CHAPTER 3

EXPERIMENTAL METHODOLOGY

3.1 The CO_2 Laser System

Overview

A home built [56, 57] FAF CO_2 Laser system is used for the mild steel cutting experiment. The system consists of 8 discharge tubes which can deliver maximum output power of 1.4kW with an efficiency of 22.3%. The design of this system will be discussed here. This laser system can be divided into four categories, namely the power supply, the resonator, the cooling and recirculation system and the vacuum system.

3.1.1 ***Power Supply System***

a) The Negative High Voltage Charger

The high voltage power supply (Figure 3.1.1) is a home built system based on a 3 phase D.C rectified circuit. The 3 phase power supply provides a more stable output power when compared to other single-phase power supply. This power supply can convert A.C 220-240V input to 26 kV D.C high voltage output with a current rating of 1A. This high voltage charger is designed to be a negative voltage charger.

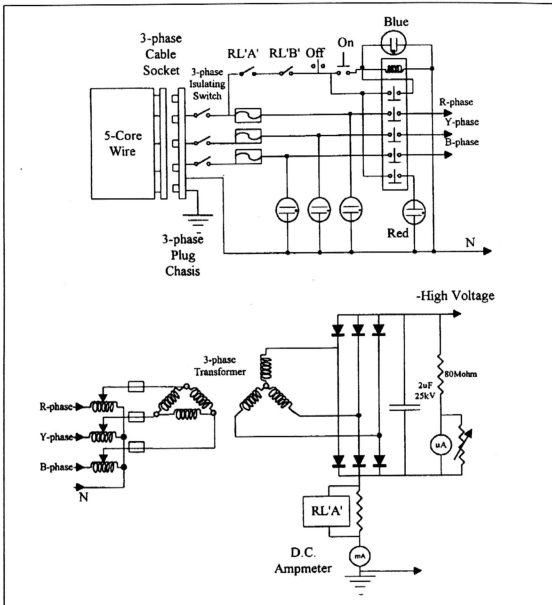

Figure 3.1.1: The DC Power Supply Circuit

b) The Preionizer

A 15kV high voltage proionizer (step up by a car ignition coil) and a high PRF of 1-4kHz (results from the combination of 555 Timer and MOSET) was integrated into each discharge tube in the laser system. In the initial breakdown stage, the preionizer generates a periodical supply of free electrons to the main discharge region which in turn reduces the main electrode gap breakdown voltage in the discharge tubes. This enhances the discharge uniformity and stability. Another main purpose of the preionizer is to maintain the discharge during laser operation. The need to use a preionizer to maintain the discharge is due to the voltage fluctuation at high operating current condition in a short discharge tube. The preionizer circuit is shown in Figure 3.1.2.

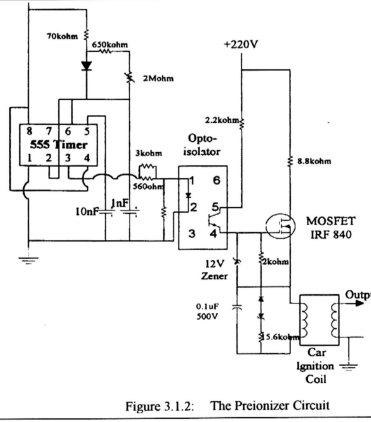

Figure 3.1.2: The Preionizer Circuit

3.2.2 *Laser Resonator*

a) Mechanical Components

The design is based on a conventional Febry-Perot resonator. The laser channel consists of 8 ceramic (Pyrex glass) tubes (Figure 3.1.3) with a high length to width ratio. The discharge length is 192cm and the diameter is 2cm. To reduce the physical length, the U-fold configuration is applied with each fold consists of 4 tubes. Both ends of each tube are mounted on a brass laser head in order to fix it in a stable position and free from vibration. The optical axis of the laser channel is designed in a 45^0 angle to the laser beam axis to adapt to the retardation angle of the Reflective Phase Retarder in the Beam Delivery System.

