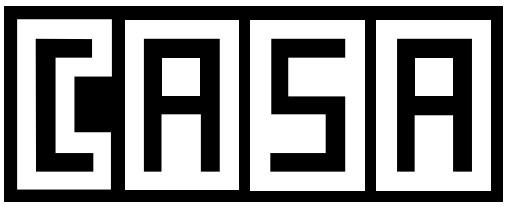


™

Intelligent - Stand Alone

User's Manual

(Firmware Configuration & Programming)

CASA Modular Systems
66-79 Fitzherbert Street
P.O. Box 38-828
PETONE
Wellington, New Zealand

Phone: 64-4-9393 777
E-Mail: sales@casa.co.nz
Fax: 64-4-9393 778

TERMS & CONDITIONS, WARRANTIES ETC.

INTRODUCTION - In the absence of other written terms (or specific contractual conditions) the following general terms and conditions of business shall apply in respect of goods **manufactured** by CASA Modular Systems. Goods of other manufacture are subject to the terms and conditions of our individual supplier's details of which can be provided upon request.

QUALITY - All goods manufactured by CASA are deemed to be suitable for the purposes for which they are designed, will perform within practical engineering limits and the specifications described on CASA's published literature.

WARRANTY - Within a period of 12 months from the date of supply, any goods, which prove to be defective will be repaired or replaced FREE OF CHARGE (fair wear and tear accepted). CASA' s liability is limited to one of the following: -

i) the repair (by our staff on our premises, or by our duly appointed agent) or:-

ii) the replacement with new or substitute goods or (failing the availability of effective repair or replacement):-

iii) the refund of the current value of the goods, not exceeding the original purchase price.

Return, and/or, replacement of goods will be at the discretion of CASA and subject to our inspection.

RETURNS - Any goods returned under warranty must be accompanied by proof of purchase (copy of our invoice or sale docket), an adequate description of the fault, suitably packed and consigned freight pre-paid to our office. Within New Zealand, return freight on warranty goods will be paid by CASA.

OWNERSHIP - Until full payment is received, legal title to goods supplied resides with CASA Modular Systems (or its appointed Agents).

PAYMENT - Unless an "Approved Credit Account" already exists, payment in full must be made prior to delivery. (Applications for CREDIT TERMS require to be made on CASA's special form).

PRICES - Published prices are subject to change without notice. Unless prices are established by written quotation the prices charged will be those in effect at the date of dispatch.

GENERAL - Should conditions arise which are not covered by the above, interpretation will be made according to current N.Z. Legislation.

These general terms and conditions are subject to change without notice. Every endeavor will be made to advise and update Customers as appropriate.

REVISION DETAILS : EDITION V1.06©

Edition Date - 26 May 1999

CONTENTS

INTRODUCTION

CASA SMCU

CASA's Intelligent ***Stepper Motor Controller Unit*** is designed to provide an integrated, versatile and flexible Microprocessor Based Controller for Single Bipolar Stepper Motors to serve a wide range of automated linear and rotary functions at an economical price.

SPECIAL FEATURES

Compatible with most Standard Stepping Motors

The SMCU may be used with almost all industry standard available stepping motors which are readily available from CASA's stock or by indent order.

Adaptable to Many Functions

The powerful and sophisticated features provided in the Standard SMCU facilitate its use in many applications (limited perhaps only by user's imagination and practical considerations). CASA is committed to adding new features and enhancing the Standard units to facilitate NEW Customer applications.

Simple to Operate

4 buttons START - STOP - PAUSE - RESET on the front panel (or externally wired to a remote location) provide all processing commands.

Depending on the requirements of the process, the operator of a machine or process using the CASA SMCU can achieve full control provided with one **START** button, or up to 3 additional buttons **PAUSE - STOP – RESET.**

Simple to Program

Simple ASCII characters address all built in Programming Commands and may be entered from any PC or Terminal (desktop or hand-held) via a RS232C port (connected via a 9 pin D-Sub connector).

Single Package Does it All

A single self-contained aluminum box (typically as small as 250 x 193 x 55mm) contains all the SMCU Electronics to facilitate comprehensive control of 4~8 wire Bi-polar driven Stepper Motors and may include the Power Supply for motors up to 4 Amps.

Flash-ROM Program Storage

64K of Flash-ROM facilitates the permanent storage of up to 5000 operational commands together with the SMCU's internal program. This enables complete stand-alone operation (once programming terminal has been disconnected) which can be re-programmed up to 100,000 times.

Variable Acceleration Ramp

Variable acceleration profiles can be made to suit any stepping motor and application load conditions such that resonance and load inertia can be optimized.

Remote I/O Switching

External connections to the work place enable interaction with the process to suit most operating applications and conditions.

Chopper Encoded Input

This future option provides monitoring of the STEPs in critical applications / processes and is soon to be available (and can be retrofitted to standard units).

Custom Programming

Special programs can be developed to facilitate non-standard features etc. Units can be supplied pre-programmed or re-programmed at our office to meet special applications.

Custom Packaging

Special enclosures can be made for IP rated environments etc.

New Zealand Designed

The CASA SMCUs are designed and manufactured in New Zealand by a company with 25 years of service to New Zealand's Professional Electronics Industries.

PRODUCT OVERVIEW

CASA's Stepper Motor Controller Unit is designed to provide an intelligent, integrated, versatile and flexible Microprocessor Based solution for the driving of single Bipolar Stepper Motors to serve a wide range of Industrial Automation and other processes at an economical price. Future expansion into Dual and Triple motors together with "Daisy-Chaining" of multiple SMCUs will enable complex systems to be engineered.

Brief Specification

- Step rate: 50 - 10,000 steps / second, with adjustable start and maximum speeds. Lower step rates are available to order. However, step rates in excess of 10,000 usually require "micro-stepping" and this is a future option reserved to applications requiring very high speed positional changes emulating servo type functions with closed loop encoding or special limit switches and appropriate overtravel such that the system can correct for overshoot etc.)
- Variable ramp up / down rate
- 4 motor phases
- Full or Half step modes
- Ability to store up to four programs of step sequences each typically in excess of 5000 commands in non volatile FLASH memory. An extra initialising program is also available in a fifth program area.
- 4 push button inputs (START, STOP, PAUSE, RESET)
- 4 indicator lights (LEDs)
- Configured from standard ASCII terminal / PC running emulation software.

