THE UNIVERSITY OF LIVERPOOL

SENSOR BASED AUTOMATIC CONTROL SYSTEM FOR NARROW GAP TIG WELDING

by

XIAO-QI CHEN

Thesis submitted in accordance with the requirements of the University of Liverpool for the Degree of Doctor in Philosophy.

Department of Electrical Engineering and Electronics

May 1989

ACKNOWLEDGEMENTS

I am very grateful to my supervisor Professor J. Lucas for his advice and continuing support throughout my research. Without his supervision and encouragement, this work is impossible. I wish to thank Dr. A. Parker for his support during my thesis preparation. I am indebted to a number of individuals who assisted in my work. In particular, I wish to thank Dr. B. J. Corlett for helpful discussions and supplying some pieces of programs, and Mr. D. Harvey and Dr. W. Lucas in the Welding Institute for their technical advice and assistance in preparing experimental tests.

My thanks are also extended to all staff in the department for their enthusiastic support throughout my study. In addition, I express my appreciation to the British Council and the State Education Commission of China for their sponsorship of my research activities in the United Kingdom.

After all, my special thanks are due to my wife Qing-Fang for her love, support, and excellent typing.

To my parents and my wife

ABSTRACT

A 16-bit single board computer has been developed to control TIG arc oscillation in narrow gap welding. Sidewall penetration is improved by magnetically weaving the arc across the gap. The custom-built computer can work either as a stand-alone arc oscillation controller or in communication with a host IBM PC. The voltage sensor resident on the computer board supplies information about arc length and arc position in the gap which are related to the arc voltage. Simple close-loop arc oscillation control and seam tracking are therefore possible. A robust vision system has been developed for advanced seam tracking and arc control in narrow gap TIG welding. The welding scene is observed by a TV camera without artificial illumination. Information on positions of sidewalls, electrode, and welding arc is extracted in real time by analysing grey level images. Edge finding and template matching techniques have been developed to accomplish this task. Real-time seam tracking and arc oscillation control are implemented based on the vision information.

CONTENTS

1. INTRODUCTION

Welding is a joining process which has been long associated with manufacturing industries. It can be split into two main groups: non-fusion welding and fusion welding. A non-fusion welding process joints two parts together without melting them. A typical example is spot welding. Two contact electrodes hold plates to be welded. A current is conducted from one electrode to another, generating a resistance heating to heat the contact area between the two parts. A sufficient pressure is applied to press them together. Spot welding features simple heating process, no filling materials, and easy automation procedures. Therefore it has found wide application in the car industry.

In the area of fusion welding, the common processes are Submerged-Arc Welding (SAW), Flux-Cored-Arc Welding (FCAW), Metal-Inert-Gas (MIG) welding, and Tungsten-Inert-Gas (TIG) welding, as shown in figure-1.1. The common feature of arc welding is that an arc is struck between a consumable or non-consumable electrode and a workpiece. The workpiece and filling materials are melted by the arc energy to form a joint. SAW uses a metal wire as an electrode. The wire is continuously fed through to balance its melting rate. The arc is buried by a layer of flux, thus invisible to outside. The melted flux covers the molten weld to protect it from the atmosphere. The unmelted flux can be collected and re-used. SAW has a high deposition rate, and produces an excellent bead shape. However, its high heat input is detrimental to the joint properties. In some cases, flux removal becomes difficult. Generally, SAW is used in non-critical joints. Similarly, FCAW uses a filler wire as an

electrode. But the flux is enclosed in the core of the wire. MIG welding also employs a filler wire as an electrode. But the arc is shielded by an inert gas. As compared with SAW, MIG welding considerably improves weld quality in addition to its high efficiency. Unfortunately, it often has spattering and arc blow problems. The cost and complexity of MIG equipment also limits its application. This type of welding is mainly used in the situations requiring high productivity and reasonable weld quality. In contrast, TIG welding employs a non-consumable electrode (usually tungsten), resulting in a stable arc shielded by an inert gas. A filler wire is fed into the weld pool, independent of the welding power supply. TIG is a clean process without spattering, and produces high quality welds, and is therefore used in critical weld joints.

For thin plates, single pass is sufficient to join them together. However, for thick sections, multi-pass or multi-layer layout is inevitably employed. Applications of thick plate or pipe welding can be found in shipbuilding industry, power generation plants and pressure vessels. Conventionally, heavy sections are welded together with a large single or double bevel preparation such as single V, double V, single U, double U (figure-1.2) [1]. Other preparations include single J, double J, and asymmetric double V. The welding procedure starts from the root pass, then gradually builds up from one side (for single-bevel preparations) or from both sides simultaneously (for double-bevel preparations). This is known as multi-pass layout. The number of passes per layer increases as the gap becomes wider from the root to the top. The thicker the plates, the more passes are needed to fill the gap. In a multi-layer arrangement, the torch moves across the preparation, forming one pass per layer. It is rarely used

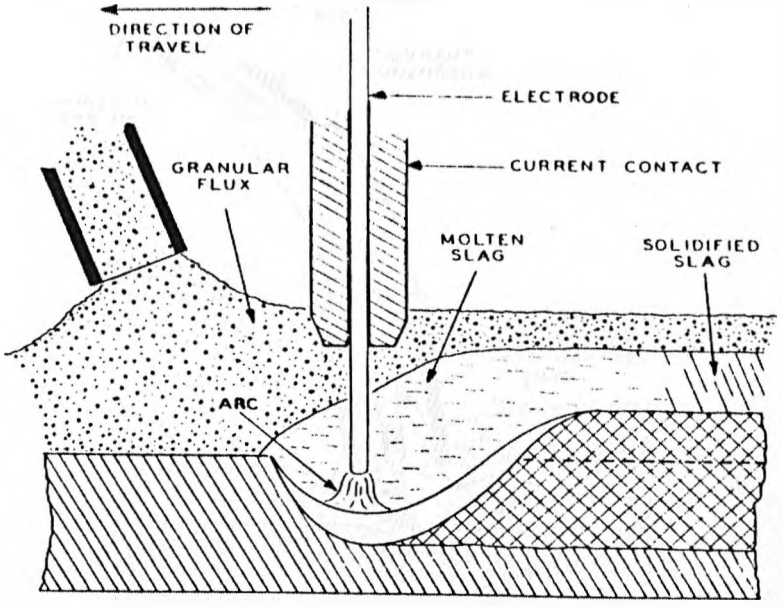
in conventional thick section welding because swinging the arc across a large gap has detrimental effects on the joint properties. Large bevel preparations enable the welding torch to readily access the sidewall, resulting in good penetration. The major disadvantages of conventional technique include low productivity and high cost.

In recent years, many researchers have investigated narrow gap welding (NGW) techniques which are regarded as the most promising processes to carry out thick section welding. In contrast to conventional thick section welding, NGW usually requires a narrow square preparation as shown in figure-1.3 [2], consequently reducing joint volumes and overall cost. For 100 mm section thickness, double V-groove preparation needs 2000 mm^3/mm weld length, while NGW requires only 800 mm^3/mm weld length with a saving of 60% welding consumables [3]. In addition to higher productivity, NGW improves mechanical properties of weld joints and reduces residual stress and distortion because of the narrower heating area and low heat input. However, NGW has the problem of poor sidewall penetration due to the small angle between the electrode and the sidewall. To overcome this problem, a number of NGW techniques has been developed [2, 3]. Common methods involve directing the arc toward the sidewall by some means or using large electrode and high heat input.

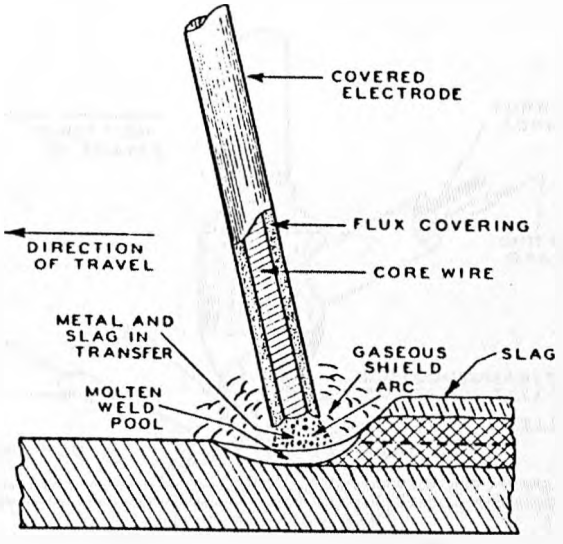
As far as real-time adaptive welding control is concerned, sensors are regarded as a crucial element for the success of intelligent and automatic welding control systems. Of particular interest is machine vision. It extracts visual information about welding conditions and feeds it back to the welding controller so that real-time control schemes such as seam tracking and welding parameters control can be imple-

mented. Most vision systems currently available to welding processes employ artificial illumination to improve the contrast in the welding scene or to provide profile information. As for image recognition, binary image processing is widely used due to its simplicity. However, a binary image contains much less information than a grey level image, and it is unlikely to provide sufficient information to implement complex control. The approaches to intelligent control lie in real-time analysing of grey level image.

The work discussed in this thesis is concerned with a sensor based real-time control system for narrow gap TIG welding. Firstly, narrow gap welding techniques and welding sensors are reviewed. Secondly, the TIG welding system is explained. The computer controlled arc oscillation is then discussed. It is followed by real-time control using the vision sensor. Finally, experimental results, and conclusions and future work are presented in later part of the thesis.

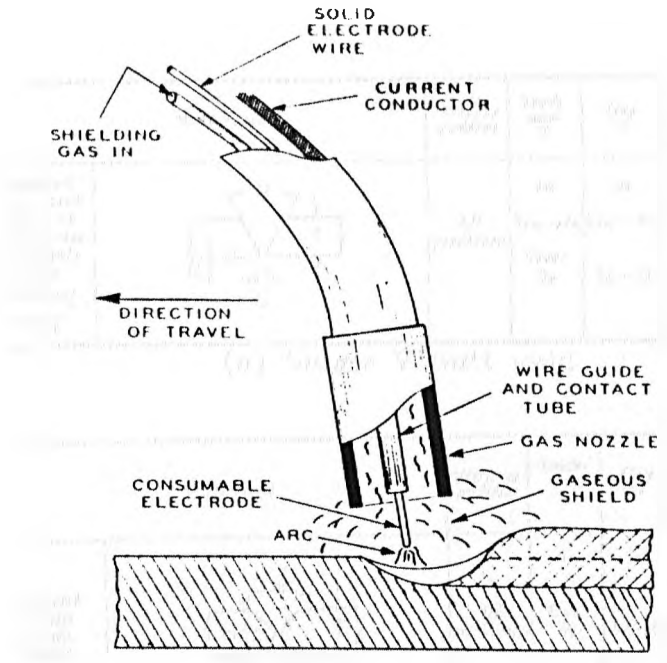


(a) Diagrammatic sketch of submerged arc welding

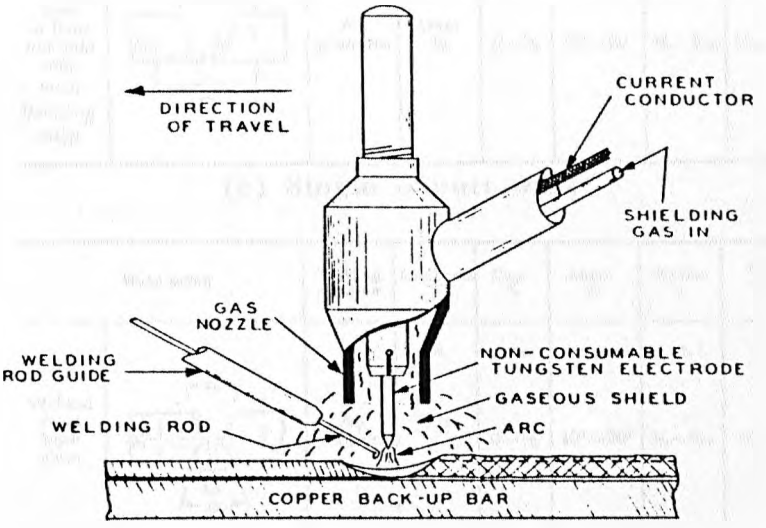


(b) Diagrammatic sketch of flux cored arc welding

Figure-1.1 Common arc welding processes

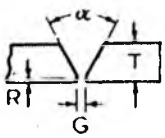


(c) Diagrammatic sketch of metal inert gas welding

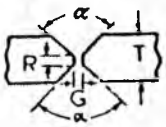


(d) Diagrammatic sketch of tungsten inert gas welding

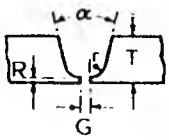
Figure-1.1 Common arc welding processes (cont.)

Weld detail		Welding position	Thick-ness T	Gap G	Angle α (min.)	Root face R
Welded from both sides or from one side only with backing strip		All positions	in. 3/16—3/8 Over 3/8	in. 1/16—3/16 1/8—1/4	60° 60°	in. 0—1/16 0—1/8

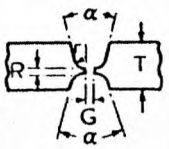
(a) Single V butt weld

Weld detail		Welding position	Thick-ness T	Gap G	Angle α (min.)	Root face R
Welded from both sides		All positions	in. Over 1/2	in. 1/8—1/4	60°	m. 0—1/8

(b) Double V butt weld

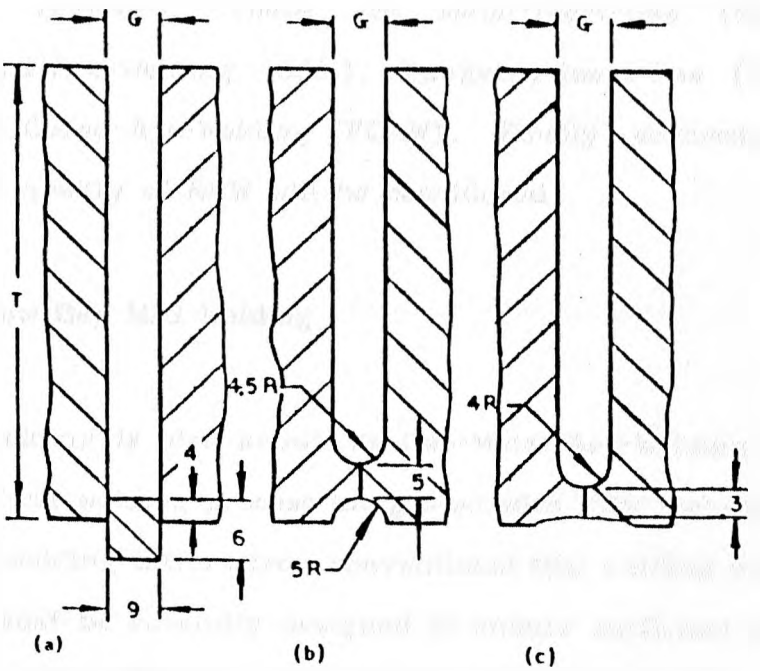
Weld detail		Welding position	Thickness T	Gap G	Angle α	Radius r	Root face R
Welded from both sides or from one side only with backing strip		All positions	in. Over 1/2	in. 0—1/8	10°—20°	in. 3/8—3/16	in. 1/16—3/16

(c) Single U butt weld

Weld detail		Welding position	Thickness T	Gap G	Angle α	Radius r	Root face R
Welded from both sides		All positions	in. Over 1	in. 0—1/8	10°—20°	in. 3/8—3/16	in. 1/16—1/8

(d) Double U butt weld

Figure-1.2 Conventional edge preparations for butt welds



T	G
≤ 30	10
30 < T ≤ 75	11
75 < T ≤ 150	13
150 <	15

Figure-1.3 Typical narrow gap welding preparations

2. NARROW GAP WELDING TECHNIQUES

2.1 Introduction

In this chapter, the primary advantages and problems of NGW will be discussed, and four major welding processes associated with NGW will be reviewed. These are Metal-Inert-Gas (MIG) welding, Submerged-Arc-Welding (SAW), Tungsten-Inert-Gas (TIG) welding, and Flux-Cored-Arc-Welding (FCAW). Finally, economical advantages and joint quality of NGW will be considered.

2.2 Narrow Gap MIG Welding

MIG welding is also known as Gas-Metal-Arc-Welding (GMAW). It was the first welding process integrated with NGW techniques. Narrow gap MIG welding differs from conventional MIG welding in that the wire feeding must be carefully designed to ensure sufficient sidewall penetration. Up to date, a variety of wire feeding techniques have been developed to control sidewall penetration of NGW [4 - 8]. Generally, these techniques fall into four main groups: curved fixed wire, arc oscillation, arc rotation and straight fixed wire. All these techniques are reviewed in narrow gap MIG welding.

Curved Fixed Wire. This technique normally employs a thin electrode wire ranging from 0.8 mm to 1.6 mm and a long contact tube. The wire is curved toward the sidewall to achieve sidewall penetration. There are two approaches to curve the wire: precast fixed wire and bent fixed torch. The former method was developed by Battelle [4], as

shown in figure-2.1. Double-wire arrangement and bi-pass bead layout are adopted. The wires are bent around the drive roll in the direction perpendicular to the sidewall before entering the contact tube, but still preserve curvature after exiting the contact tube. The main advantage of this technique is that a very narrow gap (6.3 - 9.5 mm) can be used regardless of the plate thickness. This allows the use of low heat input and much less filler material. However low heat input may lead to possible incomplete sidewall penetration. Another problem is the difficulty of maintaining consistent bending and orientation of the electrode wires because the thin curved wires have to pass a long contact tube.

The technique "Bent Fixed Torch" was invented by Sciaky SA [4]. It differs from the above technique in that the electrode wire is bent toward the sidewall in a special contact tube with an angled tip (figure-2.2). Because the wire is bent near the arc, more repeatable wire curvature can be achieved in comparison with the Battelle NGW. By increasing the gap to 14 - 20 mm two torches can be used, directed at opposite sidewalls. To further increase the joint filling rate, a third electrode wire is added in the centre of the groove to bridge two filler welds performed by other two torches. It is possible to produce monopass, bi-pass and tri-pass deposition layout with up to three electrode wires.

Arc Oscillation. In this group of NGW techniques, the sidewall penetration is achieved through oscillation of the arc across the gap. The arc is oscillated by either swinging torch, or alternate rotation of contact tube with a bent tip, or prebending electrode. The "swing torch" technique employs a straight wire and a long straight contact

tube. The torch swings across the gap horizontally. Nakayama et al [9] describe two arc weaving methods: NOW-B and NOW-HB. (figure-2.3). In NOW-B process the arc weaving follows a general sine curve motion perpendicular to the welding direction, while NOW-HB has a weaving angle (α) of 30° - 60° to the welding direction. A single electrode and monopass are required. Nippon Steel [10] developed a NGW technique involving a swivelling torch. The oscillation of the arc is realised by swivelling a straight contact tube inside the groove, as shown in figure-2.4. The sidewall penetration is assured by sufficient angle between the electrode and the sidewall. To accommodate sufficient swivelling, the gap is wider than that in the Battele technique and ranges from 12 mm to 14 mm, and the plate thickness is limited to 80 mm. Monopass bead layout is provided by one or two electrodes.

The wire feeding technique involving alternate rotation of a special contact tube with a bent tip was developed by Hitachi [4]. The tube tip is bent by 15°. When the contact tube rotates back and forth, the feeding wire describes an arc across the gap. A similar technique is presented by Halmøy et al in [11] (figure-2.5). For the small angles involved in this technique, the weaving amplitude of the arc is proportional to the angle of rotation of the contact tube. A monopass bead layout is achieved with a single electrode.

NGW techniques based on prebending electrode wire have been developed by a number of manufacturers. Nippon Steel [4] developed the "loop nap" technique, as shown in figure-2.6. The wire is prebent by rollers on the loop panel which sets a curvature plane for the prebent wire. On exiting from the straight contact tube, the wire is curved just in the plane of the loop panel. Swinging the loop panel

(up to 70°) periodically produces the arc oscillation. Monopass and a single electrode are typically employed. Mitsubishi Heavy Industries developed "Zig-Zag Prebent Wire" technique [4]. The wire is periodically deformed between two bending gears (figure-2.7). The wire exiting from the contact tube preserves the curvature of the original wire configuration. As the wire is fed through the tube, the arc oscillates across the gap. By changing the shape of the deforming gears, the wire curvature and the arc oscillation pattern (oscillation frequency, width and dwell time) can be controlled. A monopass, single electrode and flat position are preferred for this technique. The main problem in this technique is the dependence of the wire configuration on the groove width. A variation of this technique is "Sinusoidally Prebent Wire" developed by Babcock-Hitachi K.K. [4]. The wire is plastically and periodically deformed into a sinusoidal shape by a special oscillation panel (figure-2.8). The oscillation frequency and amplitude control the sidewall penetration. Iversen and Palussek [12] describe a narrow gap MIG welding system based on "Sinusoidally Prebent Wire" to produce circumferential welds of nickel alloy. It is claimed that the welds are free from defects with wall thickness up to 126 mm.

Arc Rotation. This group of techniques is similar to those of arc oscillation, except the sidewall penetration is controlled by arc rotation. Most NGW techniques based on arc rotation involve deforming the electrode wire. In the Hitachi-Zosen NGW technique [4], the wire is plastically deformed by a bending roller which is rotated about the wire axis and exits the contact tube with a spiral shape. Kobe Steel [4] used "Twist Arc" to rotate the arc, as shown in figure-2.9. Arc rotation is obtained by feeding a special twist electrode through a straight contact tube without special rotating or bending devices. The

twist electrode consists of two intertwined wires of the same diameter. When fed into the groove, the arcs at the tips of the two wires define a continuous arc rotation.

An innovative technique was developed by Nippon Kokan [4]. Arc rotation is achieved by feeding a 1.2 mm straight wire through an eccentric guiding hole in a straight contact tube rotated at a extremely high speed up to 7200 rpm about the contact tube axis (figure-2.10). The sidewall penetration is controlled by adjusting the radius of arc rotation and the rotating speed of the contact tube. Monopass, single electrode and a flat position are standard for this technique.

Straight Fixed Electrode. The main features of this group of techniques are: a relative thick electrode wire ranging from 1.6 mm to 5 mm, a long and straight electrode extension inserted in the centre of the groove, and high heat input. The contact tube and the nozzle are usually above the workpiece, and the electrode is stiff enough to be inserted into the groove without a sophisticated contact tube with water cooling and electrical insulation. A monopass bead layout, a single electrode arrangement and a flat position are recommended. High deposition rates up to 10.8 Kg/hr are possible due to high heat input. The control of sidewall penetration is achieved through manipulating welding parameters rather than electrode wire. It is best illustrated in the technique developed by Linde [4], as shown in figure-2.11. The sidewall penetration is controlled by using a straight polarity DC power supply. The electrode wire ranges from 2.0 to 3.2 mm in diameter and requires a groove about 12.5 mm wide. The Mitsubishi M.N.N. technique [4] employs a pulsed arc and reverse polarity DC. A spray transfer is achieved instead of globular transfer which is typical for

the straight polarity Linde technique. E.O. Paton Electric Welding Institute [4] used a short pulsed arc with high current peak (900 A) and reverse polarity DC for welding high alloy steels. Groups of pulses with frequency 25 Hz are superimposed on the background current (350 A), making the arc column pulsate and improving sidewall fusion.

Narrow gap MIG welding becomes attractive due to its high welding efficiency, high weld quality and cost effectiveness. However, its application is limited by sensitivity to sidewall defects, spattering, arc blow, and cost and complexity of the equipment. It is mainly applied in welding fabrication requiring high productivity but reasonable weld quality.

2.3 Narrow Gap SAW

SAW has been long associated with welding of thick plates, particularly in pressure-vessel fabrication. However, it was not until the early 1980's narrow gap SAW was intensively investigated. In contrast to narrow gap MIG welding, electrode feeding techniques used in most narrow gap SAW are limited to "curved fixed wire" and "straight fixed wire" [13, 14]. The former technique controls sidewall penetration by directing the electrode toward the sidewall and/or high heat input. The latter one achieves sidewall penetration through a large electrode (from 2 mm to 5 mm) and high heat input.

Bead deposition layout selection in narrow gap SAW depends on many factors, but flux removal is one of the most important considerations. Monopass layout is used only with self-detachable fluxes specially de-

veloped for easy removal of slag from a narrow groove. It has much higher joint filling rate, but has the disadvantages of using non-standard fluxes, accurate electrode positioning, strict limitation on gap variations, and sensitivity to fluctuations of welding conditions and to solidification cracking. Monopass layout is widely used in Japan, while outside Japan multipass layout is more popular. The latter brings about lower joint filling rate, but it is more flexible, reliable and less sensitive to weld defects. More importantly, multipass layout allows the use of standard, or slightly modified fluxes and a conventional SAW welding procedure.

In Japan, multi-electrode arrangement is widely used. It dramatically increases the deposition rate, but produce excessive heat input. It also produces a large weld pool and prolongs metal-slag interaction which makes slag detachment even more difficult. For these reasons, multi-electrode arrangement is not recommended for root layers and for grooves narrower than 13-14 mm. In fact, a single-electrode arrangement is preferred outside Japan.

Narrow gap SAW has the advantages which conventional SAW possesses:

(1) Producing welds with good bead shapes, no spattering.

(2) Controlling the mechanical properties of the weld by selection of suitable fluxes.

(3) High deposition rate.

However, the high heat input employed by SAW is detrimental to weld quality. Flux detachability also poses a problem. In general, narrow gap SAW is used in the situations requiring high productivity but not critical weld quality.

2.4 Narrow Gap TIG Welding

TIG welding is also called Gas-Tungsten-Arc-Welding (GTAW). It uses a tungsten (non-consumable) electrode, while the filler wire is fed into the weld pool separately. Therefore weld pool formation and wire feeding can be independently controlled. Sidewall penetration in narrow gap TIG welding is achieved by either arc oscillation or arc rotation, as shown in figure-2.12.

(a) A straight tungsten electrode is vertically inserted into the groove, and a monopass layout is achieved through the lateral oscillation of the electrode.

(b) A straight tungsten electrode is inserted into the groove at a slight angle (2.0 - 3.5°) to the sidewall. A bi-pass bead layout is performed.

(c) The arc is rotated through an alternate rotation of the tungsten electrode with a tip bent at 20 - 25°, resulting in a monopass bead layout. Alternatively, the bent tip of the tungsten electrode is directed toward the sidewall to achieve sidewall penetration. Multipass bead layout is therefore provided [15].

Grinin and Shtrikman [16] describe an automatic narrow gap TIG welding system employing two electrodes. Both electrode tips are bent toward the sidewall to control sidewall penetration. The filler wire used for TIG is usually 1.6 - 2.5 mm in diameter. The wire is fed into the weld pool either from the rear side of the arc at an entry angle 50 - 75° to horizontal or from the front of the arc at an entry angle 15° to horizontal. For the dual-electrode arrangement, the wire is fed between the two electrodes.

In recent years, there is increasing interest in hot wire narrow gap TIG welding [17 - 19]. This is largely due to the higher deposition rate of this process in comparison with cold wire TIG welding. As show in figure-2.13, the filler wire is subject to resistive heating which provides approximately 80 - 90% of the energy necessary to melt the wire. To establish a stable arc, the filler wire must be fed at a speed balancing the melting rate. To maximise the penetration, some schools [20, 21] propose a combination of laser (typically a few kilo watts) and TIG or MIG welding (figure-2.14). It has been reported [20] that a combined process narrows the arc root and avoids undercutting even at high speed.

2.5 Narrow Gap FCAW

FCAW utilises a flux-cored wire without shielding gas. As compared with MIG welding, FCAW has certain advantages: smoother metal transfer and stable arc, better bead shape, higher deposition rate, simpler equipment, and higher tolerance to electrode guiding accuracy. Like SAW, it can adjust weld properties by selection of cored flux. Another attractive feature of FCAW is elimination of gas shielding. Significant advances have been achieved in recent years, and improved consumables have emerged providing fabricators with new options for FCAW applications traditionally carried out using MIG welding [22]. Despite the above merits and progress, FCAW is less used for NGW application because of the presence of flux in the wire core. Slag removal and risk of hydrogen content are potential problems associated with flux-cored wire. Furthermore, fabrication of this type of wire is difficult and expensive.

Narrow gap FCAW normally employs "straight fixed wire" technique. Sidewall penetration is provided through manipulation of welding parameters. Oerlicon [4] claims that an extremely high deposition rate 14.5 Kg/hr is achieved with high current (700 A) and tandem arrangement of two flux-cored electrode wire. In 1975 the Engineering Laboratory of Canadian General Electric Co. [4] developed a narrow gap FCAW technique utilising a DC straight polarity at high voltage (34 - 42 V) to ensure sidewall penetration. High current (440 - 600 A) provided a high deposition rate, but might cause arc blow. To avoid the problem of arc blow, Sumitomo Metal Industries [4] has developed a new technique which employs AC power supply. A special flux-cored wire is designed to produce a very thin film of slag on the weld metal surface. It is said that slag can be easily removed in plates up to 120 mm thick.

2.6 Assessment of NGW

Advantages of NGW has been briefly mentioned in chapter 1 and in the above discussions of NGW techniques. The remainder of this chapter assesses NGW from two general aspects: economical advantages and joint quality.

2.6.1 Economical Advantages of NGW

Compared with conventional welding, NGW shows attractive economics. The thicker the weldment is, the more attractive the NGW application. The basic economical features of NGW are as follows:

(1) Joint Preparation. A square joint preparation is usually required for NGW. This type of groove simplifies plate cutting and grinding of the cut surfaces before welding. One cut and grinding of two surfaces

are sufficient. However for double-V groove preparation of SAW, five cuts and grinding of four surfaces are necessary.

(2) Joint Filling Rate. Joint filling rate (JFR) is a criteria to assess welding productivity. It can be defined as a joint length per unit time:

$$JFR = D / (A \times G) \qquad \text{(equation-2.1)}$$

where D stands for deposition rate, A for groove cross-section area, and G for gravity of the weld metal. The influence of plate thickness, deposition rate and welding technique on JFR has been investigated by Butler et al [23]. Figure-2.15 shows joint filling rates for NGW welds at two different deposition rates. Comparison of JFR between NGW and conventional welding is illustrated in figure-2.16. The JFR of NGW can be as high as that of twin wire electroslag welding, and much higher than that of SAW and electrogas.

(3) Consumables Consumption. Reduction in consumables is directly due to narrow gap preparation. It may play a dominant part in reducing overall fabrication costs in some NGW cases.

(4) Electric Energy Consumption. NGW requires less energy because of smaller weld beads, lower heat input and more liberal requirement to preheat and postweld heat treatment.

(5) Heat Treatment. NGW produces improved fine microstructure in both weld and heat affected zone (HAZ). Consequently, postweld heat treatment (PWHT) may be unnecessary or reduced.

2.6.2 Joint Quality of NGW

NGW has superior joint quality due to favourable effects produced by the combination of specific features inherent to NGW.

(1) Distortion and Residual Stress. NGW employs lower heat input, smaller weld pool and heating area, with the results of less distortion and residual stress.

(2) Tensile Strength and Ductility of NGW Joints. Tensile strength of the welds is usually high enough to meet all requirements. Joint ductility is sufficiently high due to favourable fine microstructure.

(3) Fracture Toughness of NGW Joints. The fracture toughness of a NGW joint is considerably improved. This is particularly true of the fracture toughness of the heat affected zone which has always been a problem in conventional welding of high strength low alloy (HSLA) steels utilised in pressure vessel fabrication. The improvement of fracture toughness of NGW joints is convincingly demonstrated in figure-2.17 [24].

(4) Hydrogen Distribution. Excessive content of diffusible hydrogen in welds may lead to fatal hydrogen cracking. Desirably, the hydrogen content in NGW welds is greatly reduced. Figure-2.18 compares the distribution of diffusible hydrogen content in weld metals for narrow gap MIG and conventional SAW.

In this chapter, four major NGW processes have been discussed. The advantages and problems of NGW have been presented. The principal difficulty with NGW is sidewall penetration control. Either excessive or incomplete sidewall fusion is unacceptable. It is therefore essential to have close-loop feedback control, requiring sensors for supplying feedback information. The next chapter will review different types of sensors for welding control.

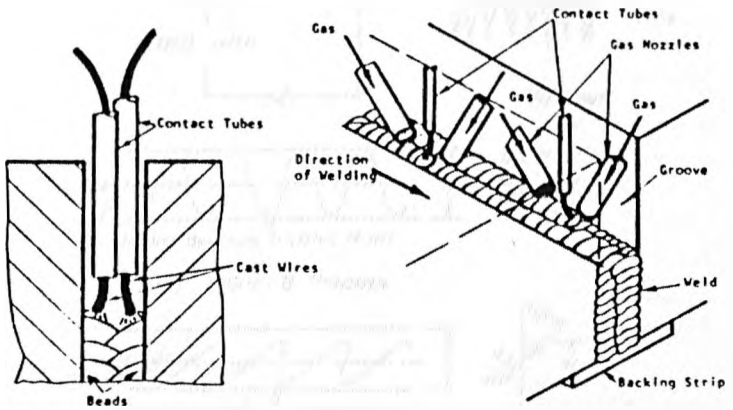


Figure-2.1 Principle of the Battelle (precast fixed wire) technique

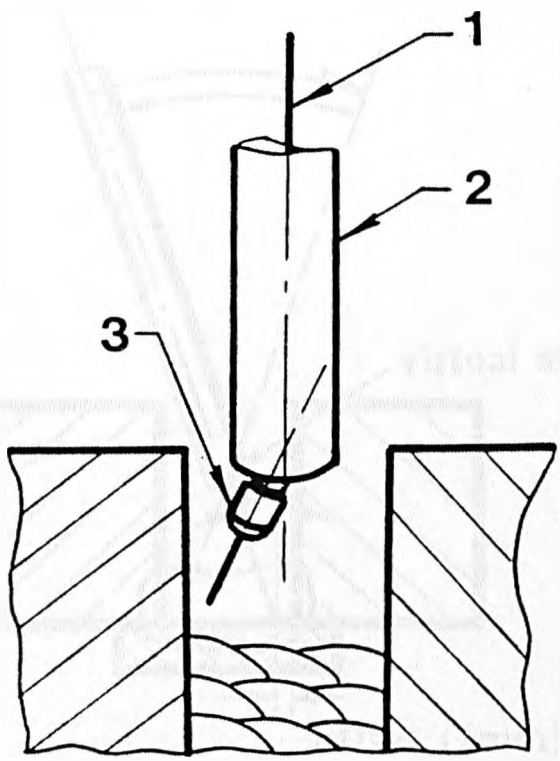


Figure-2.2 Sciaky SA (bent fixed torch) technique. 1, wire; 2, contact tube; 3. contact tip

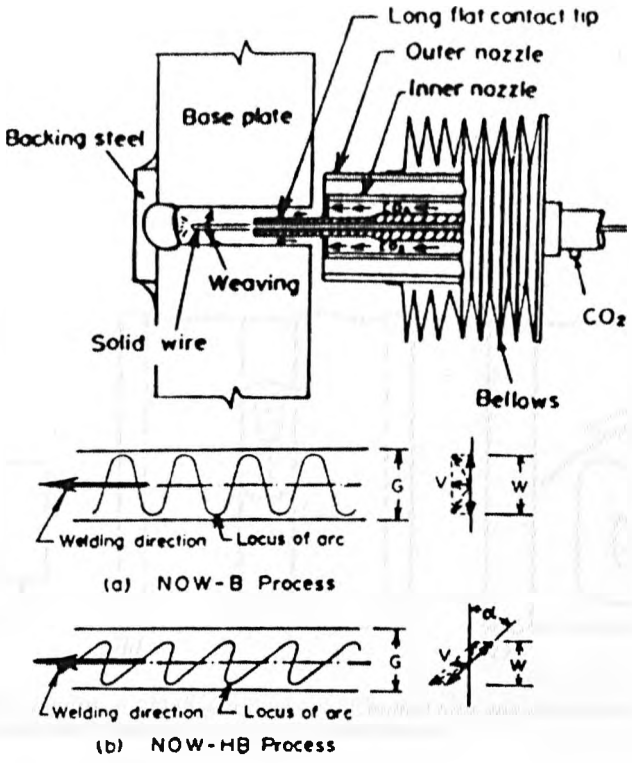


Figure-2.3 Principle of "NOW" (swing torch) technique

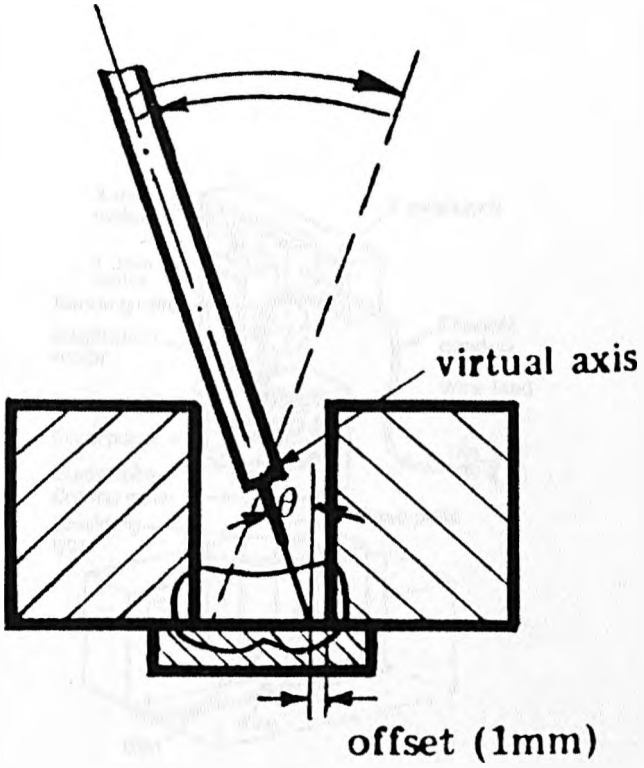
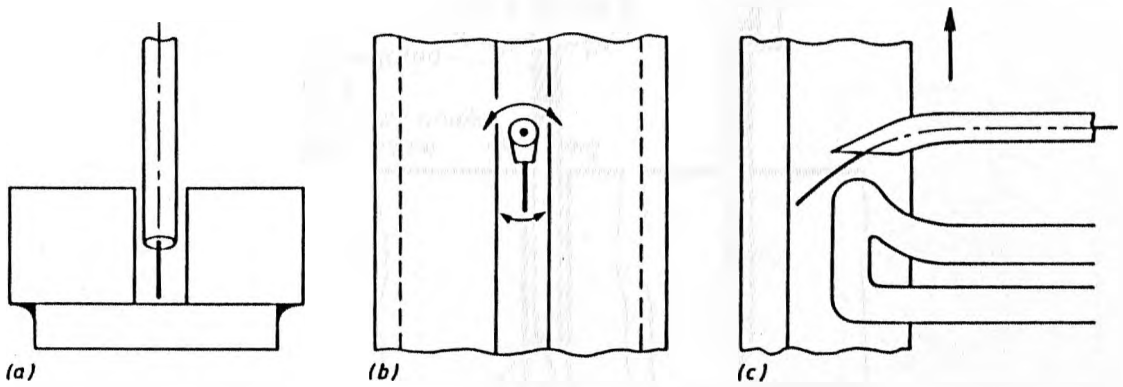


Figure-2.4 Principle of Nippon Steel (swivelling torch) technique



1 MIG narrow gap test weld configuration: (a) end view of workpiece, contact tube, and electrode, (b) top view indicating oscillating motion, and (c) side view including copper bend weld pool support