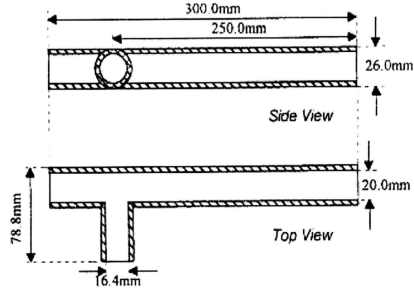

Figure 3.1.3: Dimensions of Pyrex Glass Tubes

b) Optical Components

The optical resonator (Figure 3.1.4) is built from four mirrors: the Back Mirror, the Output Coupler, and the two Folding Mirrors. The output beam from the entire configuration will be S-polarised.

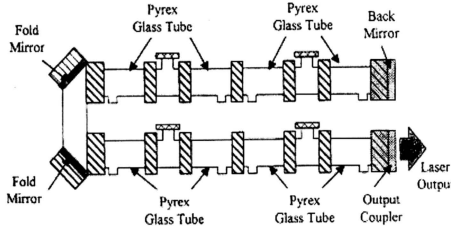

Figure 3.1.4: The Laser Channel

c) The Supporting Structure

The resonator is supported by its supporting structure (Figure 3.1.5) which can keep the optics in the best possible alignment with one another during operation. The structure consists of two flat brass plates which carry the resonator mirrors and are held apart by four brass tubes. Brass is a low thermal expansion material which can minimize the changes in discharge length during operation. The spacer tubes are filled with flowing water to further stabilize their temperature and isolate them from temperature changes inside the laser enclosure.

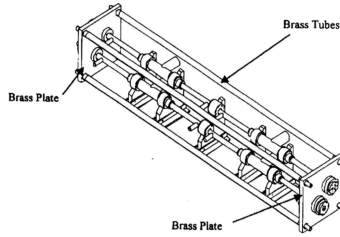

Figure 3.1.5: The Supporting Structure

d) Electrodes

Each tube is connected with a 50kΩ ballast resistors (4*200kΩ parallel) through the cathode and powered with a negative high voltage. The ballast resistors on each tube stabilize the discharge current during laser operation in the negative dynamic plasma resistance. The side pin anodes are mounted perpendicular to each of the glass tube as the positive high voltage electrode. A set of perforated plate and washer forms the side pin anode and is designed to generate turbulence, when the gas mixture is injected at the speed of sound through the plate. Turbulence flow plays an important part in:

i. Improving the homogeneity of the discharge current or electron density.

ii. Increasing the local diffusion rate and recombination rate of the charge particles near the wall by turbulent diffusion process. Hence, the maximum input power is increased.

iii. Reducing the current density near the anode by putting a washer on the side pin anode. Hence uniformity could be improved especially at higher current.

3.1.3 Vacuum and Recirculation System

c) Gas Supply Unit

The gas mixture used is LaserCut 34156. The ratio of the gas mixture is CO_2 3.4%, N_2 15.6%, and the remaining is He.

d) Vacuum Pump

A rotary vacuum pump (E2M2, Edwards, UK), with a pumping capacity of $5.6m^3h^1$ is used in the system. The main purposes of the vacuum pump are:

i. To evacuate the laser chamber to a low pressure level before filling with the CO_2 gas mixture.

ii. To reduce the gas contamination species which are generated from chemical reactions during electrical discharge by removing a small amount of the used gas to the atmosphere. Equal amount of fresh gas mixture is then injected into the system.

iii. To ensure the pressure of the gear chamber of the root blower is lower then the laser chamber. This is important to prevent oil from the gear chamber from entering the laser chamber.

e) Root Pump

A root pump (Hick Hargreaves MB53, Figure 3.1.6) with a displacement capacity of $2580m^3h^{-1}$ at 50Hz flow rate is used to recirculate the gas

medium between the discharge channel and the heat exchangers. This type of root pump is a positive displacement unit by using two balanced impellers. One of the impeller is motor driven. A pair of timing gears in the root pump casing transfer drive to the other impeller. These two balanced impellers rotate in opposite directions and do not touch each other as well as the casing but are separated a few tenths of millimeter apart. Gas is drawn in at the suction connection, trapped between each impeller and the casing. Then it will be ejected to the discharge channel after passing through another heat exchanger. In order to protect the root pump from overheating, a variable frequency converter (ABB SAMI GS) is used. Meanwhile, this converter is used to control the rotating speed of the root pump motor by varying the AC supply current to the motor. When the gas temperature rises above the upper limit, the root pump will be switched off by the frequency converter automatically.