Basic Concepts

Position

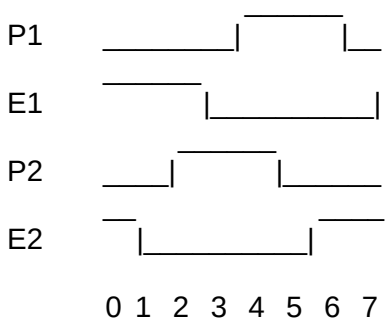
Using the IF command in association with limit switches, it is possible to associate an internal position or step counter with a known physical position. Since there is, as yet, no encoder input implemented, the system then depends on the ability of the motor to not slip to accurately know its position.

Direction

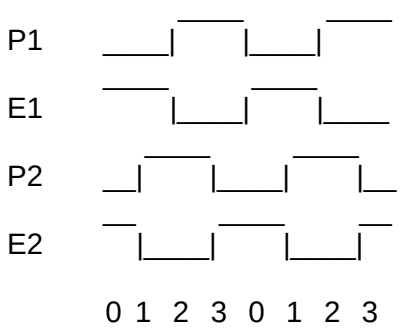
There are commands to move the motor *Forwards* and *Backwards*. The direction that these actually equate to is dependent on the motor wiring (changing any two phase leads will reverse the effective / relative direction) therefor the terms "Clockwise" or "Anti-clockwise" could be meaningless. Instead they are defined here in terms of a progression through the step pattern of the 4 motor phase outputs as shown below. Reverse is defined as the opposite way through the same sequence.

Step Sequence - Forwards

(half step mode)



(full step mode)

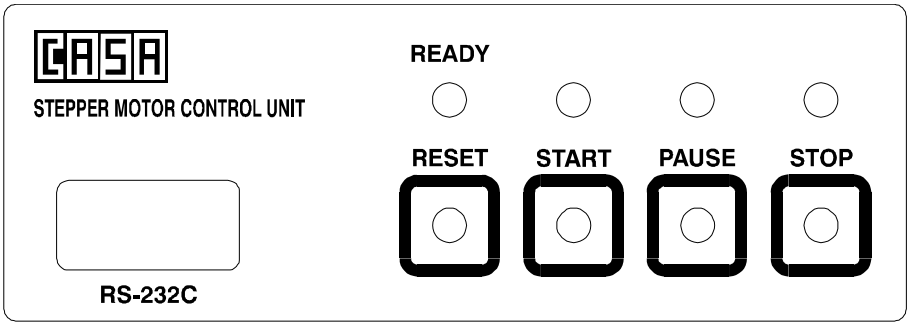


Comparison of ½ *Step & Full Step* modes

Step	Torque	Speed	Resolution	Acceleration	Resonance	Comment
Half	Less	Higher	Higher	Best		
Full	More	Lower	Lower	Poor		

FRONT PANEL BUTTONS & INDICATOR LIGHTS

The front panel of the controller incorporates four tactile MOMENTORY push button switches and four LED indicator lights.



Buttons

The four switches have the following function:

RESET Button

Stops any step sequence currently being executed, turns motor current off, resets the position counter and places the controller in the “ready-to-run” state.

Note- Upon installing the NC Software option RESET may also be used to return the mechanism to the “HOME” position (typically defined by a limit switch). Under these conditions the switch may be re-labeled “Reset - Home”.

START Button

Provided that the system is already in the “ready-to-run” state (achieved by RESET above) this button initiates execution of whatever sequence is currently loaded in main memory. If no sequence is loaded then this button has no effect.

Note- This button is also used as the “wait-for” condition to terminate wait periods.

PAUSE Button

Pauses execution of the **present** sequence or step command. If the motor is actually turning then it is first decelerated (at whatever rate is the present deceleration setting, refer “DE” command, below) to the start up speed before it is stopped. This prevents any slippage, which might occur if the motor were simply stopped suddenly from a high speed.

To re-start, press either **START** or **PAUSE** a second time, execution then resumes from the point in the sequence at which it stopped. It is assumed that position has not changed. The motor will be accelerated back up to the currently defined “Maximum” speed at the present “Acceleration” setting (refer “A” command below).

The controller will remain in the **PAUSED** state until either a restart (ie START / PAUSE as above) or a **RESET** are issued.

Motor Current is held ON during the Pause (a future option may reduce current or switch it off after a nominal time, otherwise, over-temperature sensors will be deployed to turn current OFF should the temperature of the heatsinks exceed 55 deg C).

If, when the PAUSE button is pressed, the controller was executing a WAIT ("w" command) then the re-START will terminate this WAIT period regardless of whether the required time has elapsed or not. (In the future, consideration may be given to alternative methods should it be preferred to continue or repeat the WAIT time in critical applications.)

Note - By the sequential use of PAUSE & Un-PAUSE & PAUSE etc … a "JOG" type function can be created which may be useful during initial set-up and testing of a program sequence. (In future software development it is proposed to implement the JOG function such that when the PAUSE button is held down for 3 seconds or more while/then pressing START the system will JOG through the NEXT step sequence in the program list).

In the case of RESET the paused sequence is simply aborted and the controller returns to the "Ready-to-Run" state (indicated by the "READY" LED).

STOP Button

Stops any currently running sequence or step cycle, ***immediately.*** This is intended as an emergency stop function. The motor output lines are frozen at the present position with current **ON** to provide maximum detent and slow down as quickly as possible. After a delay of one second the current is turned **OFF** so that the motor may be moved by hand if necessary.

Under some application conditions it is possible that the motor may slip during this process and so the absolute position may be lost. The controller will stay in a "stopped" state, prohibiting further stepping, until the **RESET** button is pressed.

Indicator Lights

The lights are arranged one above each switch (or at the side if the unit is mounted vertically) and will always pulse on (approx. 500ms) to acknowledge the pressing of the associated button. Additionally the lights are used to display status information as follows:

READY Light

Indicates that the controller is in the "ready-to-run" state.