Figure-2.5 Principle of "bent rotating torch" technique

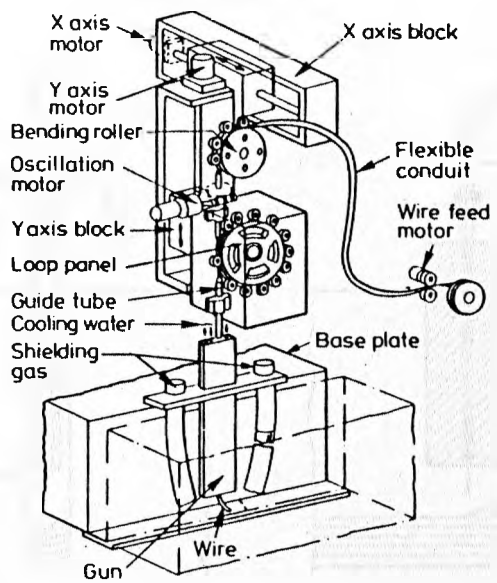


Figure-2.6 Principle of loop nap technique

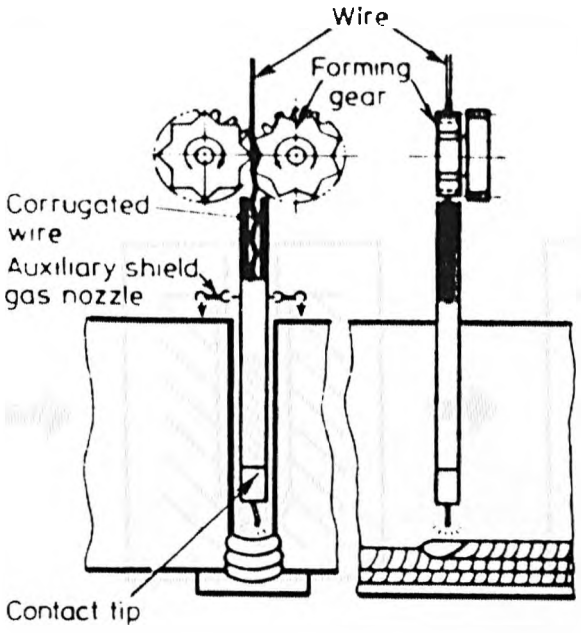


Figure-2.7 Principle of zig-zag prebent wire technique

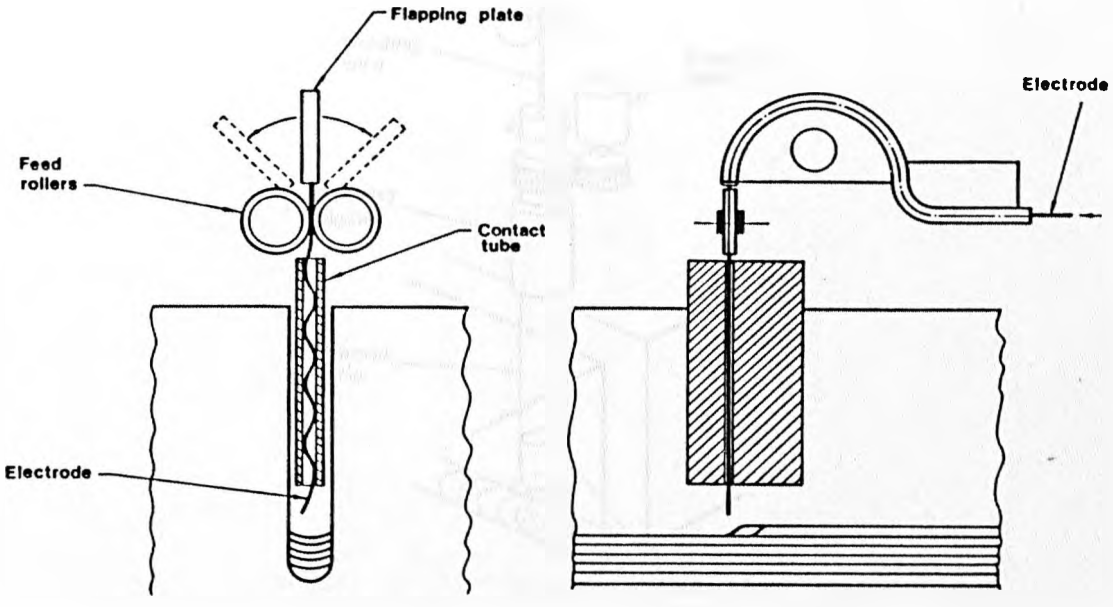


Figure-2.8 Principle of "sinusoidally prebent wire" technique

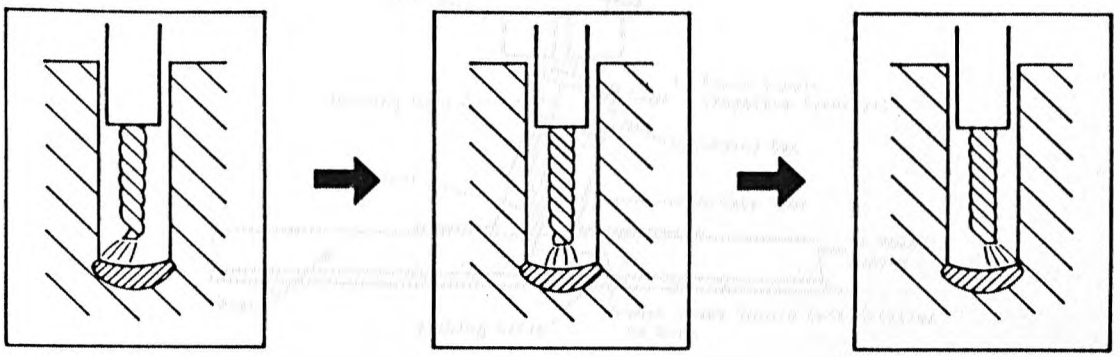


Figure-2.9 Principle of twist arc technique

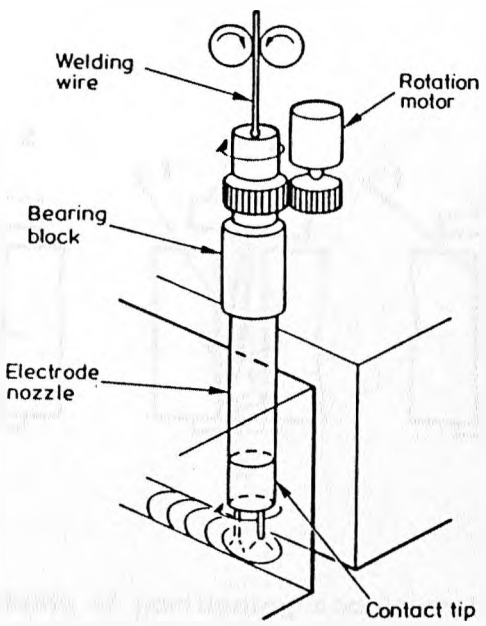


Figure-2.10 Rotation of an eccentrically bored wire guide tube

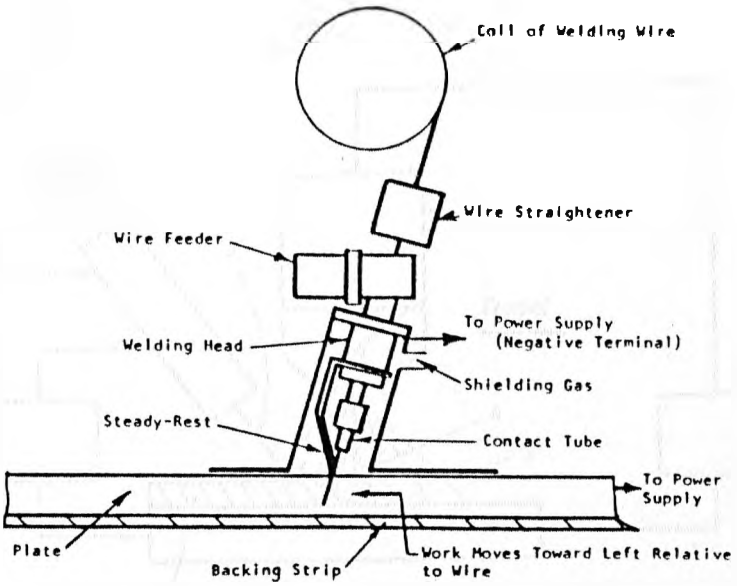


Figure-2.11 Straight fixed wire technique

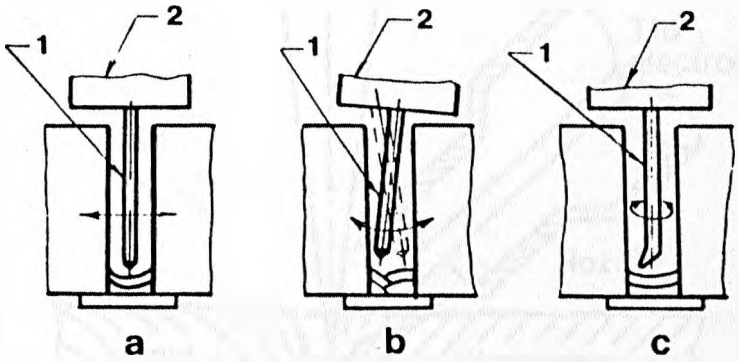


Figure-2.12 Methods of positioning and lateral oscillation of tungsten electrode in the groove in narrow gap TIG welding. a, vertical electrode; b, inclined electrode; c, electrode with a bent tip; 1, tungsten electrode; 2, shielding gas nozzle

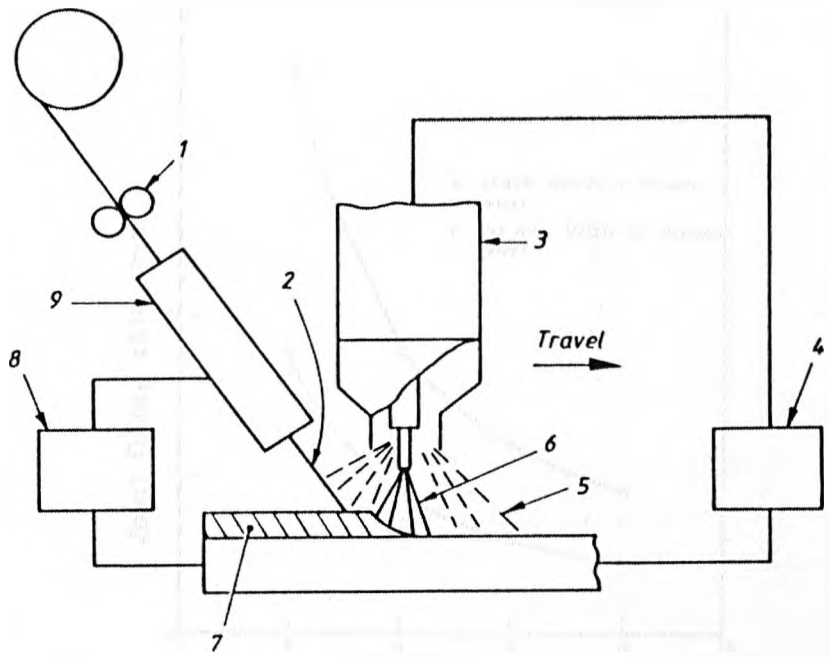


Figure-2.13 Principle of hot wire TIG welding process.
1, wire feeder; 2, hot wire; 3, torch; 4, DC power; 5, gas shield; 6, arc; 7, weld; 8, AC hot wire power; 9, contact tube

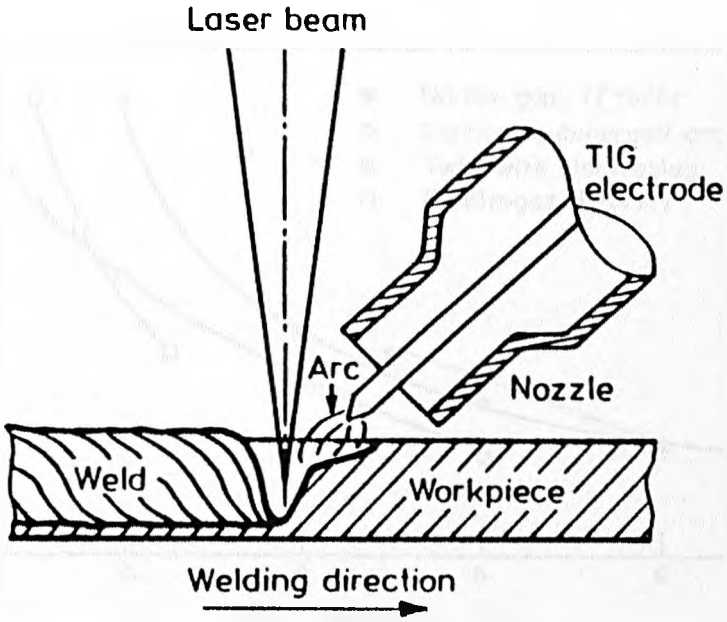


Figure-2.14 Principle of laser TIG welding

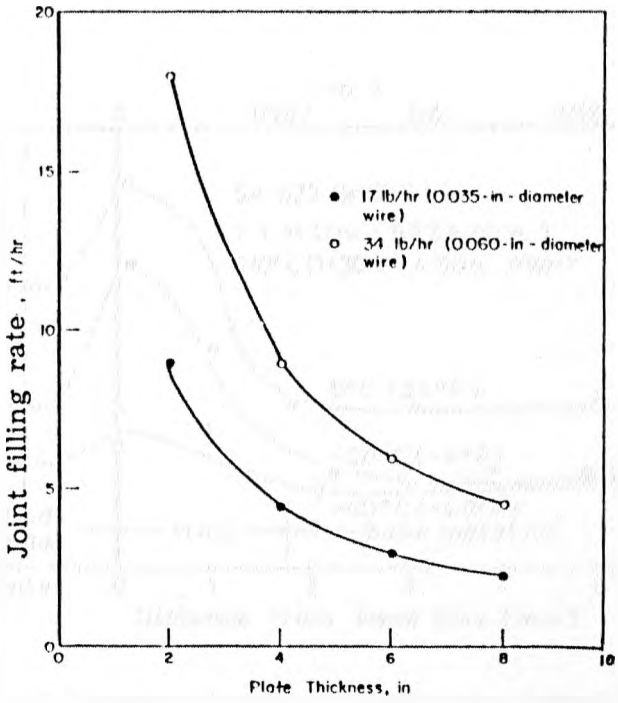


Figure-2.15 Joint filling rates for narrow gap welding at two different deposition rates

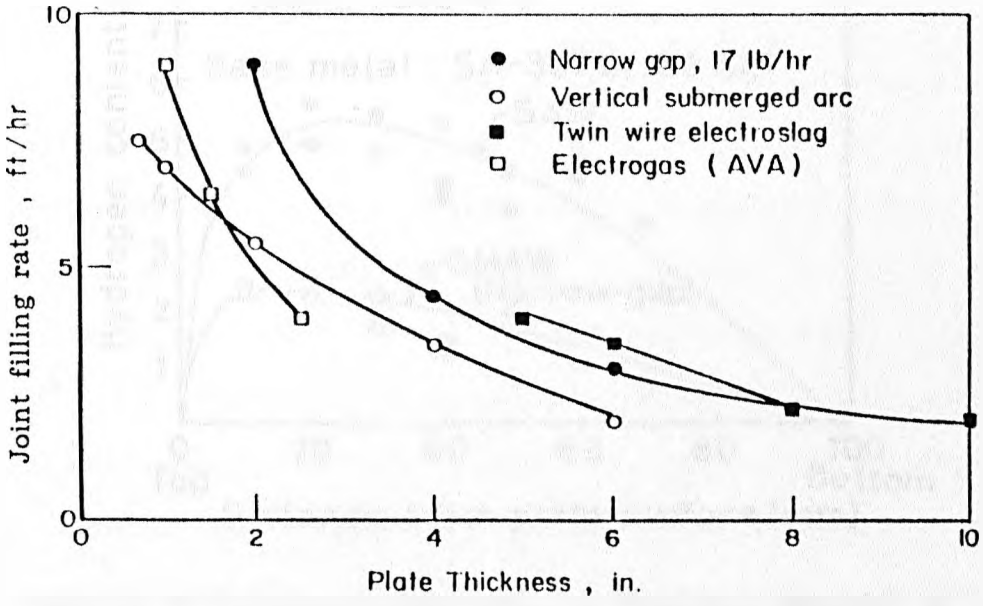


Figure-2.16 Comparison of joint filling rate between NGW and conventional welding

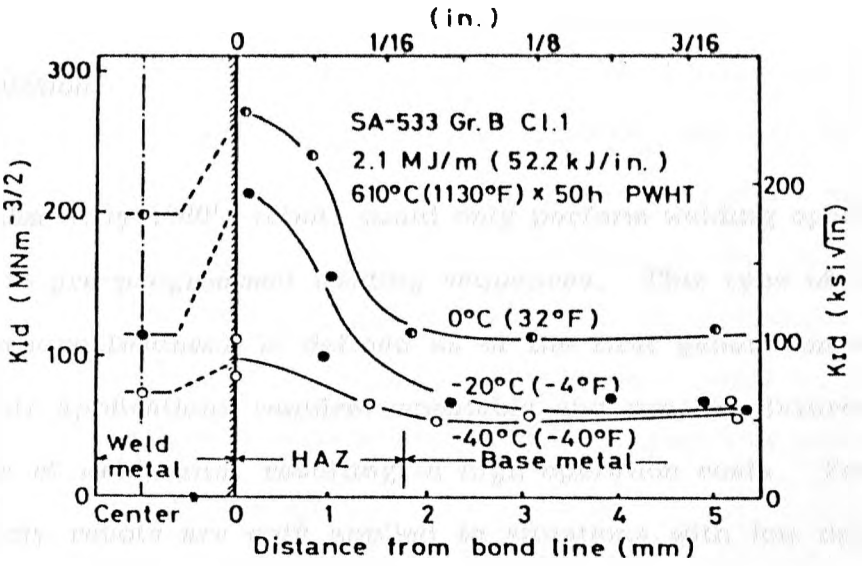


Figure-2.17 Dynamic fracture toughness distribution of narrow gap welded joint

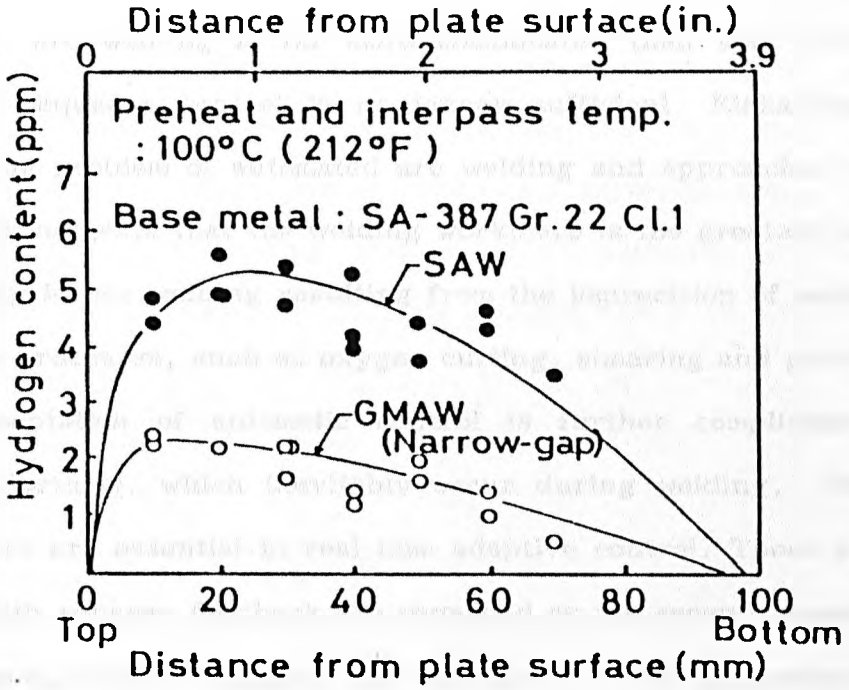


Figure-2.18 Hydrogen distribution in thickness direction of 100 mm thick weld. Elapsed time start to finish of welding narrow gap process-3.5h; submerged arc-8h

3. SENSORS FOR WELDING CONTROL

3.1 Introduction

Before the early 1980's robots could only perform welding operations according to pre-programmed welding sequences. This type of robots without sensory feedback is defined as of the first generation of robots. Their applications require repeatable and precise fixtures and preparation of weldments, resulting in high operation costs. For this reason, early robots are only applied to situations with low demands of precise positioning. A typical application is spot welding robots which account for a large population of the robots market.

However, arc welding is far more complicated than spot welding, and simple sequence control is no longer sufficient. Richardson [25] considers the problem of automated arc welding and approaches to its solution. He suggests that the welding workpiece is the greatest source of variability in arc welding resulting from the imprecision of weldment preparation processes, such as oxygen cutting, shearing and grinding. The implementation of automatic control is further complicated by welding distortions, which inevitably occur during welding. Therefore, sensors are essential to real-time adaptive control. Those robots equipped with sensory feedback are regarded as the second generation robots. Some [26] even suggest the concept of third generation arc welding robotics which hinges on the development of sophisticated sensing systems, capable of seam tracking, penetration control and bead shape control.

In general, the automation problems associated arc welding can be divided into two main areas: seam tracking and penetration control. To meet these objectives, a variety of sensing systems have been developed. Hanright [27] reviews arc welding adaptive control and classifies welding sensors into three main categories: through-the-arc, preview and direct arc sensors. Other classifications can be made in relation to the sensor devices employed. This chapter intends to overview major sensing systems available at the present time: through-the-arc, laser optical, and direct vision sensors. Other sensing systems such as tactile and ultrasonic sensors are presented in the later part of the chapter.

3.2 Through-the-Arc Sensing

Through-the-arc sensing techniques are based on the relationship between the arc signals (i.e. arc voltage and arc current) and the arc length. Depending on which arc signal is acquired, they are defined as current sensor or voltage sensor. Welding control based on current sensor is often called arc current control (ACC). Likewise, arc voltage control (AVC) is referred to as control systems based on the voltage sensor.

3.2.1 Current Sensor

Current sensors are normally used in consumable arc welding processes, typically Metal-Inert-Gas (MIG) welding. This is associated with the characteristic of a MIG power supply. Usually a MIG power supply has a constant or flat characteristic, as show in figure-3.1. Any small change in the arc length results in a minor change in welding

voltage, but a marked change in welding current. This is the basis of current sensor. Figure-3.2 shows the current waveform as the arc oscillates in a groove. With the current information simple control algorithms can be designed to carry out automatic seam tracking. Munezane [28] proposes the method of using peak currents of the current wave form. The difference ($\triangle$I) between the peak current at left groove side and that at right groove side is used as the tracking error. The torch is adaptively moved to the groove centre once the error occurs. The method used by Kobe Steel [28] is a little more complicated. First the difference between the left peak current and background current and that between the right peak current and background current are computed. Then the difference between them is used as the tracking error. The joint is tracked when the error is zero. Laing [29] describes a through-the-arc sensor utilising the multiplication of arc current and arc voltage for narrow gap MIG welding. Since the penetration is related to the welding energy (V $\times$ I), sidewall penetration control can be achieved in addition to seam tracking.

3.2.2 Voltage Sensor

In contrast to the current sensor, the voltage sensor samples voltage signals during arc oscillation. It has found applications in both consumable and non-consumable arc welding. The principle is based on the relationship between the arc voltage and the arc length (figure-3.3) [30]. For the arc gap from 1 mm to 10 mm, the arc voltage (V) is proportional to the arc gap (la) at a constant current:

$$V = \eta_1 + \eta_2 la \qquad \text{(equation-3.1)}$$

where η_1 and η_2 are coefficients depending on the welding current. This equation indicates that the arc length can be detected by sampling the arc voltage, which forms the foundation of voltage sensor.

A number of tracking systems based on voltage sensor has been developed. Cook et al [31] patented a weld tracking system, which measures the voltage drop between the electrode and the sidewall of the groove as the arc approaches the sidewall. Fujimura et al [32, 33] have applied through-arc-sensing for both joint tracking and weave amplitude control. Essers [34] describes arc control with pulsed GMA welding by measuring the arc voltage during the pulse period in each cycle as the control reference. Measurements and control of the arc length are made in real time. An arc-controlled seam tracking system with a different arc oscillation is proposed by Eichhorn and Gompel [35]. The torch is mechanically oscillated around a centre of rotation rather than moving transversely, as shown in figure-3.4.

A novel development of arc sensing was achieved by Madigan et al [36]. It employs natural oscillation of the weld pool to control weld penetration. By periodically pulsing the welding current, an impulse of arc pressure is generated, which excites the molten pool into natural oscillation at a frequency inversely proportional to the pool size. This results in changes in the arc length, hence the arc voltage. Information about the pool size and the pool oscillation frequency is obtained by analysing the voltage waveform. The full-penetration oscillation is thought to have longer wave length, therefore lower frequency. It is claimed that when the current is pulsed at 15 to 25 Hz partial-penetration weld pools oscillate at 150 to 400 Hz and full-penetration weld pools at 38 to 188 Hz.

Hollinggum [37] describes a direct arc sensing system for weld quality assessment developed by Philpott. It involves using the radio frequency (RF) component in the arc voltage waveform. Changes in welding conditions are reflected in the RF signals. In particular, a breakdown in the gas shielding of the arc produces a marked increase in amplitude of the RF component. Variations in the wire feed are also detected. Insufficient wire feed leads arc breakdown, and a drop in the RF component is sensed.

3.2.3 Features of Through-the-Arc Sensing

Through-the-arc sensors have following advantages:

(1) Simple electronic hardware. An arc signal sampler and a computer controller are sufficient to perform the control work.

(2) Simple control algorithms. The control error is computed from the arc signal waveform. Linear control schemes are generally applicable.

(3) Cheap to use. No specialised skill is needed to operate arc sensing systems. They work rather reliably with stable welding conditions.

However, through-the-arc sensors are limited in use because of the complexity of arc welding phenomena. There are many factors influencing the arc stability, which makes sampling of arc signals more difficult. These factors include:

(1) Welding power supply and welding position.

(2) Electrode conditions such as electrode stickout, electrode geometry and electrode contamination.

(3) Workpiece processing and weld preparation.

(4) Shielding gas composition and flow rate.

(5) Metal transfer mode. This becomes more complicated in MIG welding. Generally, spray transfer stabilises the arc while globular and short-circuiting transfer destabilise the arc.

(6) Conditions of wire feeding.

(7) Magnetic arc blow. This can be caused by unsymmetrical magnetic field in the workpiece induced by the welding current or the residual magnetic field in the workpiece.

3.3 Laser Optical Sensors

3.3.1 General

The optical sensor is probably the most successful sensor used in welding control. So far a variety of optical sensors have been developed, as summarised by Cook [38] and Jones et al [39]. A general layout of optical sensing system is shown in figure-3.5. There are three main considerations in an optical sensing system: the object to be viewed, illumination, and sensor device. In arc welding, the viewing region can be either of pre-weld (i.e. weld preparation), post-weld, backface and arc zone.

The selection of appropriate sensor device depends on the objective of the sensing. To measure a single dimension of weld pool, a linear array of photodiodes may be used. Vroman [40] used a linear array of 64 photodiodes to observe the GTA weld pool. The pool width was determined by counting the number of brightly lit diodes. Smith [41] used a backface sensor to detect the pool size and welding position, as shown in figure-3.6. Other applications of photoelectric elements

for weld pool detection are found in [42 - 45]. More complicated sensing devices such as phase shift detector, position transducer and TV camera, are reviewed below.

Unless the sensor is observing the arc itself, arc illumination is usually undesirable, because of its high intensity and variability in both position and intensity inducing a wide range of light levels to which sensors are unable to respond. Therefore artificial illumination is often used:

(1) To minimise the effects of arc illumination.

(2) To provide a controlled and steady intensity in a range acceptable to the sensor.

(3) To improve the contrast of the feature to be viewed.

(4) To provide profile information by using structured light to intercept the object.

Among many illumination sources, laser light is most widely used because of its constant nature and high intensity. It is exemplified by the laser optical sensor based on shadow-cast technique, which is discussed later.

3.3.2 Time-of-Flight Technique

The time-of-flight technique is one of the most powerful techniques, giving true 3-D picture. It detects the object by calculating the time period during which the laser beam travels from the transmitter to the object, then back to the receiver, as illustrated in figure-3.7 [46]. The time-of-flight can be determined by using a pulsed laser and measuring the elapsed time or by measuring the phase shift. This technique overcomes some of the "hidden part" problems encountered with tri-

angulation since the transmitter and receiver beams are usually co-axial. However, it has long scanning time and insufficient accuracy. It takes three minutes to collect a 128×128 range picture. Such a time is too long for real-time control. Therefore, this time-of-flight technique needs more active research and development in order to meet real-time requirements.

3.3.3 Triangulation Technique

The principle of the range finder based on triangulation and position transducer is shown in figure-3.8. A point source of laser light is projected onto the surface. The scattered light deflected from the surface is focused by a lens onto a position-sensitive photo detector. The distance from the laser source to the surface is measured by calibrating the light spot position on the photo detector. By scanning the light across the joint preparation, 2-D information is obtained. The third dimension can be obtained when the laser source travels along with the welding torch. Edling and Porsander (ASEA) [47] apply this technique for adaptive control of torch position and welding parameters. A different type of range finder has been developed by Mitsubishi [48]. The laser source is rotated round the torch, rather than oscillated across the gap. Perhaps, the most successful laser tracking system based on triangulation and position transducer is the system called Seampilot developed by Oldelft [49, 50]. Figure-3.9 shows the system diagram. The principle of seam tracking is explained in [51 - 53]. A beam deflector mirror and viewing mirror are attached to the axis which is located above the joint and rotated by a stepping motor. The laser light is deflected onto the surface by the deflector and received by the photo detector via the receiving mirror and focusing lens. When

the stepping motor rotates, the laser light swings across the joint and 2-D profile information is obtained by the line photo detector. Again 3-D information is obtained as the camera moves along the weld direction.

Sensors based on triangulation and position transducer have distinct advantages as follows:

(1) Real-time sensing.

(2) Adapting to any joint geometry and welding process.

(3) Seam tracking and process control.

(4) Relatively simple hardware.

(5) Simple image processing algorithms.

For these reasons, this technique becomes very attractive in some cases.

3.3.4 Shadow-Cast Technique

This is perhaps the most widely used technique at present. Unlike the above two methods, shadow-cast technique uses a structured light illumination to form a high contrast profile image which is captured by a TV camera or an area array. Figure-3.10 shows two basic shadow-cast arrangements. Joint tracking systems based on shadow-cast technique are commercially available. Two MetaTorch systems, MetaTorch 200 and MetaTorch 500, have been developed by Meta Machines [54, 55]. The unique advantage of these systems is that the camera sensing head is enclosed inside the torch, providing the reliability and capability of operating under very strict conditions. Beattle [56] further explains the application of these systems for multipass welding. A similar tracking system is presented by Kawahara [57]. He explains the

image processing method to obtain a binary picture from the original picture. In the system described by Bamba [58], a two-dimensional position-sensitive photo detector, rather than a TV camera, is utilised to detect the image pattern formed by the structure laser on the workpiece. A further advance was made by Niepoid and Brummer [59]. By scanning the weld preparation, the system called PASS can pre-select the welding parameters in addition to joint tracking. The same system configuration is also used to observe the molten pool, thus enabling corrective movements of the torch and adaptive control of the heat input.

Instead of a flat sheet of laser light, a cone-shaped laser strip is used in the sensing system developed by Agapakis et al [60]. Feature points of the joint profile are extracted from the image for seam tracking and real-time adjustment of welding variables during single or multipass arc welding. A practical system based on this principle is introduced in [61, 62]. Dufour and Begin [63] describe a shadow-cast sensor with a pre-processor. Fast image recognition was intended. Inoue [64] used a rectangular spot of light to illuminate a very narrow joint gap. Detection of a thin weld line less than 0.5 mm width is said to be possible. In [65] he presents an automatic control system for horizontal narrow gap welding. By projecting a light strip onto the narrow gap the measurement of the groove width, detection of welding start and stop positions and tracking weld line are determined.

Similar to the technique based on the triangulation and position transducer, the shadow-cast technique has the advantages of:

(1) Real-time operation.

(2) Adapting to any joint geometry and welding process.

(3) Seam tracking and process control.

However, its widespread use may be limited by the following drawbacks:

(1) High hardware costs.

(2) Sensitivity to smoke and spatter.

(3) More sophisticated image processing.

(4) Slow sensing speeds. Very high welding speed can be difficult to achieve.

3.4 Direct Vision Sensor

For laser optical sensors, 3-D profile information about the object is possible with a single camera in collaboration with a structured laser light. On the contrary, direct vision sensor views an object without artificial illumination. Therefore a single-camera sensing system can only supply 2-D information. To obtain the third dimension, two cameras are needed to observe the object from different angles. LaCoe and Seibert [66] applied this idea for a 3-D vision guided welding robot system. It was able to locate the part to be welded. The problem with this system is that the vision process takes a long time to extract 3-D geometric information from two pictures. Therefore this type of configuration is rarely used for real-time welding control. Most direct vision systems available at the present time employ a single camera.

Gordon et al [67] used a closed-circuit television (CCTV) sensing system to monitor the weld pool. The pool image was captured via a so-called through-the-torch optical system, as shown in figure-3.11. A similar vision arrangement was adopted by Richardson [68] in order to view the GTA weld pool. The joint edge preparation, pool width

and the tungsten electrode can be identified by the vision system. Butt joint tracking and weld pool control are therefore possible. Baheti (GEC) [69] used this type of vision system to compute weld puddle geometry parameters for GTA welding of thin metal plates. The objective was to accommodate disturbances resulting from heat sinks and to produce acceptable welds.

Arata et al [70] used a vision system to observe the arc zone from outside the torch, so that the joint configuration, weld line and molten pool were detected. A considerable amount of research work has been done by Inoue [71, 72] for real-time examination of the welding process. He also applied direct arc observing vision methods for automatic control of horizontal narrow gap welding [73]. Binary image processing was often used in Inoue's work because of its simplicity. It did not cause much problem because the arc illumination was rather bright and the scene pattern was not complicated.

Direct vision is a powerful technique because it simulates the human eyes and shows a direct live image of the welding process. It can be either used for process monitoring or applied to seam tracking and process control. In the former case, no complicated image recognition is needed. However, for seam tracking and process control, the captured image must be processed in real time. Binary image recognition simplifies this problem, but is not applicable in such situations as poor contrast and complex control. Apparently, the power of direct vision systems relies on the fast speeds of grey level image recognition.

3.5 Other Sensing Techniques

Tactile sensor is an alternative way for seam tracking. Usually, a guide probe named tactile finger is placed in front of the electrode. The finger can have one, two or three degrees of freedom. Each of them senses displacement in one direction. A deflection proportional transducer is installed to feel the displacement at the probe tip. The output from the transducer is an electrical signal such as a voltage. This signal is fed back to a controller to correct the torch position. Pandjiris and Weinfurt [74] present a tactile sensor using this principle. Bollinger [75] discusses some algorithms and the application of tactile sensors to the guidance of welding robots.

Another way of tactile sensing is to code the probe displacements in an optical way. Presern [76, 77] presents a one-finger tactile sensor with two degrees of freedom. The position in each direction is coded by a set of six phototransistors, i.e 2×6 array of binary data for two degrees of freedom.

For the tactile sensor discussed by Cullen [78], the weld wire itself is used as the tactile finger. It is energised to a low voltage and the robot slowly approaches the object. Once the wire touches the grounded object, a drop in the voltage is sensed by the power supply, hence the workpiece is located.

Tactile sensors require simple electronics, with the result of low costs and simple control schemes. It works reliably with butt and fillet welding. However, bad conditions of the groove edge may lead to wrong displacement being recorded at the tactile finger.

Ultrasonic sensors have been investigated by a number of research workers. Ultrasonic detection of a weld pool is detailed by Lott [79] and Herdt and Katz [80]. Fenn and Stroud [81] used a pair of contact ultrasonic probes to measure the bead depth, and to detect the weld line. By doing so, a penetration control signal, a seam tracking signal and a quality control signal were generated. In [82] Gunnarson and Prinz employed non-contact ultrasonic transducers to measure the distance from the transmitter to the surface. In the system described by Tan and Lucas [83] a pair of transmitting and receiving transducers were located above the workpiece and rotated across the joint. The workpiece is sensed by calculating the phase difference between the transmitter and the receiver signals.

An ultrasonic sensor directly detects the weld pool or joint profile without much interpretation. No sophisticated electronic devices are involved. Usually, a microcomputer can efficiently analyse the ultrasonic signals, and supply control signals. The principal disadvantage of this type of sensors is that they are heavily dependent on the material conditions. Higher temperatures can produce errors using ultrasonic methods.

An infrared vision system employs an infrared camera to observe the welding process from either the front face or back face. In the vision system described by Chin et al [84], an infrared camera with a resolution of $\pm 0.2\,^{\circ}C$ was used to observe the weld pool during welding. Arc misalignment, groove geometry faults, variations in penetration, and impurities were detected by analysing the thermography. Seam tracking may be based on the difference in radii of isothermal curves to the left and right of the joint. A simple linear control action can

be used to move the arc until both radii are equal. An adaptive control system of GMAW process based on infrared vision was proposed by Begin and Boillot in [85]. Feature extraction from the infrared field surrounding the welding area enabled seam tracking, and mass and energy balance adjustment. Lukens and Morris [86] used this technique to sense the cooling rates for arc welding control. All the above methods look very encouraging. However temperature thermography is very sensitive to workpiece conditions. Furthermore, extracting geometric information from thermography is rather complex. For these reasons, infrared vision systems are not widely used in welding processes.

To sum up, a variety of sensors have been developed for welding control, but no universal sensor is possible to carry out all-round control in a welding operation. The selection of an appropriate sensor is largely dependent on the characteristic of a particular welding process and the objective of the sensor. A simple sensor such as a voltage sensor may be sufficient, while in some situations a vision system is necessary to implement complex control. This will be reflected in later parts of the thesis concerning real-time feedback control. Before that, chapter 4 will discuss experimental techniques and apparatus.