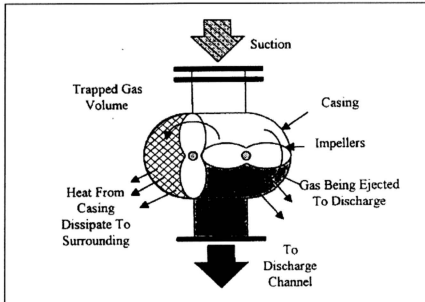

Figure 3.1.6: The Root Pump

3.1.4 Gas Cooling System

Two heat exchangers are used for cooling the gas. The heat exchangers are made of two mild steel containers of volume around $0.08m^3$ each. One of the heat exchanger is used for cooling the gas mixture from the discharge tube before entering the root pump, while the other heat exchanger is used for removing the heat generated by compression of the root pump. Both of them are cooled by flowing chilled water from a water chiller at a temperature below 10^0C. The minimum temperature of the gas that can be cooled down was around 20^0C for this system without discharge of the gas. The gas circulation and cooling system is shown in Figure 3.1.7.

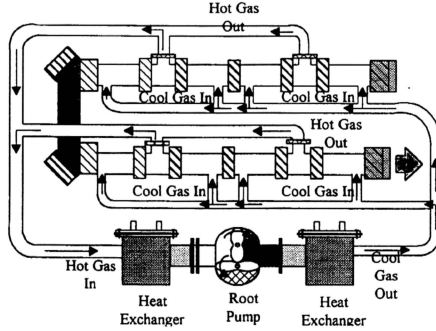

Figure 3.1.7: The Gas circulation and cooling system

3.2 Beam Delivery System

The beam delivery system used is a 3-mirror system [58, 59, 60]. The laser beam is bent 90^0 three times before focussing to the workpiece as shown in Figure 3.2.1. The first mirror is a quarterwave reflective phase retarder aligned 45^0 to the output coupler. It is used to convert the S-polarised beam to a circular polarised beam. Then the beam is bent twice through two zero phase total reflectors before entering the focusing lens. The focussing lens is a ZnSe positive meniscus lens with a effective focal length of 5.0 inches. The diameter of the lens is 1.5 inch with a thickness of 0.12 inch. It can stand a maximum pressure of 5.5 bar. The focusing lens is mounted in a laser cutting head and is water-cooled to reduce the thermal lensing effect, which in turn will add to the aberrations of focusing power.

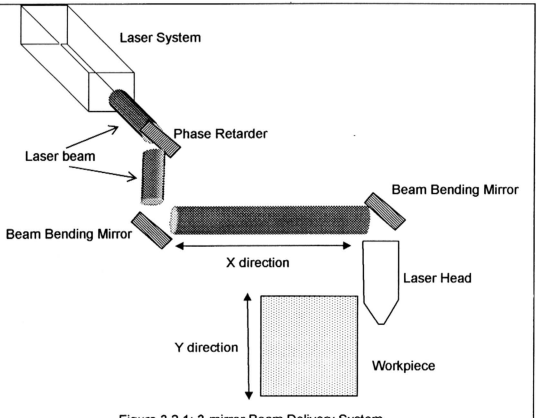

Figure 3.2.1: 3-mirror Beam Delivery System

3.3 Motion Control System

3.3.1 XY Table

The existing hybrid positioning system is used for the 2 axis positioning. A hybrid system is a solution to reduce the complications of a moving mirror system. The moving laser beam is taken arbitrarily as the X-axis while the moving table is taken as the Y-axis. A combination of these two orthogonal axes generates any desired contour. The maximum cutting area is 550mm*550mm. The design of the XY table used is shown in Figure 3.2.2.

