Sefton Industrial and Machine control systems chop saw run function

Chop Saw run routine

A run routine or function is a program that designed to operate a process, in this case the process is to cut material as defined in the menu program we wrote earlier. The parameters are passed to the run function in registers this is similar to that of functions in 'C' please see the example below:

Define the function

```
void Run( float x, float y, int qty)
{

// X is a value passed on the stack representing the X input
// Y is a value passed on the stack representing the Y input
// qty is the integer quantity
        int g;

        Move(x,y)
        cut(qty);
}
```

Usage

```
        Run(x, y, qty);
```

Map functions inherit all the registers of the parent process and therefore there is no need to pass variables the the function. Therefore the MAP equivalent would be invoked from your menu using an OnKey command as shown below:

OnKey run.mcp

The most important thing we need to do when considering writing a program for a machine is to consider safety. Safety is the most important thing in any environment, this machine obviously has a saw that can be extremely dangerous to the operator.

Normally any process follows similar considerations. The most obvious consideration is making sure that before there is any movement the machine is safe. It may well be that the power was turned off while the machine was running. If this is the case it might well be that cutting tools may be actively in location. For example if we imagine a drilling machine which was in the process of drilling when the machine was turned off it is likely that the drill head will still be in position. Any attempt therefore to move any axis must start by withdrawing the drill first. If we were to send the X axis home before we brought the drill home obviously this would break the drill. Chop saw is different in that it is unlikely to have any issues when an axis goes home regardless of the order. On the chop saw, the X axis (which is the axis used to push the material towards the saw and is called the back gauge) always goes home away from the saw blade and therefore cannot cause any damage. Likewise the saw blade can only go vertically up or down, bringing the saw down if it is not running is not a good idea however returning it to the up position is safe at any time.

In this machine if we assume it is a two axis machine it is safe to home both axis at the same time.

Therefore the first thing we would do when passing control to our run program is to ensure that the axis returned home and that any pneumatics used for example for clamping the material are deactivated. To deactivate the outputs all we need to do is to write 0 to the output port this will automatically turn all the drivers (a driver is a transistor which controls an output), this should also turn the saw off. After turning off the drivers the axis can be sent to their zero position. Note some control systems require the programmer to home the axis themselves. In map the programmer can rely on the automatic homing system. If the axis have not been homed (meaning is moving the axis to a safe zero position) then map will automatically home the axis in the order they have been set up in the homing menu. However programmers should be aware that some languages will require homing to be part of the program they write.

For our chop saw project we are going to assume that there is a sensor which can inform us when material is present or not. When the operator wants to load material in the saw, they should arrange that the back gauge is in the zero position (as far away from the cutting blade as possible) to allow the operator to put the material in the machine. When the operator presses the start button the controller must ensure that the saw blade is in the position and turned off, the back gauge is then moved until the sensor indicates that there is material present. Normally the sensor will be placed ahead of the cutting blade, in this case we will assume 25 mm ahead of the cutting blade. The type of sensor chosen is also very important, if this saw was to only cut metal we could use an inductive proximity switch, however our saw can also cut plastics and wood therefore we need to use a sensor which can detect non-metallic materials, normally this would be using infrared light. Although there are different types of sensor probably the most common typical type is shown below.

Through beam sensors as the name suggests emit infrared light from a transmitter mounted on one side of the sensing zone and on the other side there is a receiver. The output can be either high or low for inactive e.g. there is nothing in the sensor. The sensors therefore have two output states either blocked where the light cannot flow from one side to the other or unblocked. In our example we will look for the sensor to become blocked e.g. the material has entered the sensor. When we see that the sensor has become active, we need to move the material forward by the distance of the sensor from the blade plus the thickness of the blade and a safe cutting distance.

Cutting blades such as saws and even guillotines have a safe cutting distance. The reason for this is that if we were to cut insufficient material then there is not pressure on both sides of the cutting blade. Because saw blades are thin they are not designed to cut on one edge and therefore cutting I dimension which is equal or less than the width of the saw can be very dangerous it can result in the saw blade bending and even breaking. To avoid this dangerous situation it is wise to cut a minimum of the width of the blade plus a safety margin which as a rule of thumb should be at least 6 mm. With some materials a very thin slice can be cut possibly only 1 mm thick, but great care must be taken to ensure that there is no detrimental effects with cutting small slices. Note the material must be firmly held in position or the saw can grab the material and this can be very dangerous.

Therefore once we have moved to the place where we wish to start cutting, we must activate the clamps to hold the material firmly in position. Most pneumatic clamps are relatively fast and normally would activate within 0.1 of the second. If we assume for your application that the clamps

are pneumatically operated all you need to do is to activate the clamps and you can assume that they are active. Before any movement of the saw takes place the blade must be started. Normally there will be a run-up time, this is the time taken by the saw to reach operating speed typical times are around 1.5 seconds. Once the saw has been activated the blade may be brought down at a constant speed and for your project we can assume that the movement is 200 mm. In reality we would need to calibrate the position where the saw is fully down, this would require was writing another menu to calibrate the saw.

During cutting and once a saw has reached the maximum point of 200 mm the saw blade is brought home again. Normally soaring applications also require a feed rate, this is the rate at which the saw blade advances, this can be dealt with in the calibration menu or can be added to the operator screen or included in a description of the material (this is a file which contains cutting speed, played width and other important cutting information) again we will not include this information in our project as it is purely a learning exercise.