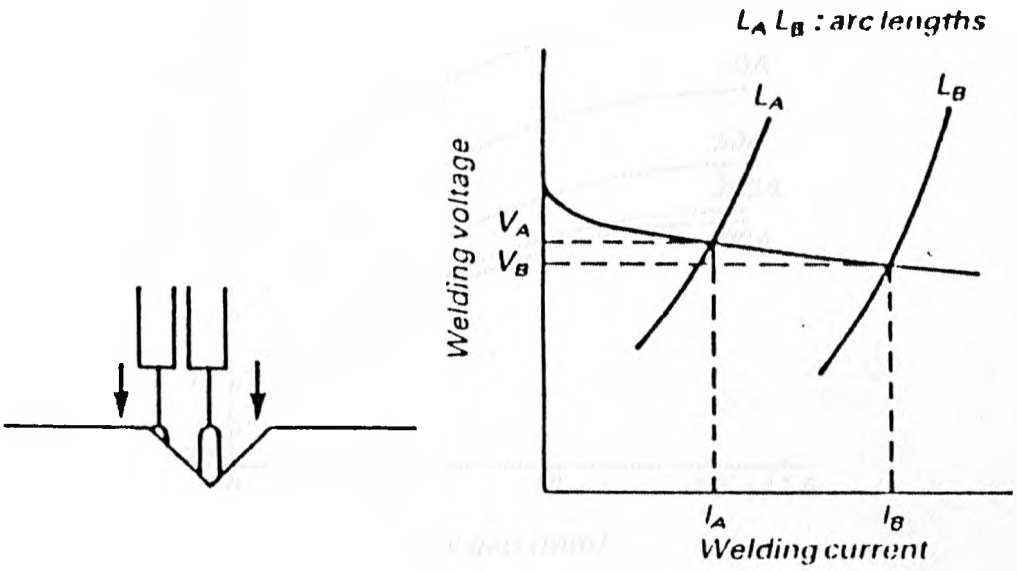


Figure-3.1 MIG power supply and its self-adjusted characteristic

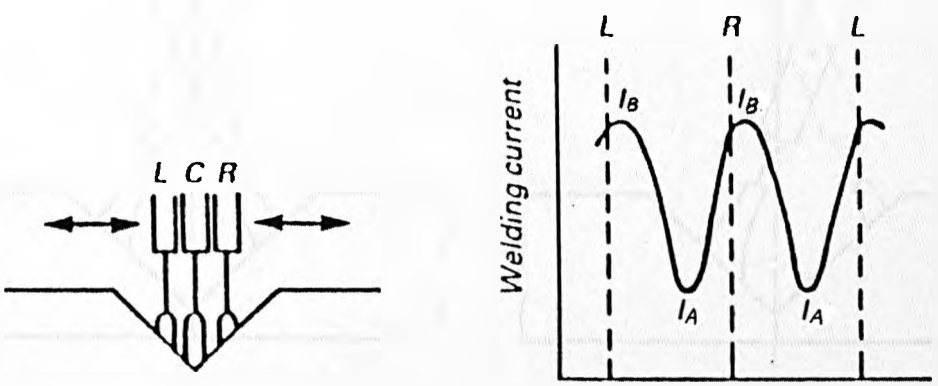


Figure-3.2 Change of welding current with oscillation

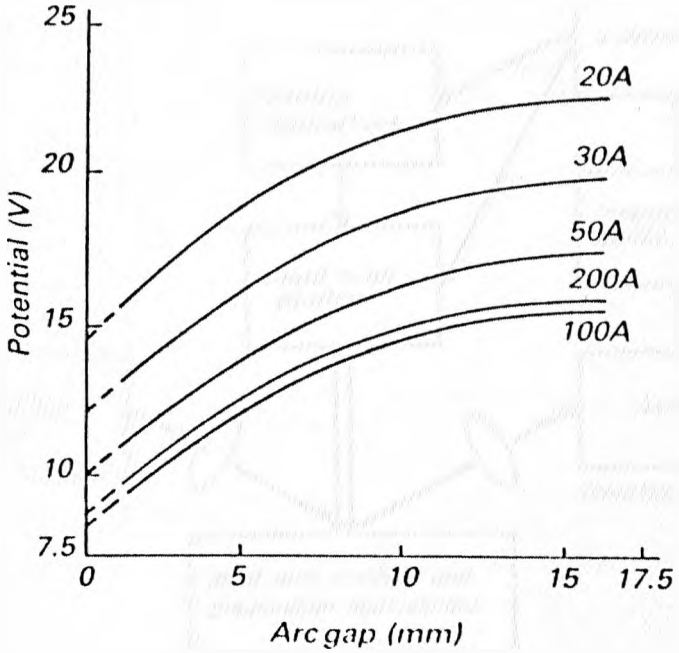


Figure-3.3 Variation of potential with arc length

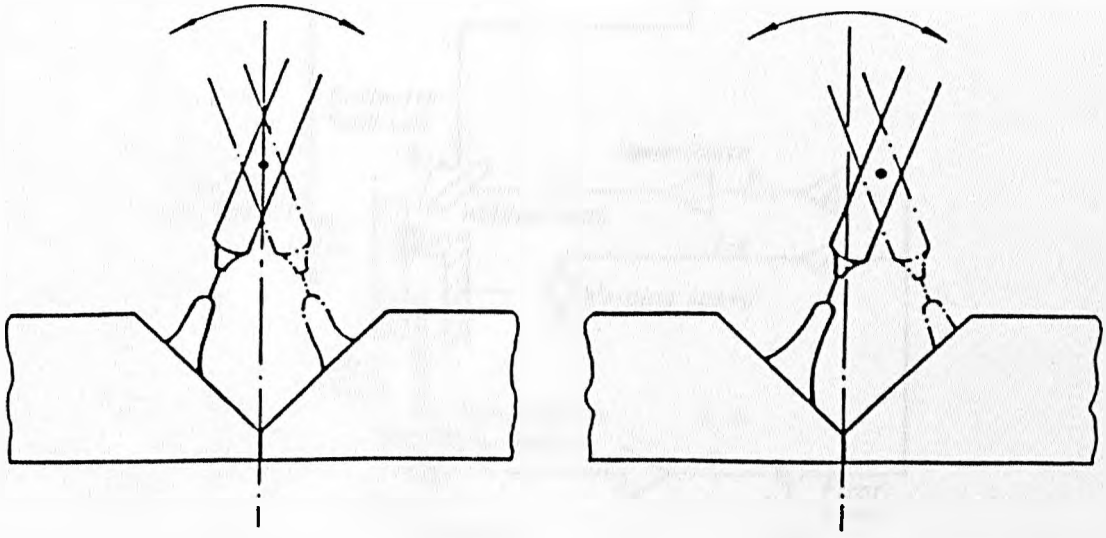


Figure-3.4 Oscillation torch around a centre of rotation

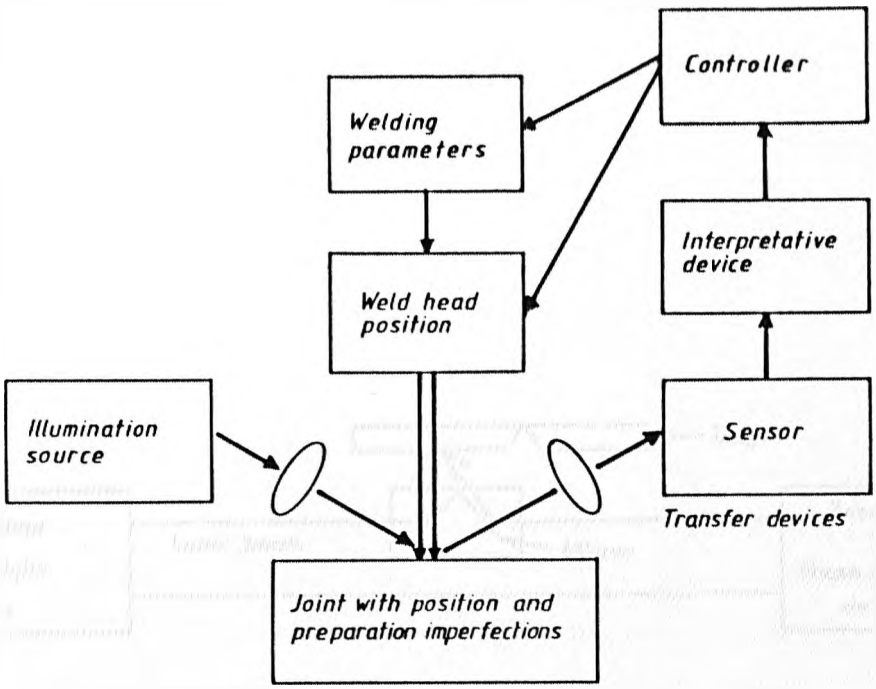


Figure-3.5 Schematic layout of optical sensing system

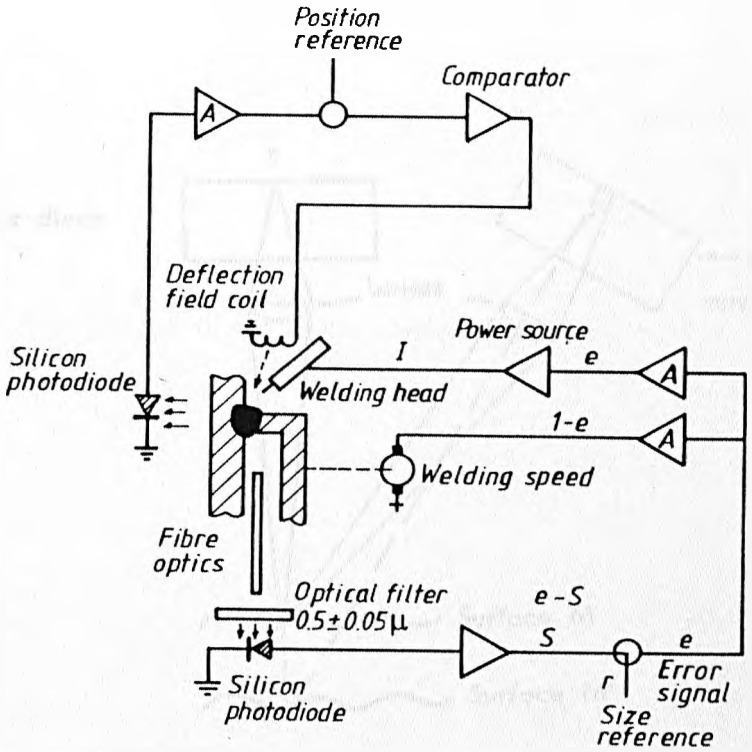


Figure-3.6 Backface sensor to detect the pool size and welding position

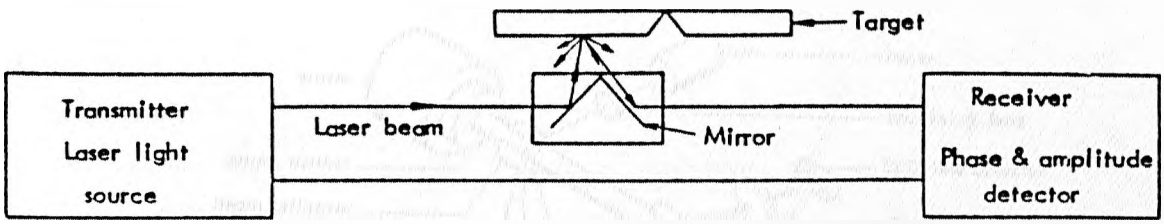


Figure-3.7 Block diagram of a phase shift range sensor

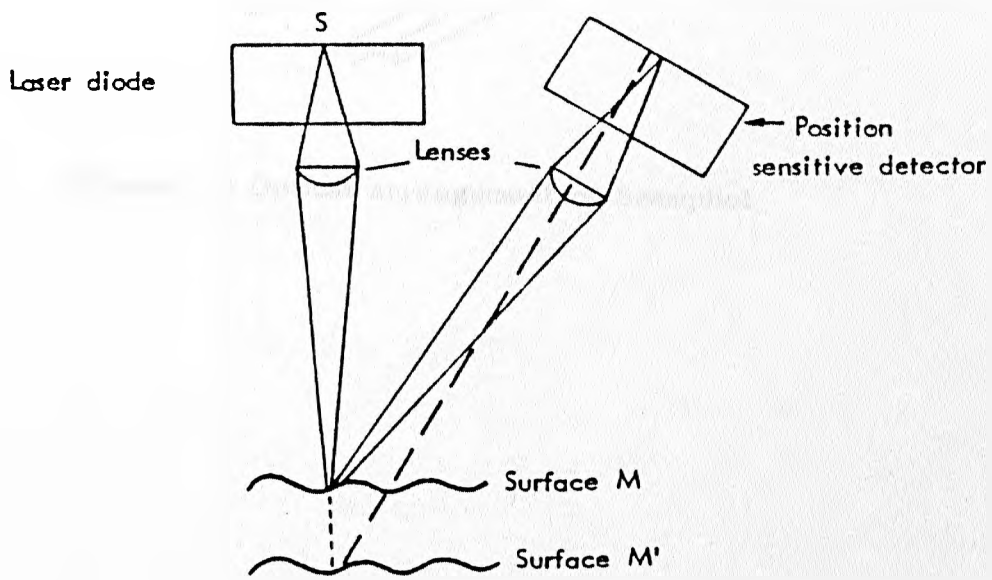


Figure-3.8 Principle of the range finder based on triangulation

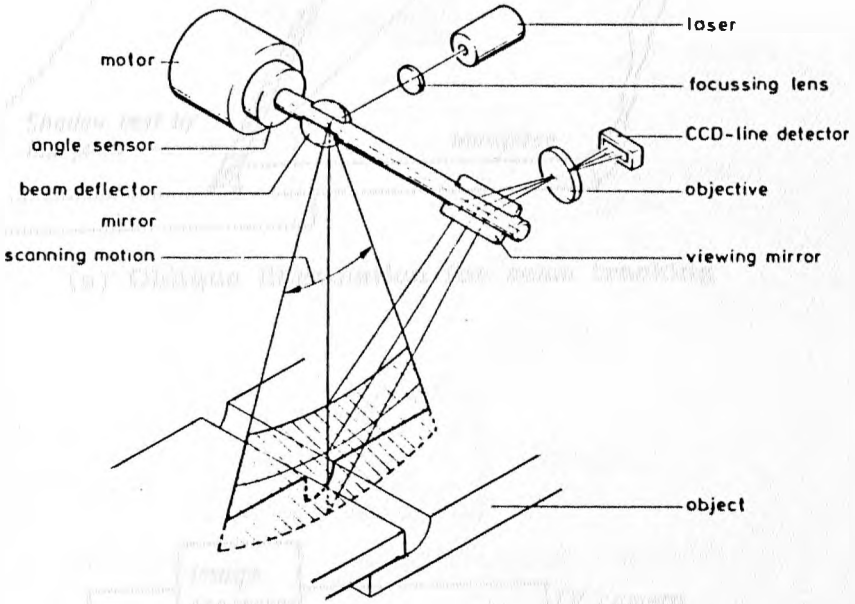
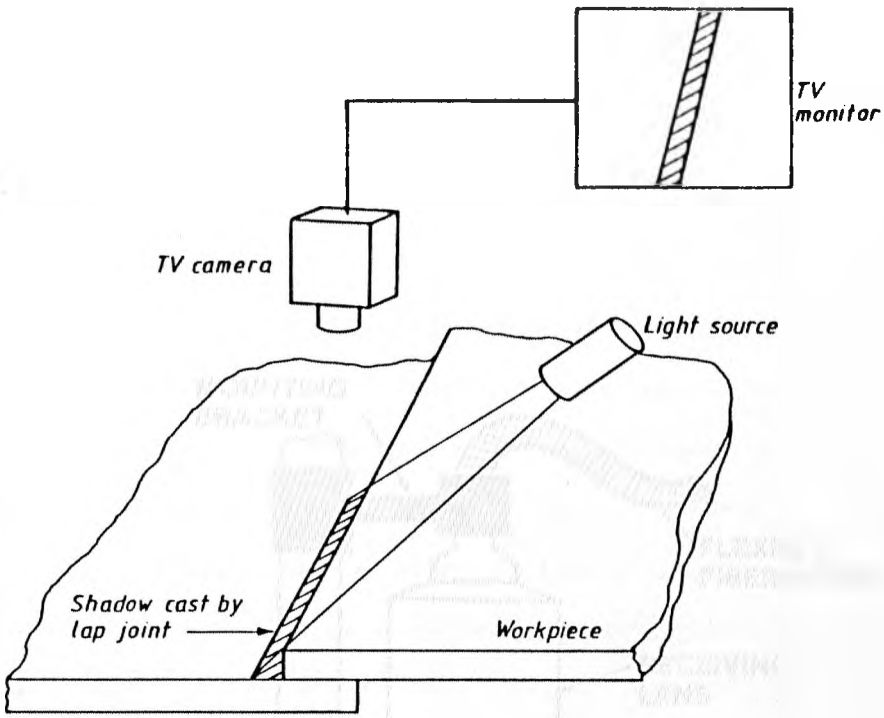
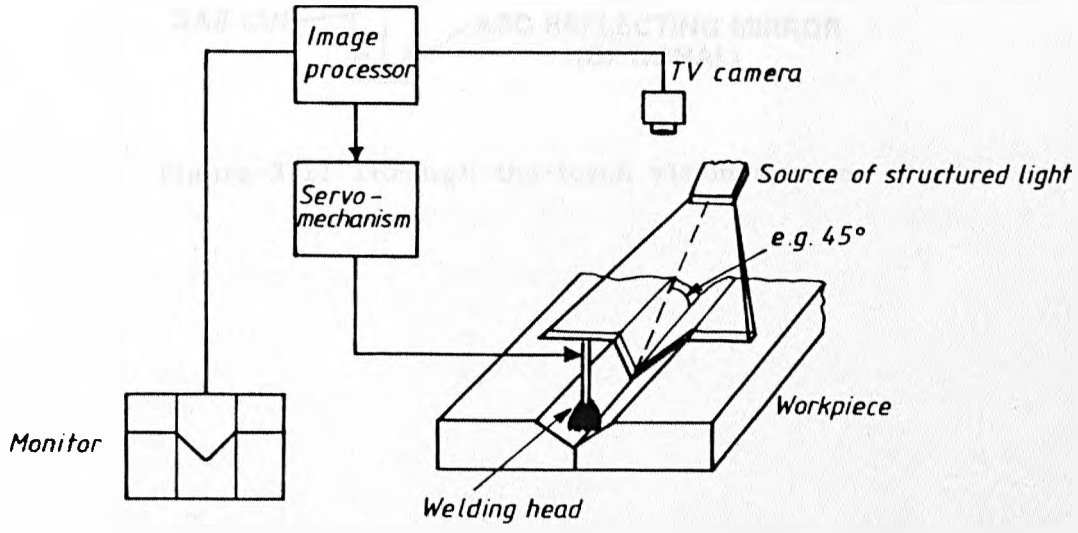


Figure-3.9 Optical arrangement of Seampilot



(a) Oblique illumination for seam tracking



(b) Structured light system for seam tracking

Figure-3.10 Arrangements of shadow cast technique

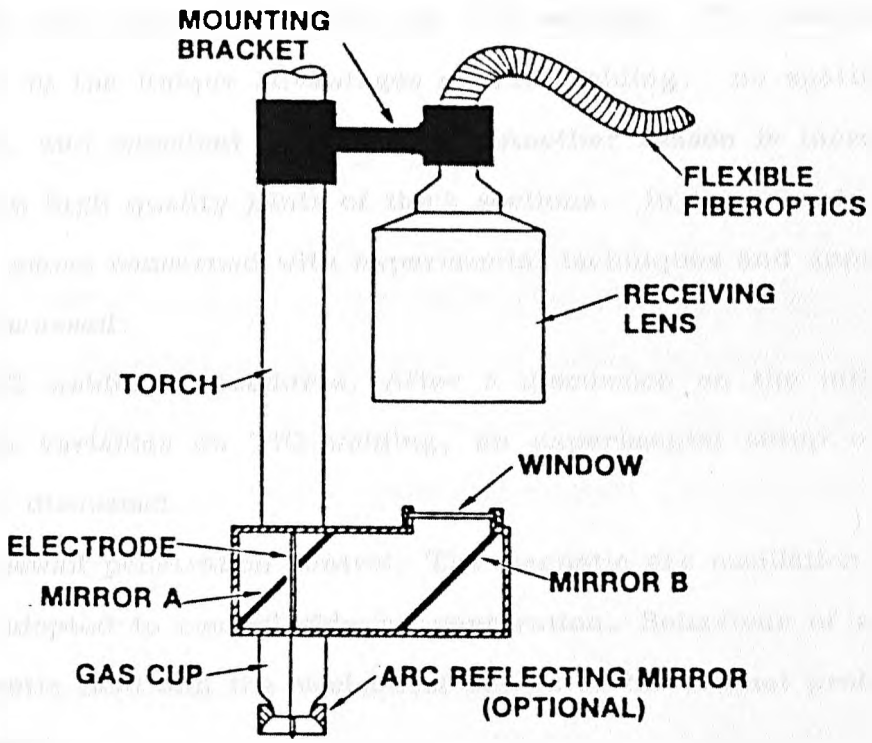


Figure-3.11 Through-the-torch vision system

4. A PROTOTYPE NARROW GAP TIG WELDING SYSTEM

4.1 Introduction

In chapter 2, major NGW techniques were discussed. Of particular interest in this project is narrow gap TIG welding. The initiative is taken due to the unique advantages of TIG welding: no spattering, stable arc, and excellent weld quality. Another reason is increasing demands on high quality joints of thick sections. In this chapter, the following issues concerned with experimental techniques and apparatus will be discussed:

(1) TIG welding procedures. After a discussion on the influence of process variables on TIG welding, an experimental setup of TIG welding is discussed.

(2) Sidewall penetration control. The magnetic arc oscillation technique is adopted to control sidewall penetration. Behaviour of an arc in a magnetic field and the mechanical design of the magnet probe will be discussed.

(3) Experimental apparatus. The drive circuit for electromagnet, the host computer, the welding rig and its controller, and the pendant controller will be discussed in the later part of this chapter

4.2 TIG Welding Procedures

As introduced in chapter 1, a TIG welding arc takes place between the non-consumable electrode (tungsten) and the weldment, and shielded by an inert gas. TIG is a relatively clean process without flux and spatter. Since the wire filler is fed into the pool separately,

considerations of TIG welding focus on the arc behaviour itself while metal transfer is a simpler issue. Generally speaking, a TIG welding process involves the following variables: arc voltage, arc current, arc standoff, electrode stickout, composition of electrode materials, electrode geometry, torch angle, and shielding gas. All these factors have profound influences on the arc and fusion characteristics, which are discussed as follows.

4.2.1 Process Variables of TIG Welding

The arc characteristic, i.e. the relationship between the arc voltage and arc current, is shown in figure-4.1 [30]. When the current is lower than a certain limit, the arc conductivity exhibits negative resistance, i.e. the arc voltage falls with increasing current ($V=\xi_3/I$). When the arc current exceeds a certain limit the arc conductivity demonstrate positive resistance, i.e. the arc voltage increases with the arc current ($V=\xi_1+\xi_2 I$). The complete V-I relationship is

$$V=\xi_1 + \xi_2 I + \xi_3/I \qquad \text{(equation-4.1)}$$

where V and I are the arc voltage and current respectively, and ξ_1, ξ_2 and ξ_3 are positive coefficients depending on the arc standoff. In a normal welding operation,

$$V=\xi_1 + \xi_2 I \qquad \text{(equation-4.2)}$$

plays a dominant part in the arc characteristic.

The influence of process variables on the arc and fusion characteristics has been thoroughly investigated. Kenyon and Boyce's work [87] suggests that both the voltage level and the gradient increase with the arc length, and short stickouts tend to give a higher voltage at low current, but a lower voltage at higher current. It is thought [87] that the effect of electrode geometry on the level and the shape of the arc characteristic is not great except for the thinner electrodes. However, Savage et al [88] conclude that with a straight polarity arc operating in the cathode spot mode, the included angle at the conical tip of the tungsten electrode has a significant influence on the arc characteristic and the weld penetration and width. As the tip angle increases the voltage level decreases (figure-4.2). The bead width falls with the increasing tip angle (figure-4.3), but the reverse applies to the bead penetration (figure-4.4). It is found [89] that the torch angle has a definite effect on weld penetration. A trailing angle produces a high and narrow bead. The influence of the type of shielding gas is generally agreed, where a mixture of argon and 2% H_2 results in higher arc voltage than pure argon [90]. The influence of dual gas shielding on the TIG arc has been studied by Schultz [90]. The analysis of voltage readings shows that the constriction of the arc is intermediate between TIG and plasma. Another important factor of the TIG process is the type of welding current and polarity. The welding power supply can be either direct reverse polarity, direct straight polarity, or alternate current. With direct reverse polarity, the tungsten electrode is connected as an anode and subject to overheating. Therefore, the arrangement is rarely used although it has the function of removing oxidisation from the work surface. Direct straight polarity avoids overheating of the tungsten, and produces a great deal of heating on the workpiece with the result of narrow, deep-penetrated

bead. It also stabilises the arc due to no change of current polarity and the tungsten cathode. Thus most welding applications utilise the direct straight polarity. A compromise of the above two is alternate current, which may be used for welding aluminium and its alloys.

The behaviour of a pulsed plasma-pulsed GTA arc has been thoroughly studied [91, 92]. The pulse parameters: peak current, background current, peak time and background time have considerable effects on the penetration characteristic. The important implication is that the weld bead geometry, and probably the microstructure, can be controlled via the selection of these parameters. Cook et al [93] discuss the effect of high-frequency (typically 5 KHz) pulsing of a welding arc. It is shown that the arc pressure, hence arc stiffness, of the pulsed arc may be as much as ten times of the equivalent dc arc. The effect of pulsing arc exhibits greater penetration and capability of overall control.

4.2.2 TIG Welding Setup

A transistor d.c. welding power supply, APW T300EC, was used to carry out experimental work. The direct straight polarity was employed, i.e. the tungsten electrode was connected as a cathode and the workpiece as an anode. The advantages of this arrangement are:

(1) Less heat on the cathode, avoiding overheating of the tungsten.

(2) Stable arc due to the tungsten electrode acting as a cathode.

(3) About two thirds of total heating energy on the workpiece, in favour of bead penetration.

A constant current was preferred. It was preset each time before welding started. The tungsten dimensions and setup were changed in

different sets of experiments, but Hi-lite shielding gas was always used.

In a TIG welding system, the wire filler is fed into the weld pool, independent of the power supply. With a single tungsten electrode, the wire can be fed either from the rear of the pool, or from the front of the pool. In this work, the wire was fed in the latter way.

4.3 Sidewall Penetration Control

A variety of NGW techniques were detailed in chapter 2. Most of them involve mechanical weaving or rotation of the torch, or pre-deforming the filler wire. The introduction of mechanical movement of the torch clearly complicates the welding process. The repeatability of arc oscillation or rotation is often made worse by pre-deforming the filler wire. It was decided in this project that the sidewall penetration control should be implemented by means of magnetic arc oscillation. As a result, little or no mechanical movement of the torch is involved. The following discusses the behaviour of a welding arc in a magnetic field, and the technical setup of a magnetic arc oscillator.

4.3.1 Behaviour of a Welding Arc in a Magnetic Field

It has been long realised that the welding arc can be distorted or forcibly directed away from the point of welding. This phenomenon, know as magnetic arc blow, was thoroughly explained by Jannings and White [94]. The consequence of arc blow is destabilisation of the arc, hence defective welds. However, an external magnetic field can be used to control the arc, if properly designed. In fact, this subject has

drawn much attention from researchers, particularly in USSR, in the 1960's [95 - 98].

In relation to the welding direction, an external magnetic field can be applied in three different ways (figure-4.5) :

(a) A longitudinal magnetic field, i.e. one parallel to the electrode axis.

(b) A transverse magnetic field parallel to the welding direction.

(c) A transverse magnetic field perpendicular to the welding direction and the electrode.

The influence of a longitudinal magnetic field on an electric arc has been studied by Levakov and Lyubavski [95]. The interaction of this type of magnetic field can rotate the arc, or produce a conical arc column, or result in an unstable arc with variable shape. It was found that the conical arc maintained its stability, and was an efficient source of heat for welding slender tubes into a plate.

Kovalev and Akulov [96] discuss the stability of a TIG arc in a transverse magnetic field and methods to improve the arc stability. As a result of tests made on different materials using a GTAW arc subject to a transverse magnetic field, Hicken and Jackson [97] pointed out that beneficial effects were only evident when the arc was deflected forward with respect to the welding direction. Arc deflection in a transverse magnetic field has been investigated by Kovalev [98], Serdyuk [99], and Backelis [100]. In general, it was related to the magnitude of the magnetic field, welding current, arc standoff and the type of parent material. Practical uses of external magnetic field have been reported in earlier years. Serdyuk [101] describes the magnetic rotation

of a welding arc between concentric electrodes (figure-4.6). The tube to be welded was used as the inner electrode, the outer electrode was a hollow ring of electrolytic copper which was concentric to the tube. Once the arc was struck between the concentric electrodes, it was rotated and controlled by a transverse magnetic field. Consequently, a circumferential weld bead was produced. Hicken et al [102] applied a transverse magnetic field parallel to the weld line to a GTAW arc. The arc was therefore deflected across the joint. A similar arc deflection system is described by Deminski and Dyatlov [103], but applied to arc welding with a consumable electrode. Most recently, Hughes and Walduck [104], both in Coventry Polytechnic, have developed robot plasma welding of thin plates employing electromagnetic arc control. The beneficial effects were high-frequency weaving to compensate for poor joint fit-up, and improvement of bead appearance. It is generally agreed that the application of a magnetic field has the following advantages:

(1) Improving weld surface appearance.

(2) Increasing the welding speed at which undercut-free welds can be made.

(3) Improving weld pool solidification conditions, hence the mechanical properties of welded joints.

4.3.2 Magnetic Arc Oscillation in NGW

To realise magnetic arc oscillation, it is essential to design a magnet to generate a magnetic field. The electromagnet was built with an iron rod wrapped by 100-turn coil (figure-4.7). It is enclosed in a cylindrical aluminium case. A water cooling tube is attached to the case to prevent overheating of the magnet. An electromagnetic field is gen-

erated once a current passes through the magnetic coil. Adoption of an electromagnetic field allows flexibility in controlling the field pattern, hence the arc deflection. The design employs a single-pole probe rather than conventional double-pole. Thus the magnet can be made compact, and easily attached to the torch in a narrow groove. Although a single-pole electromagnet produces less concentrated magnetic field than a double-pole, later experimental results show that a sufficient magnetic field is generated at the tungsten tip to deflect the arc as required.

The field intensity is determined by the coil length (l), number of turns (N) and coil current (I) [105], that is,

$$B = NI/l \qquad \text{(equation-4.3)}$$

The flux density(H) [105], which will be used to represent field strength, can be expressed as:

$$H = \mu_0 \mu_r NI/l \qquad \text{(equation-4.4)}$$

where μ_0 is 1.26×10^{-6} $(T \times m/A)$, and μ_r is relative permeability. Clearly, for a selected coil the field strength is only affected by the coil current.

It must be noted that VHF (around 200 KHz) signals are widely used in welding power supplies, such as arc striking. When passing capacitors and inductors, these signals induce an extremely high voltage, leading to damage of the computer. Therefore, both the iron

core and magnet case must be grounded. Grounding the iron core also prevents possible arc striking between the tungsten and itself.

The electromagnet is placed behind the electrode, generating a transverse magnetic field parallel to the weld line. As a result, the arc is magnetically deflected in the direction perpendicular to the welding direction. By applying an a.c. current to the magnetic coil, an alternating magnetic field is produced, deflecting the arc towards left sidewall and right sidewall alternatively. Therefore, control of both sidewall penetrations is realised.

The distance between the magnet and the electrode is crucial in terms of the flux density across the arc. Since the flux density decays in proportion to the square of the distance, a short distance between the electromagnet and the electrode is needed to produce a strong arc deflection. However, this risks arc striking between the electrode and the magnet, and overheating of the magnet. It was found that a distance of 5 mm between the electromagnet and the electrode is a convenient compromise.

4.4 Control System of Narrow Gap TIG Welding

The overall control system for narrow gap TIG welding is shown in figure-4.8. It consists of a host computer, a welding rig and manipulator, a pendant controller, an arc oscillation controller (containing a voltage sensor) and a vision sensor. The former three subsystems are discussed in the remaining part of this chapter while the arc oscillation controller (hardware and software) and the vision system are the subjects of next three chapters.

4.4.1 Host Computer

In the control system shown in figure-4.8, dedicated microcomputers have been built for environment-oriented control. These custom-built hardwares are supported by machine-dependent softwares, therefore inefficient to carry out more complicated and flexible processing. It was decided to use a host computer to assist the control system.

For a consideration of economy and efficiency, a personal computer (PC) is preferred to a small system. To date, many PC families are available in the market, 8-bit and 16-bit microcomputers being two main categories. A 16-bit PC is faster and more efficient than a 8-bit PC, but more expensive. Among many manufacturers, IBM dominates the PC market. The most popular family is IBM-PC AT/XT. Many other manufacturers produce IBM compatible machine. In this project a IBM-PC AT is selected as the host computer due to its competitive performance.

The host IBM PC AT [106] has a 16-bit Intel 80286 CPU, and an 80287 coprocessor which enhances the computing speeds. The CPU has a clock frequency of 6 MHz. The PC is running under the DOS 3.00 [107, 108]. A powerful feature of an IBM PC is that a variety of commercial software packages are available for it, which undoubtedly facilitates the user's application.

The host computer performs the following tasks:

(1) Work as a central processor in communication with custom-built computers such as the rig manipulator and pendant controller.

(2) Work as a sensing processor. Sensor data such as arc voltage are passed on to the host PC for real-time processing. The host computer needs to support a frame store which digitises and stores an image.

(3) The user works on the machine to write, compile, debug and test programs.

4.4.2 Welding Rig and Manipulator.

A three degree of freedom test-rig (figure-4.9) was used to manipulate the welding torch. It can move the torch up and down (axis A), left and right (axis B: across the gap), and back and forth (axis C: welding direction). Each movement is driven by a stepping motor, and controlled by a manipulator based on 8086 CPU [109]. The manipulator can also control two additional stepping motors, one for workpiece movement and another one for wire feeding. The manual control of the test rig is facilitated with a pendant controller, which is discussed in the following.

4.4.3 Pendant Controller

To facilitate welding tests, a pendant controller has been designed. Figure-4.10 shows the 8085 computer board. Its block diagram is illustrated in figure-4.11. Basically, it consists of an Intel 8085 CPU [109], a 2764 (8k x 8) RAM as a store for temporary data, an 6264 (8k x 8) ROM storing control software, and I/O peripherals. The detailed circuit diagram is shown in figure-4.12, and the components used are listed in appendix-A1. Two 74LS138 TTL chips [110] were used to generate all chip selects as follows (in hex):

Chip Select	Address
2764 (ROM)	0000-1FFF
6264 (RAM)	0800-9FFF
8256 (UART)	0000
μPD7002 A/D	0200
74C922	0400
DMX402 Display	0600

I/O peripherals include a serial channel, an A/D converter, a keypad and a display. The serial channel enables communication between the pendant controller and the host IBM PC. It works via the CPU hardware interrupt RST6.5. When the pendant controller sends data to or receives data from the host IBM PC, an interrupt signal is generated at RST6.5, the 8085 CPU halts the main program until all data is sent or received. Likewise, the hardware interrupt RST7.5 of 8085 CPU was used for keypad reading. Once a key is struck, an interrupt is pending. The CUP halts the main program and reads the key code, then returns to the main program. In contrast, potentiometers reading and writing to the display were implemented via polling status. The CPU continues polling till the I/O devices are ready, then carries out I/O reading or writing. A μPD7002 A/D [111] was used to interface the joysticks to the system bus. Its four channels were connected to four potentiometers. Whenever the CPU enters the reading routine, it does not exit until all four channels are read. The display is a slow device. To ensure sufficient time for writing to it, a wait clock cycle was generate via hardware. The timing waveform is shown in figure-4.13.

The pendant control functions is illustrated in figure-4.14. These are further explained as follows:

(1) DRIVE: drive each axis using the joysticks, axis locations are simultaneously displayed. In model axis A, B, C and D are movable, in mode 2 axis C, D and E are movable. Figure-4.15 shows the flow chart of the drive model.

(2) PRESET: set up axis locations.

(3) DATUM: move the torch to the initial position.

(4) GO: move the torch to the preset position

(5) I_WAVE: select welding current wave parameters, i.e. peak current, peak time, background current and background time.

(6) WELD_PA: select such parameters as welding speed, wire feed speed, weaving width and weaving frequency.

(7) SENSOR: switch on/off sensors.

The software implementing the above control function is written in 8085 assembler [112] and listed in appendix-B1

In this chapter, TIG welding procedures and the technique to control arc oscillation has been discussed. Experimental apparatus such as host IBM PC, welding rig and manipulator, and pendant controller are also covered. The pattern of the magnetic field, hence the fusion characteristic, is controlled by a custom-built microcomputer. Hardware design of the computer controller will be discussed in the next chapter.