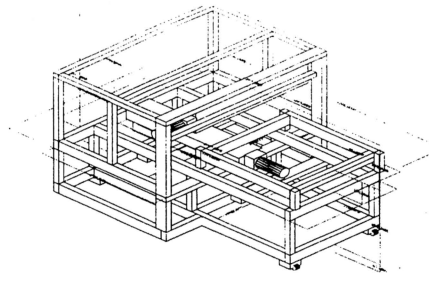

Figure 3.2.2: The XY-Table

*3.3.2 **Controller Card***

The existing PC based positioning controller, the NextMovePC controller [61] is used to control the servo motor. Some important features of the card are:

- Floating point *Digital Signal Processor* at the core of the NextMovePC controller which is capable of executing 33 million floating point instructions per second.
- Four axes of servo control with 12 bit analog outputs and incremental encoder feedback; 500 to 1000 microsecond servo loop update rate.
- 16 bit ISA bus interface via 2kB dual port RAM.
- 512kB Volatile RAM.
- Incremental encoder feedback, with ability to latch encoder position within 30 microsecond.
- 24 inputs, 12 outputs optoisolated PNP or NPN (jumper configured). Software configurable as limit inputs, home inputs, enable outputs for amplifiers or general purpose I/O.
- Eight 12 bit analog inputs, jumper configurable as 4 differential inputs. Readable as 0-5V, +/- 2.5V, or +/- 10V.

Figure 3.2.3 shows how the NextMove PC controller card is connected to the computer (host).

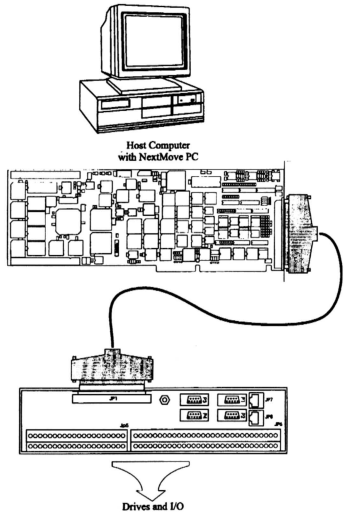

Figure 3.2.3: The Host Computer with NextMove PC controller card

3.3.3 *Servo Motor Driver*

The function of DC driver is to provide a variable current into the motor in order to control the torque. A single axis four quadrant pulse width modulation drive (Max 500 Electrocraft) for permanent DC brush motors has been used. The control inputs and outputs are optically isolated. It has high performance velocity or torque controller accepting either DC tachometer feedback or can generate velocity data from a two channel position encoder. Figure 3.2.4 shows how the Servo Motor Driver is connected to the controller card.

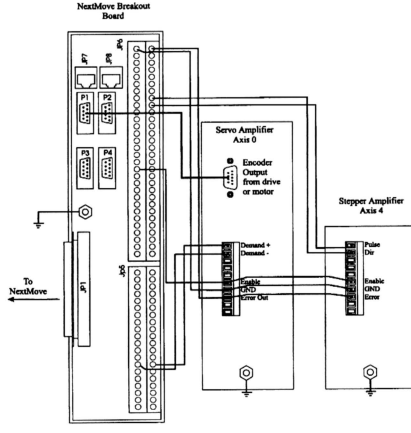

Figure 3.2.4: The Motor Driver connect with controller card

3.3.4 Motor, Lead Screw and Encoder

Two existing DC Servo Motors are used, one in driving the X-axis and the other the Y-axis. Servo motor is ideal for a wide variety of high performance applications. They offer rapid acceleration, fast response, and extremely smooth travel.

One end of the motor is attached to the lead screw. Lead screw convert rotational motion into linear motion. It has the advantage of accuracy in fast positioning. The lead screw is 1000mm long.

The other end of the motor is attached with an encoder. An encoder is an electromechanical device used to monitor the motion or position of an operating mechanism and to translate that information into a useful output. The output can take the form of a simple system status indication or it can provide feedback information or in other ways interface with related devices. An incremental encoder with 4000 counts per revolution is used.

3.4 Development of 2D CAD/CAM Program

Overview

CAD/CAM has been acknowledged as the key to improving manufacturing productivity and the best approach for meeting the recent critical design requirement [62]. Since the existing laser cutting system is only meant for 2 dimensions, a simple 2D CAD/CAM is sufficient for controlling the XY-table. The developed software includes some basic functions like Click and Move feature, Virtual Material Positioning, real time Board Status Retrieval, Speed Control Motion, and Database System. The program is ready for linear, arc, circle, and corner cutting experiment.

