4.5 Introduction to the interface

The minimum requirement for the understanding of the C code included in appendix b is basic programming knowledge, conversion between binaries, hexadecimals and decimals, bolean logics in addition to the functions controlling the printer parallel port introduced in section 4.2.

4.5.a Basic interfacing functions:

Based on the analysis conducted in sections 4.3 and 4.4 two *C* codes were written to position the controlled point on the *x*-*y* plane. The function linecar (x_{final}, y_{final}) sends the point from an initial position (x_o, y_o) to a final position (x_{final}, y_{final}). The function circle (x_c, y_c, θ) moves the point on the circumference of a circle centered at (x_c, y_c) from an initial position, that is located on the circle, an angular distance θ.

A rectangular drawing area within the allowable drawing range specified in section 2.2 was selected in order to set the limits for the inputs to the functions linecar and circle. As can be shown in figure 4.10 $x_{minimum}$ = --131.15 mm, $x_{maximum}$ = -58.75, y_{minmum} = 332.79 mm and $y_{maximum}$ = 408.63 mm. This applies that the interface will not accept values for *xfinal* and *yfinal*, input to linecar or computed from the inputs to circle out of this range. Also, circle will not accept to rotate around a centre out of the range.

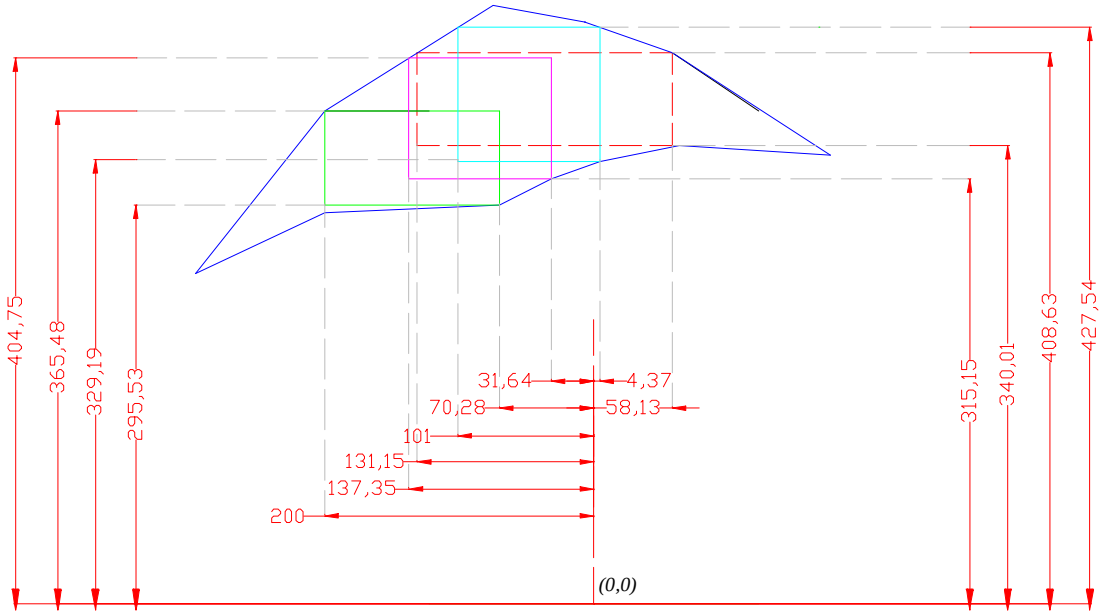

Figure 4.9 Rectangular drawing area dimensions.

Two other secondary functions were made. The function linepol (s_{total} , α) computes the final position (x_{final}, y_{final}) from the total displacement (s_{total}) and the direction of motion α, as follows

$$x_{final} = x_o + s_{total}.\ cos\ \alpha \qquad (4.43)$$

$$y_{final} = y_o + s_{total}.\ sin\ \alpha \qquad (4.44)$$

before using linecar (x_{final}, y_{final}).