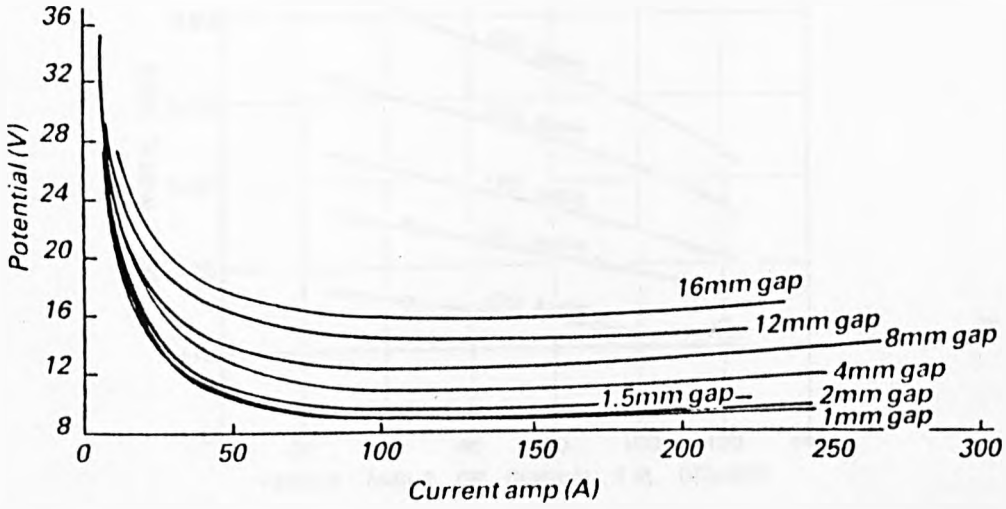


Figure-4.1 Typical arc characteristic for an argon arc

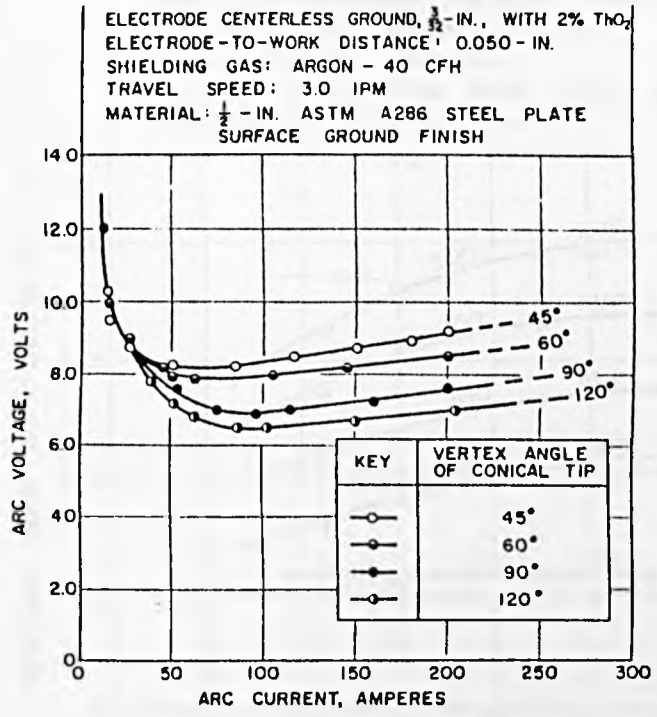


Figure-4.2 Effect of vertex angle of conical tip on arc characteristic

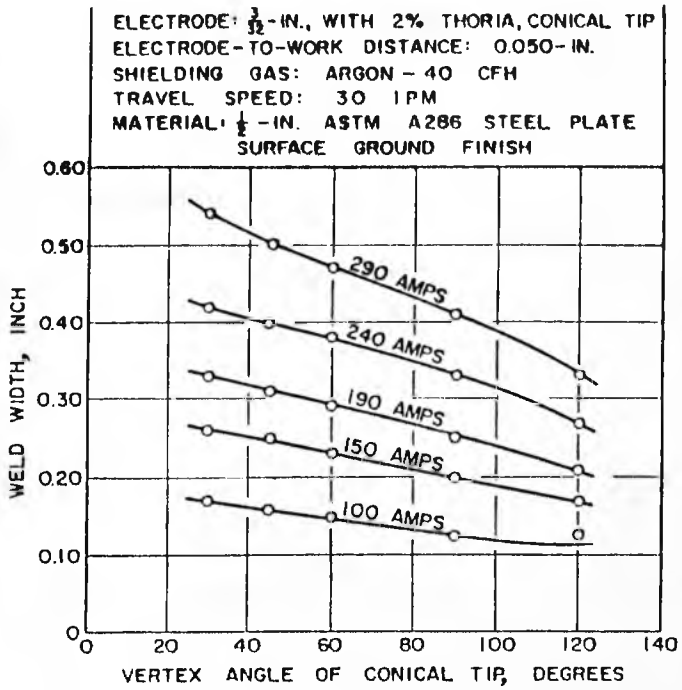


Figure-4.3 Effect of vertex angle of conical tip on weld width

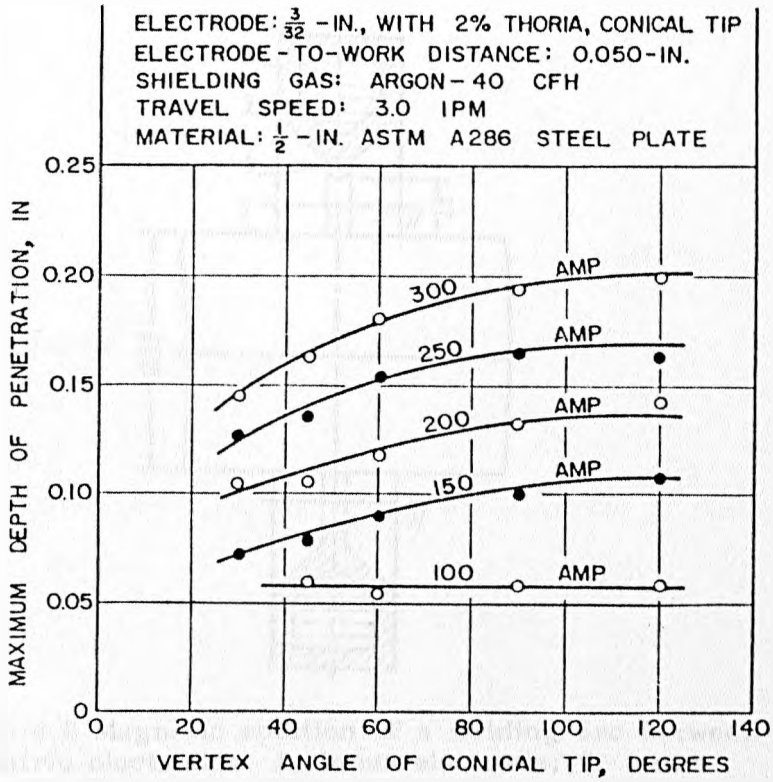


Figure-4.4 Effect of vertex angle of conical tip on weld penetration

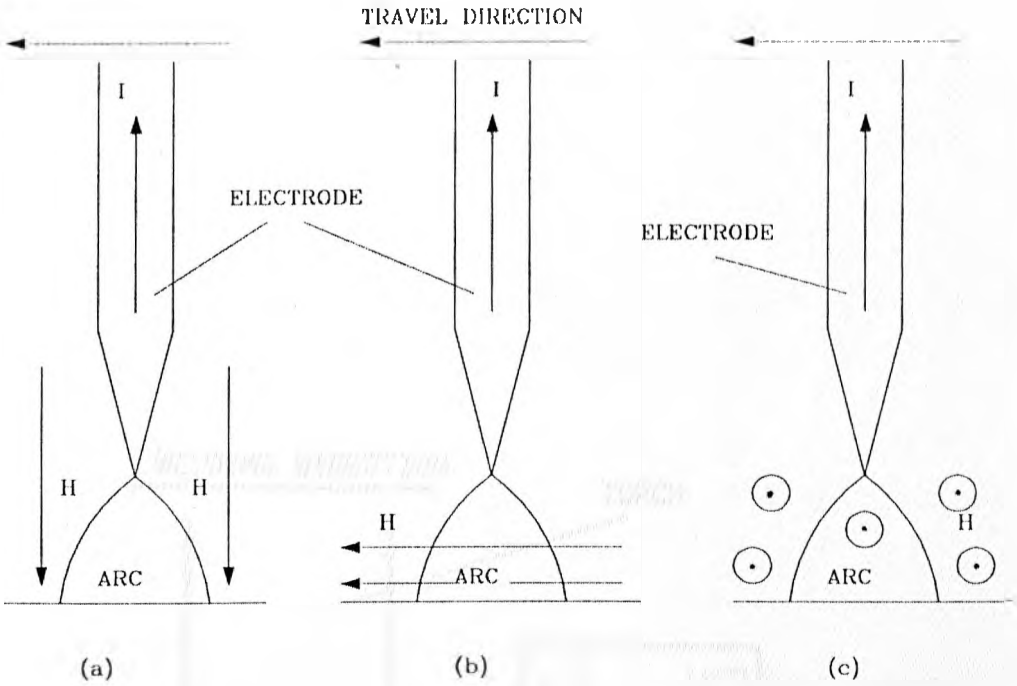


Figure-4.5 Three types of magnetic fields in relation to the welding direction

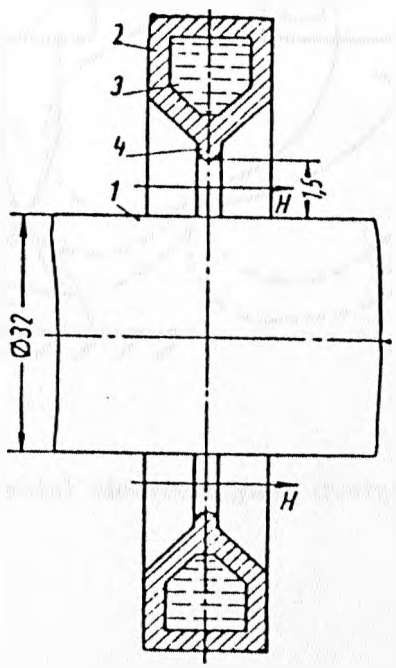


Figure-4.6 Magnetic rotation of a welding arc between concentric electrodes. 1, inner electrode; 2, outer electrode; 3, water cooling duct; 4, internal flange

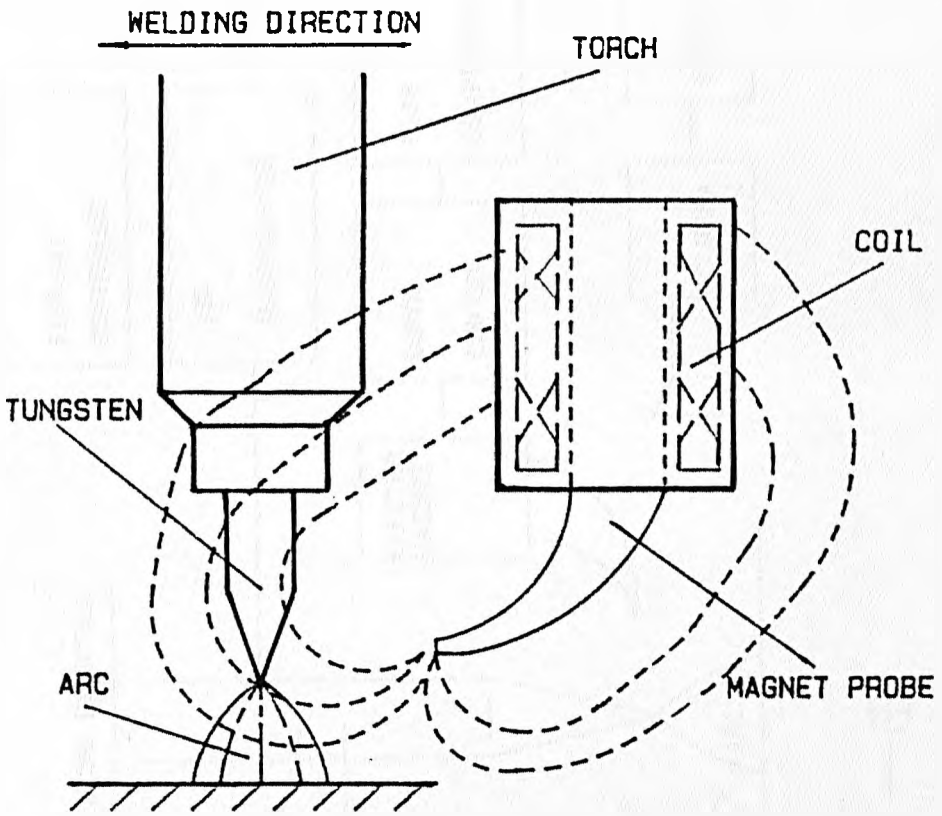


Figure-4.7 Experimental electromagnet designed for arc oscillation

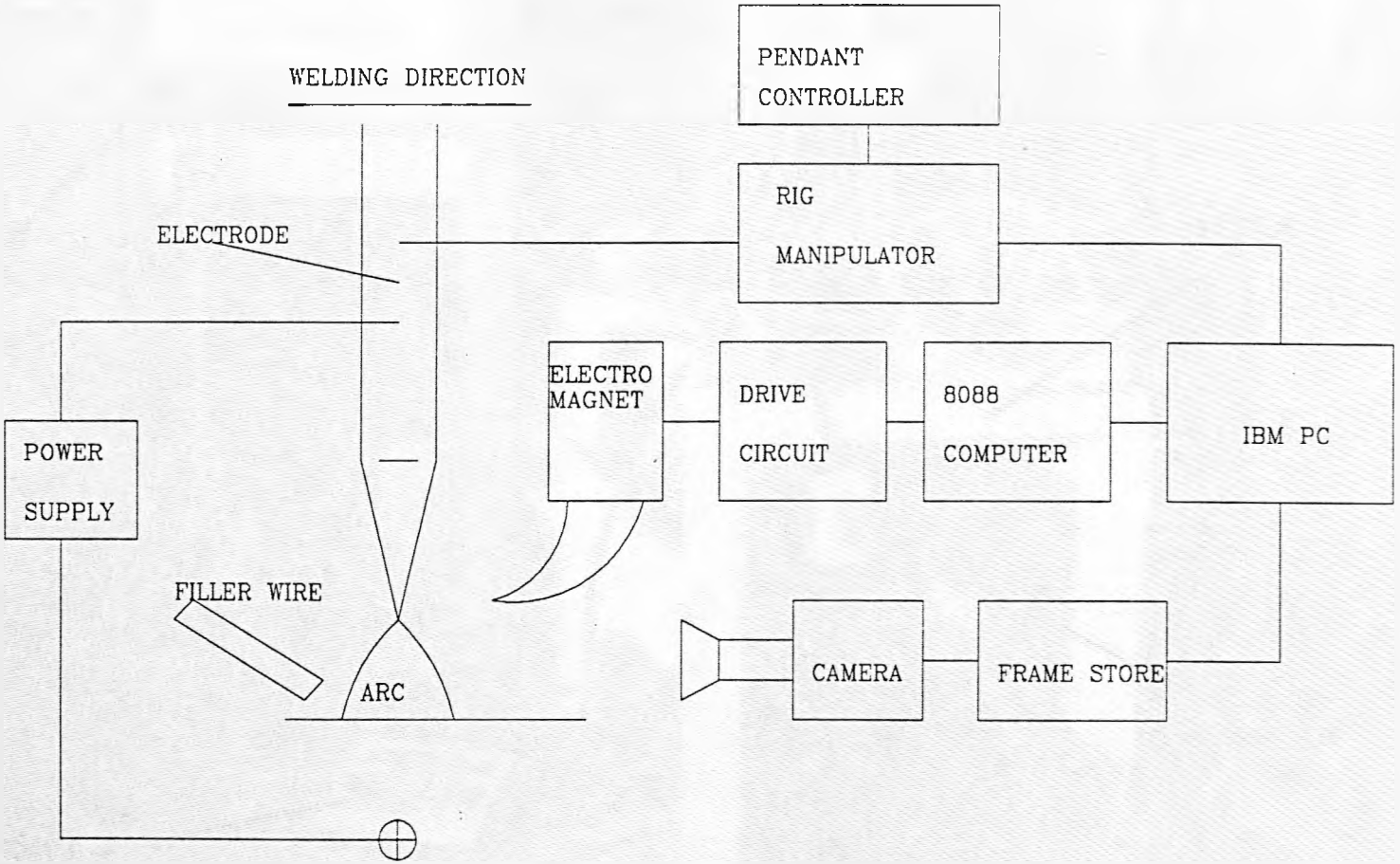
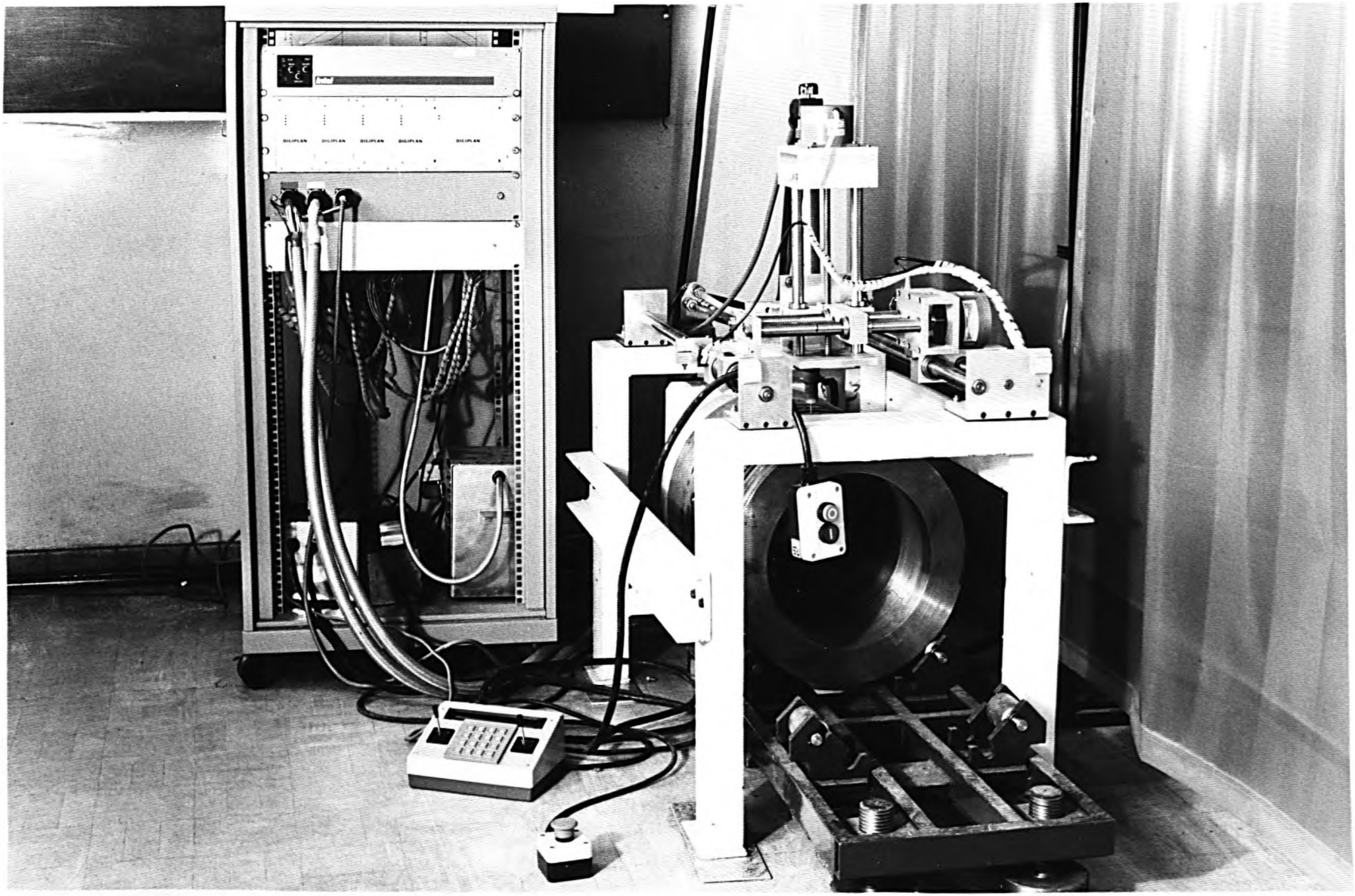
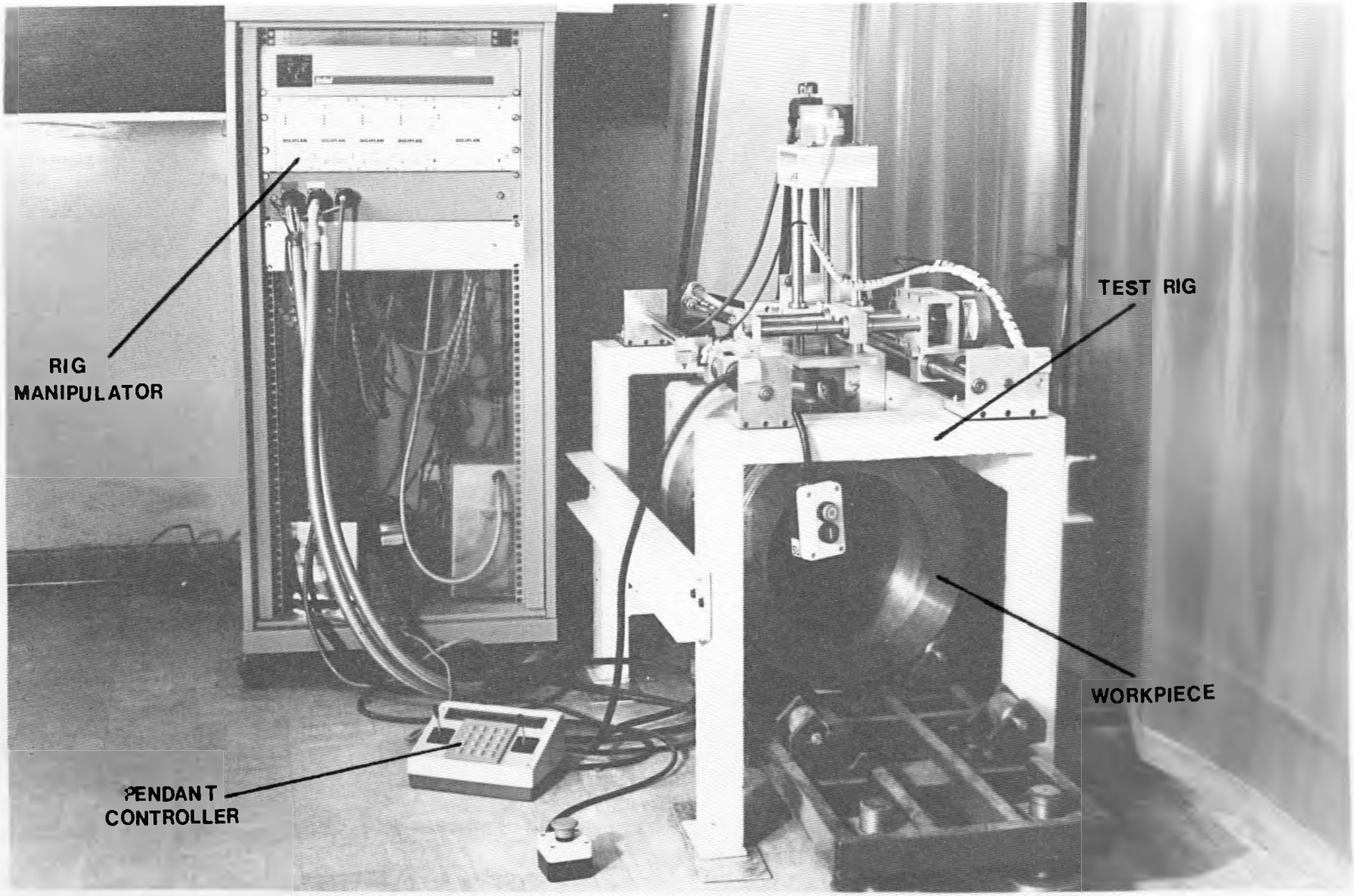


Figure-4.8 Overall control system for narrow gap TIG welding

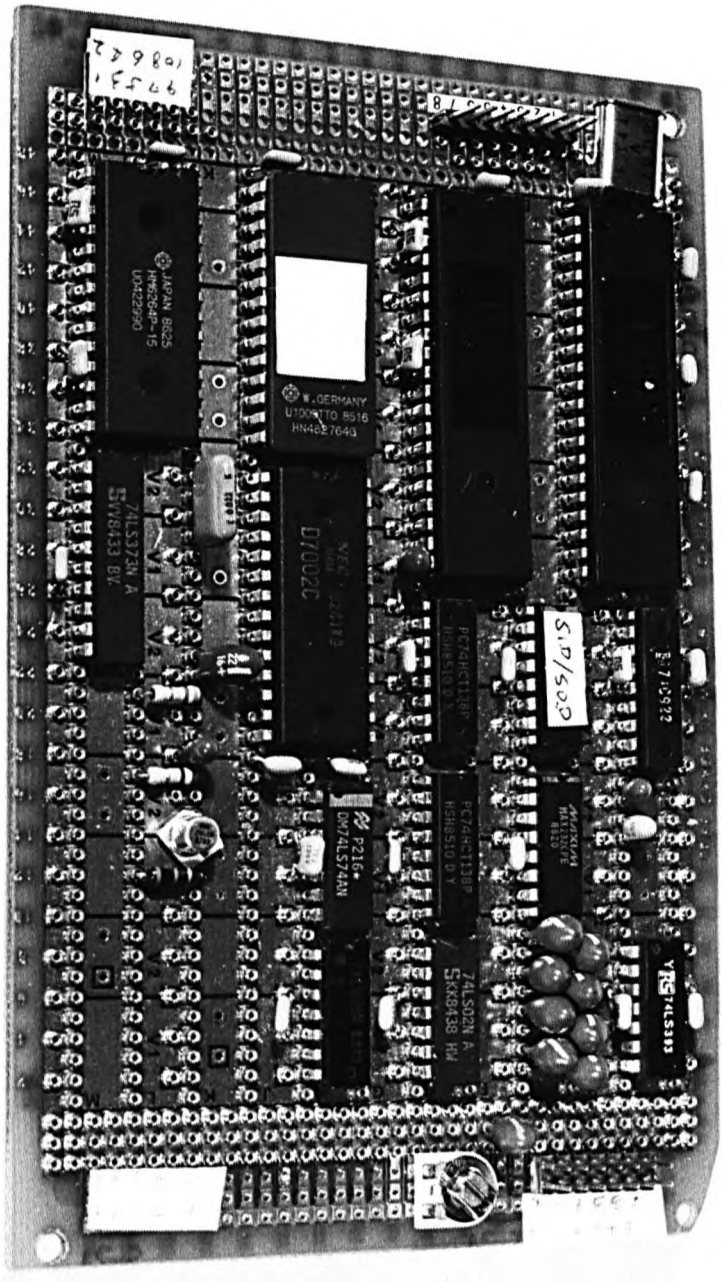
FIGURE–4.9 WELDING TEST RIG





FIGURE–4.9 WELDING TEST RIG

FIGURE – 4.10 8085 COMPUTER BOARD


W.GERMANY
D7002C
74LS373N A
74LS02N A
74LS393

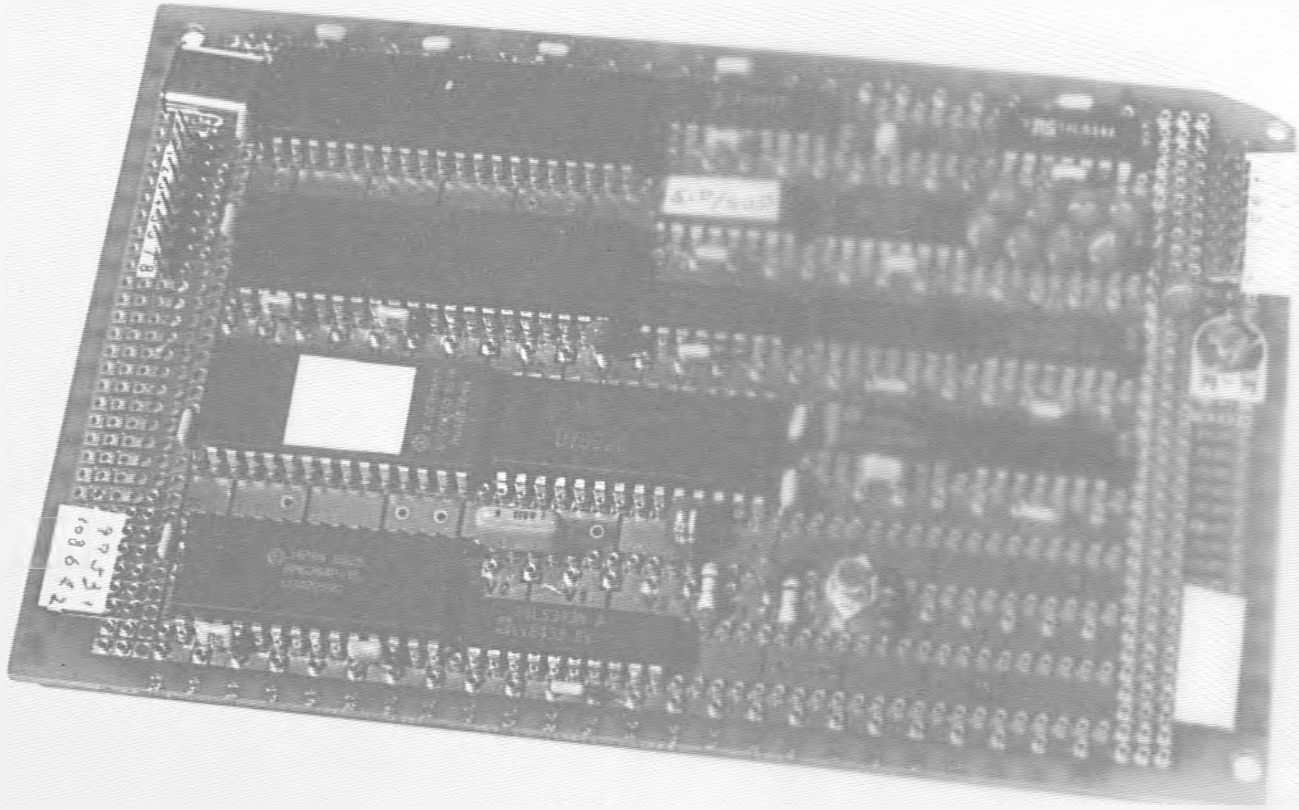


FIGURE – 4.10 8085 COMPUTER BOARD

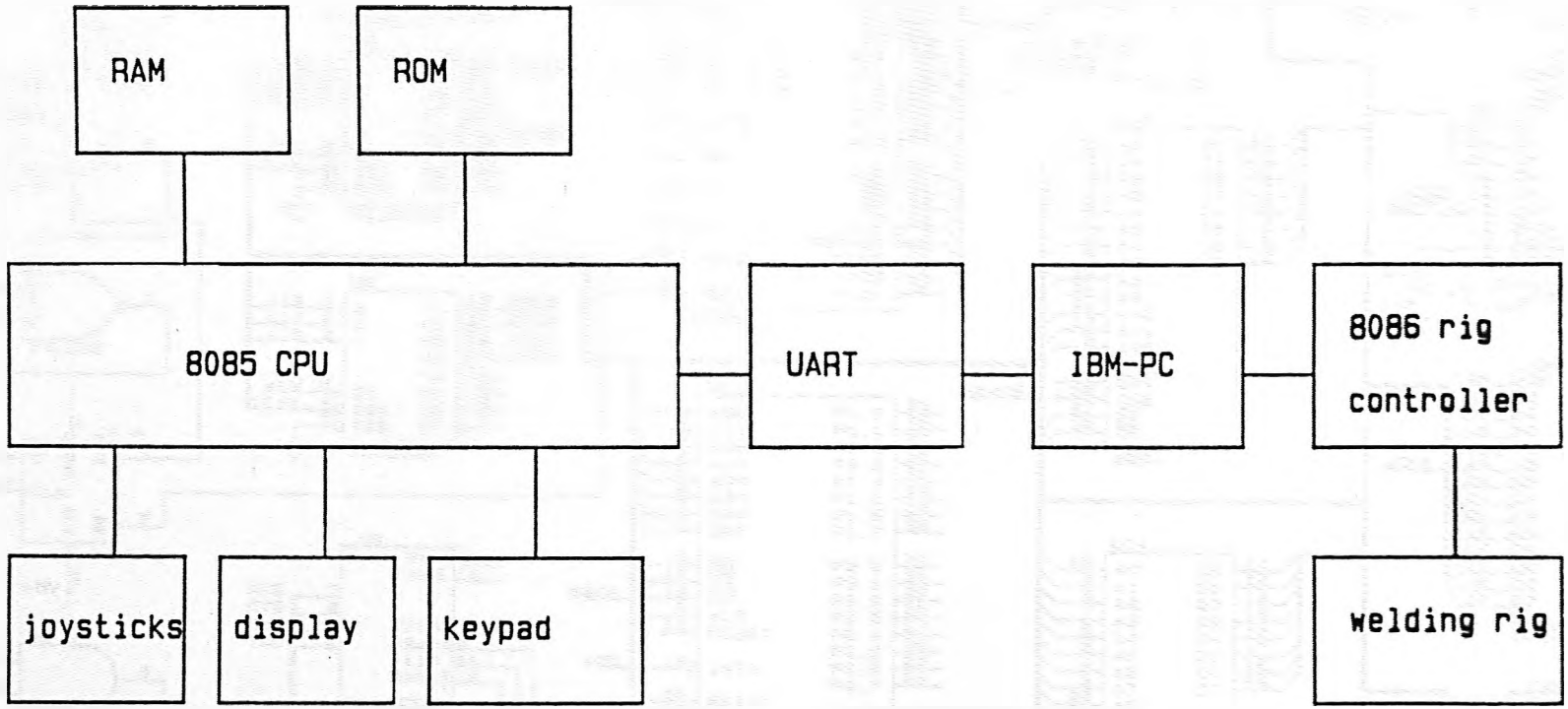


Figure-4.11 Block diagram of pendant controller

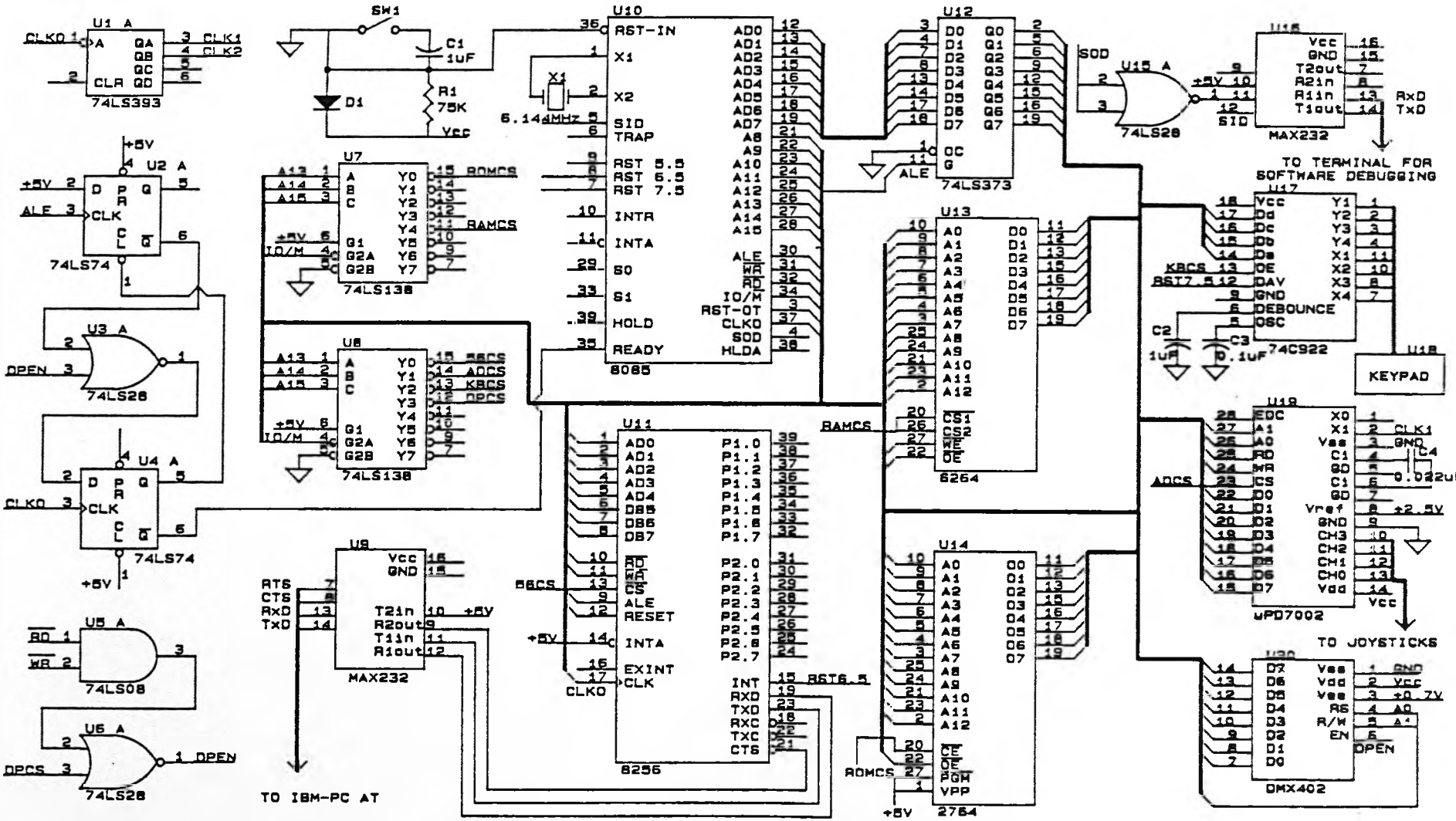


Figure-4.12 Circuit diagram of pendant controller

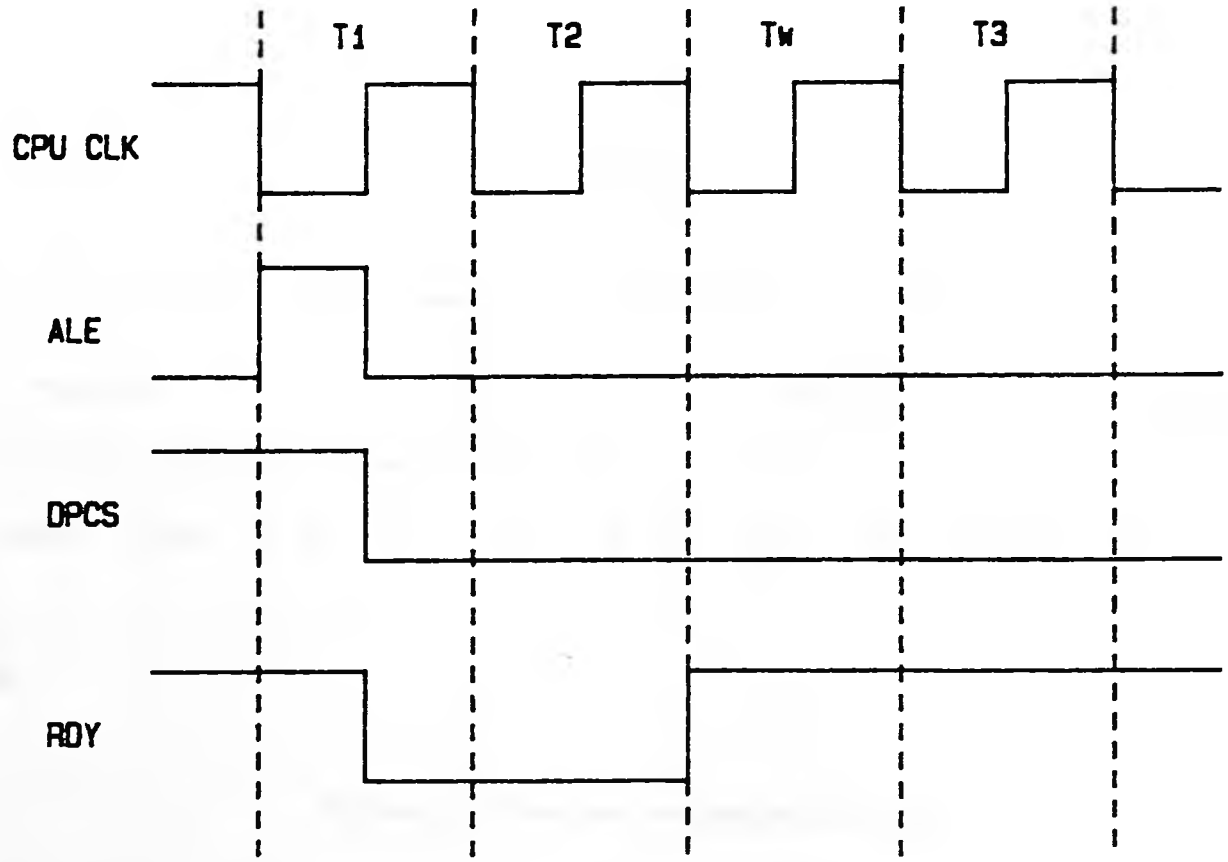


Figure-4.13 Timing waveform for the 8085 computer

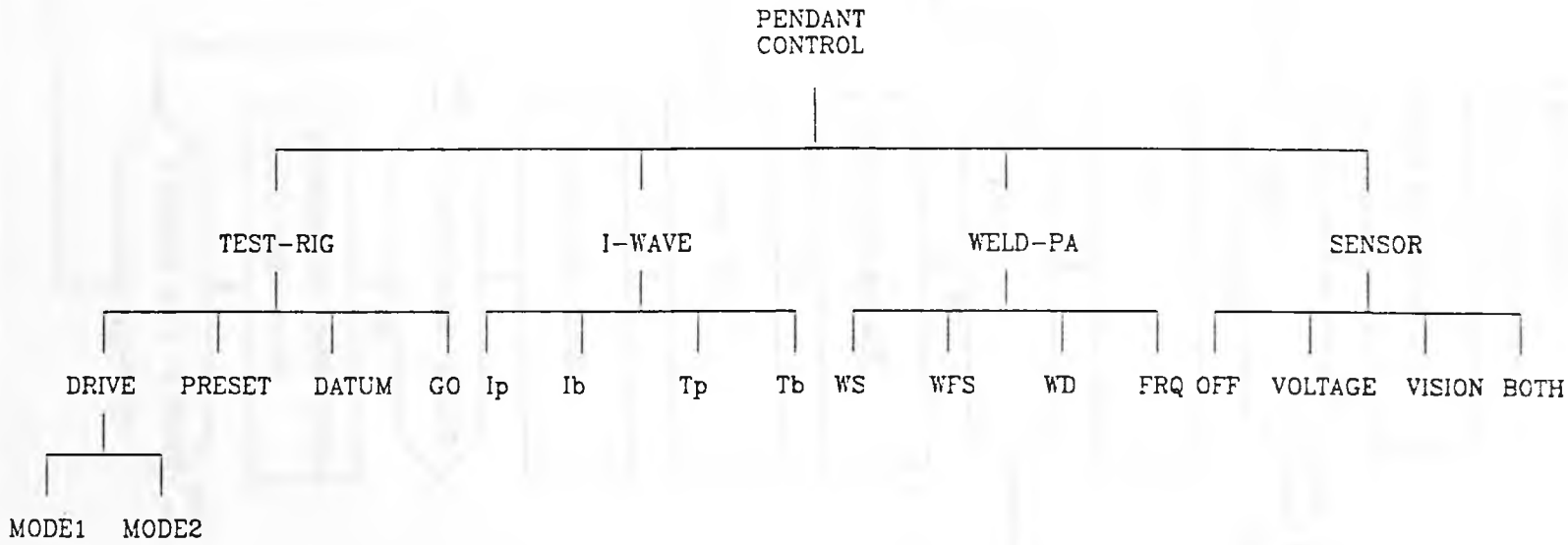


Figure-4.14 Functions of pendant control

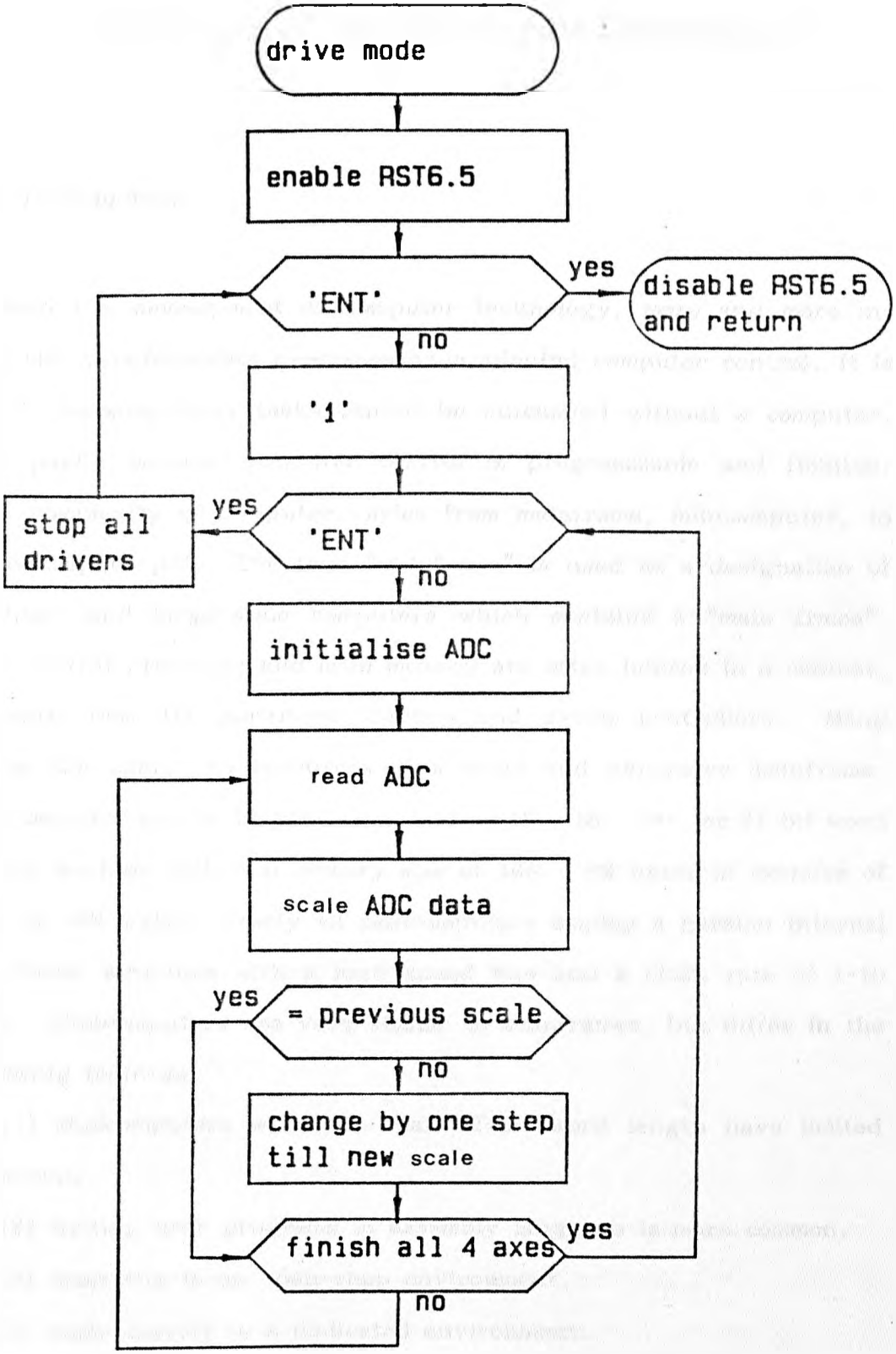


Figure-4.15 Execution procedure of drive mode1

5. HARDWARE OF ARC OSCILLATION CONTROLLER

5.1 Introduction

With the development of computer technology, more and more industrial manufacturing processes have adopted computer control. It is partly because many tasks cannot be automated without a computer, and partly because computer control is programmable and flexible. The complexity of computer varies from mainframe, minicomputer, to microcomputer [113]. The term "mainframe" is used as a designation of medium- and large-scale computers which contains a "main frame". The central processor and main memory are often housed in a cabinet, separate from the peripheral devices and device controllers. Many users can share the resources of a large and expensive mainframe. Minicomputer can be broadly classified as 12-, 18-, 24-, or 32-bit word length machine with real memory size of 16K - 8M bytes in modules of 16K or 64K bytes. Nearly all minicomputers employ a parallel internal processor structure with a high speed bus and a clock rate of 1-10 MHz. Minicomputers are very similar to mainframes, but differ in the following features.

