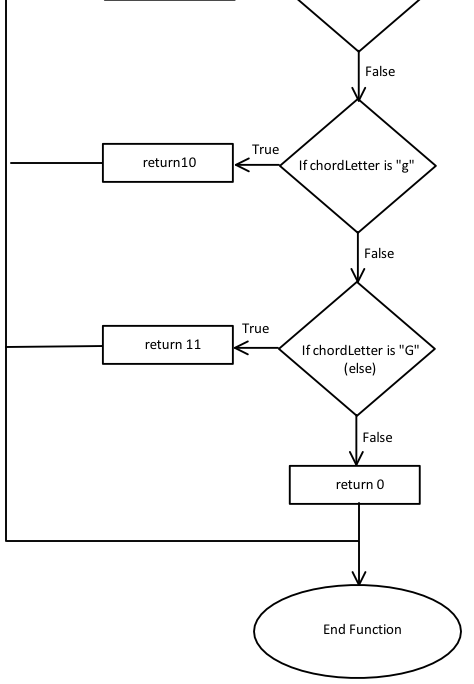
PARAMETER: char chordLetter (pass by value)

RETURN TYPE: Integer

TEST CASES:

* MAKE A MAIN FUNCTION TO PRINT OUT EVERY POSSIBLE CONDITION
* MAKE A LIST OF PREDICTED OUTPUT
* COMPARE ACTUAL OUTPUT WITH PREDICTED OUTPUT



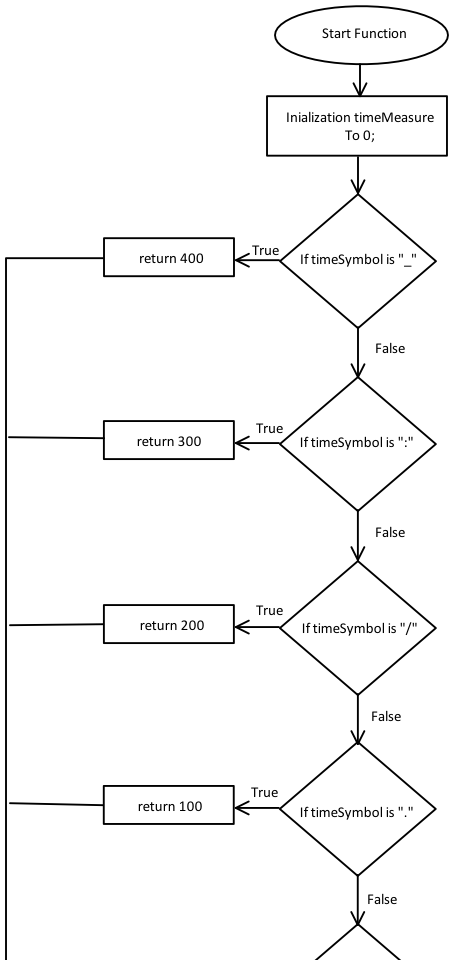
**getTimeValue Flow Chart**

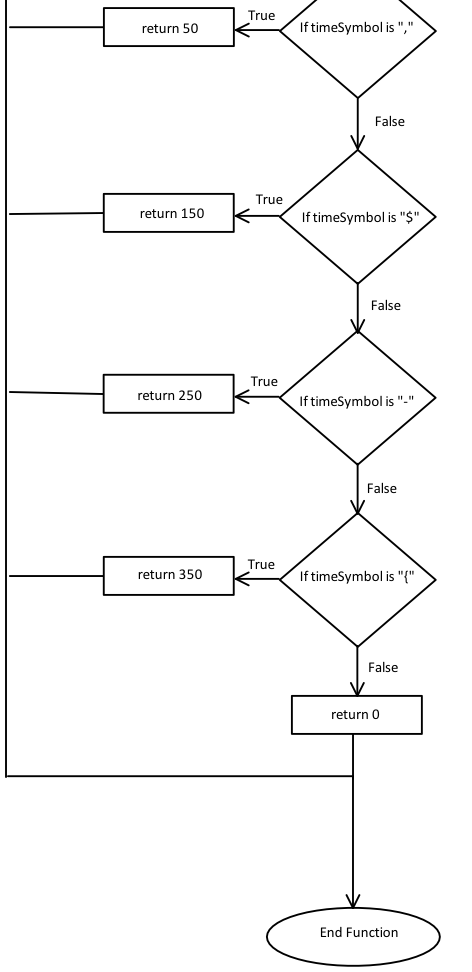
PARAMETERS: char timeSymbol (pass by value)