3.4.1 Visual Basic 6 Programming Language

Visual Basic [63, 64] is a Microsoft Windows programming language. Visual Basic programs are created in an *Integrated Development Environment* (*IDE*). The *IDE* allows the programmer to create, run and debug Visual Basic programs conveniently. *IDEs* allow a programmer to create working programs in a fraction of the time that it would normally take to code programming without using *IDEs*. The process of rapidly creating an application is typically referred to as *Rapid Application Development* (*RAD*). Visual Basic is the worlds' most widely used *RAD* language. The existing program is written using Microsoft Visual Basic Learning Edition. It provides fundamental programming capabilities but cannot be compiled to native or machine code.

1. Visual Basic Coordinate Systems

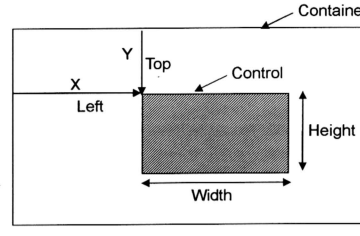

Figure 3.4.1: The visual basic coordinate system

Figure 3.4.1 shows the definition of the default Visual Basic Coordinate System. The X coordinate represents the distance from the left edge of the drawing area to the point, and the Y coordinate represents the distance from the top edge of the drawing area to the point. The upper left corner will have the coordinate value of (0, 0). By default, Visual Basic uses the vbTwips ScaleMode. In order to use the custom coordinate system we should convert the default coordinate system to suit our application by setting the ScaleMode to vbUser. Then use the keyword ScaleLeft, ScaleWidth, ScaleTop and ScaleHeight to change the value.

2. *Graphic Methods for Objects*

There are 3 kinds of graphic methods [65] used extensively throughout the project, namely Line, Circle and Pset Method.

a) Line Method

The Line method is used to draw line segments using the following syntax:

Line Step (*x1, y1*) – step (*x2, y2*), *color*, BF

Step	If the first Step keyword is included, the coordinates of the line's starting point are relative to the end of the last drawing command.
(*x1, y1*)	The coordinate of the point at which the line should start. If you omit this point, the line starts where the previous drawing command ended.
Step	If the second Step keyword is included, the coordinates of the line's ending point are relative to the line's starting point.
(*x2, y2*)	The coordinate of the point at which the line should end.
Color	The color the line should have
B	If the B flag is included, the Line method draws a box with corners at the start and end points.
F	If the F flag is included the Line command fills the box with corners at the start and end points, using the same color used to draw the box. In this case B flag must also be included.

b) Circle Method

The circle method draws circles, ellipses, and sections of circles (arc) and ellipses. It uses the syntax:

Circle Step (*x, y*), *radius, color, start, end, aspect*

Step	If Step keyword is included, the X and Y coordinates of the circle's center are relative to the end of your last drawing command.
(x, y)	The coordinates where you want the center of the circle.
Radius	The radius if the circle.
Color	The color of the circle
start, end	The starting and ending angles for drawing an arc or pie slice.
Aspect	The aspect ratio (height divided by width) for drawing ellipses.

The angles used by the Circle method are measured counterclockwise in radians where a horizontal line to the right lies along the angle 0. Figure 4.4.2 shows how Visual Basic measures angles.

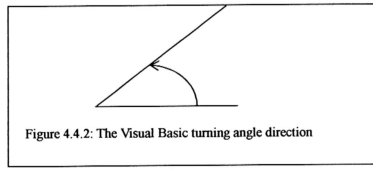

Figure 4.4.2: The Visual Basic turning angle direction

The Circle method also requires that its angle lie between -2π and 2π. If specifying angle outside this range, Visual Basic generates an error. Note that the difference between drawing a sector and an arc is the minus sign character, -. The minus sign does not correspond to negative angles which are not supported by Visual Basic.

c) Pset Method

The Pset method set the color of a single pixel at a specific position. Pset has this syntax:

Pset Step (x, y), color

Step	If Step keyword is included, the X and Y coordinates are relative to the end of the last drawing command.
x, y	The coordinates of the point you want to color.
Color	The color for the point.