Once blade has been returned home we can check to see if there are any further cuts that need to be made. If there are further cuts to be made, we need to release the clamps, and then we need to advance the back gauge by the length of the material needed and the thickness of the blade. We then continue the above until the quantity has been satisfied (we have made all the pieces we need).

If your menu has multiple cutting lists at this point you would move onto the next cutting list if not it is necessary to park the saw in the position, only when the saw is in the upper position release the clamp and turn the saw off. Note the clamp must never be released while the blade is coming down or returning up as the blade may hit the material which may cause the material to be thrown out of the machine where there to be a short remnant of material remaining in the feeder. Once the cutting list is complete with the saw in the safe position, the back gauge can be returned to the home position, the power can be turned off and run can return back to the user input menu you designed earlier. The commands you will need to complete this project are shown below note you all have a copy of the manual and your free to use whichever commands you wish. For the purposes of our project please assume that the through beam sensor will provide logic zero when the beam is present and logic one when the beam is blocked.

ChangeOp (change output)

Map **ChangeOp** command run time function Allows the user to change a single bit of the output port does nothing during simulation.

Data field usage:
Data[0] = the number of the bit to change data is constant long or register
Data[1] = the value of the state where 0 = reset the bit and none zero is set data is constant long or register

DoMove (start motion and enter a do while loop)

MAP **DoMove** command is used to undertake motion and start a while loop at the same time The while loop is the same as a normal while except that it is terminated if the condition becomes false or the motion completes. If a move is terminated early by the action of the while loop, the position is taken when the axis stops and written into the destination register so that the next move will correctly start from the point where the command terminated there may be a small error due to the granularity of the encoder resolution. In relative axis this error may build up where there are large numbers of do moves consideration should be given to this effect
if an axis does not need to move.

Data field usage:
Data[0] = X position constant or register (float).
Data[1] = Y position or register (float).
Data[2] = Z position or register (float).
Data[3] = axis 3 position or register (float).

LoadReg (load a register with a constant or variable data)

Map **LoadReg** command used to load a value into a register from a selection of sources where data is loaded into the register specified by data[0]
Data[0] = Constant register number of the register to be loaded (destination register).
Data[1] = is either the value to load if CC = VorR or an index for other condition code values this may be a constant float or register.

Data field usage:
Data[2] = Constant register number of the second register to be loaded (destination register).
Data[3] = is either the value to load if CC = VorR or an index for other condition code values this may be a constant float or register.

Condition codes:

0	VorR	: load data. Loads a constant or the contents of a register.
1	Apos	: load actual position in MM for the indexed axis.
2	Encr	: loads the encoder position register data for the indexed channel.
3	Ferr	: Load the sign adjusted positive following error from the indexed axis.
4	Torq	: Torque output this is the motion controller demand signal for the indexed axis.
5	Port	: Load input port. Loads the contents of the input port into the destination register. Data[1] has no effect .
6	IP.b	: Load input port bit, where data[1]= is the number of the bit to test.
7	Ikey	: Load the last key value from the keypad. if the Key is ESC or F8 the program aborts. Data[1] is not used and has no effect
8	Jpos	: Load the jog position register to the destination register. Data[1] specifies which register SMC values are 0-1 PMC4 0-2
9	T->R	: Load the current tool number into the destination register. Data[1] is unused.
10	(R)R	: Load register indirect, the contents of a register pointed to by the value contained in another register are loaded into the destination register. Data[1] is ignored.
11	RPOS	: Actual position relative to the origin example if the absolute position is 120 and the origin is 100 this will return 20.
12	AOPC	: Axis operation count: every time an axis completes a movement the counter is incremented mainly used with valved axis.
13	Dkey	: The debounced key the command automatically eliminates multiple key presses etc.

Output (output data to the outputs)

Map Output command run time function Allows the user to change the contents of the output port does nothing during simulation.

Data field usage:

Data[0] = the port output data which is constant long or register.

Move (linear interpolated move)

The **Move** instruction run time function uses the current feed rate or speed and moves to positions specified in the 4 data fields on completion continues to next instruction.

Data field usage:
Data[0] = X position constant or register (float).
Data[1] = Y position or register (float).
Data[2] = Z position or register (float).
Data[3] = axis 3 position or register (float) (PMC4 only)

While (part or a do while loop)

Map WHILE command run time used to do logic compare and jump has two functions when preceded by do if true will jump to the line in the do_vars array else will loop locally until the term is false.

Data field usage:
Data[0] = the base resister and is a constant register number
Data[1] = the compare register and is a constant long or register number to compare.

Condition Codes:

0	==	: Equal.
1	!=	: Not Equal.
2	>	: Greater than.
3	>=	: Greater or Equal.
4	<	: Less than.
5	<=	: Less or Equal.
6	BitL	: Register Bit = 0.
7	BitH	: Register Bit = 1.
8	PortL	: Input Bit = 0.
9	PortH	: Input Bit = 1.
10	FALSE	: Always Jump FALSE.
11	OKACT	: OnKey Function activated branch if a function was called.
12	IpAct	: Input Bit Active (if an input is active e.g. if PNP and input is logic high or NPN and input is logic low) branch on false.
13	InMot	: In motion, an axis is moving.