The function circle2 (r, x_c, y_c, θ) will use the function linecar to place the point on a circle of radius r, centered around (x_c, y_c). linecar will take the following arguments:

$$x_{final} = x_c - r.\ cos\ \psi \qquad (4.45)$$

$$y_{final} = y_c - r.sin\ \psi \qquad (4.46)$$

$$: \psi = arctan[((yo - yc)/ (xo - xc)] \qquad (4.47)$$

(x_{final}, y_{final}) of linecar is then used as initial position (x_o, y_o) used by circle to continue the rest of the operation.

The functions penup and pendown will be used to left the drawing pen or put it down. Sending a +5volts from the 9th pin in the parallel port to the relay will allow passing a current from the +12V source to energize a solenoid pushing the pen down. Setting pin 9 voltage to zero will lead to de-energizing of the solenoid. The pen will be pulled up by a spring action. Sending logics to 9^{th} pin (Data 7) will contradict with the logic that be should be sent to the Data port in order to drive the motors every time the pulse sending logic is applied. To overcome this problem, the hexadecimal representing the logic to be sent to pin 9 should be always added to the hexadecimal sent to pins 2 and/or 4, when using the function outportb.

4.5.b Return 2 datum System:

Each basic position function requires that the coordinates of current position are know, so that the function can be executed. Usually, the functions use the end position reached in a previous step as the initial position for current one. One may ask: how can the first operation, that has no previous steps, be executed? And what to do if for some reason, the robot reached to an unknown position?

We can always have a known initial position that will be called the datum of the robot. Controlled point will return to this datum by returning back the actuators to their minimum lengths. However, how much each actuator should be shortened until reaching their minimum length will be unknown, and thus, we can't specify number of steps for each motor to move until reaching the minimum actuators lengths. Instead, we can install a limit switch that will detect the instance when the actuator reaches its minimum length, when a nut attachment presses the limit switch, sending logic one to an input bit on the parallel port (Status 5 for actuator *a* and Status 4 for actuator *b*). The computer will stop

sending pulses, and thus, driving the motors when logic one is read on the input pin. See figure 4.10.The code controlling this mechanism is called ret2datum.

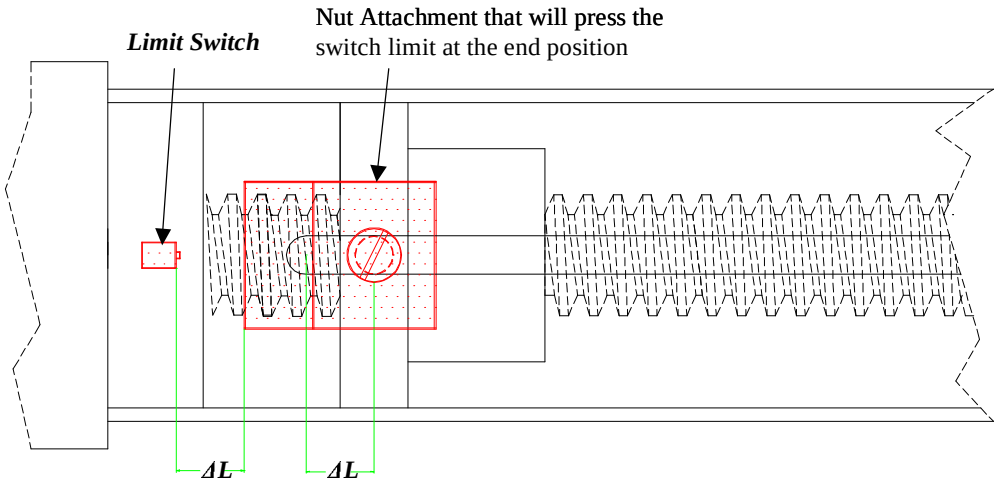

Figure 4.10: Return 2 datum mechanism.