RETURN TYPE: Integer

TEST CASES:

* MAKE A MAIN FUNCTION TO PRINT OUT EVERY POSSIBLE CONDITION
* MAKE A LIST OF PREDICTED OUTPUT
* COMPARE ACTUAL OUTPUT WITH PREDICTED OUTPUT





**readFile Flow Chart**

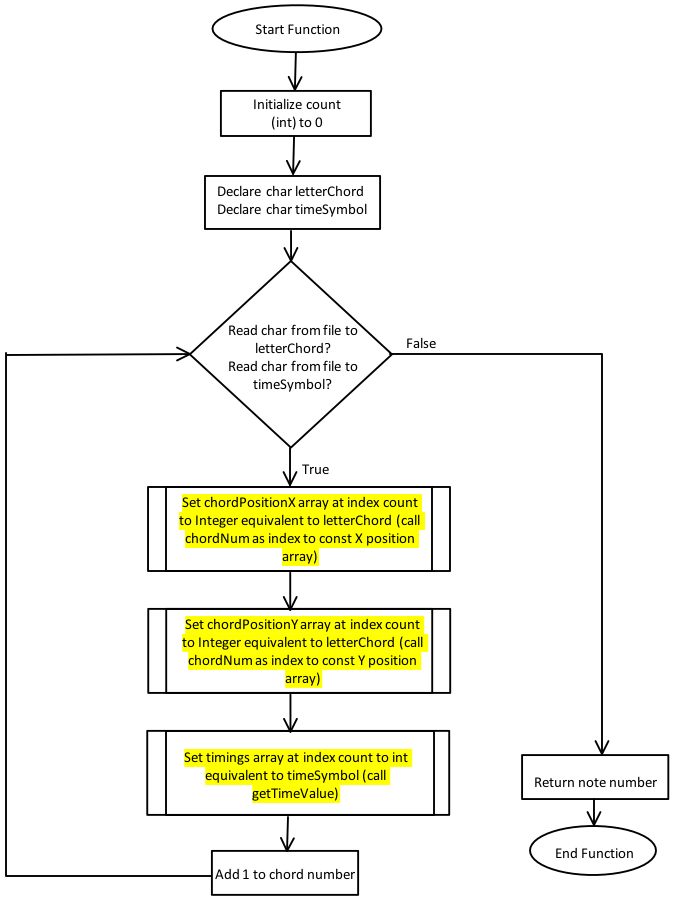
PARAMETERS:

* TFileHandle song file (pass by reference &)
* int chordPositionX array (pass by reference[])
* Int chordPositionY array (pass by reference[])
* Int timings array (pass by reference[])

RETURN TYPE: Integer (number of notes to be played)

TEST CASES:

* READ IN FILE TO TWO ARRAYS
* CREATE EXPECTED OUTPUT
* PRINT OUT ARRAYS
* COMPARE ACTUAL OUTPUT WITH EXPECTED OUTPUT



**setGantryPos Flow Chart**

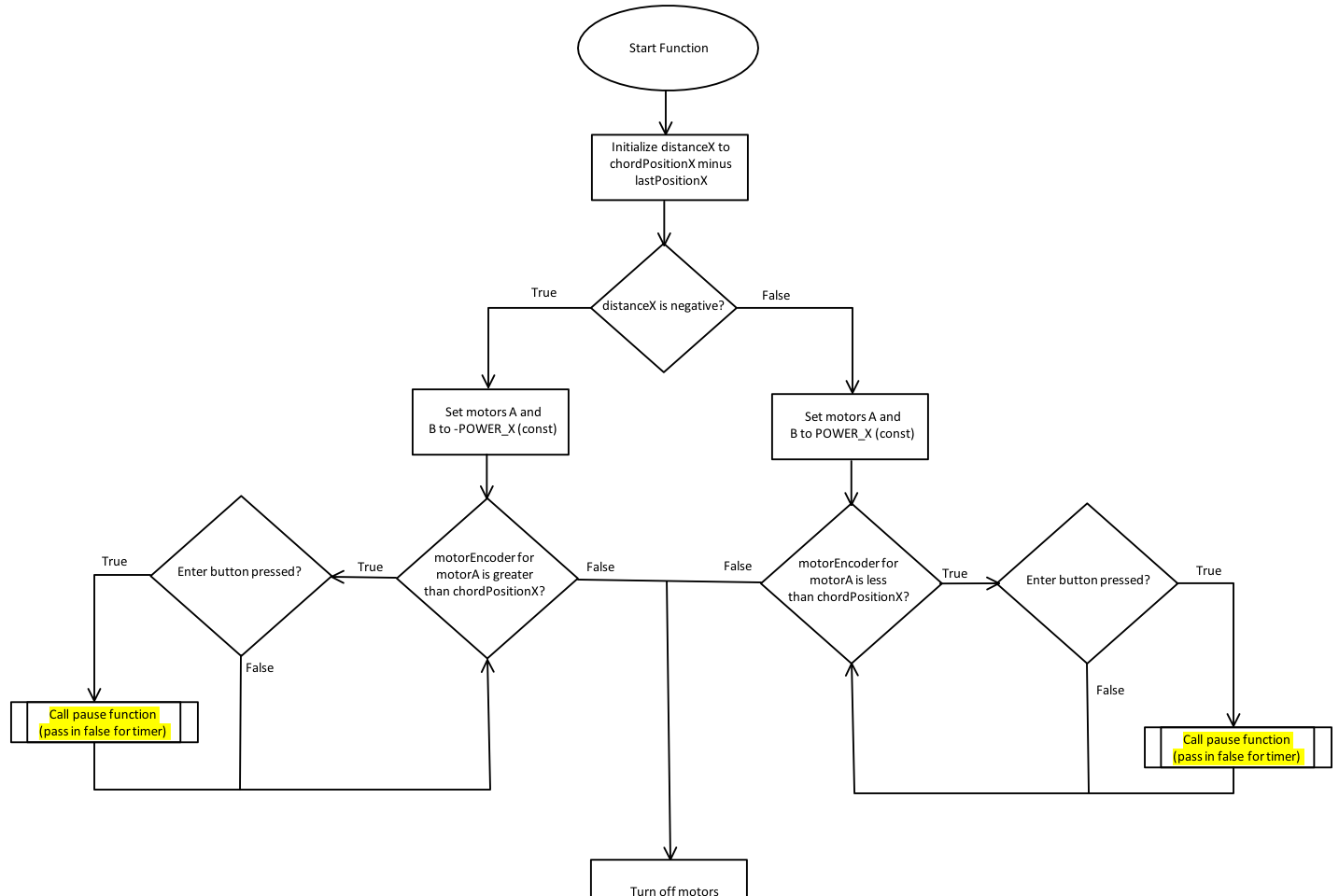
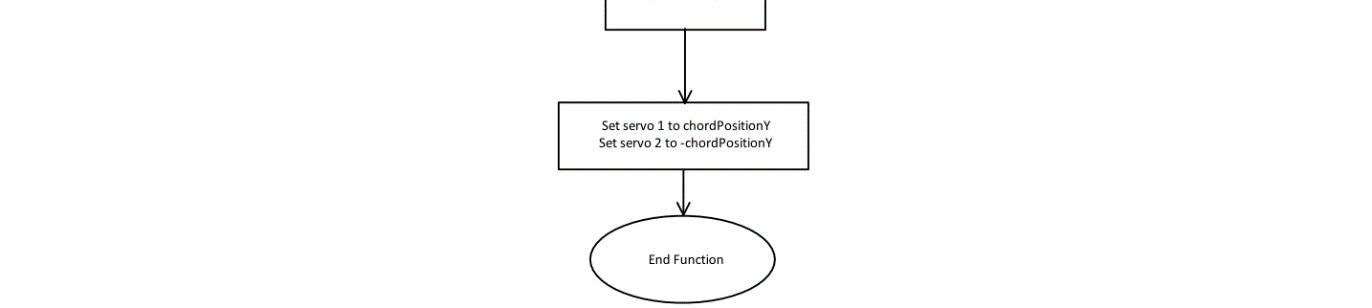
PARAMETERS:

* Int chordPositionX (pass by value)
* Int chordPositionY (pass by value)
* lastPositionX (pass by value)

RETURN TYPE: void

TEST CASES:

* GIVE CHORDS AT DIFFERENT ENCODER POSITIONS FROM MULTIPLE ENCODER POSITIONS AND CHECK TO SEE IF THEY GO TO THE RIGHT POSITION. MUST BE CHECKED FOR A LONG PERIOD OF TIME TO ESURE THAT IT IS NOT OFFSET DRASTICALLY OVER TIME.

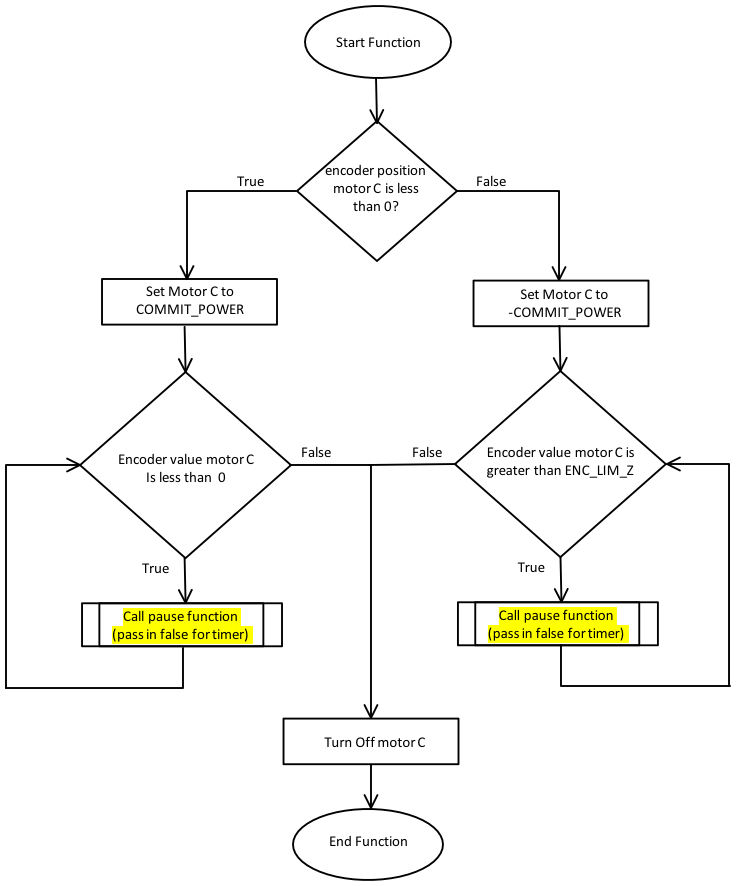
**commitOrRelease Flow Chart**

PARAMETERS: None

RETURN TYPE: void

TEST CASES:

* TEST FUNCTION BY RUNNING THE CODE OF THE FUNCTION THROUGH A TASK MAIN
* RAISE AND LOWER THE Z AXIS MULTIPLE CONCURRENT TIMES TO TEST RESPONSE AND DURABILITY OF HARDWARE.



**strum Flow Chart**

PARAMETERS:

* Bool isLeft (pass by reference &)

RETURN TYPE: void

TEST CASES:

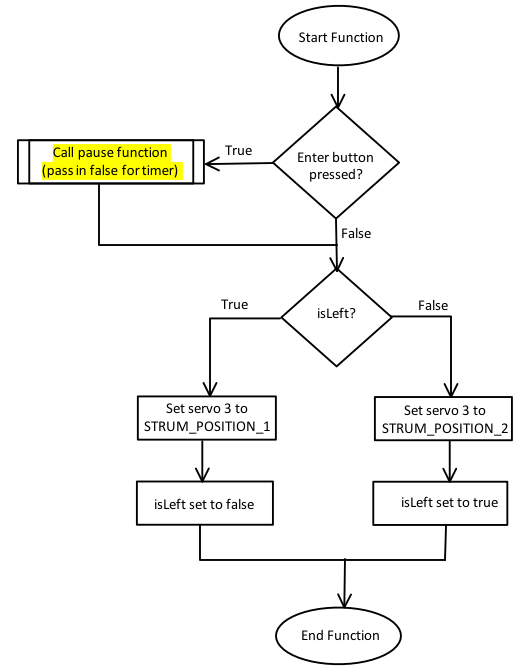
* PASS VARIOUS TIMES TO THE FUNCTION

AND OBSERVE MOVEMENT OF THE ARM AND

ACCURACY OF THE STRUM IN TERMS OF

VARIOUS TEMPOS CONSISTING OF LONG

WAITS, SHORT WAITS, AND NO WAITS.