(1) Minicomputers with less than 32-bit word length have limited precisions.

(2) Writing user programs in assembly language is more common.

(3) Some run in an open-shop environment.

(4) Some operate in a dedicated environment.

(5) No large and complex operating systems.

Since mainframes and minicomputers are large and expensive, in many case it is inefficient to use them to control a small industrial process.

In contrast, microcomputers have found wide application in industrial automation.

A microcomputer is made up of a microprocessor, memory, I/O interface units, and other logic chips. A microprocessor, or microprocessing unit (MPU), is a central processing unit (CPU) built as a single semiconductor chip. It contains the arithmetic logic unit (ALU) and control logic circuitry necessary to perform the operations of a computer program. A microcomputer can appear in either of the two forms: commercial personal computer (PC) and custom-built microcomputer. A commercial PC, such as the IBM PC used in this project, has universal I/O peripherals (e.g. monitor, keyboard, printer, etc). It is easily expandable --- both internally and by addition of peripherals. It supports commercial software packages and utilities, and is often used as a general-purpose machine for personal programming. On the other hand, a custom-built microcomputer is oriented to the control tasks defined by a particular process. It is often made on a single board, so is called single board computer (SBC). Although standard 8- and 16-bit single board computers are commercially available, it was decided to develop a dedicated microcomputer to control arc oscillation in this project.

Since the central processor used in this work is an IBM PC AT which has a 16-bit 80286 CPU, a CPU from the 8086/8088 family [114] is preferred for the custom-built computer. An Intel 8088 CPU was then selected to construct the microcomputer. Figure-5.1 shows the photographies of the computer. The microcomputer consists of an Intel 8088 CPU and its supporting components including memories, I/O interface units, and erasable programmable logic devices (EPLD), as

shown in figure-5.2. The detailed circuit diagram is drawn in figure-5.3. Appendix-A2 lists major components for the computer board. The following details the hardware design.

5.2 8088 CPU and Its Supporting Components

The Intel 8088 CPU [109] is a 16-bit microprocessor. It features an 8-bit data bus, but 16-bit internal processing. The 20 address lines are able to address 1Mbyte memory space. The CPU, working in the minimum mode, is driven by the clock generator and driver 8284A. The input clock to the 8284A is a 14.31818 MHz crystal, therefore providing a 4.773 MHz system clock. To ensure that the CPU is capable of driving all chips, the data bus is buffered by an 8286 transceiver and the address bus is latched by three 8282 latches. The 2764 (8K × 8) EPROM [111] is selected as a permanent store for data while the M5M5256AP-70 (32K × 8) RAM [115] used to store temporary data.

In a computer system, programmable clock frequencies are often used to clock data conversion (A/D or D/A) and data transmission. This can be implemented by a 8254 Programmable Interval Timer (PIT) [116]. The 24-pin dual-line chip has three independent rate generators. The output clocks are equal to the input clocks divided by the values held in the registers. The three input clocks are tied to the system clock (i.e. 4.773 MHz). Desired rates can be obtained by writing appropriate numbers to the 16-bit registers:

Register(hex)	Output Frequency	Purpose
C0: 0B6F	out0: 100 Hz	waveform generation
C1: 001F	out1: 154 KHz	TxC/RxC of 8251
C2: 001F	out2: 154 KHz	not used

The out1 serves as data transmitting and receiving clocks for serial communication. The 154KHz TxC/RxC ensures 9600 baud rate (number of bits per second). Timing frequency of waveform generation is programmed with reference to the frequency of arc oscillation. The higher the timing frequency, the smoother the waveform, but more processing time is required. For arc oscillation up to 10 Hz, the reasonable timing frequency is 100 Hz. For extremely fast arc oscillation (e.g. 50-100 Hz), a clock of 1,000 Hz is probably necessary.

There are three common approaches for I/O reading and writing: programmed I/O, interrupt I/O, and Direct Memory Access (DMA). The third method is efficient in transferring a large block of data, but not in byte reading or writing. Programmed I/O is simple, requiring no extra circuitry. The CPU examines the status of an I/O interface according to a programmed procedure. It continues the status checking routine unless the ready bit is active. The drawback of this method is sequential checking of I/O status. As a result, when the device being accessed is not ready, other devices have to wait even if they are ready. A more efficient method for I/O reading and writing is interrupt I/O, which is adopted in this work. By means of interrupts, the status checking procedure is eliminated. Instead, an I/O device requires CPU service via hardware interrupts. When the CPU is informed of an interrupt, it halts the main program, and executes the interrupt service routine (ISR). On completion of ISR, the CPU

continues the execution of the main program. An extra circuitry is needed to manage all interrupts from the I/O devices. This is done by a single chip --- 8259 Programmable Interrupt Controller (PIC) [116]. The PIC provides 8 level interrupts designated as IR0...IR7. IR0 is programmed as the highest priority interrupt, while IR7 as the lowest priority. The following table lists all the interrupt sources:

IR0: timer(100 Hz)
IR1: keyboard
IR2: μPD7002 A/D
IR3: voltage sensor
IR4: spare
IR5: 8251 RxRdy
IR6: 8251 TxRdy
IR7: spare

IR0 provides timing control for the waveform generation. A 100 Hz timing frequency derived from out0 of 8254 is designed as the interrupt source. IR1 is driven by DATA AVAILABLE of the keyboard decoder 74C922 [111]. IR2 handles potentiometers reading. The capture of arc voltage signal is implemented in the IR3 service routine. IR5 and IR6 facilitate handshaking serial communication between the computer and the host IBM PC.

5.3 Erasable Programmable Logic Device --- EP600

The hardware design of the computer adopts the advanced semiconductor technology, Erasable Programmable Logic Device (EPLD). Three 24-pin dual-line EP600 chips [117] were employed to generate chip

selects and logic signals. The EP600 has high density (over 600 gates) and high speed (typical delay time: 25ns), together with two synchronous clock inputs. Four general inputs and 16 input/output pins are available. On the digital board three EP600 chips are employed to generate all chip selects and logic signals which are normally implemented by up to 20 standard logic chips. The EPLD chips save space on the board, and additionally provide the designer with much flexibility. The chips are programmed by a PAL programmer. Source files are edited in the IBM PC and compiled by the CUPL software package [118]. Figure-5.4 shows the layouts of EP600 chips. Chip selects and their addresses (in hex) generated by the EPLD chips are listed as follows:

Supporting chips	
8259	F000
8254	8000
74LS574 (D-type flip-flop)	1000
Memory	
2764	FE000...FFFFF
M5M5256	00000...07FFF
I/O ports and interface	
8251 (USART)	6000
427 A/D (voltage sensor)	3000
428 D/A	0000
7542 D/A (waveform generation)	5000
7002 A/D (pots)	D000
74C922 (keyboard decoder)	4000
DMX402 (display)	E000

Apart from chip selects, EPLD also generates other logic signals:

8286 OE = DEN & !8259EN

Clock-A: 1/2 CPU clock

Clock-B: 1/4 CPU clock

Clock-C: 1/8 CPU clock

RDY signal for 8284

The RDY signal tells the CPU if the I/O device is ready. For slow devices such as 8251, display and 7002 A/D, the read or write cycle is lengthened by one extra clock cycle. The timing waveform is shown in figure-5.5.

5.4 Input/Output Interface

The computer has such I/O peripherals as a serial communication channel, a voltage sensor, a waveform generator, a keyboard and a display. Figure-5.6 shows external devices connected to the computer. The 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) [116] is designed to interface the microcomputer with the host IBM PC. During the development stage the serial channel was particularly useful in dumping the control codes from the IBM PC to the computer board. The control program can be tested and modified until it satisfies the user's requirements. The finalised control code can then be loaded to the EPROM and the single board computer can work as a stand-alone arc oscillation controller.

To generate a continuous waveform to control the arc oscillation, a 12-bit D/A converter (7542) [111] is interfaced to the CPU. The 7542

chip has a buffered input, simplifying its interface with the CPU. A common practice is to connect the 7542 D/A data lines to the system data bus. The output from the D/A converter is treated as the input to the driver circuit to be discussed later.

Four potentiometers were used to preset wave parameters. They are interfaced to the computer via a 7002 four-channel A/D converter [119]. An improved design is to reduce four potentiometers to two which are made in a remote control box.

The interface unit for voltage sensor is basically an 8-bit A/D converter (ZN427) [111]. The arc voltage is extracted by a divider with an appropriate ratio, then buffered by an op-amp. The CPU reads the sampled voltage from the A/D.

A 16-key (4 row × 4 column) keypad [119] and a dual-line (2 × 40) DMX402 display [119] are used as I/O peripherals enabling the user to talk to the computer. A keyboard decoder 74C922 [111] is used to interface the keypad to the CPU. The 16 keys are decoded from 0 to F (hex). The keyboard ASCII codes are software selectable. All characters corresponds to ASCII codes except artificial "ENT" which is coded as 7F (hex). As far as the display is concerned, no special interface unit is needed. It can be directly connected to the system bus.

5.5 Drive Circuit

The drive circuit is not part of the computer circuit, but has a close link with the computer. It is designed to power the computer generated

waveform, supplying a current wave to the coil which in turn generates an electromagnetic field. The circuit uses two normal op-amps (741) [111] and one power op-amp (165) [119], as shown in figure-5.7. The first stage op-amp serves as a feedback unit to the 7542 D/A, resulting in a uni-polar waveform. The second stage op-amp buffers the output from the first op-amp, and more importantly converts the uni-polar waveform to bi-polar one. The variable resistor R_4 is used for zero adjustment. It is done by simply outputting 0800 (hex) from the CPU to the D/A and tuning R_4 until the output from the second stage op-amp is zero. The variable resistor R_6 controls the voltage gain (G):

$$G = (R_5 + R_6) / (R_3 + R_4) \qquad \text{(equation-5.1)}$$

The voltage output from the second stage op-amp (V_{ctr}) ranges from $-V_{ref}G$ to $+V_{ref}G$ where V_{ref} is 2.5 volts.

Because an op-amp like the 741 can only supply tens of milliamperes current, further power amplification is needed to output higher currents for the magnet coil. The 165 power amplifier is chosen to meet this need because of its low cost and as high as 3A current output. As figure-5.7 shows, the 165 amplifier is configured as a current source. The current passing through the coil, designated as coil current, is dependent on V_{ctr}:

$$I_{coil} = V_{ctr} / R_7 = V_{ctr} \qquad \text{(equation-5.2)}$$

By adjusting the voltage gain (G), i.e. tuning R_6, V_{ctr} is limited to the range from -1.5V to +1.5V, i.e. I_{coil} from -1.5A to +1.5A. A Farnell NA075402 power supply [120] is used to supply +12 V, -12 V to

the power op-amp. The current ratings at +12 V, -12 V are 3A and 2A + 0.5A, which are sufficient to power the analogue circuit. In addition, 8A at 5V can be supplied to the computer board. It is essential that a heat sink is attached to the power op-amp in order to prevent overheating.

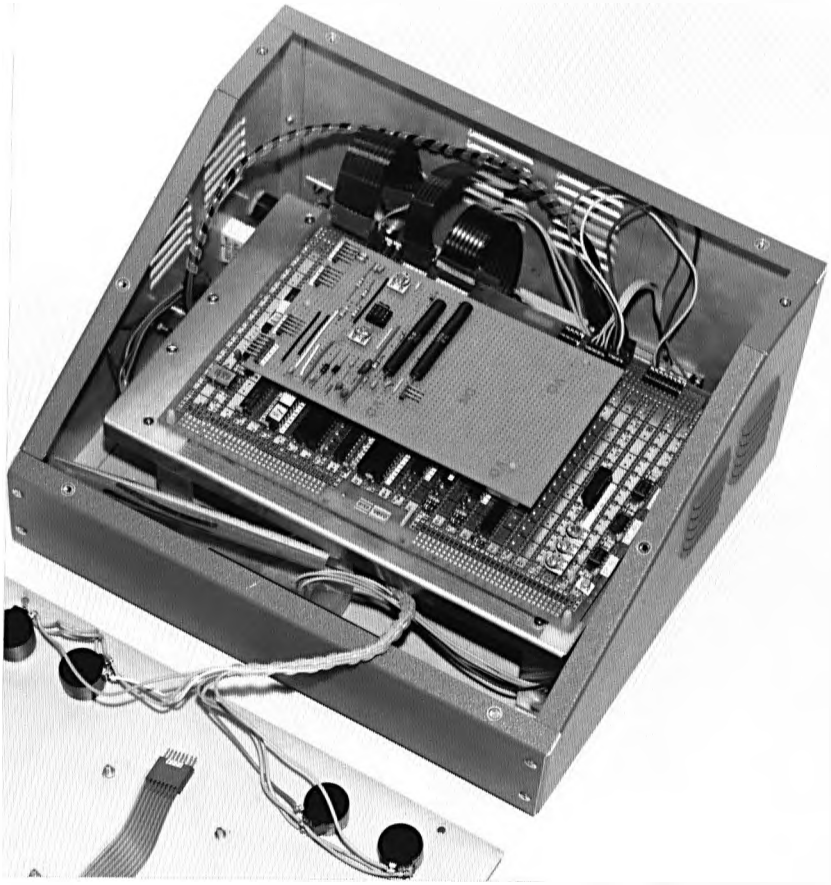
In constructing the drive circuit board, it must be noticed that signals output from a normal op-amp are only a few milliamperes and readily pick up electrical noise when being transmitted through a long distance (half meters, say). Practically, the feedback op-amp is placed as close to the 7542 A/D converter as possible. The second stage op-amp is also put on the computer board. The power amplifying circuit is built on a separate bread board in order to reduce the interference of analogue signals with the computer board. It is stacked above the computer board, as shown in figure-5.1(a). The power op-amp with the heat sink is separate from the two boards, and linked to the analogue board via short and thick wires.

5.6 VHF Protection

Measures must be taken to isolate high frequency sources from the computer. The computer must be placed in a grounded aluminium box which deflects and absorbs very high frequency signals. Figure-5.1(b) shows the appearance of the controller with the microcomputer board and drive circuit inside. Filtered mains and connectors are recommended to isolate very high frequency sources from the main power supply. For further protection, the computer board and the drive circuit board can use two separate power supplies. Connections between them are implemented via opto isolators. Under the conditions,

even if the very high frequency signals enter the drive board, they are cut off from the computer.

In this chapter, hardware design of the arc oscillation controller has been discussed. It includes a microcomputer built with a 16-bit Intel 8088 CPU, and a drive circuit supplying a current waveform to the magnetic coil. The employment of erasable programmable logic devices made the computer board very compact, and design work flexible. The computer can work either as a stand-alone device or in communication with the host IBM PC. Software design of arc oscillation control will be the focus of next chapter.



(a) Inside the box

(b) The front panel

Figure-5.1 Photographies of the 8088 computer

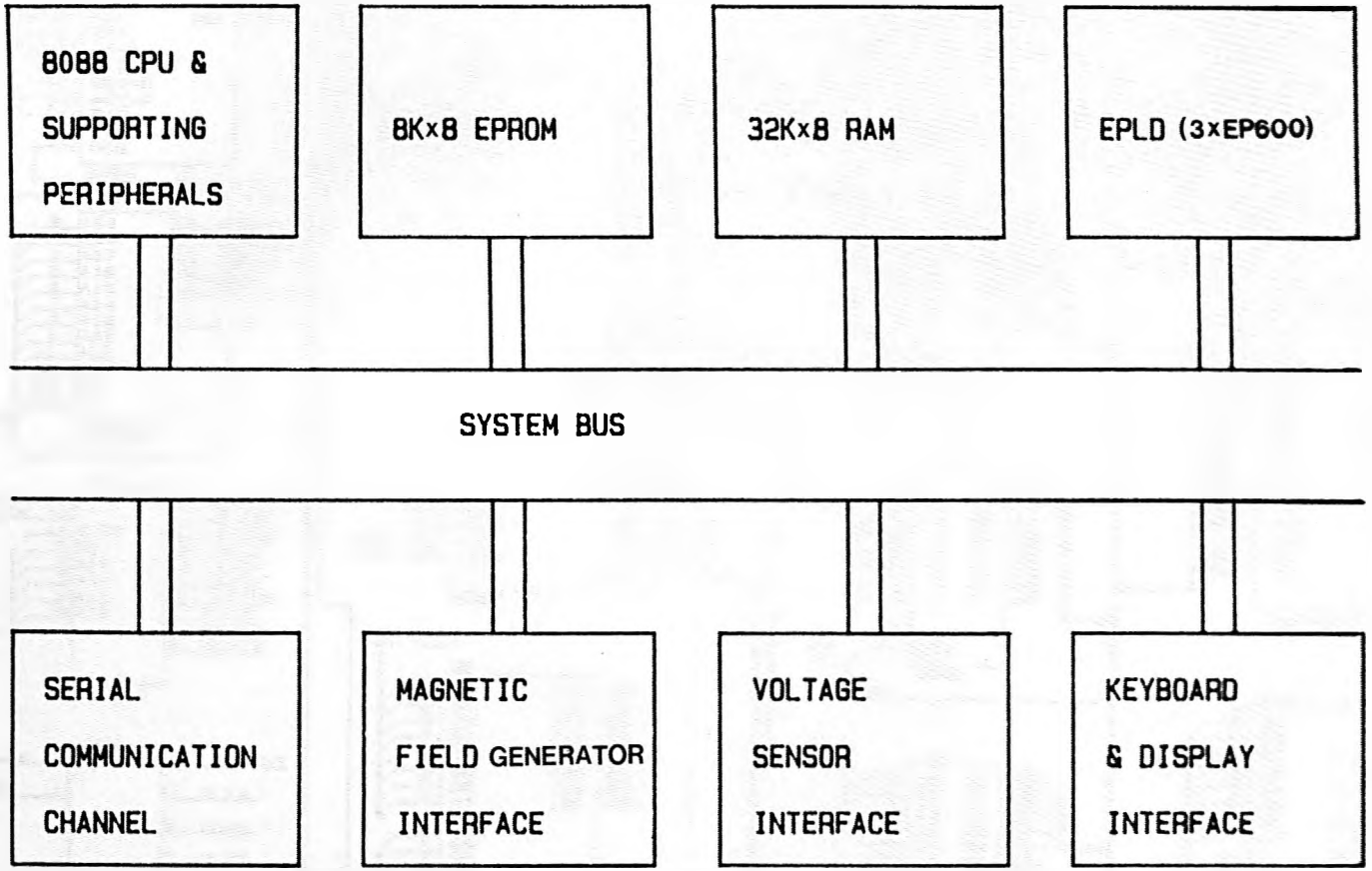


Figure-5.2 Block diagram of the 8088 computer

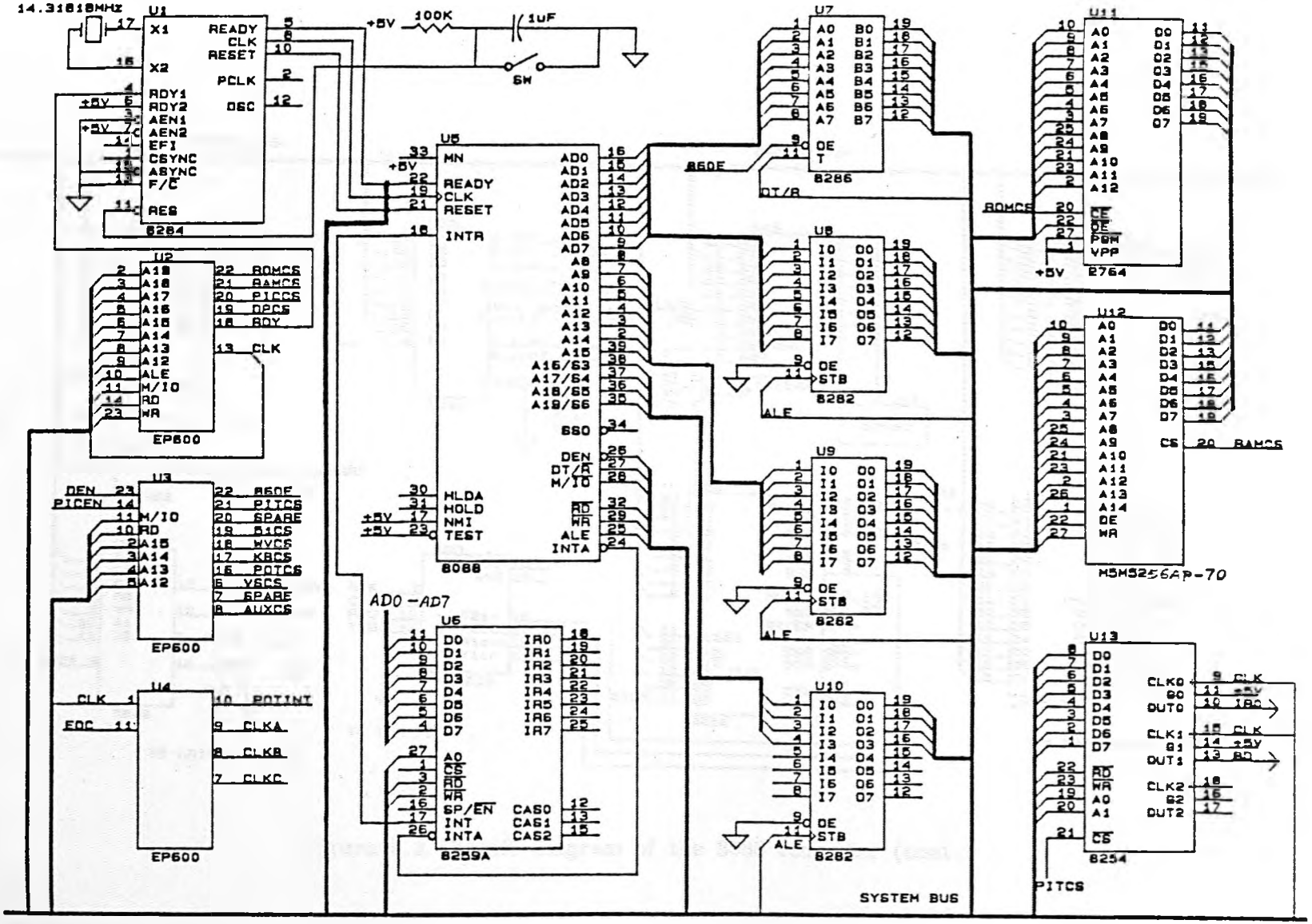


Figure-5.3 Circuit diagram of the 8088 computer

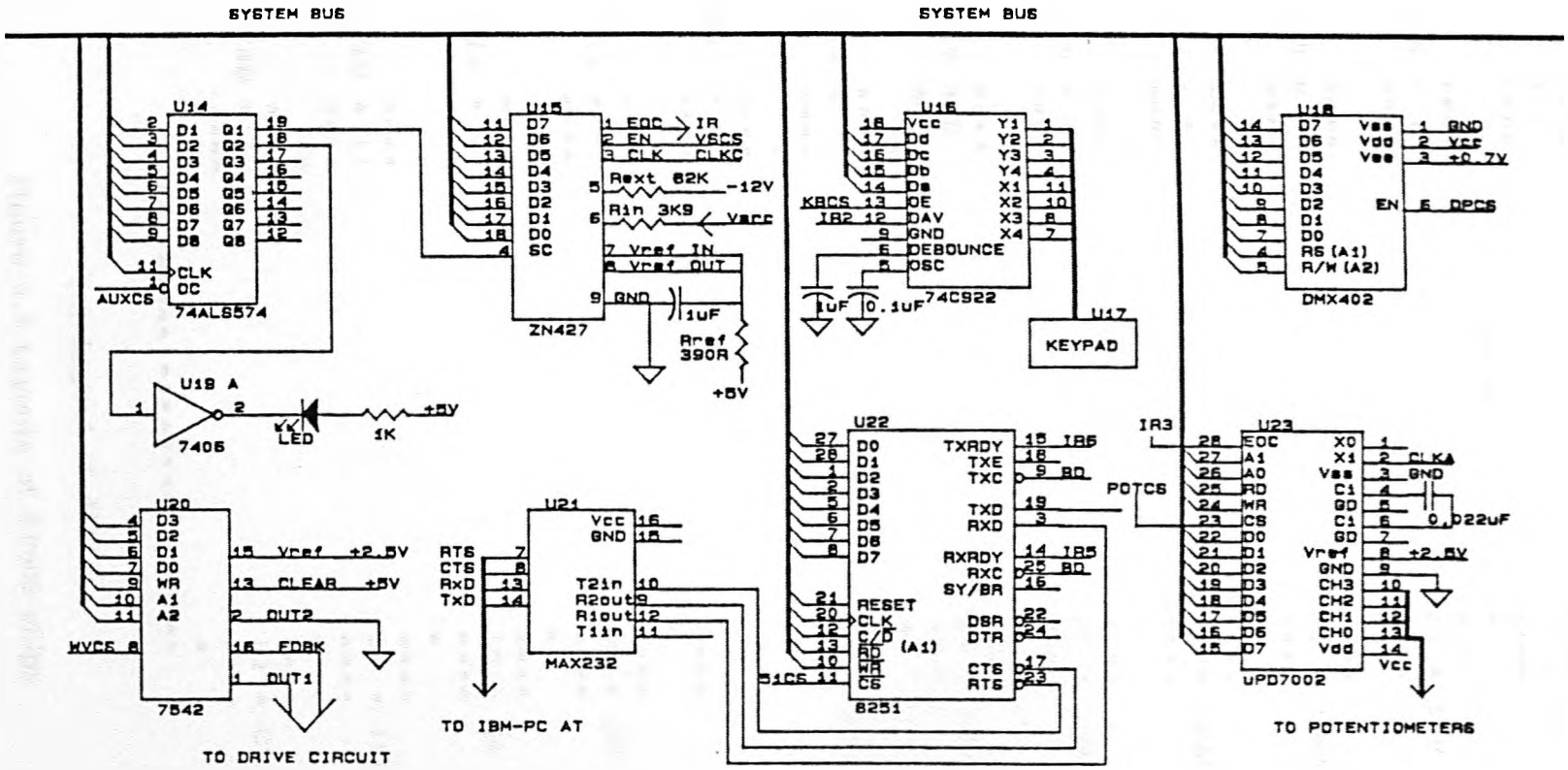
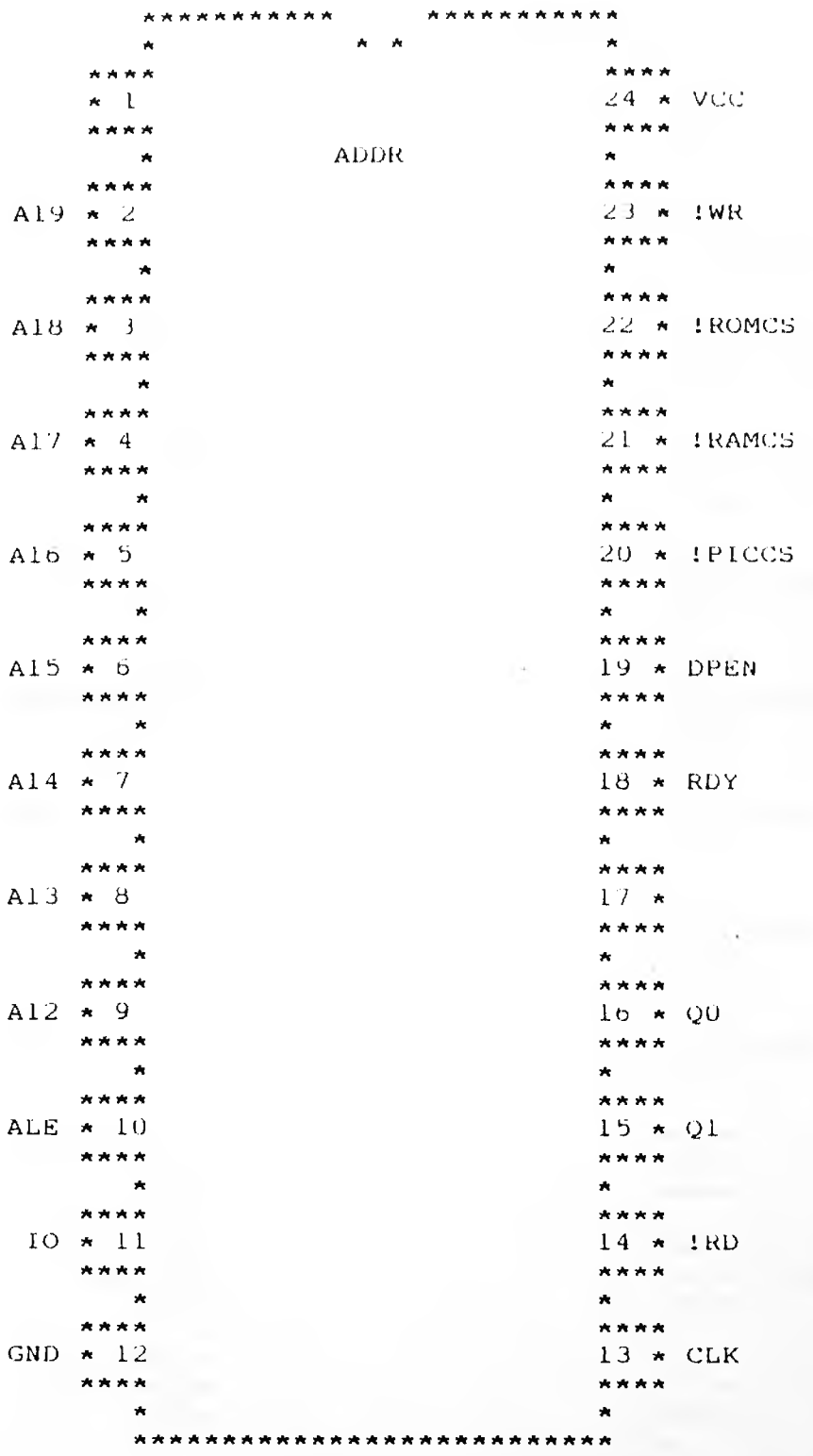
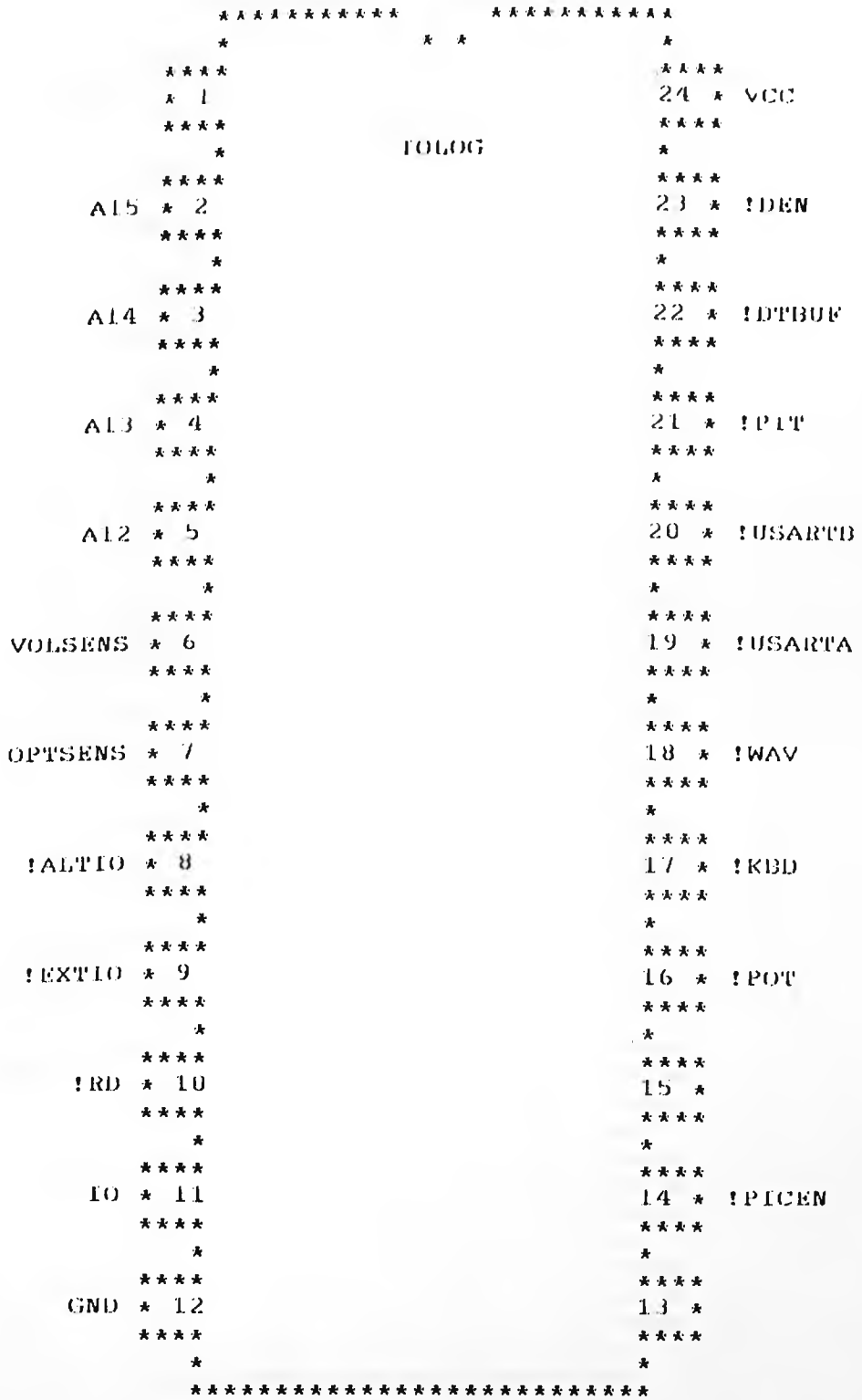


Figure-5.3 Circuit diagram of the 8088 computer (cont.)



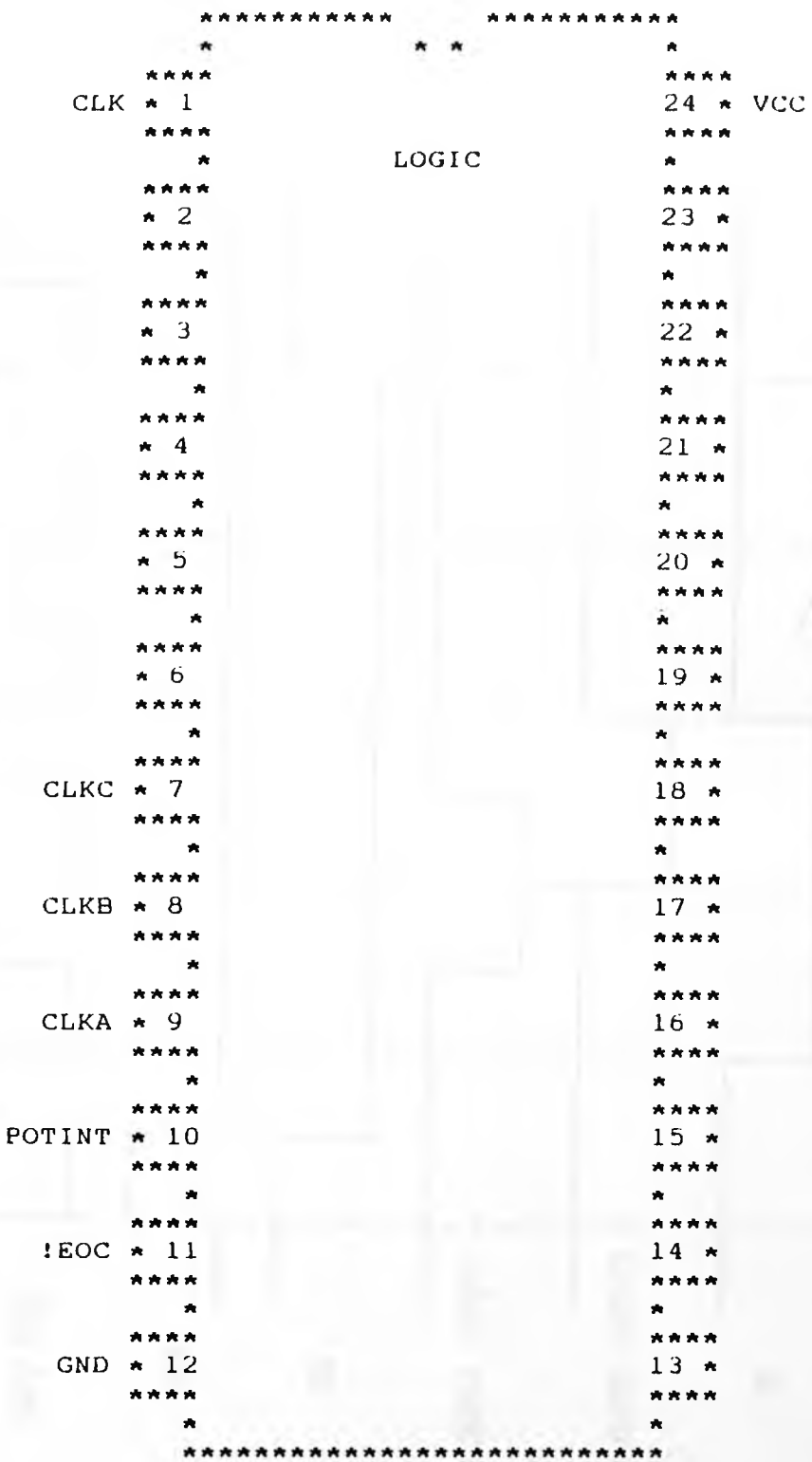
(a) Layout of ADDR

Figure-5.4 Layouts of EP600 chips



(b) Layout of IOLOG

Figure-5.4 Layouts of EP600 chips (cont.)