3. Techniques of Drawing in Visual Basic

a) Make a draw permanent part of the picture

Sometimes it is necessary to make for example a Line or a Point to be the permanent part of the drawing object so that the Cls command does not affect its existence. To do so, the following code is applied:

picFrom.Picture = picFrom.Image

where picFrom is the drawing object.

b) Draw a Rubber-band Line

In order to draw a rubber-band line, the DrawMode property should be changed to vbInvert. Drawing a same line twice in vbInvert erases the line, so this erases the previous line between those points. Finally draw the line in vbCopyPen to make the line normal again. The same applies to the Circle method.

3.4.2 MINT Programming Language

MINT is the programming language [66] used to program the controller in order to meet the requirements of specific applications. MINT provides control of the I/O and motion control aspects of the controller using Basic-like programming structures and keywords. MINT programs consist of two files. The *Configuration Buffer* stores information relating to the machine setup, for instance the servo loop gains. The *Program Buffer* stores the actual motion control program. In fact the two files are the same and can contain the same instructions, except that the configuration buffer is only 8K maximum size, while the *Program Buffer* can be up to 100K. In addition, there is a third method of storing information in the controller, in the form of array data, which can be used as variables within the program.

1. The Configuration File

The *Configuration Buffer* is used to store appropriate defaults for a particular system. Once gains and speeds have been found, these should be incorporated into a *Configuration Buffer* for the system.

Whenever the program is RUN, the *Configuration File* is executed first. The following provides a list of parameters necessary for setup:

- Servo loop gains – to tune the system response.
- Scale factor – to set the units of measure of the application.
- Maximum following error (MFOLERR) – to set a safe maximum difference between the actual and the desired positions.

- Default speeds (SPEED), accelerations (ACCEL) and decelerations (DECEL) for the system – to determine the shape of the trapezoidal velocity profile (RAMP).
- Error features such as hardware and software limits (FWLIMITIN, FWSOFTLIMIT, REVLIMITIN, REVSOFTLIMIT), stop inputs (STOPINPUT) and error inputs (ERRORIN).

These parameter are generally to be setup only once for an application, but can at any time be altered in the *Program buffer*.

2. *The Program File*

The MINT program is written in much the same way as in other programming languages. It can contain Arrays, Functions, Subroutines, Loops and Conditional Statement. MINT program accepts most of the relational and mathematical operators. The syntax and keyword for motion control can be found on the reference manual.

3. Creating Motion

a) Scale Factor

Before any motion is created it is important to have a correct scale factor to allow each axis to be scaled for ease of use, it is a multiplicative factor that is applied to all motion variables for that axis (speed, acceleration, move distances, etc.). In order to scale, one must know the encoder line number.

b) Establishing a Datum

The HOME keyword is used to establish a datum position on the axis. This function is useful for all applications that require a consistent zero point from which moves are referenced. The home position can be established on the basis of the opening of contacts on a home switch, or on the first index pulse of the encoder after opening of the switch. The first method is quicker but less accurate than the second. However, the second method requires a three channel encoder, with a separate marker (index) pulse. In our system, the first method is used.

c) Motion Profiles

To move motors in a controlled way, it is necessary to understand the basic principles behind motion. The motion of moving objects can be specified by a number of parameters, which together define the Motion Profile. It is usual to define acceleration, deceleration and move time. To

make it easier to understand motion profiles, it is usually a good idea to represent them in a graphical format. This can be achieved by plotting a motion property like position or speed against time. The most useful of these graphs is velocity against time.

Most move types are executed using a commonly referred to as a trapezoidal profile [67] (Figure 3.4.3), because of the shape of the speed profile. It has constant acceleration and deceleration.

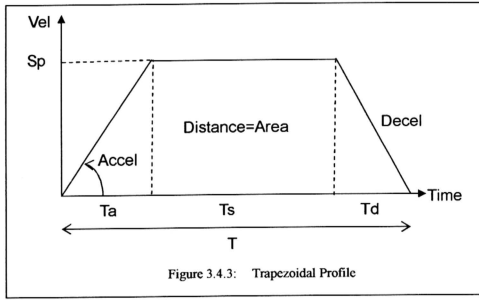

Figure 3.4.3: Trapezoidal Profile

From this graph all of the relevant properties can be clearly identified.