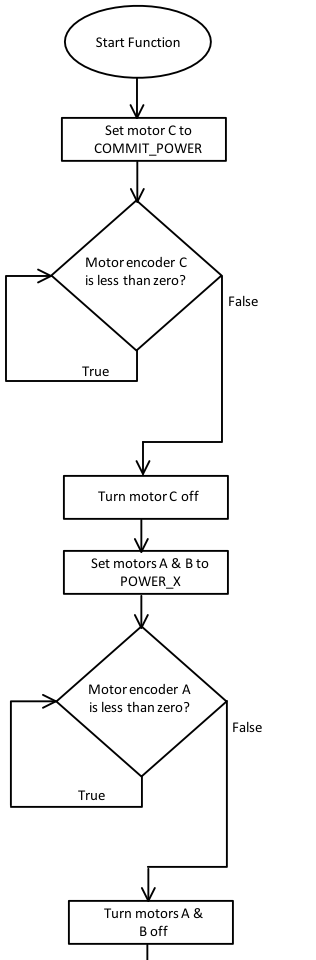
**terminate Flow Chart**

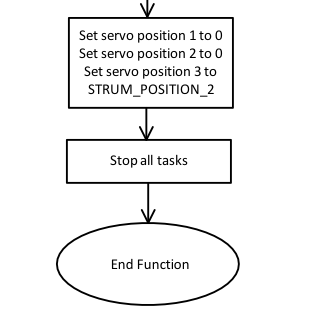
PARAMETERS: None

RETURN TYPE: void

TEST CASES:

* CREATE A MAIN THAT MOVES THE GANTRY TO A CERTAIN POSITION, COMMITS AND STRUMS
* WHEN THE ENTER BUTTON IS PUSHED, CALL TERMINATE TO RETURN THE EVERYTHING TO THE INITIAL POSITION.
* CALL IT AT DIFFERENT INSTANCES THROUGHOUT THE TEST CODE.





**displaySong Flow Chart**

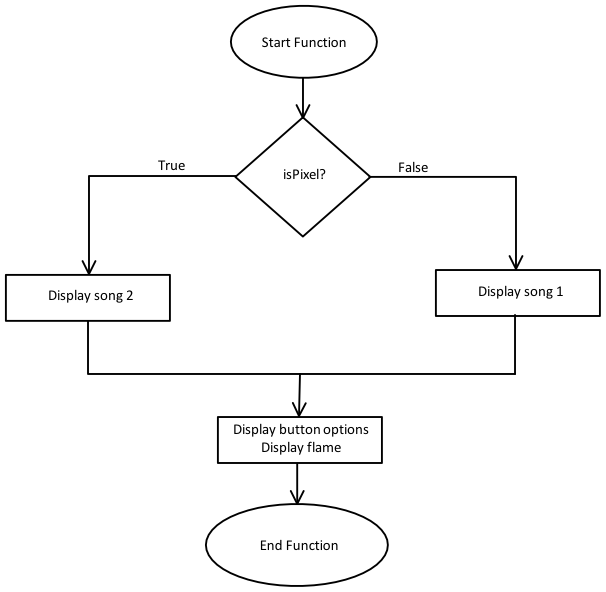
PARAMETERS:

* Bool isPixel (Pass by value)

RETURN TYPE: void

TEST CASES:

* DRIVER THAT TESTS BOTH SONGS TO BE PRINTED TO THE SCREEN



**pause Flow Chart**

PARAMETERS:

* Bool timerNeeded

RETURN TYPE: void

TEST CASES:

* TESTED WITHIN OTHER FUNCTIONS