(c) Layout of LOGIC

Figure-5.4 Layouts of EP600 chips (cont.)

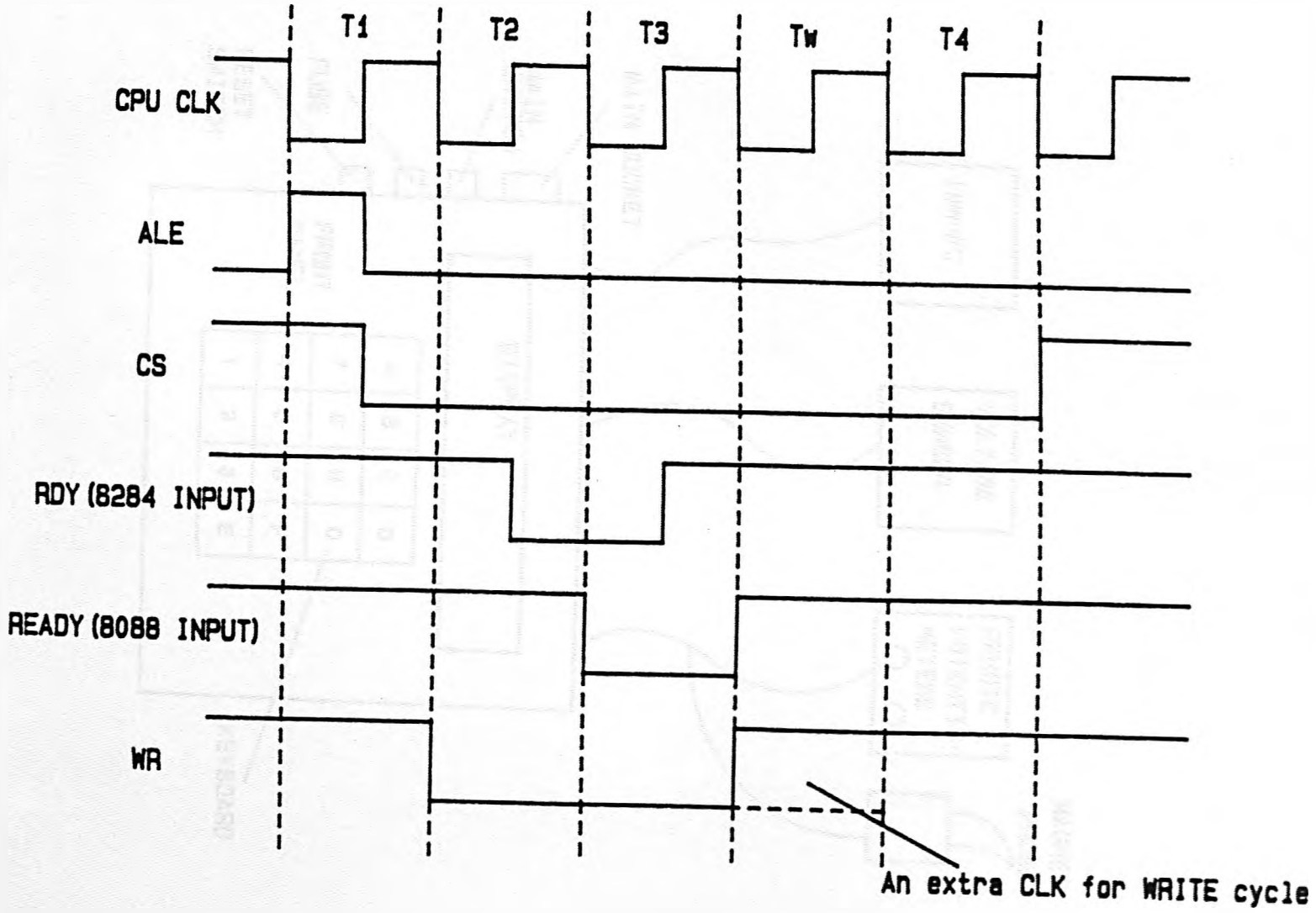


Figure-5.5 Timing waveform of the 8088 computer

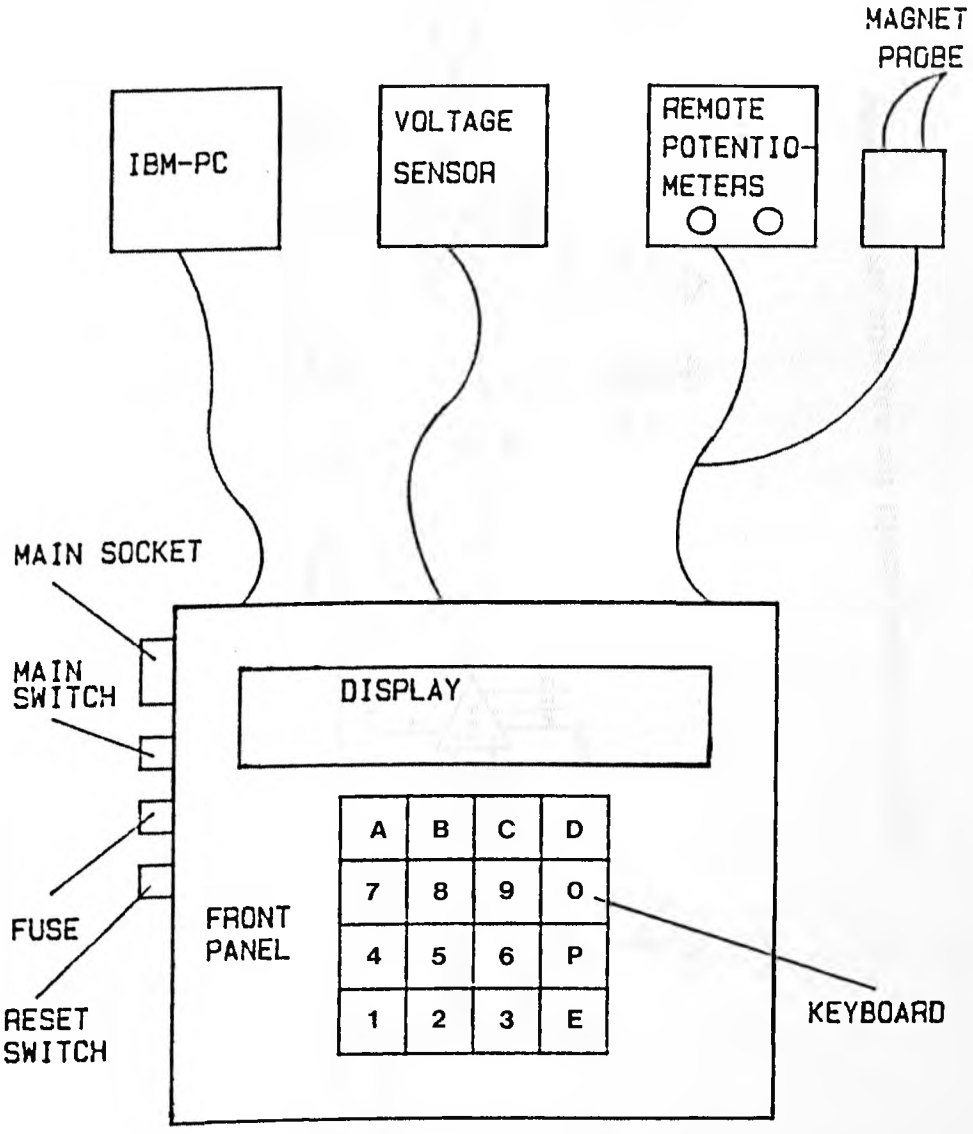


Figure-5.6 Interconnections between the 8088 computer and external devices

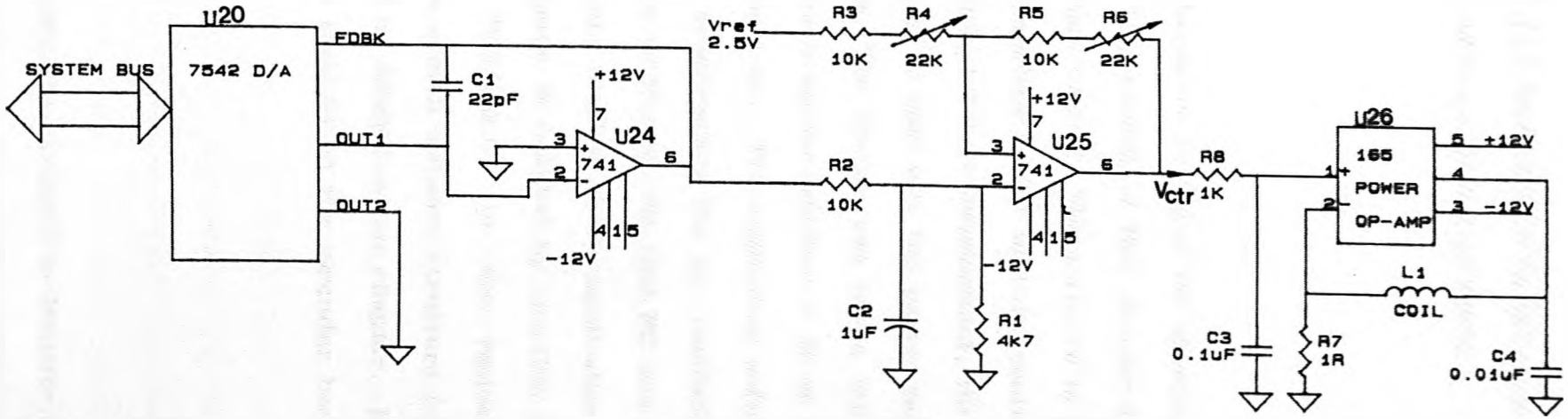


Figure-5.7 Drive circuit for magnetic arc oscillation

6. SOFTWARE IMPLEMENTATION OF ARC OSCILLATION CONTROL

6.1 Introduction

In the last chapter, hardware design of the microcomputer and drive circuit was discussed. The subject of this chapter is software implementation of arc oscillation control. The software is designed to generate a programmable waveform which in turn controls the magnetic arc oscillation. To facilitate software development, the control software in the application computer is split into two parts: residential software and application software. The former was blown into the ROM, and therefore resident in the computer whether it is on or off. It boots the computer once switched on. The application software generates a digital waveform which synchronises the arc oscillation. In order to be modified readily, it is written in the IBM PC and allocated in the RAM of the 8088 computer. Loading of the application codes from the IBM PC to the 8088 computer is fulfilled by the IBM PC software and the loading program "BOARD.C" in the residential software. Figure-6.1 illustrates the overall software structure for arc oscillation control. Each module will be detailed in this chapter. Real-time control using the voltage sensor resident on the computer board is discussed afterwards.

6.2 IBM PC Software

This part of the software was designed to transfer a file from the IBM PC to the 8088 computer via the serial channel com1. The main

program "TRANS.C" was written in the C language [121], as listed in appendix-C1. Since the program runs in the IBM PC, an MS-DOS supported C compiler can be employed to compile it. The Lattice-C compiler [122] was selected due to its support of versatile routines. Additionally, a library of user routines PCLIB.A86 (appendix-C2) was designed to cope with low level I/O of the IBM PC. Since these routines are closely associated with the IBM PC hardware, they are written in the assembly language [123]. The two pieces of programs are compiled separately, producing object files: TRANS.OBJ and PCLIB.OBJ. Two object files are linked together by PLINK86 [123] to generate an executable file TRANS.EXE which can be run in the IBM PC. This procedure is illustrated in figure-6.2.

The communication channel is configured as: 9600 baud rate, 8-bit character, 1 stop bit and no parity. DTR and RTS are forced active before data communication. Data transmitting and receiving are carried out through polling the status of control registers. For the IBM PC to receive a character from the 8088 computer, it polls the Data Ready bit (bit0) of the Line Status Register (LSR). If the Data Ready bit is active, the IBM PC reads the data from the Receiver Buffer Register (RBR), as shown in the function rdcom1() in appendix-C1. Likewise, to send a character from the IBM PC to the 8008 computer, the Clear to Send (CTS) bit (bit3) of the Modem Status Register (MSR) is tested. If the CTS is active, a character is written to the Transmitter Holding Register (THR). To ensure the transmission more reliable, a delay routine can be appended at the end of each of character transmission. This clearly slows down the transmission speed. An improved method is that the IBM PC sends data line by line, then waits for an acknowledgement from the 8088 computer. Detailed procedure is:

(1) Read a character from the file. Write it into the buffer "buf0" and print it on the screen. If not carriage return (CR), loop (1). Otherwise, go to (2).

(2) Send the line of characters in the buffer "buf0" to the 8088 computer one by one.

(3) Wait for an ($) acknowledgement from the 8088 computer, then go to (1) for the next line unless the end of file (EOF) is encountered.

This piece of software is particularly useful at the development stage. It downloads the application codes (in the Intel hex format) to the 8088 computer. A loading program in the 8088 computer receives these codes and load into the RAM. The application software can be readily changed during testing.

6.3 Residential Software

This part of the software is resident in the 8088 computer whether the computer is on or off. It consists three modules: BOOT.A86, BOARD.C, and ARCLA.A86. BOOT.A86 is a booting program (appendix-D1), and closely related to the computer hardware. Its functions include:

(1) Allocate memory space for variables.

(2) Set the stack point (SP) to the address 8000 (hex), just above the RAM.

(3) Initialise the following devices: 8529 PIC, 8254 PIT, 8251 USART and display.

(4) Set timer to zero.

(5) Set interrupt vector.

(6) Send stars to an IBM PC terminal, wait for

T: Send current time to the terminal, or

C: Enter the loading program.

The loading program BOARD.C (appendix-D2) receives control codes from the IBM PC and loads them to the RAM. Since it dealt with a large amount of character processing, the C language was chosen to construct this module. The 8088 computer receives control codes (in hex) line by line from the IBM PC, separated by a carriage return (CR) character. In other words, it buffers characters from the IBM PC till a CR character is received, then carries out processing. It checks if the received characters is in the range from 0 to F, and tests the checksum. If the received line is right, it sends an acknowledgement ($) to the IBM PC, and waits for the next line. Otherwise it requests the IBM PC to re-send the line. Once the end of file is reached, the file transfer is terminated.

ARCLA.A86 (appendix-D3) is a collection of user routines which can be called from other modules. It contains routines such as character and string input/output via the serial line, low level data transfer, read current time, i/o read and write, interrupt vectoring, and interrupt handler.

Since the residential software runs in the application environment, the executable codes must be burned into the ROM in a certain format. The Intel Software Development Tools [125] have provided utilities to accomplish this task. Figure-6.3 shows the procedure of compiling, linking, locating, and hex conversion. The booting program and user routines are compiled by the Intel 8086 assembler (ASM86), generating object files with extension .OBJ. The loading program is complied by

the Intel C compiler (CC86) [126], resulting in an object file BOARD.OBJ which can be linked to those generated by ASM86 by LINK86. The link file (BOOT.LNK) is allocated from the address FFFF0(hex) downwards by LOC86, then converted to a hex file BOOT.HEX. The hex codes are then burned into the ROM by a programmer. In the linking procedure, the booting object file BOOT.OBJ is the first one to be linked, therefore at the top of the ROM. Once the computer is switched on, it jumps to the address FFFF0 (hex), executes the booting program. If the IBM PC is connected to the 8088 computer and configured as a kermit terminal, stars are continuously displayed on its screen. If "T" is pressed in the terminal, the current time in the 8088 computer is displayed. If "C" is struck, the execution procedure jumps from the booting program "BOOT.A86" to the loading program "BOARD.C". BOARD displays the starting address on the IBM PC monitor. If the address is the starting address of the application codes, execution of the application program is started by pressing "S". Otherwise, press space bar, exit from kermit. Then execute TRANS in IBM PC for downloading the application codes from the IBM PC to the 8088 computer.

6.4 Application Software

The application software is designed to generate a computer waveform which synchronises the arc oscillation. It includes a user library of "C" routines ARCLC.C (appendix-E1) and a main program WEAVE.C (appendix-E2). Both are compiled under the Intel C compiler (CC86), resulting in the object files: ARCLC.OBJ and WEAVE.OBJ. The linking, locating, and converting into a hex file are exactly the same as the residential software. The only difference is that during

the development stage ARCLC.OBJ,WEAVE.OBJ and ARCLA.OBJ are linked into WEAVE.LNK, and allocated in the memory space of the RAM (figure-6.4), then downloaded from the IBM PC to the computer via the serial line. The IBM PC software TRANS.C and the loading program BOARD.C carry out this work, as discussed above. The finalised application software can be linked with the residential software, and burned into the ROM. The entire program includes BOOT.A86, ARCLA.A86, WEAVE.C and ARCLC.C (figure-6.5). In this case, the computer can work independently of the IBM PC.

The library of user routines ARCLC.C serves the main program. It can be divided into the following groups:

(1) Masking interrupts. Individual interrupt can be enabled or disabled.

(2) Programming the rate frequencies of the 8254 PIT, particularly the timing clock of waveform generation: pit0_rat(i).

(3) Keypad reading routines.

(4) Wave parameters reading routines from the potentiometers.

(5) Wave parameters reading via keypad.

(6) Display routines.

(7) Waveform generation routines.

The Waveform generation routines are most important in the control software. Basically, every time the timing clock (out0 of 8254) interrupts the CPU (IR0), the CPU halts the main program, then enter the routine wavout() shown in appendix-E1. It calculates the positive integer i to be output, and calls the routine wv_write(i), listed in appendix-E1, which writes i to the 7542 D/A converter. Then the CPU returns to the main program. With the timing clock of a constant

frequency (100 Hz, say), a discrete wave is produced, which is smoothed in the drive circuit to form a continuous waveform. It is necessary to consider the design of the waveform to meet the requirements of the arc oscillation.

By means of triangular waveform, the arc oscillates across the central line without dwelling at each side. In the case of a square waveform, the arc flicks from one side to other, probably causing incomplete fusion of the central line. It was then decided to use a trapezoidal waveform to control the arc oscillation. Four parameters are used to describe the wave: field bias (fb), field amplitude (fa), weaving time (tw). and dwell time (td), as defined in figure-6.6. They can be selected from either the keyboard or potentiometers. The field strength is scaled from 0 to 10, where 10 corresponds to the peak field in one direction (or coil current = 1.5A). and 0 to that in the reverse direction (or coil current = -1.5A). The term "field bias" is used to indicate the central position of arc oscillation. Fb=5 represents symmetrical arc weaving. Fb=0 stands for the extreme bias to one side wall while fb=10 for that to the other side wall. The weaving and dwell time range from 0.000 to 5.000 seconds. The period (T) is,

$$T = 2(tw + td) \qquad \text{(equation-6.1)}$$

The maximum and minimum fields can be respectively expressed as:

$$fmax = fb + fa/2 \qquad \text{(equation-6.2)}$$

$$fmin = fb - fa/2 \qquad \text{(equation-6.3)}$$

Since fa and fb are preset independently, either fmax or fmin can exceed the valid range from 0 to 10. In this case, fa is re-evaluated, and the fa setting overridden. If fmax > 10, fa is evaluated as $2 \times (10\text{-fb})$, and fmax is equal to 10. If fmin < 0, fa is evaluated as $2 \times$ fb, and fmin is equal to 0.

The function of digital output as time within a period can be written as follows:

$$A = \begin{cases} fmin + fa \times t / tw & (0 \le t < tw) \\ fmax & (tw \le t < tw + td) \\ fmin + fa \times (T - t) / tw & (tw + td \le t < T - td) \\ fmin & (T\text{-}td \le t < T) \end{cases}$$

(equation-6.4)

The above calculations are implemented in the routine wavout(). Figure-6.7 illustrates the flow chart of this routine.

Having constructed a library of user routines, the main program "WEAVE.C" to control the arc oscillation can be easily designed. Most functions in the main program are called from the user library except manual display routines. Figure-6.8 shows the procedure of software execution. Starting from the top, it has four modes: PRESET, WEAVE, REMOTE and FDBACK.

(1) PRESET. The four wave parameters can be preset from the keyboard.

(2) WEAVE. The arc is oscillated according to the preset parameters.

(3) REMOTE. The wave parameters can be changed during arc oscillation.

(4) FDBK. Feedback control, voltage sensor and/or vision system. is switched on/off.

By means of PRESET, WEAVE and REMOTE modes, open-loop arc oscillation control is implemented. However, close-loop control is possible with the assistance of the voltage sensor and vision system. The voltage sensor resident on the computer board can be effectively used to control the arc length and carry out seam tracking.

6.5 Real-Time Control Via Voltage Sensor

6.5.1 Arc Length Control

The principle and methods of arc voltage sensing have been reviewed in the chapter 2. Two common applications are arc length control and seam tracking. The former is particularly attractive in TIG welding. Generally, a constant arc standoff is always desired in TIG welding. Figure-6.9 shows open-loop and close-loop control of arc length. In open-loop control, the arc standoff is usually set up by the welding operator according to the welding specification before welding. However the arc length can be altered by an external disturbance such as an irregular joint preparation and workpiece distortion. Therefore a close-loop feedback control based on sensing information is important in high-quality TIG welding.

To realise a close-loop feedback control, the control variable (arc voltage) must be sampled at intervals, and fed back to the torch controller. In this particular application, the voltage sensor resident

in the 8088 computer samples the arc voltage. Since there exists a linear relationship between the arc voltage and arc standoff, as mentioned in chapter 4 , the following equation can be derived:

$$\Delta l = Kp \times \Delta V \qquad \text{(equation-6.5)}$$

where Δl is the variation in arc length, ΔV is that in arc voltage and Kp is a proportion constant depending on the welding conditions, e.g. welding current. This equation indicates that arc length correction can be made in a proportional way when the change in arc voltage is captured by the sensor. Because the arc voltage is rather sensitive to welding conditions, it is necessary to use some kind of digital filtering to process the arc voltage signal. A simple method is to average arc voltage readings in an oscillation cycle.

6.5.2 Seam Tracking

Experiments show that on approaching the sidewall, the arc voltage drops down. This is because the arc flicks towards the sidewall when being close to the sidewall. If the torch is oscillated across the gap, the arc voltage waveform will look like figure-6.10. The two dips at the left (Vl) and right (Vr) sidewalls indicate the symmetry of the torch movement with reference to the central line of the gap. As long as the torch is oscillated equally towards to both sidewalls, Vr is equal to Vl and no correction is needed. However, if the torch movement is off the centre and closer to the left (right) sidewall, Vr is greater (less) than Vl. In either of these case, correction of torch movement is necessary. Take Vr > Vl as an example, the correction procedure looks like:

(1) Calculate the error $\Delta V = Vr - Vl$.

(2) Calculate the number of steps to move the axis of the torch movement towards the right sidewall.

(3) The 8088 computer sends the information to the torch manipulator for a substantial correction.

Real-time control via the arc voltage sensor is relatively simple. It involves simple electronic hardware and control algorithms. However the uses of through-arc sensors are limited by their sensitivity to welding conditions.

In this chapter, software implementation of arc oscillation has been discussed. IBM PC software, residential software and application software were detailed. At the end, real-time control using the arc voltage sensor was discussed. However, a fast vision system is necessary for advanced control. This will be the focus of the next chapter.

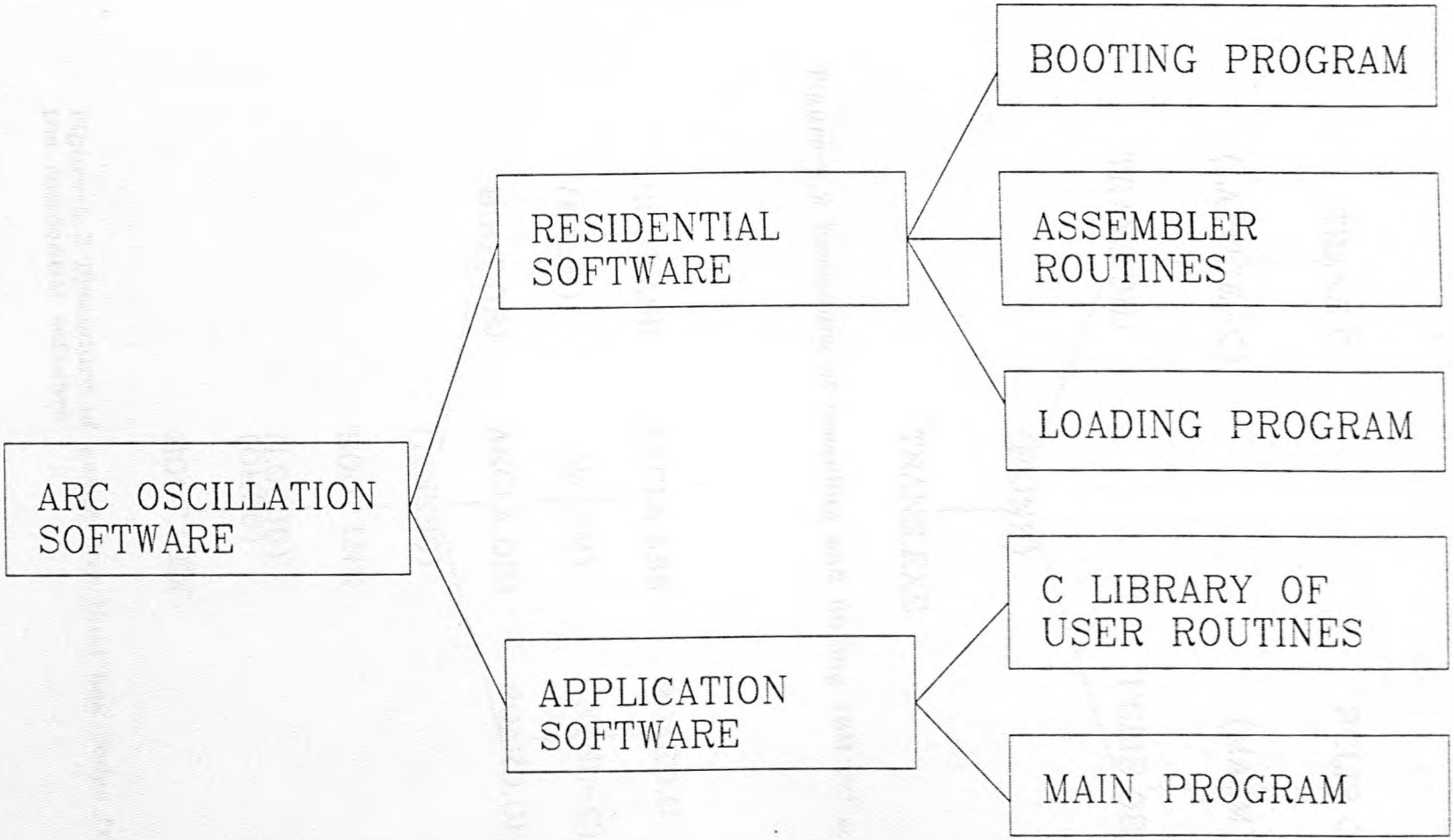


Figure-6.1 Software structure for arc oscillation control

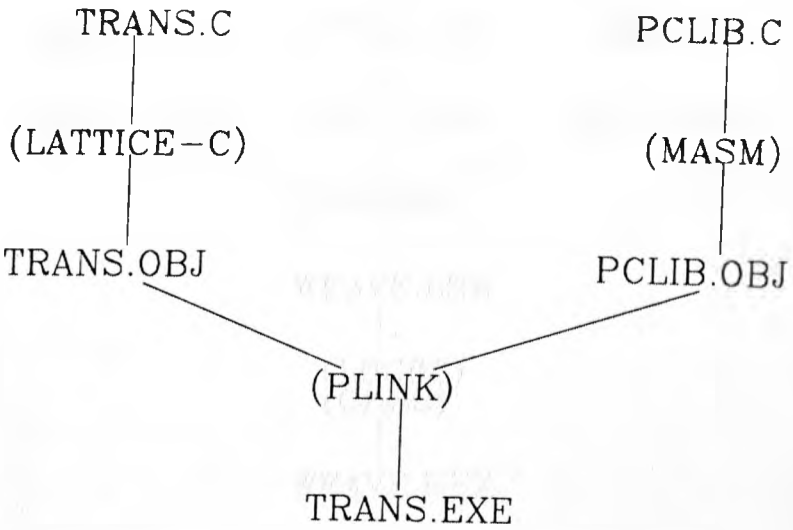


Figure-6.2 Procedure of compiling and linking IBM PC software

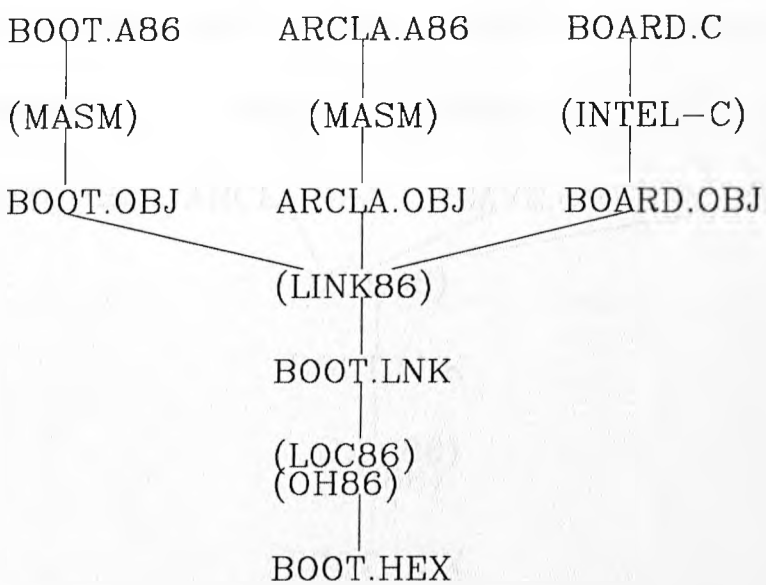


Figure-6.3 Procedure of generating Intel hex codes from the residential software

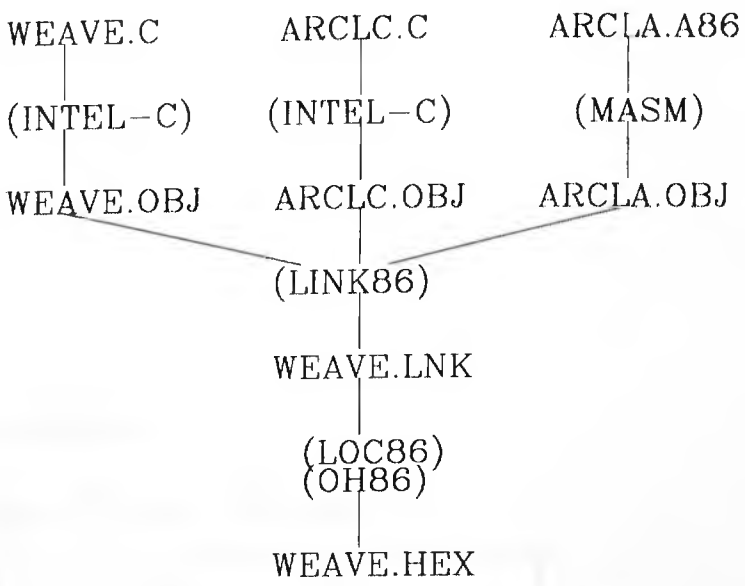


Figure-6.4 Procedure of generating Intel hex codes from the application software

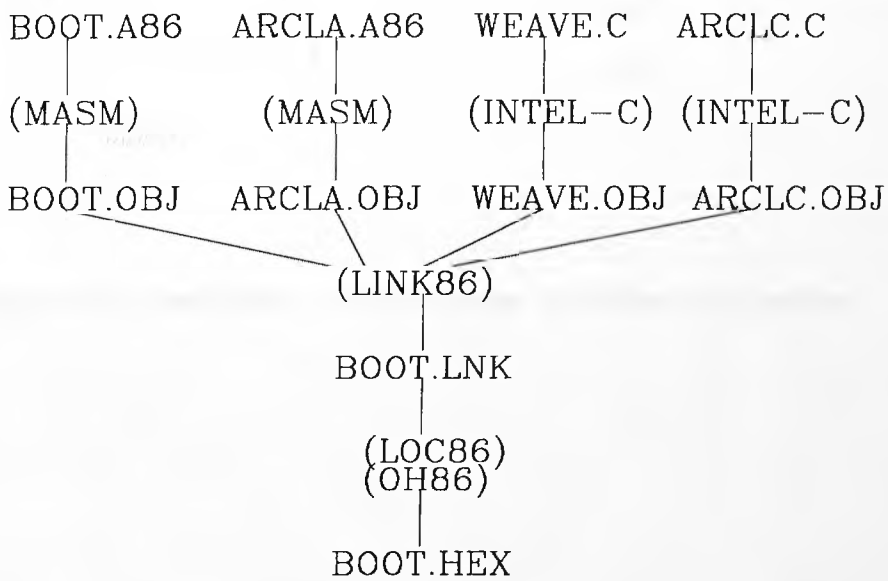


Figure-6.5 Procedure of generating finalised control codes

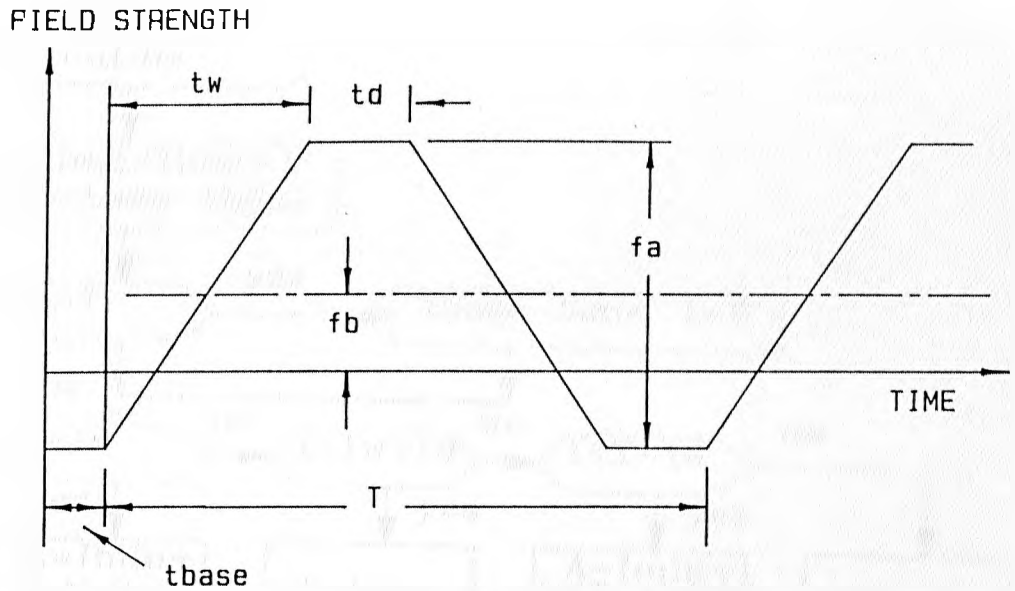


Figure-6.6 Definitions of oscillation waveform parameters

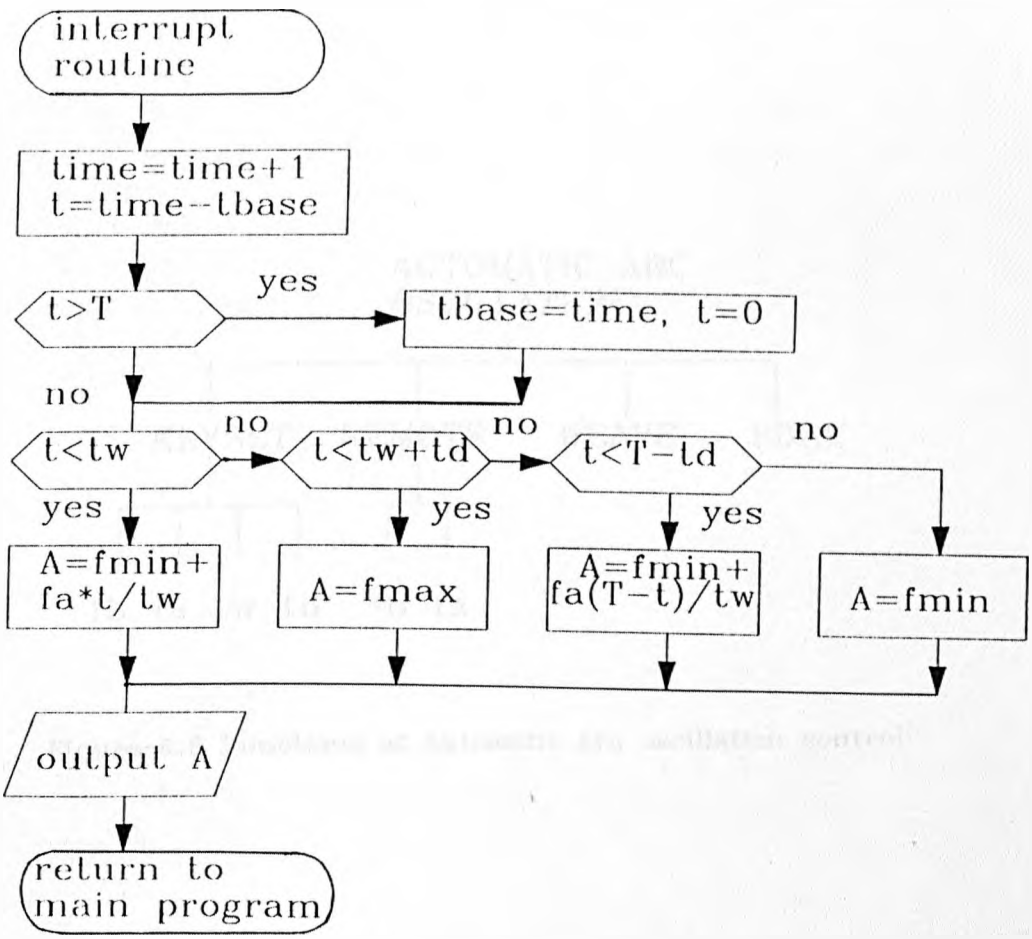


Figure-6.7 Flow chart of waveform generation routine

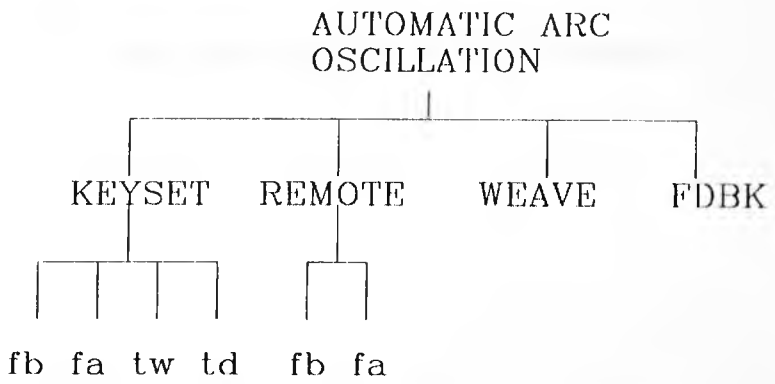
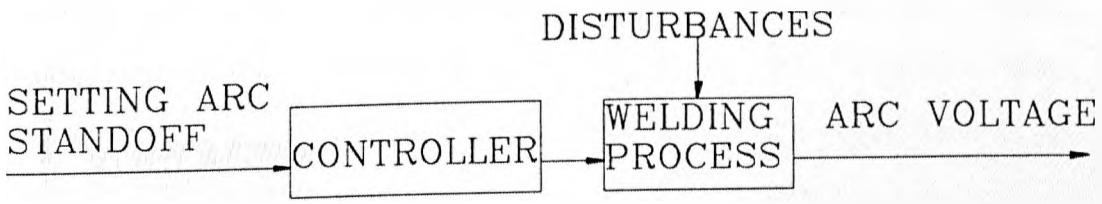
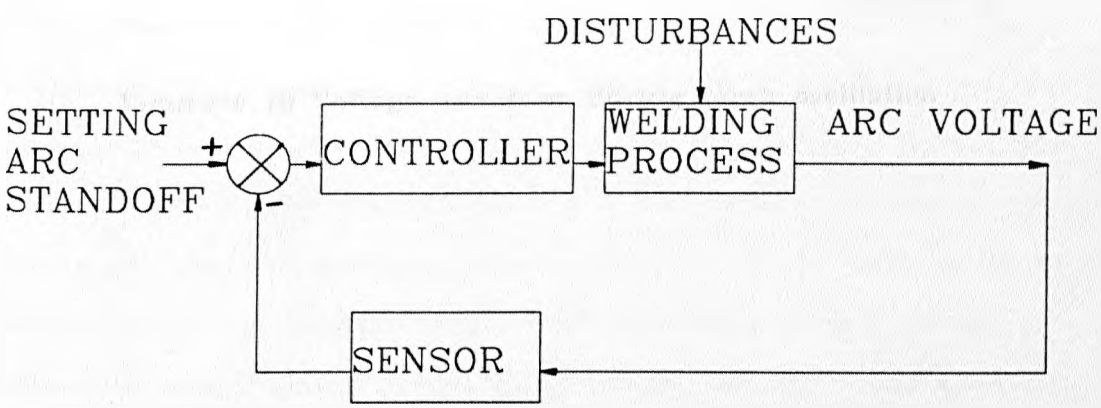


Figure-6.8 Functions of automatic arc oscillation control



(a) Open loop control of arc length



(b) Close loop control of arc length

Figure-6.9 Open loop and close loop control of arc length

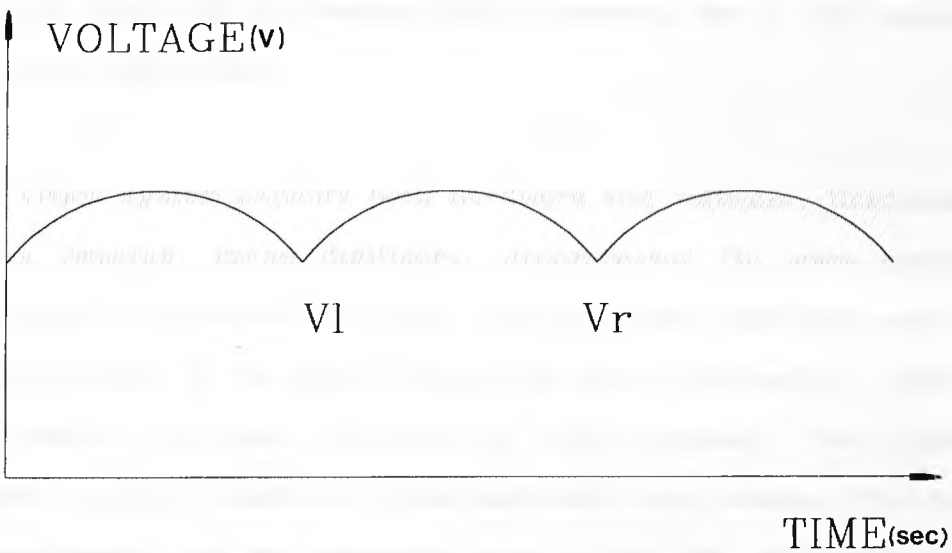


Figure-6.10 Voltage waveform during torch oscillation

7. VISION-BASED REAL-TIME CONTROL

7.1 Introduction

For many tasks that a robot performs, vision is the most important source of information about its environment. The recognition of surrounding objects and the perception of certain relations among these objects provide vital information for robot control. Since the late 1960s machine vision has experienced many advances, but is still considered relatively undeveloped.