NOTE - If the READY Light flashes this indicates that the internal thermal protection has been activated and has put the SMCU into the PAUSE condition (decelerates the system, turns on the PAUSE Lamp), turns the Motor Current off and flashes the READY Lamp fast. When the temperature falls the READY Lamp will flash slower at about one flash a second, the PAUSE Lamp will remain lit and the system may be Un-Paused by pressing the **PAUSE** Button or the whole system restored by pressing the **STOP** or **RESET** Buttons.

START Light

Indicates when the controller is in the process of executing a command sequence (and remains lit until the end of the present program or until **PAUSE** or **RESET** are pressed.

PAUSE Light

Indicates that the controller is currently in the *paused* state and is awaiting either a re-start (START or PAUSE) to continue or a **RESET** to abort the paused sequence.

STOP Light

Indicates that the controller has been stopped and is awaiting a **RESET** before returning to the "ready-to-run" state.

Power-Up Sequence

At power-up the following "Self-Check" sequence should be observed:

STOP ***LED*** ON -> ALL ***LEDs*** ON -> ALL ***LEDs*** OFF -> READY ***LED*** ON

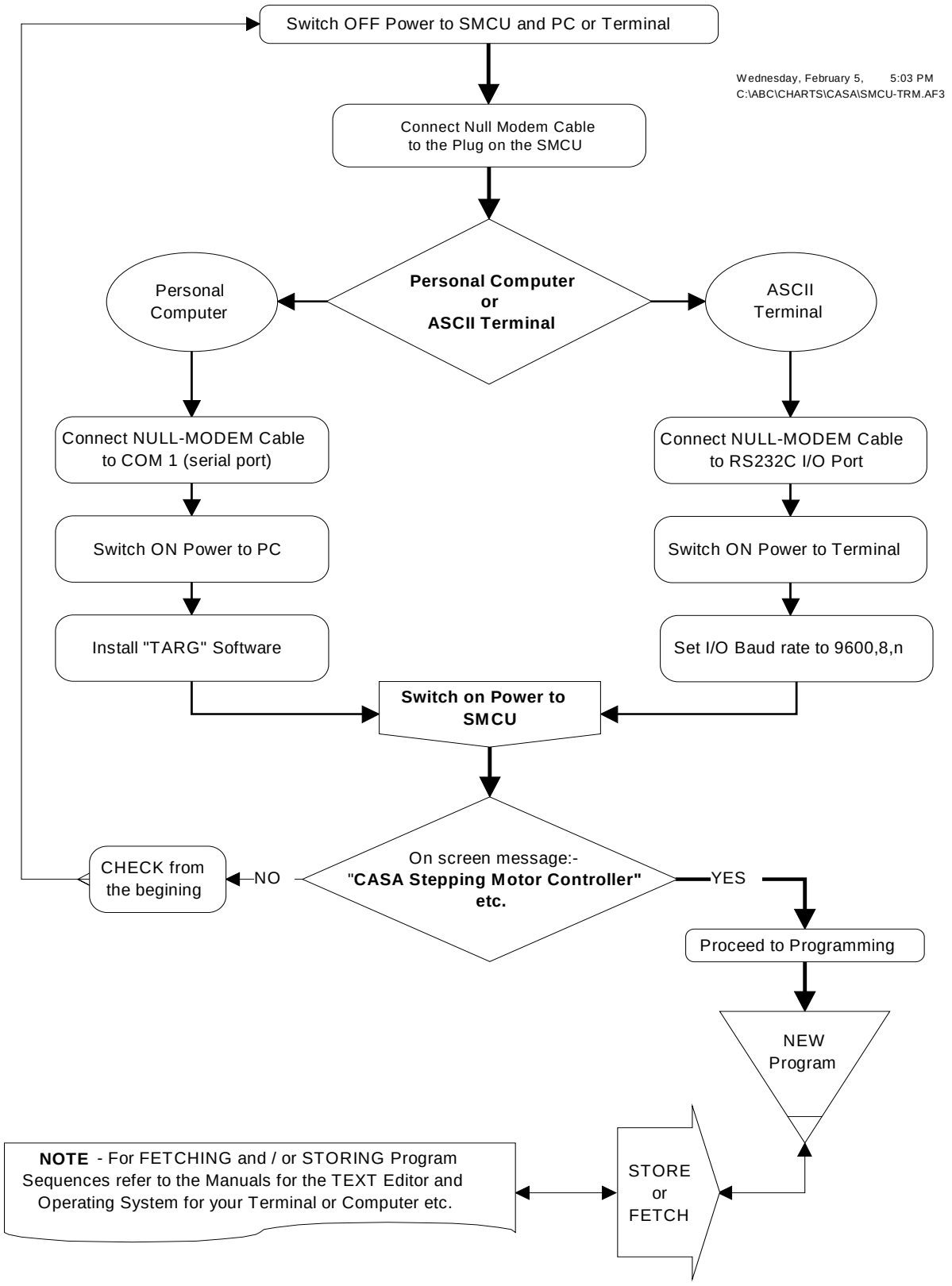
If this sequence is not observed this indicates a failure of the system to boot properly. Refer to the Installation and Diagnostic Flowcharts before referring to CASA Modular Systems with a clear description of the fault etc..