- T is the total move time, Ta is the acceleration time, Ts is the slew speed time and Td is the deceleration time.
- Sp is the value of slew speed.
- The rate of acceleration is the gradient of the line during the acceleration part of the profile and can be calculated using the formula

  Accel = Sp / Ta.

- Similarly the rate of deceleration is defined as the slope of the speed line during the deceleration part of the graph, Decel = Speed / Td.
- The distance moved is defined as the area under the speed line.

Knowing this information, it is possible to use simple calculations to determine any information that is not known to enable us to define a move. We could use the standard equations of motion for uniform acceleration, like:

$$v = u + at$$

$$s = ut + 1/2\ at^2$$

$$v^2 = u^2 + 2as$$

where u = initial speed, v = final speed, a = acceleration and t = time.

d) Interpolated Moves

In interpolated moves, the axis speed and acceleration refer not to individual axis speed, but to the path along which motion is in progress. Interpolated moves are positional moves and so use the trapezoidal motion profiles. Two kinds of interpolated moves are essential, namely the Linear Interpolation and the Circular Interpolation.

Linear interpolation provide interpolated motion on two or more axes using the keyword VECTORR, a relative vector move and VECTORA, an absolute vector move. Circular interpolation provide circular interpolated move on two orthogonal axes, for instance the XY table using the keyword CIRCLER, a relative move, and CIRCLEA, an absolute move.

e) Contoured Moves

The keyword CONTON (Contouring On) allows a series of interpolated moves to be executed without decelerating at the end of each positional move. Thus, the path is traced out at a constant rate of motion. The contouring routines maintain a constant vector speed throughout the profile, deceleration will not occur until MINT detects that the final vector has been reached. This gives rise to the following constraints:

- Contouring should only be attempted around smooth shapes. Corner should be rounded, a right angled cornered will jar the machine since one axis must instantaneously stop and the other instantaneously start.
- The first axis specified is taken as the master axis, its speed, acceleration and deceleration being used for the path. The same master axis should be used throughout a series of contoured moves.
- In order to maintain a constant path speed, the total move distance remaining in the move buffer should be greater than the distance required to decelerate using the current DECEL rate. If the move distance in the buffer is less than the distance required to decelerate to a halt, the axis will start to slow down.
- If the move buffer is of insufficient size or new moves are not loaded frequently enough, then the path speed will be seen to vary. The move buffer size can be changed with the MOVEBUFFER keyword. The size of the move buffer to set depends on the vector lengths being used.

- It is recommended that vector should be greater than 4 to 6 milliseconds in length. When a linear or interpolated move is being executed, the target speed of travel is taken from the SPEED keyword. If a number of moves are loaded into the move buffer and the SPEED parameter is changed, that change takes place immediately and affects all further moves in the buffer.
- With FEEDRATE, the target speed is attached to each move individually. This means that moves of varying speed can be loaded and the change in speed will take place when that move is executed. FEEDRATEOVERRIDE allows the slew speed of all moves in the buffer to be changed together.

3.4.3 MINT Interface Library

The MINT Interface Library [68] is a common API (Application Program Interface) for the range of MINT based controllers. The MINT Interface Library is DOS and Windows based with support for Windows 3.11, Windows 95, and Windows NT via a dynamic link library (DLL). A number of development platforms are supported including Visual C++, Borland C++, Borland Delphi, and Visual Basic. Features of the library include:

- Ability to upload and download MINT programs and configuration files.
- Ability to interrogate the MINT command line.
- Updating of new firmware into FLASH or RAM.
- Support for the MINT Comms Protocol, whereby data can be transferred to an executing MINT program by means of a protected datapacket.
- Ability to read Dual Port RAM locations on the NextMovePC controller.

The MINT Interface Library is supplied as a standard edition and as a developer edition. In development of the CO_2 Laser Cutting Program the developer edition is used. The most important feature for the developer edition is the Immediate Command Mode (ICM) functionality for NextMovePC. This is the mechanism for controlling motion directly from the host. For example the host calculates 2D vectors and sends the demand coordinates to NextMovePC as a series of multi-axis vectors in real time, while the MINT program handles I/O.