A vision system exploits both hardware and software. Hardware includes cameras, image digitisers, preprocessor (in some systems), computers and interface equipment. Software deals with image analysing and conversion of the digital image into useful information. Reference [127] outlines hardware available for vision systems. Two types of cameras commonly used are vidicon and solid-state cameras. The former is inexpensive and has relatively high speed. The disadvantages of vidicons are their fixed scan pattern, limited spatial resolution and the nonlinearity of the scan due to distortions in the electrical field in the scanner itself. Solid-state cameras are of two types: charge-coupled devices (CCDs) and charge-injected devices (CIDs). Both of these devices are charge-transfer devices (CTDs). They have excellent geometric accuracy. Linear arrays may have up to 512 × 512 elements per device, but are quite expensive. The video signal from a camera is input to the image digitiser known as the frame store. The entire image is divided into a two-dimensional array (typically 128×128, 256×256, 512×512) of pixels, namely picture elements. Binary video

requires only 1 bit per pixel. Eskenazi [128] describes a TV digitiser that reads in the binary image at video frame rates. Gray level video may have as many as 256 levels and require 8 bits per pixel. For a $256 \times 256 \times 8$ frame store, 64Kbyte memory is needed. Image analysing is the most important task to extract vision information from an image. Principles and methods of image processing and pattern recognition can be found in a variety of sources [129, 130]. Dawson [131] overviews basic image algorithms and explains concepts of point, area and frame processes. A point process is an algorithm which changes a pixel value based only on that pixel value. An area process changes a pixel value based on the value of pixel and values of neighbouring pixels. Algorithms that change pixel values based on comparing two or more images are called frame processes. Another important image analysing method is image segmentation which divides an image into different regions of certain properties. It is in detail discussed in [132 - 134]. In general, image segmentation techniques can be categorised into three classes: characteristic feature thresholding or clustering, edge detection, and region extraction. Cunningham [135] discusses binary image segmentation techniques. Weszka [136] surveyed threshold selection techniques. There are two types of thresholding techniques: global and local. The former is based on analysis of the image grey level histogram, while the latter on the grey levels in a neighbourhood. Rosenfeld [137] reviews image pattern recognition from a technique-oriented standpoint. Image enhancement is discussed by Woods & Gonzalez in [138]. Principle of edge finding technique is explained in [139]. Corby [140] examined robotic vision and its relation to computer vision. The role of vision as a robotic sensory process is discussed and compared with other robotic sensory processes.

Most vision systems currently available to welding processes employ artificial illumination to improve the contrast in the welding scene, or to provide profile information. As far as the image recognition is concerned, binary image processing is widely used due to its simplicity. However, a binary image contains much less information than a grey level image, and it is unlikely to implement complex control. This chapter is concerned with a real-time vision system. Information about the sidewalls, welding arc, and tungsten electrode is extracted through analysing the grey level image for the purpose of seam tracking and close-loop arc oscillation control. Firstly, the configuration of the IBM PC based vision system will be presented. Secondly, two methods of grey level image analyses: edge finding and template matching will be discussed. It is followed by vision software. Finally, real-time control schemes via the vision system will be covered.

7.2 IBM PC Based Vision System

The block diagram of a vision based control system is shown in figure-7.1. The vision sensor is the major part of the control system. It is made up of a CCD or vidicon camera, frame stores, and the host IBM-PC AT. Two frame stores were built on the prototype expansion board which was plugged into the IBM-PC AT. Each frame store has 256 × 256 pixels, and one pixel takes one byte. Therefore 64Kbyte RAM is needed for one frame store, or 128 Kbyte RAM for two frame stores. Both frame stores are memory mapped. The first frame store is from C0000 to CFFFF (hex), and the second from D0000 to DFFFF (hex). The IBM-PC AT directly access the frame stores by reading from or writing to the memory, which clearly maximises the image access speed.

Once the frame store is in the capture mode, the captured picture is digitised at the video rate, 50 frames/second. A 7-bit A/D converter is used to convert the video signal into digital numbers, known as grey levels, which are from 0 (black) to 127 (bright). The lower 7-bit of the memories in the frame stores contains the grey level while the top bit may be used as binary image bit. The welding scene is directly viewed by a TV camera without artificial illumination. Information about positions of sidewalls, electrode, and welding arc is extracted in real time using edge finding or template matching, which will be discussed in the following.

7.3 Edge Finding

Edge finding is often used to segment the image into homogeneous portions. In a two-dimensional image, the origin of the (x,y) coordinates is defined as the upper-left corner of the image. The x coordinate is the column number increasing from left to right, and y coordinate is the row (or line) number increasing from up to down. To detect edges the brightness (grey level) gradient vector G(x,y) at each point (x,y) is computed. The magnitude of the gradient can be expressed as [137, 140]:

$$G(x,y) = \sqrt{(\partial p/\partial x)^2 + (\partial p/\partial y)^2} \qquad \text{(equation-7.1)}$$

where p(x,y) is the grey level at point (x,y). The direction of the gradient is $\tan^{-1}(\partial p/\partial x \div \partial p/\partial y)$.

For horizontal tracking, equation-7.1 can be simplified as:

$$G(x,y) = \partial p/\partial x \qquad \text{(equation-7.2)}$$

The gradient direction is indicated by the sign of G(x,y). Positive G(x,y) means the direction from left to right, and negative means right to left. For a digital gradient, differences are taken instead of derivatives:

$$G(x,y) = \Delta p(x,y)/\Delta x \qquad \text{(equation-7.3)}$$

If a fixed pattern of pixels is considered, Δx is the same for each pixel. Therefore Δp can represent the rate of change in a horizontal line of pixels. To make the method less sensitive to noise, differences of averages, rather than of single grey levels, are employed, i.e.,

$$\Delta p(x,y) = [p(x+1,y) + p(x+2,y)]/2 - [p(x-1,y) + p(x-2,y)]/2 \qquad \text{(equation-7.4)}$$

Discarding the constant, the difference among the four pixels near point (x,y) is regarded as the change rate of grey level at point (x,y) in the horizontal direction:

$$D(x,y) = p(x+1,y) + p(x+2,y) - p(x-1,y) - p(x-2,y) \qquad \text{(equation-7.5)}$$

For an object brighter than the background, when scanned horizontally the pixel values change from low to high, then high to low. Applying equation-7.5, two differential peaks, positive and negative, are detected. The positive peak corresponds to the left edge of the object and the negative to the right edge. In the situation of TIG welding, the arc is always brighter than the background. The tungsten electrode is very hot due to intensive arc heating, and therefore

brighter than the background. The lower part of the electrode which is closer to the arc is even brighter. The pixel values of a horizontal line across the arc or the electrode follow the pattern: low - high - low. This is proved by later experiments. By means of equation-7.5, the left edge of the arc or the electrode has the positive peak rate of change in pixel value while the right edge has the negative peak rate. Figure-7.2 demonstrates edge finding of the electrode and non-deflected arc, figure-7.3 for the electrode and left-deflected arc, and figure-7.4 for the electrode and right-deflected arc.

In narrow gap TIG welding, the bottom of the groove is rather brighter due to the arc illumination. The light intensity in this region is relatively constant. When scanned horizontally, the pixel level has sharp changes at the sidewalls, but only slight variations in the groove. By applying the above equation, two sidewalls are located by looking for positive and negative differential peaks. Unfortunately, this is not the case of tracking the electrode and the arc since pixel patterns across the tungsten and the arc do not always indicate the peak pixel variations at their edges, but at sidewalls. This problem can be avoided by applying the edge finding technique to the electrode or the arc in the area defined by the detected sidewall positions. Assuming the left sidewall location is x1 and the right one is x2, x in equation-7.5 is taken from x1 to x2. Then positive and negative differentiations can be clearly identified for left and right edges of the electrode or the arc (figure-7.5). In the mean time, edge finding in the smaller area enhances the tracking speed because of less differentiations.

The advantages of the edge finding technique include its simplicity and fast tracking speed. However it requires that the object to be tracked is distinct from the background.

7.4 Template Matching

The template matching technique analyses an area of pixels, rather than a line of pixels. It tracks a target object by correlating a template with a portion of image. A square template can be used to assemble the feature of the object. The function of pixels in the template can be described as:

$$t(i,x,y,xj,yj) \qquad (0 \le xj \le xs,\ 0 \le yj \le ys)$$

where i stands for the template number, (x,y) for the location of upper-left corner, xs for the template length, and ys for the template width. Likewise, the image window can be defined in the same way, and the function of pixels in the window is,

$$w(i,x,y,xj,yj) \qquad (0 \le xj \le xs,\ 0 \le yj \le ys)$$

Template matching correlates the template with the image window. Mathematically, pixels in the template and those in the window can be treated as linear arrays of data. The correlation between the two arrays can be calculated by applying derivation and covarience. This process involves a great amount of multiplication and division, therefore uses much computation time. To maximise the processing speed, point-point differences are calculated. Their summation is considered to be the correlation between the window and template, i.e.,

$$C(i) = \sum_{\substack{xj=0,xs \\ yj=0,ys}} | t(i,x,y,xj,yj) - w(i,x,y,xj,yj) | \qquad \text{(equation-7.6)}$$

The smaller the correlation value, the better the matching between the two. If the template and the window are exactly the same, the correlation is zero.

The application of template matching involves two situations: a target object with a fixed shape such as tungsten electrode and sidewalls, and that with variable shape, e.g. the welding arc being deflected. In the first case, a single template is often set up to represent the object to be tracked (figure-7.6). The initial location of the object is decided by correlating the template against windows with the same size, but varied column (x) from 0 to 255. This means that 256 correlations are carried out. The target object is tracked by finding out the window with smallest correlation number. The substantial tracking is implemented by correlating the template with the existing window (i.e. current location of the target), the three windows to the left of the existing window, and the three windows to the right of the existing window. The window with the smallest correlation number is the new location of the target.

Since the welding arc changes during deflection, a single template cannot represent the arc feature. An alternative way is designed to overcome this difficulty. The idea is that a set of templates are set up in the range of arc deflection, and each of them represents an arc location, as shown in figure-7.7. The pixel values in each template t(i,x,y,xj,yj) are stored in a buffer. During matching, it is correlated against the window w(i,x,y,xj,yj) according to equation-7.6, resulting in a correlation value C(i). The best matched window, which has the smallest correlation, is therefore regarded as the current arc location. Figure-7.8 shows that the arc is tracked by this technique. To im-

prove matching quality, a limit of correlation value can be set. If the smallest correlation is greater than the limit, no good matching is found. In this case, either revising templates or resetting the correlation limit must be done.

The template sizes and number of templates can be decided by the user. The larger the templates the slower the processing. More templates tend to increase detection accuracy, but sacrifice the processing speed. Reasonably, eight templates with a size of 15×10 (pixel) are recommended. The tracking speed of template matching can be doubled by dividing those templates into two groups: one for left deflection and another one for right deflection. Template correlations are made only for one group of templates, which is decided by the synchronising signal (a one-bit code: left or right deflection) from the 8088 computer.

As mentioned earlier, the template matching technique processes an area of pixels. It uses a template to describe the feature of an object or to represent a location of an object. This method does not require a particular object feature, and can detect an object in a complicated scene. The major disadvantage of this technique is relatively slow speeds as compared with the previous method.

7.5 Vision Software

Choosing a suitable programming language is often a difficult decision. For real-time applications, the processing speed needs to be carefully considered. Another consideration is the flexibility, portability and programming efficiency of the language. Assembly languages have the highest processing speed. However, they tend to be less

compact, in particular when mathematical operations are implemented. On the other hand, high level languages, such as FORTRAN and PASCAL, can efficiently implement complicated mathematical operations. Unfortunately, their processing speed is too slow to meet the needs of real-time processing in most circumstances. To make a compromise, the C programming language is chosen. C is a general-purpose, expressive and versatile programming language. It features modern control flow and data structures, and a rich set of operators. C is hardware independent. Programs written in C can run on any machine that supports C. The compiler used throughout this work is LATTICE-C compiler, which runs on the host IBM-PC AT.

The vision software was constructed modularly. It consists of a library of user routines SYSLC.C, as shown in appendix-F1, and a set of main programs (appendix-F2) for vision analysing. Major user routines are listed as follows:

capture(): capture the image.

disp(): display the image stored in the frame store.

threshold(tv): threshold the image with a threshold value of tv.

df_ptr(xp,yp): define a point (xp,yp).

df_hl(xl,yl,xs): define a horizontal line.

df_vl(xl,yl,ys): define a vertical line.

df_box(xl,yl,xs,ys): define a window with left-upper conner (xl, yl) and right-down conner (xl+xs, yl+ys).

read_hl(xl,yl,xs,db): read a horizontal line of pixels into a buffer db.

write_hl(xl,yl,xs,db): write a horizontal line from a buffer db.

read_area(xs,ys,xl,yl,db): read an area of pixels into a buffer db.

write_area(xs,ys,xl,yl,db): write an area from a buffer db.

track_hl(x,y,xs,lloc,rloc,ldf,rdf): find edges in a horizontal line from (x,y) to (x+xs,y). Left edge location, right edge location, left differentiation and right differentiation are respectively stored in the buffers lloc, rloc, ldf, and rdf.

cor_tem(temp): carry out template matching and return the correlation value.

All routines can be called from the main program. Both the edge finding and template correlation routines work with the real (or live) image. Buffering image pixels is not necessary. Consequently, the processing speed is optimised.

7.6 Real-Time Control Schemes

7.6.1 Seam Tracking

To achieve a good weld quality, both sidewalls of the narrow groove must be equally penetrated, otherwise weld defects such as incomplete fusion and undercuts are likely to occur. One approach is to maintain the tungsten electrode at the centre of the groove while the arc is oscillating equally from side to side without a field bias. For a welding process without seam tracking, this requires precise groove preparation, material alignment and fixture, which increases considerably the overall cost. Even if the above requirements could be satisfied, inevitable distortion can cause the tungsten electrode to divert from the centre of the groove. Therefore, a real-time seam tracking system is necessary to ensure narrow gap welding quality.

The vision information on the locations of sidewalls and electrode can be effectively used to control the torch position in the gap. Once the torch is off the centre of the gap, an adaptive correction can be made based on the feedback vision information, as shown in figure-7.9. An indirect way to carry out seam tracking is to use the arc deflection distance as the control variable The left deflection (Dld) is defined as the distance from the left edge of the arc to the centre of the electrode, and the right deflection (Drd) as the distance from the right edge of the arc to the centre of the electrode. Without field bias the arc oscillates equally toward both side walls, provided the electrode is at the centre of the groove. As soon as the electrode is off the groove centre, the arc oscillation becomes unsymmetrical. It is detected by capturing two images at left-hand and right-hand dwelling (referred to figure-6.6), synchronised by the 8088 computer. More specifically, the left deflection at left dwelling and right deflection at right dwelling are detected using either the edge finding or the template matching technique discussed above. The difference between them is worked out, and the torch is adaptively moved to the groove centre.

7.6.2 Close Loop Arc Oscillation Control

In addition to the seam tracking, arc oscillation can be automatically controlled. The correct arc deflection angle (α) defined in figure-7.10 is crucial to side wall penetration. It can be calculated from the groove gap (G) and arc standoff (la):

$$\alpha = \tan^{-1} [G/(2 \bullet la)] \quad \text{(equation-7.7)}$$

When the vision sensor captures the images at dwelling times, the deflection distance can be measured using either edge finding or template matching technique. The successive measurements during two oscillating cycles are averaged. The deflection angle (α) can be derived from the distance by referring to a look-up table (LUT), then compared with the specified angle. The IBM-PC AT feeds back this information to the arc oscillation controller, the latter sets wave parameters accordingly in order to correct the error. Figure-7.11 shows the flow chart of close loop control of arc oscillation.

In this chapter, vision-based real-time control has been discussed. The configuration of the vision system, grey level image analysing techniques: edge finding and template matching, software implementation and real-time control schemes were covered. All experimental results concerned with arc oscillation and vision control will be presented in the next chapter.