External Connection for remote indicators / switch inputs

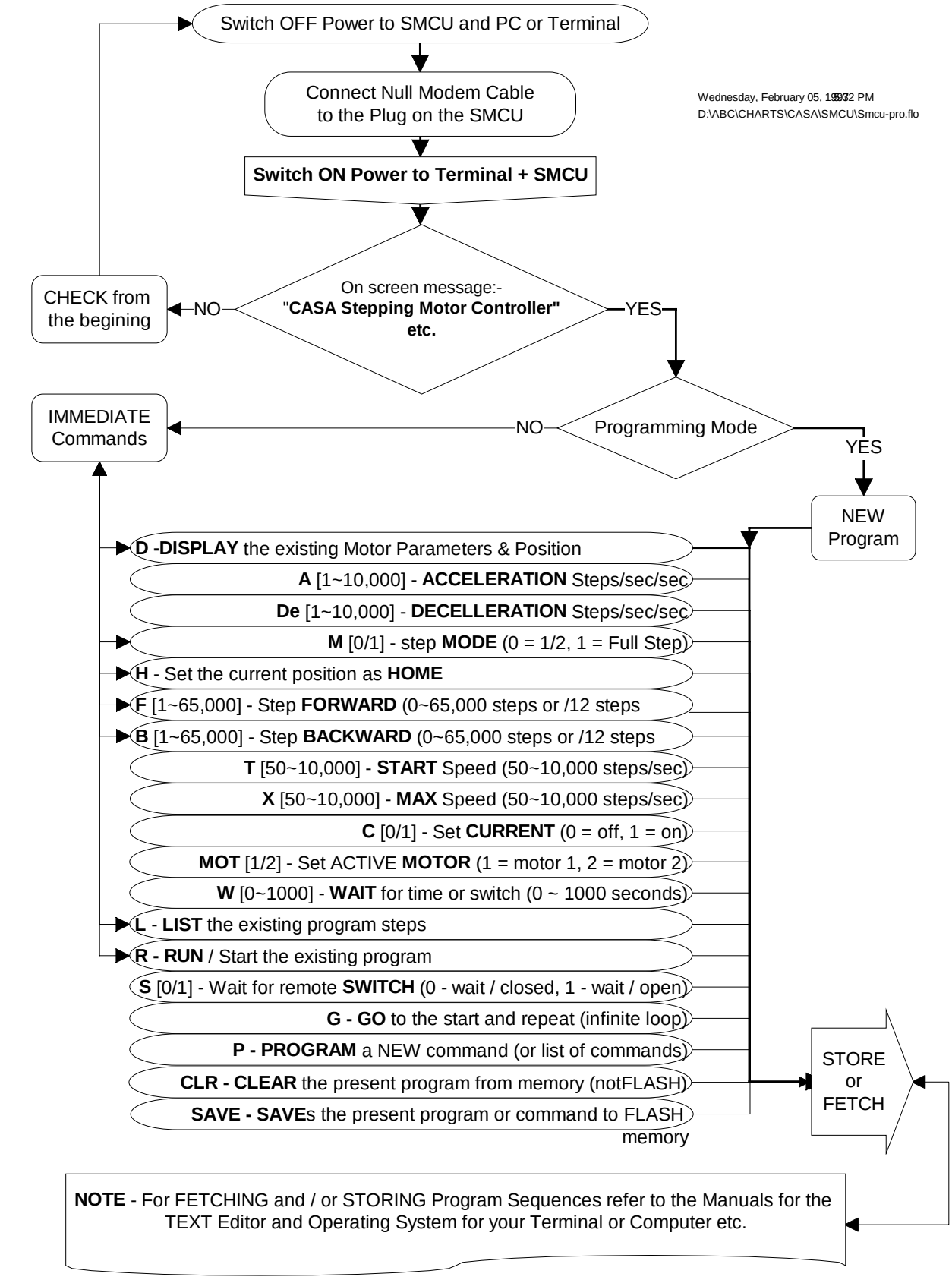
Additional inputs and outputs can be made available on the DB25 at the rear of the unit to provide remote indicators, remote operator control or process activated switches etc.

In order that automatic operation can begin or end *without* interaction with the front panel switches (or a separate remote control panel) the "S" command (wait for switch) can be used to interact with a single pole micro-switch or similar. This input has NO effect upon the front panel switches but provides for a step sequence, which may be PAUSED indefinitely or until such time as the remote micro-switch is actuated thereby permitting the NEXT command sequence to be executed.

Setting up the Terminal for the SMCU



Programming the SMCU



TERMINAL COMMANDS

The controller is configured from an ASCII terminal (or PC running "TARG" which is supplied with each unit or other simple terminal emulation software) set at 9600 baud, no parity, 8 data & 1 stop bit (9600, n, 8, 1) connected by an appropriate Null-Modem cable (available from CASA).

Command Summary

The following commands are available:

Motor Control Commands

d - **D**isplay the active step parameters (governing the motor or motors performance

m [0/1] - step **M**ode, [half/full]
x [50-10000] - ma**X** speed, [half-steps/sec or steps/sec]
t [50-10000] - star**T** speed, [half-steps/sec or steps/sec]
a [1-10,000] - **A**cceleration rate, [half-steps/sec^2 or steps/sec^2]
de [1-10,000] - **De**celeration rate, [half-steps/sec^2 or steps/sec^2]
h - set present position as **H**ome
c [0/1] - turn motor **C**urrent on or off [off/on]
f 0-65000 - step **F**orward, half-steps / steps (6/1/97 Alan suspects there may be some errors with larger values and values exceeding 65000 should be sent as an error to the terminal in future software updates)
b 0-65000 - step **B**ackwards, half-steps / steps (6/1/97 Alan suspect there may be some errors with larger values and values exceeding 65000 should be sent as an error to the terminal in future software updates)
w 0-1000 - **W**ait for time or switch pressed, secs/10, 0 = forever (or until STOP or RESET is pressed)
mot 1-2 - **MOT**or 1 or 2 active
s 0/1 - wait for remote **S**witch, 0 = wait for closed, 1 = wait for open
i 1/2 0/1 r[eset]/sta[rt]/p[ause]/sto[p]/n[o action] - **I**f near/far is open/closed then execute the specified action
o 0/1 - turn external contactor or solenoid **O**utput off or on
h 0/1 - **H**old stepping motor energised at lower current.

- Values in square brackets indicate optional arguments and their minimum and maximum values. Where such an argument is omitted the controller will display the present value of that particular parameter. eg c <enter> will return either the message *"motor current is on"* or *"motor current is off"* depending on the present status.
- Commands shown with arguments not in square brackets must be issued with an argument inside the specified range or an error message will be returned ("invalid argument"), eg F <enter> is not a valid command because the number of steps is not specified.

Programming Commands

l	- **L**ist the present program sequence
r	- **R**un the present program (same as start button)
clr	- **CL**ea**R** the active program from memory (not FLASH)
p	- **P**rogram new command sequence - appends to the present program (unless CLR entered previous)
g	- **G**o to the start and run the sequence again (indefinite loop)
save [0-4]	- **SAVE** the new program sequence to FLASH memory (non-volatile)
load [0-4]	- **LOAD** the saved program
ym [1-4]	- Commissioning motor drives
?	- display a help screen (these command lists abbreviated)

- All commands must be followed by the **<enter>** key
- Commands are **not** case sensitive and spaces are not required following the command characters.