1. *Communicating with NextMovePC*

NextMovePC controller is referenced in the MINT Interface Library by a handler. This handle allows the library to reference all the information it requires on the controller. To create the handlers we use *createNextMovePCHandle* function.

2. *Dual Port RAM (DPR)*

All communication between NextMove and the host is performed using Dual Port RAM (DPR). This is a physical block of memory on the NextMove which can be accessed by either NextMove or the host. Various locations in DPR have been set aside for special purposes such as sending control codes and passing I/O information. Other locations have been left for the user to pass any required information back and forth. The main features and uses of DPR are:

- Support for loader routines. This allows NextMove applications to be loaded into NextMove.
- Support for the MINT Comms Array. This is a method of asynchronously updating variables in a MINT program from the host.
- MINT pseudo serial buffer. This allows communication with the MINT command line, and MINT program and configuration loading/saving.
- Reporting of MINT status. The host can read whether MINT is at the command line and if not, which line is executing.

- Automatic reporting of motion variables. In every 2 miliseconds, NextMove writes various motion parameters into DPR – the speed, position, gain of each axis, the state of I/O, and errors, if any. This can be read at any time by the host. The host can also use this area to update NextMove motion variables e.g. speed or gains.
- Control of the NextMove by the host (known as Immediate Command Mode (ICM)). This allows a program running on the host to access any NextMove function.
- Interrupts. Writing to DPR location 3FF hex on NextMove causes an interrupt to occur on the host. This can either be used to trap motion errors or to cause user interrupts.
- Flags and control registers. Each NextMove application uses control registers to tell the host which features it supports. Control registers can also be used to synchronise communications between NextMove and the host.
- User area. There is an area in DPR which has been left to allow NextMove and the host application to exchange whatever application specific data is required.

3. *Loader / Initialisation Routines*

NextMove contains a monitor program held on EPROM called the loader which is always resident. The loader is used to download applications (executable files) from the host into the memory of NextMove. The loader also performs

various memory checks at power up. Routines are available allowing communications with NextMove at the lowest level, before any application has been loaded. They include routines to read the version information from NextMove, and all the routines required to load and run an application.

4. *MINT Comms Array*

The Comms array is a 99 element MINT array which is mapped to DPR. This allows floating point data to be transferred between a MINT program and a host. This method of communication is the best way of communicating data between a MINT program running on NextMove and a host at run time. It can be used for simple data transfer, or as a method of synchronizing events.

5. *MINT Pseudo-Serial Buffer*

67 words of DPR are defined as a MINT pseudo-serial receive buffer, and 67 as aMINT pseudo-serial transmit buffer. Both of these are implemented as a circular buffer with 3 control words, leaving 64 words available for transferring characters (one per DPR location). The pseudo-serial buffer provides an interface to the MINT command line and is also used in uploading and downloading MINT files. Pseudo serial buffer is used to upload and download MINT configuration and programs.

6. *Auto Update Area*

Certain areas of DPR are used by NextMove to report the current status of various motion variables. These locations are updated every 2ms by NextMove's MINT Motion Library (MML). The host can read values directly from DPR using the functions *getWord*, *getLong*, *getFloat*. It is also possible to pass values for some of the motion variables to NextMove.

7. *Immediate Command Mode Interface*

The Immediate Command Mode (ICM) is a method of allowing the host to send motion commands directly to NextMove, bypassing MINT. It allows direct access to all the commands available either in a MINT program or NextMove application. It is particularly useful if there is a large amount of processing to do (calculation of multi-axis paths) as the host can do the processing and send the commands to NextMove. Note that these functions can be used in conjunction with a MINT program. For example, a MINT program handles the I/O and the host calculates the path and sends it to NextMove using *setVECTORA*().

8. *MINT Interface Library and Visual Basic*

To access the MINT Interface Library from Visual Basic module MIL_DEV.bas is included in this project and make sure that the file MIL_DEV.dll is found in the Windows directory. The motion commands can be found in the NextMove MINT Motion Library [69].