4.5.c Letters drawing functions:

A letter drawing function is a code consisting of a combination of the basic positioning functions and the functions controlling the solenoid circuit. Pressing a button on the keyboard will call the corresponding drawing function. For example, to draw the letter S shown in figure 4.11, the following sequence of commands should be implemented:

linecar(X0,Y0) → **[** pendown() → linecar(X0+scale*7.5,Y0) → penup() →

linecar(X0+scale*7.5,Y0+scale*2.5) → pendown() →

circle(X0+scale*7.5,Y0+scale*2.5,3.1416) **]**

where *scale* = 1. The letter size can be controlled by changing the value of the variable *scale*.

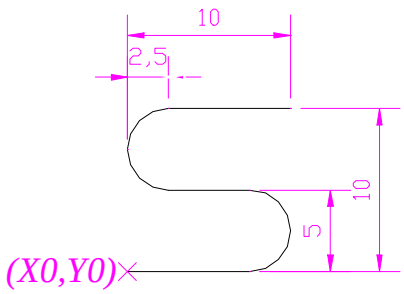

Figure 4.11: Drawing the letter S using a sequence of linecar and circle functions.

Its obvious that the input to his function is the coordinates of the lower left corner of the letter. This is called *free letter typing mode*, where the letter can by typed anywhere in the drawing plane. Another mode called *horizontal letter typing mode* will type-write letters on a horizontal line and will go to the beginning of the next line, if the end of the current line is reached.

4.5.d Drilling mode:

The objective of implementing this function is to show the multi functionality of this robot and the wide range of applications that it can be used for. The drilling mode will control both the parallel robot and the jack to make several holes on a printed circuit board (PCB) in order to have it ready for electrical components to be installed on. The manipulator will be used to move the drill to the required position to be drilled. The jack will rise the board slowly upword using the function jackup, then bring it down again using the function jackdown. If the nut is moved from the position x_1 to x_2, the board will by rised a distance $2(y_2-y_1)$ as can be seen in figure 4.12. x_2 can be evaluated from the relation

$$x_1^2 + y_1^2 = x_2^2 + y_2^2 \qquad (4.42)$$

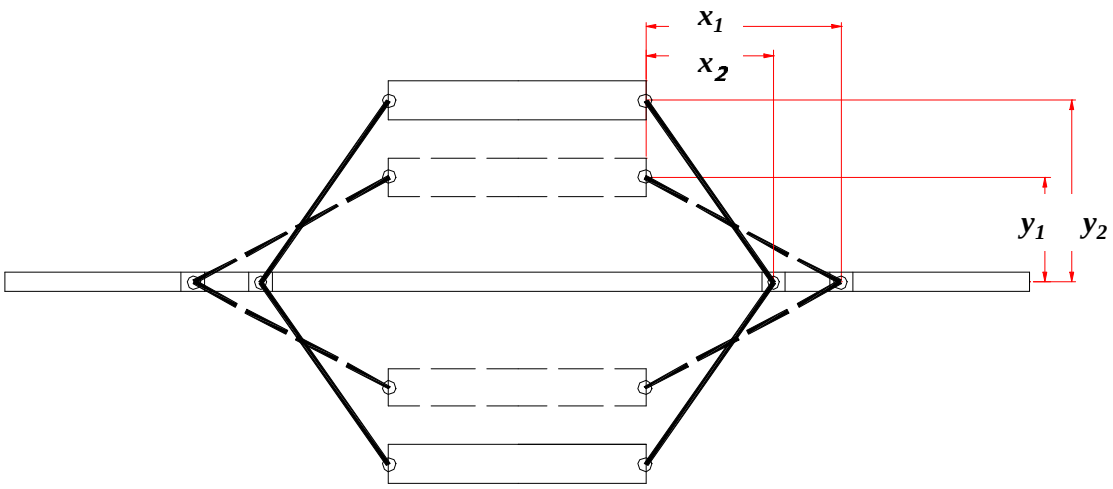

Figure 4.12: Evaluating the distance required to be moved by the nut.

In order to specify the drill path, the positions of the holes on the board should be known. The card was drawn on AutoCAD as shown in appendex.

Drilling mode will use the function drill which is a combination of the basic positioning functions in addition to jackup and jackdown. It is a part of the control code that can be found in appendix.