A detailed description and examples of each command are given below.

Command Descriptions

D - Display Active Settings

usage: **D** <enter>

Displays the present values of the basic step parameters (step mode, start speed, max speed) to the terminal. Also displays the present value of the position counter.

M - Set Step Mode

usage:	**M** <enter>	*display present step mode*
	M 0 <enter>	*set to HALF STEP mode*
	M 1 <enter>	*set to FULL STEP mode*

Sets, or displays the present setting of the **step mode**.

X - MaX Speed

usage: ***X*** *n <enter>* *n is a value between 50 - 10000*

Set, or display the present setting of the top stepping speed. This is the speed that the motor will ramp up to when issued an **F** or **B** command. The value specified is either in Full or Half Steps / second depending on the setting of M (above).

Note that Max Speed cannot be set *lower* than Start Speed (see below).

T - StarT Speed

usage: ***T*** *n <enter>* *n is a value between 50 - 10000*

Set, or display the present setting of the motor starting speed. This is the speed that the motor (together with its related load etc.) *will* start at, and ramp down to at the end of a step cycle, when issued an **F** or **B** command. This speed should be well below the pull-in speed of the target motor / load combination. The value specified is either in Full or Half Steps / second depending on the setting of M (above).

Note: The "Start Speed" cannot be set *higher* than Max Speed (see above) and it is recommended to set the MAX speed first. Tests are necessary to establish the best results and eliminate or minimize resonance in the system.

A - Acceleration

usage: ***A*** *n <enter>* *n is a value between 1 - 10000*

Set, or display the present setting of, the acceleration. This is the rate at which speed will increase from the start speed to the MAX speed (ramp rate). The value specified is either in Full or Half Steps / second^2 depending on the setting of **M** (above).

A suitable value must be established empirically based on target motor / load combinations such that the acceleration is as steep as possible without causing slippage.

Note: Future software developments will provide a general manipulation and EXERCISE routines from which it is relatively easy to determine optimum TUNING values for all programmable motor parameters and thereby quickly achieve performance efficiency.

DE - DEceleration

usage: ***DE*** *n <enter>* *n is a value between 1 - 10000*

Set, or display the present setting of the deceleration. This is the rate at which speed will decrease towards start speed (ramp down rate). The value specified is either in Full or Half Steps/second^2 depending on the setting of M (above).

Again a suitable value must be established empirically.

H - Set Home

usage: **H** <enter>

Sets the present position as the home position. (At the moment this does no more than zero the position counter - limit switch, inputs etc. are required to make concept of "home" meaningful).

C - Motor Current On / Off

usage:	**C** <enter>	*display motor current status*
	C 0 <enter>	*current off*
	C 1 <enter>	*current on*

Set, or display the present setting of the motor current switch.

This energy save function enables the power in the motor (and current limiting resistors) to be turned off when there is no useful work being done and thereby avoid overheating and wasting of power.

At power up the unit does not automatically energize the outputs and turn on current to the motor. Current is turned on via this command or automatically when an **F** or **B** command is received. Similarly current is not turned off at the end of a step cycle but must be explicitly turned off via this command if so desired. When turned on the controller assumes that the motor has not moved from the position it was last driven to - if it has the motor might "kick" slightly when turned on as it jumps to the next position with the same phase pattern as the last driven position. Detent is much

higher with the current on but both the controller and motor may get excessively hot if motor current is left on for extended periods.

Note: Motor current is turned off automatically 1 second after the STOP button is pressed.

F - Step Forwards

usage: **F** n <enter> n is a value between 0 - 65000

Causes the motor to step in a forwards direction (refer Overview, above, for meaning of forwards & backwards) for n steps. Initial step rate will be the present setting of Start Speed (T, above) and will accelerate to Max Speed (X). This Max Speed will be maintained until the sequence approaches the target number of steps to be covered when the controller will decelerate the motor, aiming to reach Start Speed again at the target position.

If the motor reaches Start Speed slightly before the target number of steps have been covered then the motor will creep the last few steps at Start Speed. If the target number of steps is covered before start speed is reached then the motor will just stop anyway. For this reason it is important that the value chosen for Start Speed be well below the pull-in limit for the chosen motor / load combination.

Where n is small and the difference between Start & Max Speeds is large it is possible that the motor may never reach Max Speed before it begins to decelerate. This is not a problem since the final point will be reached in the minimum of time.

If Start Speed = Max Speed then there will be no ramp up or down.

B - Step Backwards

usage: **B** n <enter> n is a value between 0 - 65000

Steps the motor n steps in the opposite direction to **F** above, in exactly the same manner.

MOT - Motor

usage: **MOT** n <enter> n is 1 or 2

Activates either motor 1 or motor 2. Subsequent commands drive the active motor.

H - Hold

usage: **H** f <enter> f is 0 for off, 1 for on

Holds a stepping motor at near its current position. When Hold is turned on the motor drives till a particular motor coil is energised, then it reduces the current in that coil so that the motor will be held in that position.

Useful for when two motors are being driven - one is held while the other is driven.

W - Wait

usage: **W** t <enter> t is a time between 0-1000 tenths of seconds

Waits for either start button or t/10 seconds. If t = 0 then waits indefinitely for the start (or reset) button.

If the sequence is PAUSED during a WAIT period then the WAIT is terminated by re-**START** (ie the START & PAUSE buttons). The system does NOT resume the WAITing following a re-START).

This facility is typically used during sequences where time is required for some external operation to happen where the external operation is NOT under the SMCU control.

Note: Wait may be aborted by pressing either the STOP or RESET buttons.

S - wait for Switch (Conditional Start)

usage:	**S 0** <enter>	wait for EXTERNAL switch to CLOSE
usage:	**S 1** <enter>	wait for EXTERNAL switch to OPEN

Waits for an external micro-switch (or similar) to either open or close.

If the remote switch is already in the Target State then this command will just pass straight through (with no delay) initiating the next/appropriate sequence.

The “wait for Switch” may be aborted by pressing the Stop or Reset buttons, however, it will of course abort the whole of the remaining sequence.

I - If .. Then

usage:	**I n f str**<enter>	n is 1 (near) or 2 (far), f is 1 (closed) or 0 (open), str is r[eset]/sta[rt]/p[ause]/sto[p]/t[runcate]/n[o action]

There is provision for two external switches - near and far. These may be limit switches, or wired as required.

Initially on power up there is no action taken if either switch opens or closes. By using the I command, then an action takes place if either changes state at a later time. The function of either switch can change during the execution of the installed program. Thus the near and far switches can be considered configurable external extended versions of the front panel switches.

There are six possibilities: RESET, START, PAUSE, STOP, TRUNCATE and NO ACTION. TRUNCATE is an instantaneous stop with the ability to continue immediately.

For example the program may initially drive the motor one way slowly with the near switch set to action TRUNCATE when it closes. Then the program may continue with both near and far configured to STOP when either closes again.

O - Output

usage:	**O 0**<enter>	turn output off (low voltage)
	O 1<enter>	turn output on (high voltage)

Used for driving a contactor or solenoid. The output is driven by an open drain FET power transistor.

L - List Present Sequence

usage: **L** <enter>

Displays the presently loaded command sequence. This is the sequence presently in the main memory, which will not necessarily be the "permanent" one stored in non-volatile memory.

[PS it is as yet undetermined how to page through long entries where terminals don't have built in editors.]

R - Run the Present Sequence

usage: **R** <enter>

Runs the presently loaded sequence. This is similar to pressing the **START** button.

CLR - CLeaR the Present Sequence

usage: **CLR** <enter>

Clears the sequence from main memory. Any sequence saved to non-volatile memory will be unaffected. This is required before a **new** sequence can be entered (otherwise programming only appends new commands to any existing sequence).

[PS the append feature may eventually be used to enable EXTRA commands to be added to the end of an existing sequence without the need to use a text editor]

P - Program a New Command Sequence

usage: **P** <enter>

command 1 <enter>
command 2 <enter>
command 3 <enter>
….
command n <enter>

Q

Enter the command sequence programming mode. All commands entered until **Q** will be appended to the existing command sequence. Refer Programming Mode, below.

G - Go to start of sequence

usage: **G** <enter>

Jumps the program back to the start of a sequence and begins repeating the sequence indefinitely.

This command facilitates an infinite loop, which can only be terminated by the **START** or **RESET** buttons (pause will work only to stall the loop until un-paused).

Note - It is important that "**G**" be used as the very LAST command in a sequence otherwise any commands entered after a "**G**" will never be executed.

SAVE - SAVE the Present Sequence to Non-volatile Memory

usage **SAVE** <enter>

Saves a copy of the present command sequence to non-volatile (FLASH) memory. A sequenced saved here will be copied back to main memory at power up and can then be run via **START** buttons etc. from Stand Alone Mode (refer below).

PE - PErsonality

usage: **PE [s m]**<enter> s = serial number, m = model number

Used for factory commissioning only, for configuring the software to the hardware required for that model, and to provide specialised software functions for particular applications.

The models available are:
Model 1 - standard model as described in this manual
Model 2 - DRI milk powder stirrer. This model does not have over-temperature detection, nor access to the near/far If .. then ability. The four front panel pushbutttons are used to load the four programs from non-volatile storage. The associated lamps follow the button presses.

YM - Motor Commissioning

usage: ***YM [n]***<enter> n is 1, 2, 3, or 4

Used during commissioning to single step a motor with one active coil only. Subsequent YM commands increment to the next step in the forward direction. The convention for forward is a right handed screw - clockwise rotation looking from the rear of the motor. Set MOT 1 or MOT 2 first.

? - Help

usage: **?** <enter>

Displays the above list of available commands (refer to the Command Summary, above).

OPERATING MODES

Immediate Mode

If any of the "motor control" commands above are entered from the terminal then the requested action will happen immediately, using whatever parameters are currently set. For instance typing "F 5000" <enter> at the terminal will cause the motor to be stepped 5000 steps or half-steps (depending on the present step-mode), starting at the present setting of start-speed, ramping up to the present setting of max-speed, *immediately* after the <enter> key is pressed.

This mode is useful for testing and configuring the Stepper Controller. It is suggested that new users first experiment with the controller in this mode in order to become familiar with it, before attempting to write the required sequence(s).

In immediate mode, wait and step commands can be interrupted and aborted by pressing either the **STOP** or **RESET** buttons on the controller's front panel.

Any parameters, which are changed, remain effective until the parameter is changed again, or the unit is powered off. eg "X 3000" <enter> sets the top stepping speed to 3000 steps a second. Once this has been set all step commands ("**F**" or "**B**") will ramp up to this speed. The existing values of the step parameters can be displayed with the "D" command.

Note: Future software developments will provide a general manipulation and EXERCISE routine from which it will be easy to determine optimum TUNING values for all programmable motor parameters so as to achieve performance efficiencies and economies.

Programming Mode

Any of the above motor control commands can be used to build a sequence or "program" of steps and waits etc. as required. Once loaded such a sequence can then be stored in non-volatile FLASH memory and run every time the "**START**" button is pressed (even if the terminal or PC is no longer connected).

Entering Step Sequences

To enter programming mode type "**P**" <enter> from the ASCII terminal or PC. The commands that are required to form the desired sequence can then be entered (in the required sequence). Each command must be followed by <enter>. The controller will not execute the commands when they are typed in but will just store them until SAVE or RUN are activated.

A "**Q**" (or "q") must be entered to end the sequence, and exit programming mode.

A sample sequence:

x3000
t500
m0
f50000
b50000
x5000

f 10000

w0

It is recommended that the sequence should always begin by setting up the required step parameters - *start speed* (T), *max speed* (X) and *step mode* (M).

Note: If required these parameters may be changed at any time during the sequence (as per the example above). Each parameter setting remains in force until overwritten by a new value.

Note: The commands are *appended* to any existing sequence so if a new sequence (this enables a program to be built "STEP-at-a -TIME" providing a check and repeat capability) is required then the program memory must first be cleared before the sequence is entered. Clearing the program memory is achieved by issuing the "**CLR**" command.

To review a sequence once it has been entered type the command "**L**".

Infinite Loop

Any sequence may be made to repeat endlessly by appending a "**G**" command to the end of the sequence. This will cause the whole of the preceding sequence (or sequences) to be repeated in its entirety every time the end ("G" command) is reached.

The STOP or RESET commands (but may be interrupted indefinitely by the PAUSE button) can ONLY break the endless loop thus created.

Note - It is therefor obvious that any commands entered after the "**G**" command will be effectively and completely disregarded (made ineffective).

Running the Step Sequence

The sequence may be run either by entering the "**R**" command or by pressing the "**START**" button on the controller's front panel (or remote switch if installed). During execution, status messages are echoed to the display terminal, or PC, indicating the steps as each is completed in the program sequence. This is useful when de-bugging a new sequence.

A sequence may be aborted at anytime by pressing the "**STOP**" button.

Saving the Sequence

Initially the sequence is stored in the controller's main memory (RAM). Sequences stored here will be lost when the controller is turned off. To save the sequence to non-volatile memory (i.e. permanent FLASH memory) type the word "**SAVE**" <enter>. Note that this will overwrite any previously stored sequence. Once saved in this way the sequence will be available whenever the unit is powered up, until replaced by another sequence. The save process may take a few seconds for a large sequence.

At power up only (any) sequences which was previously saved to non-volatile memory will be loaded into main memory, from where it can be run.

Note: If a new sequence is subsequently loaded to main memory from the terminal, it is this *new* sequence which will run when the **START** button is pressed (or R command issued) or listed when the "**L"** command is issued, not the sequence that is or was stored in non-volatile memory.

Note that **CLR** command mentioned above does **not** affect the sequence stored in non-volatile memory. If it is desired to clear a saved sequence then **CLR** followed by **SAVE** will do the job (ie SAVE an empty sequence).

There are four areas for programs, so up to four separate programs may be saved in the non-volatile memory, using the command SAVE 1, SAVE 2, SAVE 3 or SAVE 4. The standard SAVE command saves to area 1, and the power-up load is from area 1.

When the system is in the ready-to-run mode, it is possible to load into main memory any of the four previously saved programs using the command LOAD 1, LOAD 2, LOAD 3, or LOAD 4.

In the model tuned to DRI's milk-stirring requirements, the four pushbuttons duplicate these load commands.

There is a fifth area accessed using SAVE 0 and LOAD 0 that is run on power up only. This is used for initialising the stepping motor, to drive it to the correct position for the start of the main program.

Loading Sequences from the Terminal or PC

Alternatively, if terminal software that allows screen editing and downloading of text files is available (eg. Targ) then sequences may be written with a simple text editor and then downloaded. If the sequence is saved to disk (in text file format) then it can be recalled later. In this way a library of different sequences can be built up, saved on the terminal computer and downloaded when that particular task is required. Thus the re-configuration process is much simplified without suffering the constraints and limitations of dedicated Stepper Controllers.

This is particularly useful if either frequent sequences changes or very long, complex sequences are required.

Note: As at the 8/1/97, any loops required in the sequence must be expanded out using a standard text editor. Such a sequence file must begin with the **CLR** and **P** commands and must end with a **Q**. Between the **P** and **Q** commands should be the same parameter and step commands that would be used if the sequence were being entered directly from the terminal, eg:

```
CLR
    P
    x 3000
    t 500
    m 0
    f 50000
    b 50000
    x 5000
    f 10000
    w 0
    Q
SAVE
```

Sequence Length

There are approximately 30,000 bytes of memory available for sequence storage in each of the five areas (four non-volatile and the main memory). Each command takes a minimum of 2 bytes (command character and delimiter) with a further byte per character for any arguments - 5 bytes per command is probably a conservative rule of thumb. Using this guide there would typically be room for a sequence of 6000 commands.

Printing Program Listings

Assuming that a printer is connected to the terminal or PC, listings on the screen can be directed to the connected printer using the Terminals PRINT Function or in the case of the PC, DOS Print-Screen.

Ultimately CASA will provide utilities available from within the Terminal or PC's functions, however, future software developments will offer these facilities from within the SMCU or in association with simple add-on utilities from the public domain.

Stand Alone Mode

Once a sequence has been loaded and saved to FLASH the terminal may be disconnected.

The unit is then controlled via the front panel buttons (or remote switches) and operator feed back is provided by way of the four indicator lights (LEDs).

If the READY indicator (LED above the RESET switch) is on then the saved sequence will be run when the **START** button is pressed. It can be aborted using the **STOP** and **RESET** buttons. The indicator lights show the present state (ie the light below the **START** button will be on during sequence execution etc).

Note: Following a **STOP** the **RESET** button **must** be pressed before the sequence can be run again. This is indicated by the READY light off and STOP light on.

Motor Testing & Tuning

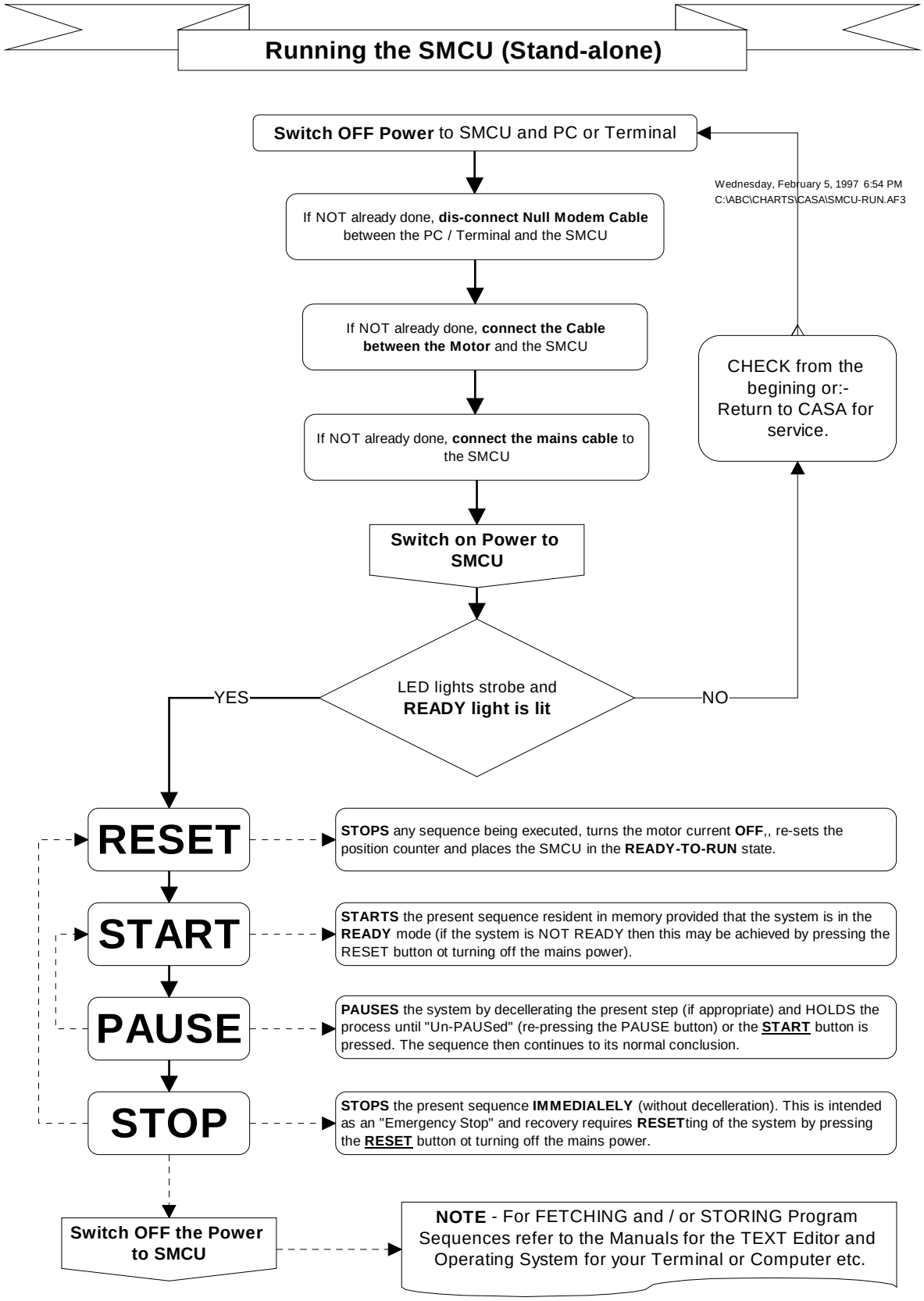
Provided that an encoder is (or can be) fitted, CASA's future software developments will provide a general manipulation EXERCISE for the motor and attached mechanics. Programmable routines will make it easy to determine optimum TUNING values for all programmable motor parameters so as to achieve performance efficiencies and economies not available other than by costly Trial & Error (suck-it-and-see) methods.

Over Temperature Alarm

Protection has been added to prevent excess heat build-up in the event that motors are inadvertently left in the IDLE condition with the current on and to protect the SMCU should it be connected to a motor which is either the wrong size or faulty. This protection is in the form of an internal Thermal Switch which operates (switch goes open circuit) when the temperature on the heatsink exceeds 55 deg C.

This condition puts the SMCU into the PAUSE condition (decelerates the system, turns on the PAUSE Lamp), turns the Motor Current off and fast flashes the READY Lamp.

When the temperature falls (the thermal switch goes closed circuit at about 40 deg C), the READY Lamp will flash slower (about 1 flash per second), the PAUSE Lamp will remain lit and the system may be Un-Paused by pressing the PAUSE Button or the whole system restored by pressing the STOP or RESET Buttons.


Running the SMCU (Stand-alone)
Switch OFF Power to SMCU and PC or Terminal
Wednesday, February 5, 1997 6:54 PM
C:\ABC\CHARTS\CASA\SMCU-RUN.AF3
If NOT already done, dis-connect Null Modem Cable between the PC / Terminal and the SMCU
If NOT already done, connect the Cable between the Motor and the SMCU
If NOT already done, connect the mains cable to the SMCU
CHECK from the begining or:- Return to CASA for service.
Switch on Power to SMCU
LED lights strobe and READY light is lit
YES
NO
RESET
STOPS any sequence being executed, turns the motor current OFF,, re-sets the position counter and places the SMCU in the READY-TO-RUN state.
START
STARTS the present sequence resident in memory provided that the system is in the READY mode (if the system is NOT READY then this may be achieved by pressing the RESET button ot turning off the mains power).
PAUSE
PAUSES the system by decellerating the present step (if appropriate) and HOLDS the process until "Un-PAUSed" (re-pressing the PAUSE button) or the START button is pressed. The sequence then continues to its normal conclusion.
STOP
STOPS the present sequence IMMEDIALELY (without decelleration). This is intended as an "Emergency Stop" and recovery requires RESETting of the system by pressing the RESET button ot turning off the mains power.
Switch OFF the Power to SMCU
NOTE - For FETCHING and / or STORING Program Sequences refer to the Manuals for the TEXT Editor and Operating System for your Terminal or Computer etc.

PROGRAM LIST

Krypton / DRI Residue Test Mixer

```
( PROGRAM 0 - UP - INITIALISE
clr
p
mot 2      ( UP
m 0              ( half step mode
t 400            ( start speed
x 700            ( max speed
a 3000           ( acceleration
de 6000          ( deceleration
h 0              ( remove hold
b 2050           ( drive up
h 1              ( hold
c 0              ( main current off
q
save 0

( PROGRAM 1 - DOWN - STIR - UP
clr
p
mot 2      ( DOWN
m 0              ( half step
t 700            ( start speed
x 1500           ( max speed
a 3000           ( acceleration
de 3000          ( deceleration
h 0              ( remove hold
f 2050           ( drive down
mot 1      ( STIR
t 700            ( start speed
x 1500           ( max speed
a 6000           ( acceleration
de 4000          ( deceleration
f 6000           ( stir clockwise
w 1              ( wait 100 ms
b 6000           ( stir anti-clockwise
mot 2      ( UP
t 400            ( start speed
x 700            ( max speed
a 3000           ( acceleration
de 6000          ( deceleration
b 2050           ( drive up
h 1              ( hold
c 0              ( main current off
q
save 1

( PROGRAM 4 - DOWN - STIR - UP - continuous loop
( same as program 1 with g added at the end
```