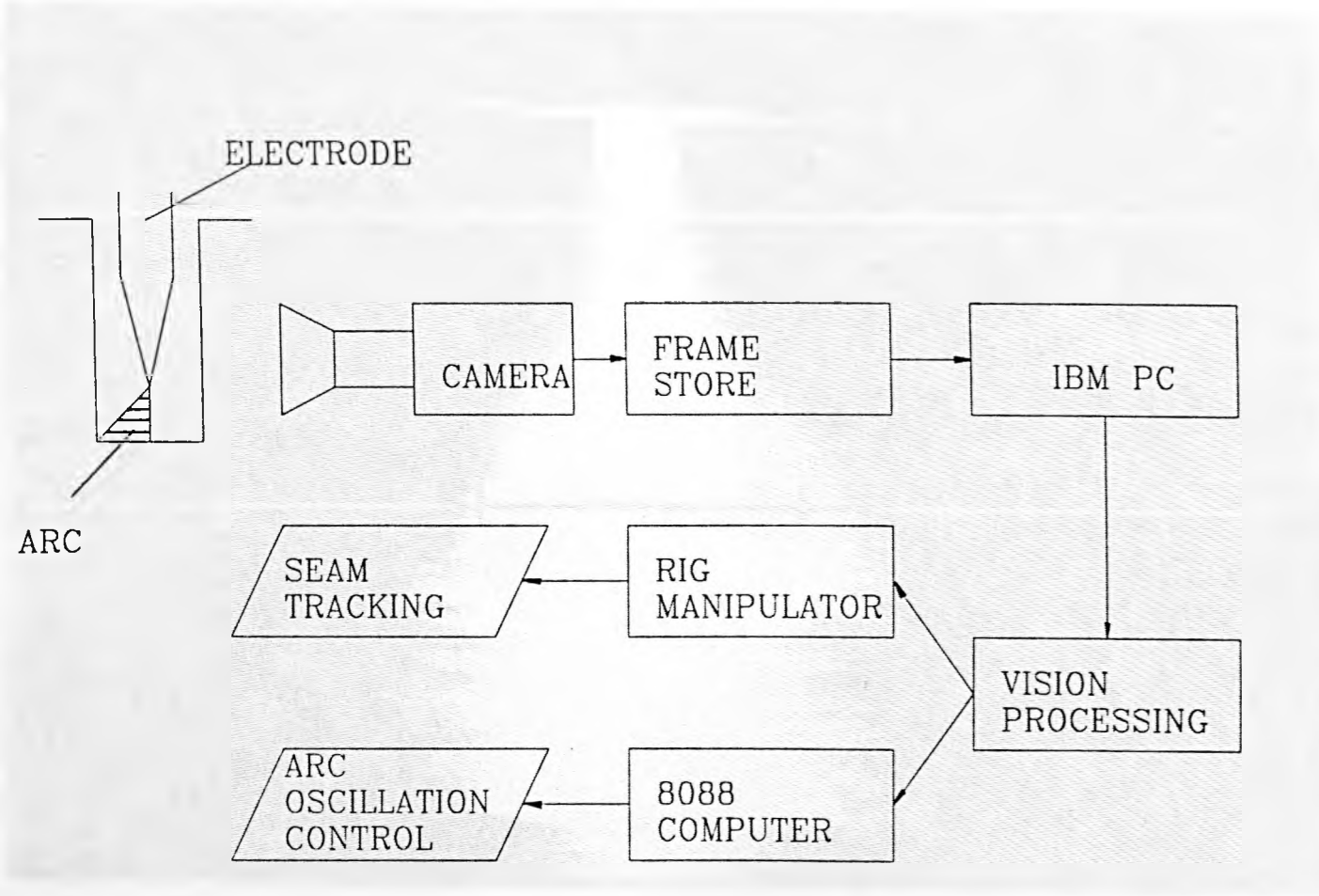


Figure-7.1 Block diagram of vision based control system

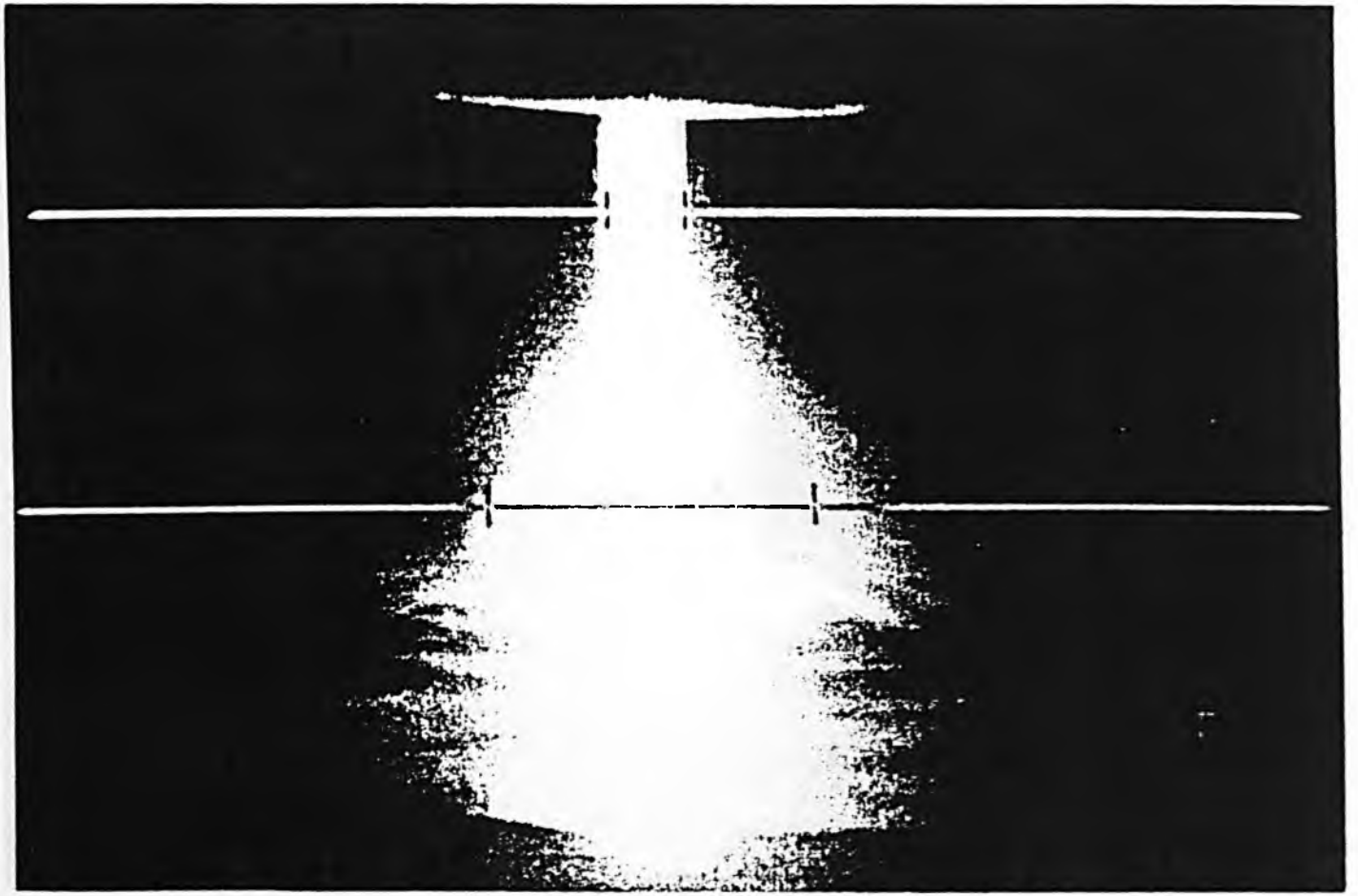


Figure-7.2 Edge finding of electrode and non-deflected arc

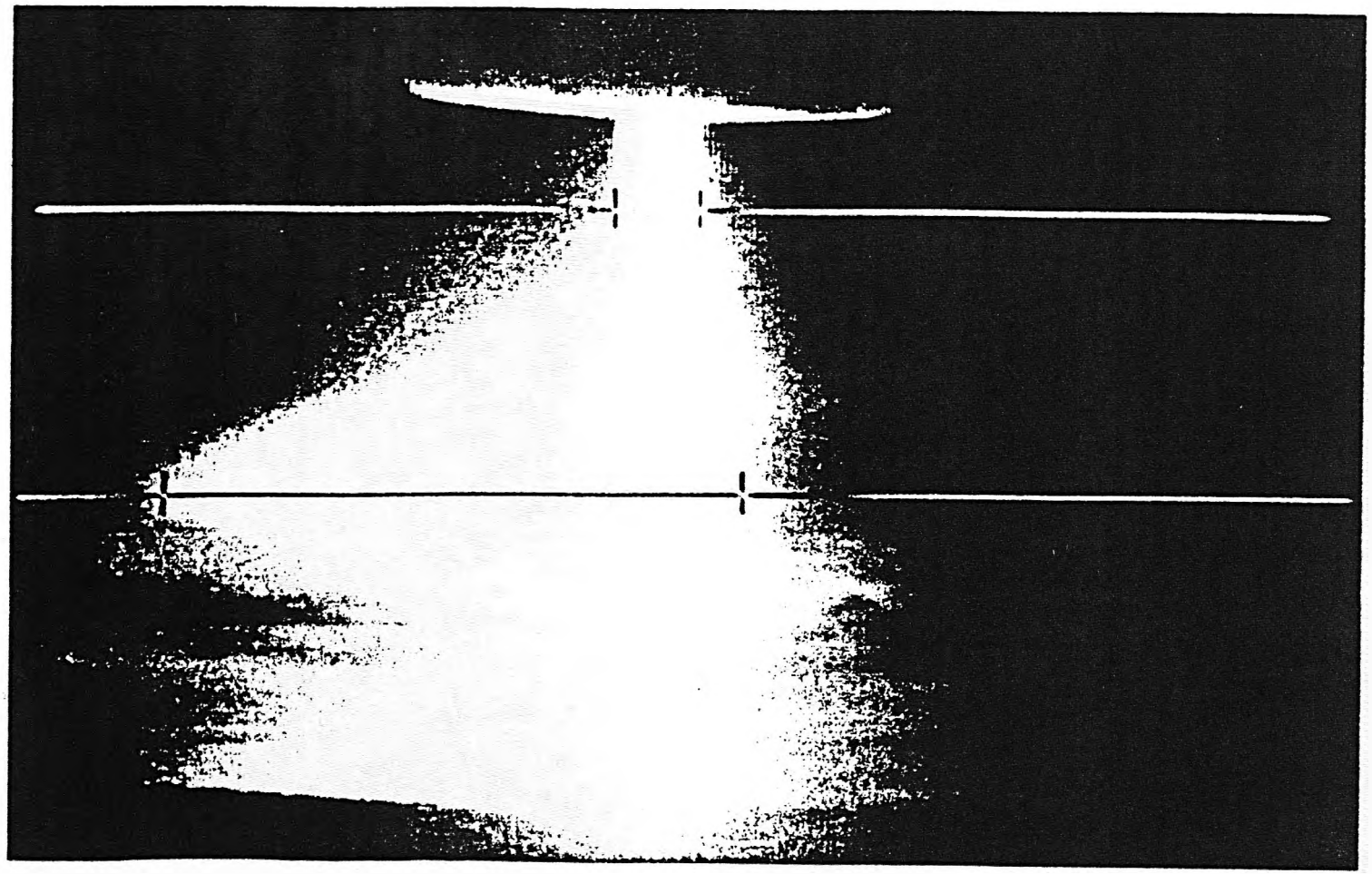


Figure-7.3 Edge finding of electrode and left-deflected arc

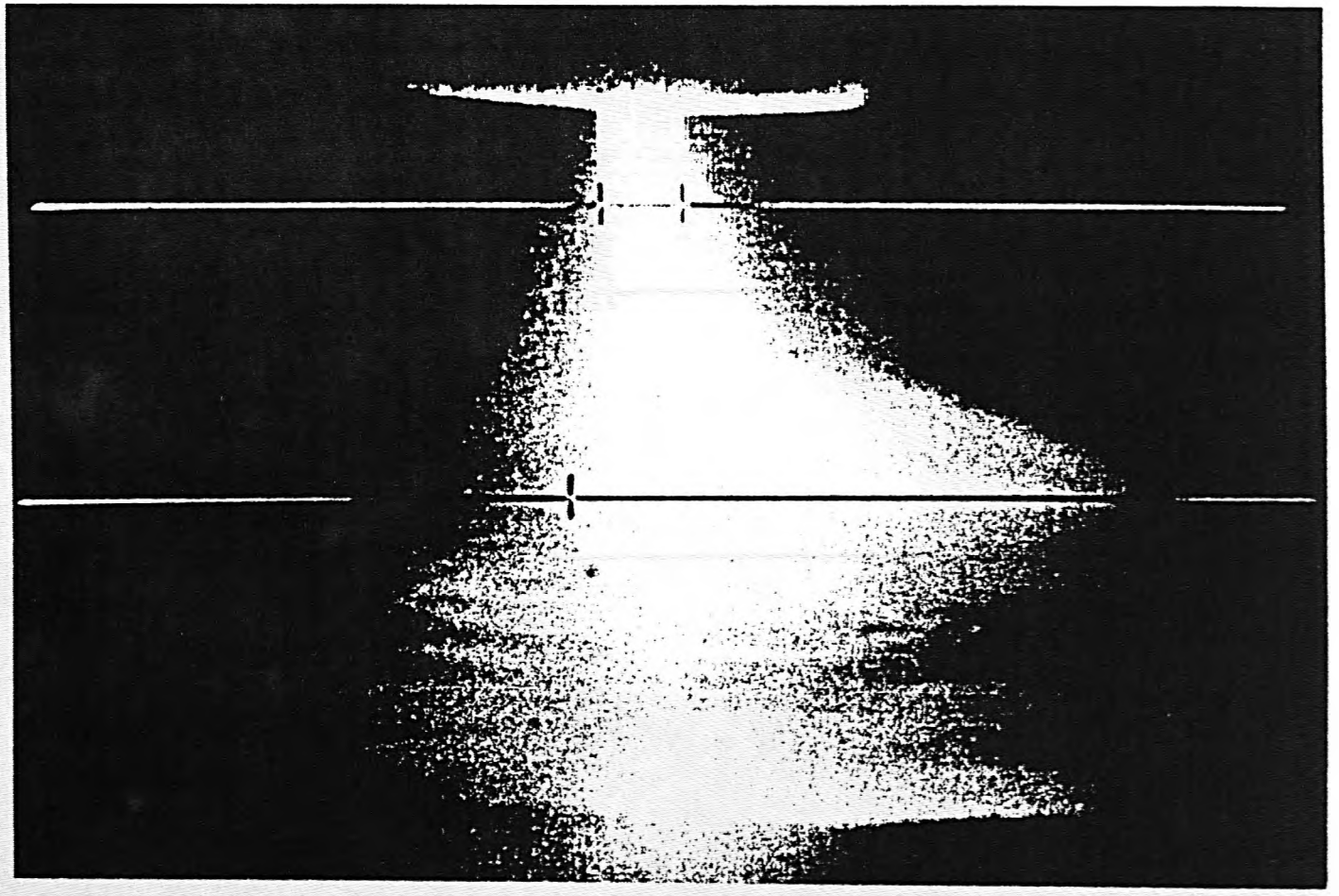


Figure-7.4 Edge finding of electrode and right-deflected arc

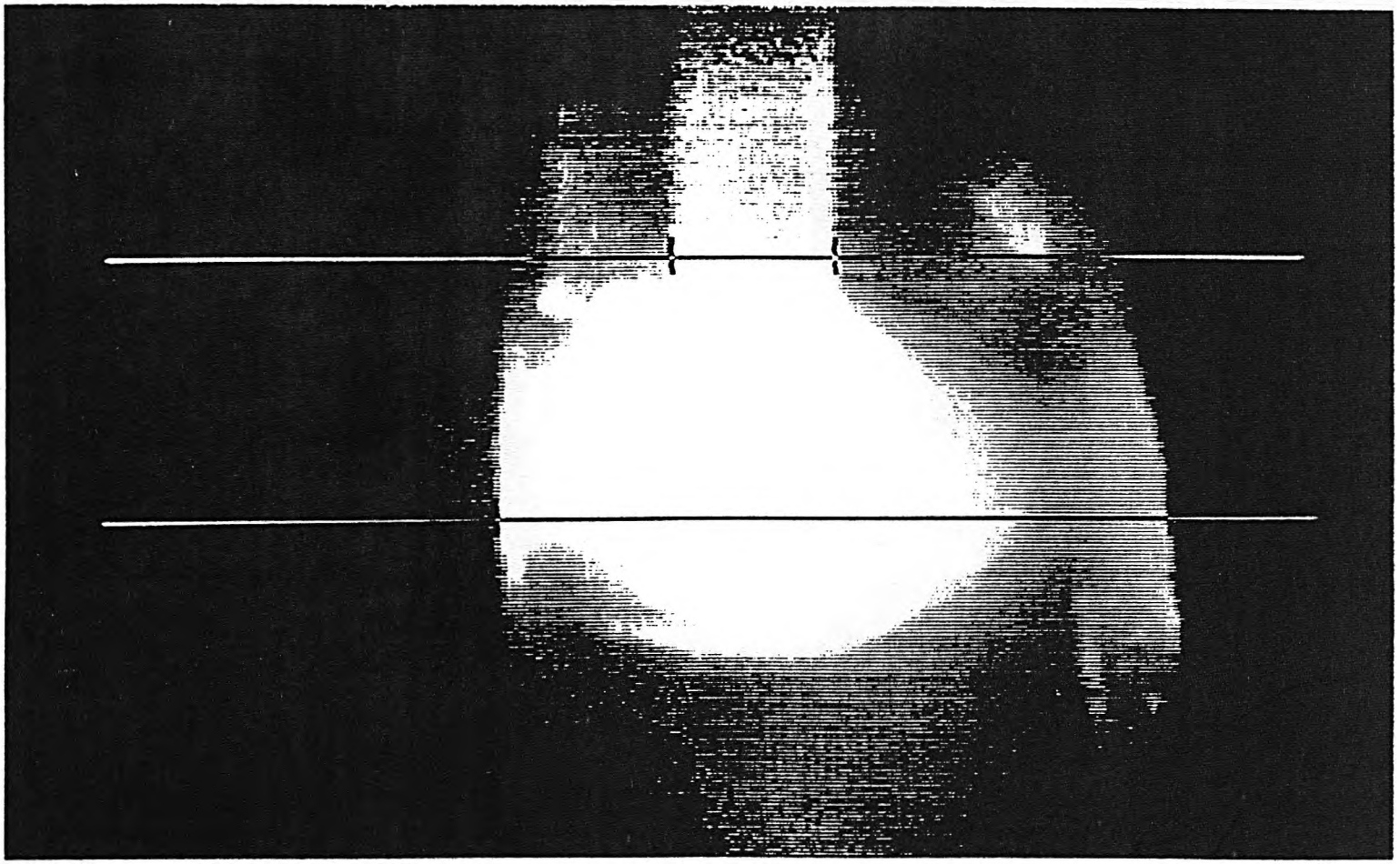


Figure-7.5 Edge finding of sidewalls and electrode

Figure-7.6 Tracking sidewalls and electrode using template matching

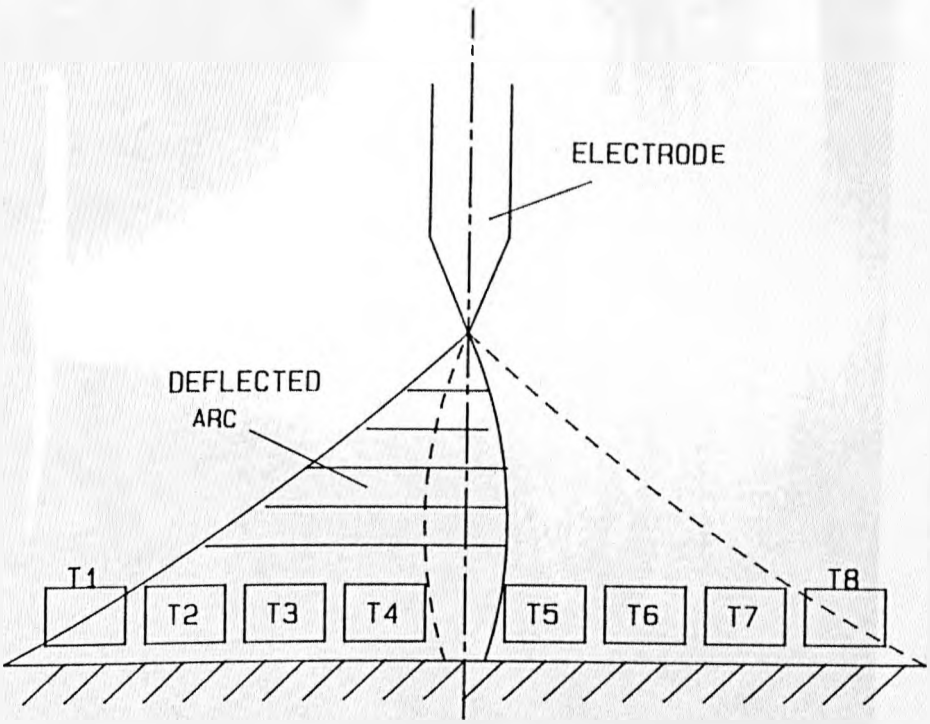


Figure-7.7 Set-up of templates for arcs of varied deflections

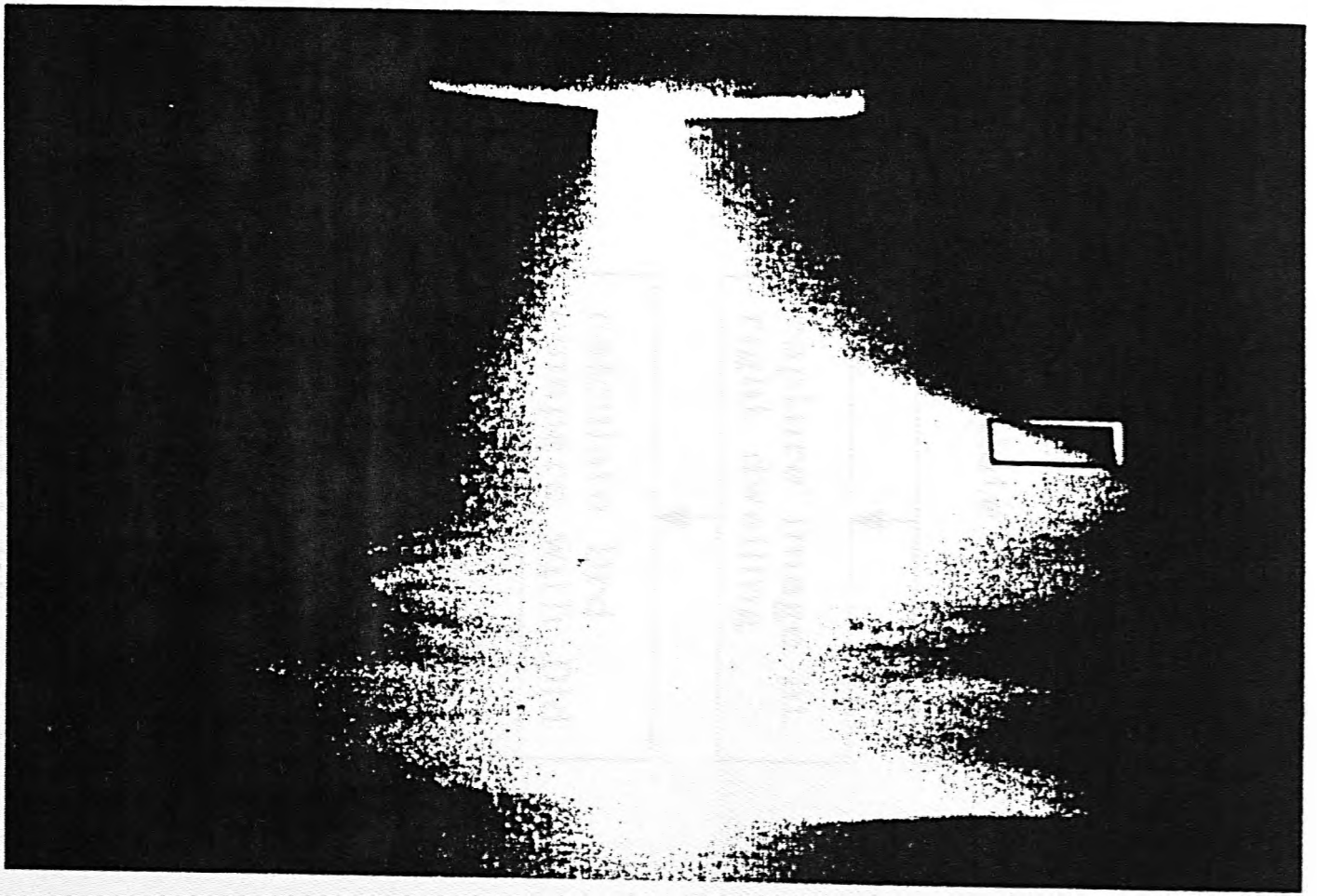


Figure-7.8 Tracking arc using template matching

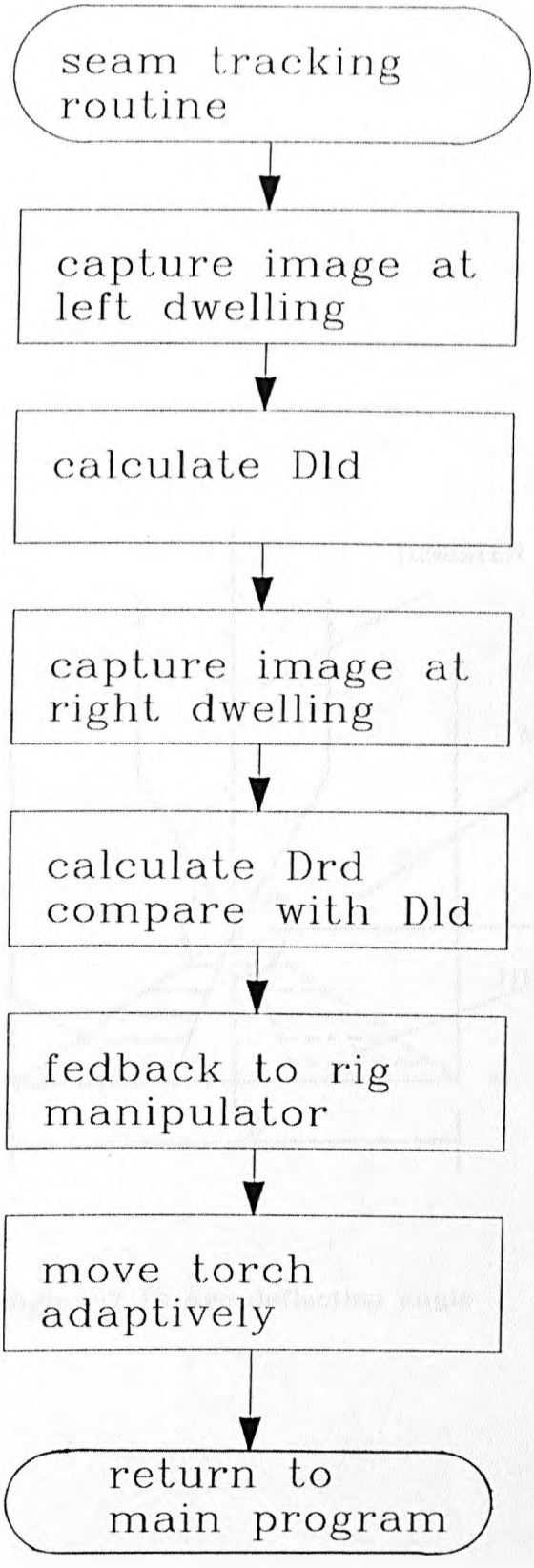


Figure-7.9 Weld seam tracking based on electrode-sidewall distance

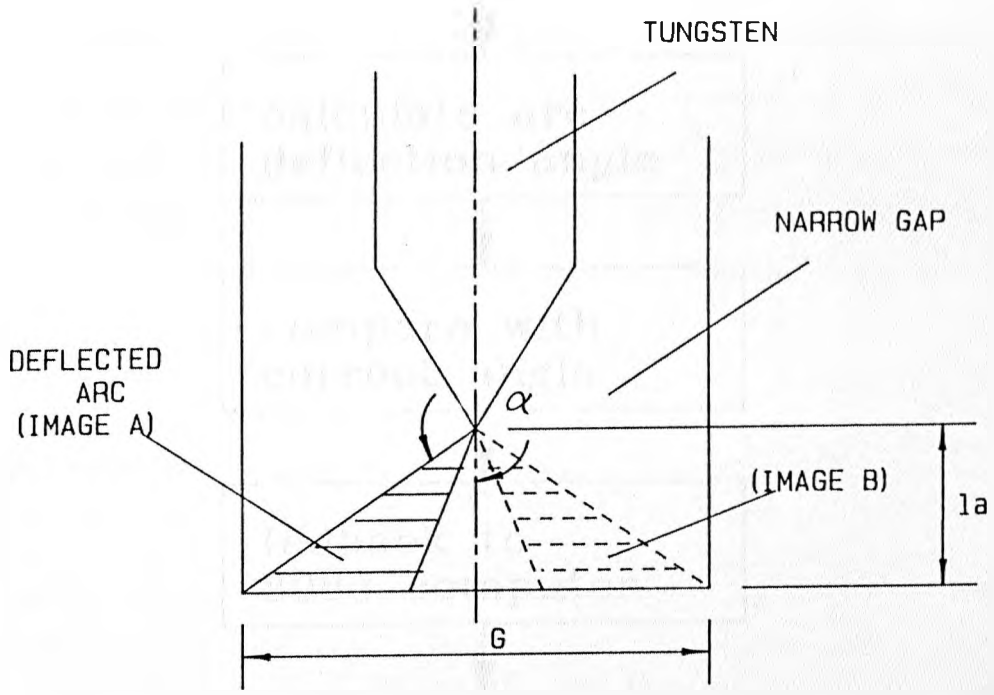


Figure-7.10 Arc deflection angle

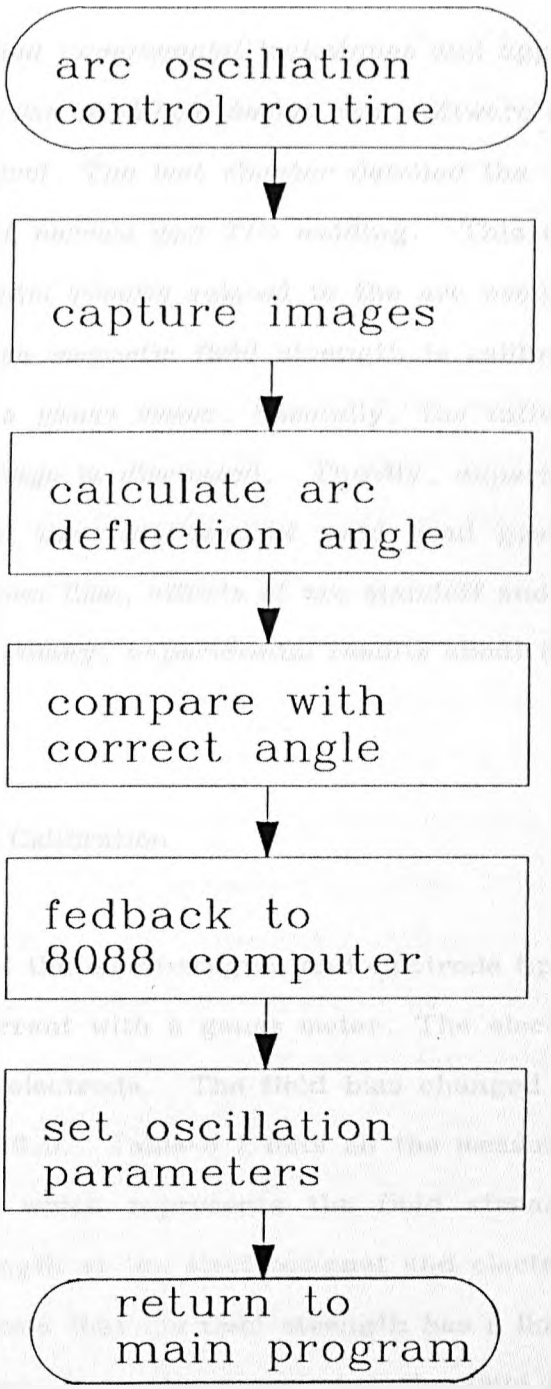


Figure-7.11 Arc deflection control using vision system

8. EXPERIMENTAL RESULTS

Chapter 4 outlined experimental techniques and apparatus. Chapter 5 and 6 discussed the hardware design and software implementation of arc oscillation control. The last chapter detailed the vision system for real-time control of narrow gap TIG welding. This chapter will presents all experimental results related to the arc oscillation and vision system. Firstly, the magnetic field strength is calibrated against the coil current with a gauss meter. Secondly, the influence of arc deflection on arc voltage is discussed. Thirdly, experiments were conducted to find out the variations of weld bead geometry with field strength. In the mean time, effects of arc standoff and welding current were considered. Finally, experimental results about the vision system are discussed.

8.1 Field Strength Calibration

Magnetic fields at the electromagnet and electrode tip were calibrated against the coil current with a gauss meter. The electromagnet was 5 mm apart from the electrode. The field bias changed from 0.0 to 9.9 while fa was set to 0.0. Table-8.1 lists all the measurements of magnetic field density which represents the field strength. Figure-8.1 plots the field strength at the electromagnet and electrode against the coil current. It shows that the field strength has a linear relationship with the coil current. It is also found that the field strength at the electrode is approximately 1/5 of that at the electromagnet under the experimental conditions. As mentioned in chapter 4, the flux density decreases in proportion to the square of the distance. The closer the

electromagnet is to the electrode, the stronger the magnetic field across the arc and the further the arc is deflected with the same coil current. The electromagnet produces as high as 60 Gauss ($T \times 10^{-4}$) field strength at the electrode tip with a relatively low coil current (1.5 A). This combination is practical, and satisfies almost all kinds of arc oscillation.

8.2 Arc Voltage Measurements

Autogenous TIG welding tests were intended to find out the influence of field strength on arc voltage. The welding test rig (figure-8.2) was provided by the Welding Institute (TWI). It was specially designed for butt welding of small pipes. The two pipes to be welded were clamped, and kept steady. The welding head, including the welding torch, wire feeding and the electromagnet, rotates around the pipes, therefore forming a circumferential weld. As seen in figure-8.2, the electromagnet is placed behind the electrode and the feeding wire in front of the electrode. Since autogenous welding tests were sufficient to find out the influence of field strength together with welding current and arc standoff on welding performance, wire feeding was avoided. Welding tests were carried out by running the welding arc on the surface of a piece of stainless steel 304 pipe (60 mm OD × 5.7 mm wall). Its chemical composition is listed in table-8.2. The welding parameters used are shown in table-8.3.

The arc voltage was recorded by the TIG power supply. Two groups of tests were made:

	Current (A)	Standoff (mm)
group 1:	40	1.5
	40	2.0
group 2:	60	1.5
	80	1.5

The coil current changed from 0.0 to 1.5 A with steps of 0.5 A. In other words, the field strength changed from 0 to 60 Gauss (T $\times$ 10^{-4}). Variations in arc voltage with the coil current are plotted in figure-8.3. Generally speaking, as the arc is deflected the arc voltage slightly increases. This effect is less obvious with a higher welding current.

Figure-8.3 (a) shows that a higher standoff results in a higher arc voltage. It is a common practice to use the arc voltage signal to control the arc length in TIG welding. Feedback arc oscillation control and seam tracking are preferably implemented by the vision system. However, with the assistance of slight mechanical weaving of the welding torch in a narrow gap, arc voltage signal can be used to carry out real-time seam tracking.

8.3 Weld Bead Geometry

Weld bead geometry is of major concern in investigating the effects of arc oscillation. Two bead dimensions are studied: bead width (W) and bead depth (D). Autogenous TIG welding tests were made with stainless steel pipe (60 mm OD $\times$ 5.7 mm wall). The welding parameters used are listed in table-8.3. The welding procedure proceeded for a preset field amplitude as follows:

Current (A)	Standoff (mm)
100	2.0
80	2.0
60	2.0
100	1.5
80	1.5
60	1.5
40	1.5

The above seven beads were made in a circle around the pipe. The field amplitude was preset from the keyboard of the 8088 computer. It changed as: 0.0, 2.5, 5.0, 7.5, 9.9. Therefore, five groups of beads were made and each of them contains the above seven combinations of current and standoff. A total of thirty five beads was laid. Figure-8.4 shows the weld beads made with 1.6 mm standoff and varied currents. Samples were sectioned to look at the bead penetration. Figure-8.5 shows those cross sections of beads with 100 A current, and figure-8.6 shows those with 60 A current.

Measurements of bead width and depth were taken with a calibrated optical microscope at a magnification of $\times 10$. The accuracy in the depth and width values is 1%. It should be mentioned that three depth measurements were equally spaced in a cross section and the average of them was taken as the "real" bead depth. All measurements are presented in table-8.4 to 8.10.

To have a close look at the influence of field strength, together with welding current and arc standoff on the bead geometry, the experimental results were grouped and graphically presented. Figure-8.7

For both "J "and narrow gap welding preparation, the root pass was made without magnetic arc oscillation. A large electrode angle (included) of 60 degrees and pulsed welding current were used. The peak current was 88 A, pulse time 0.8 seconds, background current 26 A, and background time 0.4 seconds. The torch rotation speed was 160 second/revolution.

As far as the filler passes were concerned, constant welding current, 0.8 mm filler wire and the included electrode angle of 28 degrees were used. Welding current, wire feed speed, torch rotation speed and arc oscillation parameters were different in the two situations. For "J" preparation, the welding procedure is listed as follows:

Welding parameters	Pass 1	Pass 2	Pass 3	Pass 4
Welding current (A)	89	92	96	91
Rotation speed (sec/rev)	140	140	140	140
Wire feed speed (m/min)	0.80	0.85	0.85	0.70
Arc oscillation frequency (Hz)	1.35	1.35	1.40	1.50
Dwell time (second)	0.36	0.36	0.35	0.33
Field strength at electrode tip (gauss)	26	22	15	100

For narrow gap preparation, the welding procedure is:

Welding parameters	Pass 1	Pass 2	Pass 3	Pass 4
Welding current (A)	96	96	96	68
Rotation speed (sec/rev)	145	145	145	145
Wire feed speed (m/min)	0.77	0.77	0.77	0.44
Arc oscillation frequency (Hz)	1.40	1.40	1.40	1.50
Dwell time (second)	0.35	0.35	0.35	0.33
Field strength at electrode tip (gauss)	20	20	20	123

For both preparations, the final pass was finished with strong arc oscillation, which provided fine bead appearence. Complete sidewall penetration is observed by comparing the macroscopy of cross sections and the groove preparations.

shows the function of bead width as the field amplitude. It indicates that the bead width increases with the field strength. A higher standoff tends to increase the bead width, but the effect is less obvious. The influence of field amplitude and standoff on the bead depth is illustrated in figure-8.8. Generally, a higher field amplitude and a longer standoff decrease the bead depth. It is worth noting that the bead depth is improved with arc oscillation in the lower field amplitudes (0.0 - 2.5). This is probably caused by the stirring effect on the weld pool resulted from magnetic arc oscillation.

In addition to the field amplitude and standoff, welding current was studied. Figure-8.9 shows the influence of field amplitude and welding current on the bead width. It is clearly seen that a higher current produces a wider bead. Figure-8.10 shows that the bead depth increases with the welding current. It can be therefore concluded that a high current is preferred in narrow gap welding in terms of bead penetration. Another important characteristic of a weld bead is bead appearance. Figure-8.4 shows that magnetic arc oscillation produces fine and even ripples which are desirable.

Butt welding of stainless steel 304 pipes (60 mm OD × 5.7 mm wall) has been made using magnetic arc oscillation. The weld consists of four filler pass plus a root pass. Figure-8.11 shows a "J" preparation and the cross-section of the weld. Figure-8.12 shows a "narrow gap" preparation and the cross-section of the weld.

8.4 Vision Test Results

8.4.1 Edge Finding

Experiments have been made to test the reliability of the vision sensor as well as the edge finding technique. Two programs: VIEW.C and VISN.C (appendix-F2) were designed to carry out edge finding tests. Both programs track the positions of the electrode and arc. Edges of targets were marked with crosses. The only difference is that VIEW tracks the targets continuously while VISN does only when the "ENTER" key on the IBM PC keyboard is struck. It was proved from running VIEW that the vision system tracks the targets reliably even when the arc is oscillated as fast as 10 Hz. Such a high tracking speed ensures real-time control of a welding process. Three factors are attributable to the high speed response of the vision system:

(1) Fast frame store. It digitises the image at the video rate 30 frames/second. Access to the frame store is simply for the IBM PC to read from and write to the frame memory.

(2) Effective image analysing technique. A simple differentiation equation is applied to a line of pixels.

(3) Efficient vision software. All vision softwares were entirely written in C, which is faster than other high level languages such as Fortran and Pascal.

Table-8.11 shows pixels in a line across the electrode and differentiations. The left edge and right edge are determined by the maximum and minimum differentiation. Similar results are obtained for left-deflected (table-8.12) and right-deflected arc (table-8.13). Grey

levels and differentiations of pixels are plotted in figure-8.13. Positive and negative peak differentiations are clearly defined.

8.4.2 Template Matching

Programs have been designed to track the arc being deflected by template matching. A set of templates must be established by running the program TEMP.C (appendix-F2). The program requires the user to give a file name to store the templates. The location of a template is defined by its upper-left and lower-right corners which are decided by user's moving crosses in the image. Once the template window is defined, an ID name is required for the template. The template file stores the template ID, upper-left corner point (x, y), template length (xl), template width (yl), and a list of pixels which are stored in the ASCII form.

In running the tracking program "TRAC.C", the user inputs the template file name and template IDs. Then TRAC correlates each template against its corresponding target window which assembles the same location and dimensions with the template, resulting in a correlation value. The best matched target window, which has the smallest correlation value, is considered to be the current position of the target - arc.

8.4.3 Real-Time Measurements of Arc Deflection

Experiments were carried out to observe the arc deflected in a magnetic field. Stainless steel 304 flat plate was used as testing materials. The welding current was set as 80 A. The tungsten diameter

was 1.6 mm and the arc standoff was 2.0 mm. The left deflection (Dld) and the right deflection (Drd) were defined in section 7.6.1. The arc deflection was measured by the vision system. It is a function of coil current as shown in figure-8.14. As the value of the coil current (or field strength) increases, the arc deflection increases. The maximum arc deflection can be as high as 9.0 mm which is sufficient to reach and fuse the side wall.

8.4.4 Tracking Arc Shape

By scanning line by line, the arc profile can be detected. Based on vision information, non-deflected, left-deflected and right-deflected arc are plotted in figure-8.15. It is worth noting the vision sensor can also measure the arc area which might be interesting to welding engineers. Figure-8.16 shows that the arc area increases as the arc deflects towards the side walls.

This chapter has discussed all experimental work and results. The next chapter will make conclusions about this work, and presents proposals for future development.

Table-8.1 Field strength calibration

scale	coil current(A)	field at probe(G)	field at electrode(G)
0.0	-1.5	317	63
1.0	-1.2	264	51
2.0	-0.9	206	39
3.0	-0.6	142	26
4.0	-0.3	78	13
5.0	0.0	17	1
6.0	0.3	-48	-10
7.0	0.6	-81	-17
8.0	0.9	-182	-37
9.0	1.2	-246	-50
9.9	1.5	-303	-61

Table-8.2 Chemical composition (%) of test material

- type 304 stainless steel

C:0.048	S:0.003	P:0.019	Si:0.59	Mn:1.20
Ni:9.5	Cr:18.5	Mo:0.14	V:0.06	Cu:0.08
N<0.01	Ti:0.02	Co:0.03		

Table-8.3 Fixed welding parameters for tests

Travel speed	90 mm/min
Dwell time	0.25 seconds
Weave time	0.1 seconds
Period	0.7 seconds
Frequency	1.43 Hz
Shielding gas	Argon + 1% H_2
Gas flow rate	8 l/min
Electrode diameter	2.4 mm
Electrode angle	18 degrees
Electrode stickout	8 mm
Parent material	Type 304 stainless steel (cast 3D 105), 60 mm OD × 5.7 mm wall

Table-8.4 Measurements of bead dimensions in mm

(welding current:100 A, arc standoff:2.0 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	6.5	1.8	1.8	1.6	1.7
2.5	75	15	6.7	1.6	1.8	1.6	1.7
5.0	160	32	8.5	1.5	1.5	1.6	1.5
7.5	240	48	10.1	1.3	1.3	1.2	1.3
9.9	310	62	11.0	1.1	1.2	1.0	1.1

Table-8.5 Measurements of bead dimensions in mm

(welding current:100 A, arc standoff:1.6 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	6.2	1.5	1.7	1.5	1.6
2.5	75	15	6.6	1.8	1.9	1.8	1.8
5.0	160	32	7.2	1.7	1.7	1.6	1.7
7.5	240	48	7.7	1.6	1.7	1.5	1.6
9.9	310	62	9.6	1.3	1.5	1.3	1.4

Table-8.6 Measurements of bead dimensions in mm

(welding current:80 A, arc standoff:2.0 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	4.5	1.7	1.8	1.5	1.7
2.5	75	15	5.6	1.6	1.7	1.5	1.6
5.0	160	32	7.3	1.1	1.2	1.2	1.2
7.5	240	48	9.0	0.7	0.9	1.1	0.9
9.9	310	62	9.5	0.8	0.9	0.9	0.9

Table-8.7 Measurements of bead dimensions in mm

(welding current:80 A, arc standoff:1.6 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	4.7	1.7	1.6	1.5	1.6
2.5	75	15	5.2	1.6	1.8	1.7	1.7
5.0	160	32	6.3	1.5	1.3	1.4	1.4
7.5	240	48	6.9	1.4	1.3	1.3	1.3
9.9	310	62	9.7	0.8	0.9	0.9	0.9

Table-8.8 Measurements of bead dimensions in mm

(welding current:60 A, arc standoff:2.0 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	3.8	0.7	1.1	1.1	1.0
2.5	75	15	4.8	1.1	1.1	1.1	1.1
5.0	160	32	5.8	1.0	0.8	0.8	0.9
7.5	240	48	7.0	0.8	0.7	0.7	0.7
9.9	310	62	6.9	0.6	0.7	0.7	0.7

Table-8.9 Measurements of bead dimensions in mm

(welding current:60 A, arc standoff:1.6 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	3.6	0.9	1.1	1.1	1.0
2.5	75	15	4.2	1.0	1.2	1.1	1.1
5.0	160	32	6.6	0.8	0.8	0.8	0.8
7.5	240	48	6.2	1.0	0.9	1.0	1.0
9.9	310	62	6.8	0.8	0.8	0.8	0.8

Table-8.10 Measurements of bead dimensions in mm

(welding current:40 A. arc standoff:1.6 mm)

field amplitude (scale)	field strength (Gauss) magnet	electrode	bead width w	bead depth d1	d2	d3	average depth d
0.0	0	0	2.5	0.6	0.7	0.5	0.6
2.5	75	15	3.4	0.5	0.6	0.5	0.5
5.0	160	32	3.9	0.5	0.5	0.6	0.5
7.5	240	48	4.0	0.6	0.5	0.7	0.6
9.9	310	62	4.3	0.6	0.5	0.7	0.6

Table-8.11 Pixel values and differentiations

for electrode

window dimensions

x_loc: 20, y_loc: 70, x_size: 220, y_size: 1

pixel reading

43	44	44	44	44	45	45	45	45	45	44	46	46	47	47
49	47	47	47	47	46	49	49	49	49	49	49	49	49	50
50	50	50	50	50	50	50	50	50	50	50	50	50	50	49
49	50	50	49	50	50	50	50	50	50	50	51	51	51	52
52	52	53	53	52	53	52	53	53	53	53	54	53	54	54
55	54	54	55	55	55	55	55	55	55	55	55	55	56	58
57	58	58	59	59	59	61	61	64	67	76	87	89	90	91
92	93	93	93	93	93	93	94	89	76	68	67	65	64	63
61	59	55	57	55	55	53	53	51	51	51	51	49	49	49
49	48	48	47	45	46	46	45	45	45	45	44	44	44	44
44	44	43	43	43	43	42	43	43	42	41	41	42	42	41
42	41	40	40	40	40	40	41	41	40	40	40	40	40	40
39	40	38	38	38	38	38	38	38	38	38	38	37	38	36
36	37	36	36	36	36	37	35	34	34	34	34	34	34	34
33	34	33	33	33	33	32	32	32	32					

differential computation

0	0	1	1	2	2	1	0	-1	0	2	4	4	4	3
0	-2	-2	-1	1	4	5	3	0	0	0	0	1	2	2
1	0	0	0	0	0	0	0	0	0	0	0	-1	-2	-1
1	1	0	0	1	1	0	0	0	1	2	2	2	2	2
2	2	1	0	-1	0	1	1	1	1	1	1	1	2	2

0 0 1 2 1 0 0 0 0 0 0 1 4 5 4

2 2 3 2 3 4 7 11 21 38 45 36 18 7 6

5 3 1 0 0 1 -3 -21 -43 -48 -33 -15 -8 -8 -9

-13 -12 -8 -4 -4 -6 -6 -6 -4 -2 -2 -4 -4 -2 -1

-2 -3 -5 -5 -3 -1 -1 -2 -1 -1 -2 -2 -1 0 0

-1 -2 -2 -1 -1 -1 0 0 -2 -4 -2 1 1 0 -1

-2 -3 -3 -1 0 1 2 1 -1 -2 -1 0 0 -1 -1

-2 -3 -3 -2 0 0 0 0 0 0 -1 -1 -2 -3 -2

-1 0 -1 -1 1 0 -3 -5 -4 -1 0 0 0 -1 -1

-1 -1 -1 -1 -1 -2 -2 -1 0 0

edge specification

max_dif: 45, min_dif: -48, left_edge: 100, right_edge: 114

Table-8.12 Pixel values and differentiations

for left-deflected arc

window dimensions

x_loc: 20, y_loc: 150, x_size: 220, y_size: 1

pixel reading

56	57	59	59	58	59	59	59	59	59	59	61	60	61	61
61	61	62	62	63	63	64	65	67	70	71	72	74	76	78
78	79	80	83	83	84	85	87	87	89	90	90	91	93	93
93	94	94	94	94	94	96	96	96	97	97	97	97	97	97
96	97	97	97	97	97	97	97	97	97	97	97	97	97	97
96	96	96	97	97	97	96	96	97	96	97	97	97	96	97
97	97	97	97	97	97	96	96	96	97	97	96	96	96	96
96	96	97	96	97	96	93	95	94	91	91	90	90	89	88
87	84	83	81	80	79	78	75	75	73	72	71	70	69	68
68	67	66	65	64	63	63	63	61	58	59	59	58	58	57
55	57	55	55	54	53	52	52	53	52	51	52	49	50	50
50	50	49	49	49	48	48	47	48	48	48	46	46	45	44
44	44	44	45	45	45	44	44	43	43	46	44	43	43	43
43	42	42	42	41	41	40	40	40	40	40	40	38	39	39
38	38	38	38	38	37	37	37	37	37					

differential computation

0	0	4	1	0	1	1	0	0	2	3	3	2	1	1
1	2	3	3	3	4	6	10	12	11	9	9	11	10	7
5	7	9	8	6	6	7	7	7	6	5	5	6	5	3
2	2	1	0	2	4	4	3	2	2	1	0	0	-1	-1
0	1	1	0	0	0	0	0	0	0	0	0	0	-1	-2

-2 0 2 2 0 -2 -1 0 1 1 1 0 -1 0 1

1 0 0 0 -1 -2 -2 0 2 1 -1 -2 -1 0 0

1 1 1 0 -4 -5 -4 -4 -6 -8 -5 -3 -4 -5 -8

-10 -11 -10 -8 -7 -8 -9 -9 -8 -7 -7 -6 -6 -5 -4

-4 -5 -6 -6 -5 -3 -3 -7 -9 -6 -2 -1 -3 -5 -4

-3 -2 -3 -5 -5 -5 -2 0 -1 -2 -4 -4 -3 -1 1

-1 -2 -2 -2 -2 -3 -2 0 1 -1 -4 -5 -5 -4 -3

-1 1 2 2 0 -2 -3 -3 1 3 1 -3 -4 -1 -1

-2 -2 -2 -2 -3 -3 -2 -1 0 0 -2 -3 -2 -1 -1

-2 -1 0 -1 -2 -2 -1 0 0 0

edge specification

max_dif: 12, min_dif: -11, left_edge: 23, right_edge: 121

Table-8.13 Pixel values and differentiations

for right-deflected arc

window dimensions

x_loc: 20, y_loc: 150, x_size: 220, y_size: 1

pixel reading

31	33	33	33	33	34	35	35	35	35	35	36	35	36	36
36	37	37	37	38	38	38	38	39	39	39	40	40	41	41
41	41	41	41	41	41	41	42	42	42	42	42	42	41	42
42	42	42	43	42	42	42	42	43	43	43	43	43	44	44
45	46	46	47	47	47	47	49	48	50	51	52	52	52	52
53	53	54	55	56	56	56	58	59	59	60	61	61	63	66
67	71	75	77	79	81	80	82	83	84	86	89	89	91	92
93	95	93	93	93	94	94	94	95	95	95	95	95	96	95
95	95	95	95	93	93	94	94	95	94	95	95	95	96	94
94	96	95	95	95	96	94	96	93	94	96	95	95	95	95
94	92	94	94	96	93	95	94	94	93	92	92	92	93	91
91	90	90	89	87	87	87	86	84	84	81	81	80	78	76
73	69	68	65	62	62	59	57	55	51	53	51	48	47	46
46	44	44	44	41	41	41	41	40	40	41	41	40	40	39
39	39	39	39	39	39	37	38	38	38					

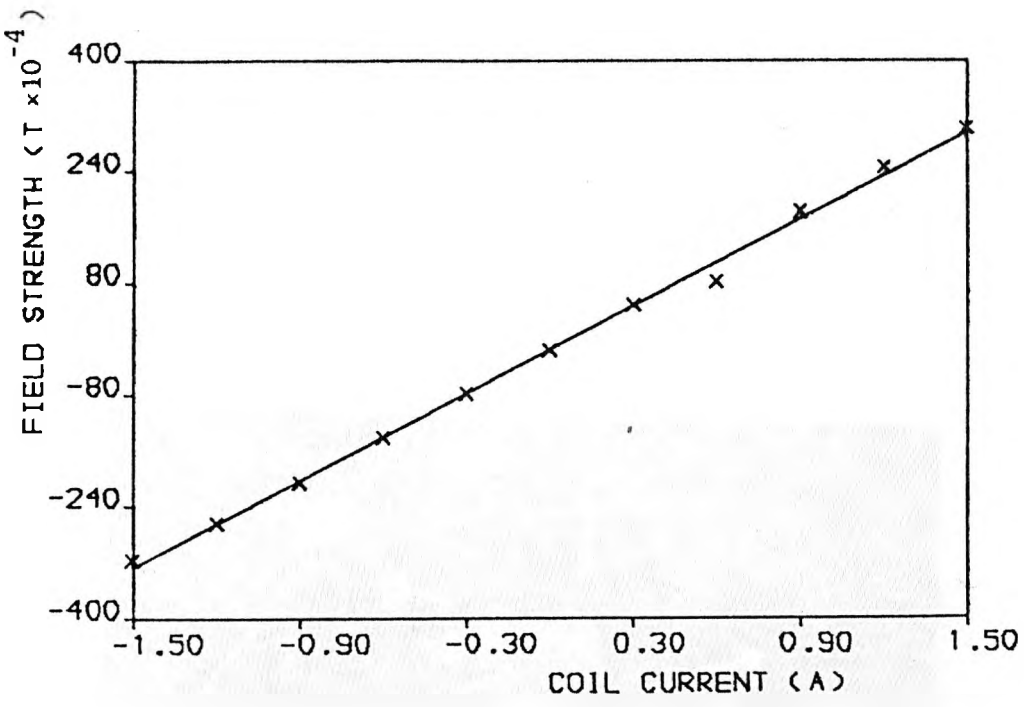
differential computation

0	0	2	1	3	4	3	1	0	1	1	1	1	1	2
2	2	2	2	2	1	1	2	2	2	2	3	3	2	1
0	0	0	0	0	1	2	2	1	0	0	-1	-1	0	1
1	1	1	0	-1	-1	1	2	2	1	0	1	2	3	4
4	4	3	2	1	2	3	4	5	6	6	3	1	1	2

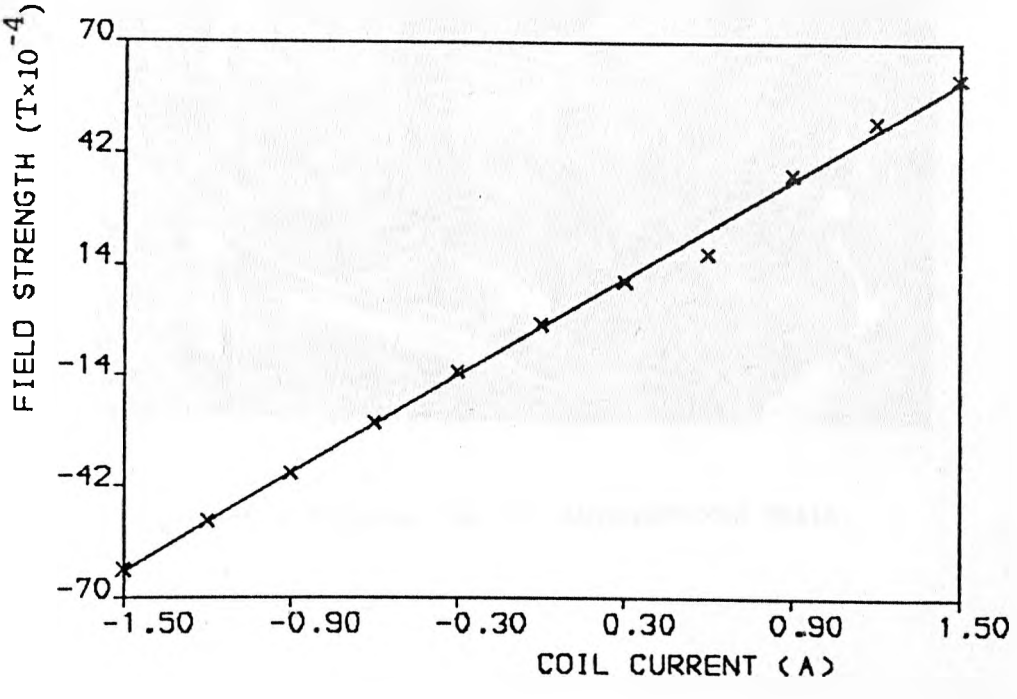
3	4	5	5	3	3	5	6	5	4	4	5	8	11	14
17	19	18	14	9	6	5	6	8	10	11	10	8	7	8
5	1	-2	-1	2	2	2	2	2	1	0	1	1	0	-1
-1	0	-2	-4	-3	0	3	2	1	1	1	2	0	-2	-1
1	2	0	0	0	0	-2	-3	0	2	3	0	-1	-1	-4
-4	-1	4	3	0	-1	-1	-1	-4	-4	-3	0	0	-2	-4
-4	-3	-5	-6	-5	-3	-4	-6	-8	-8	-7	-7	-8	-12	-16
-17	-16	-15	-13	-12	-11	-12	-15	-12	-8	-7	-9	-11	-7	-5
-5	-4	-5	-6	-6	-3	-1	-2	-1	1	1	-1	-3	-3	-2
-1	0	0	0	-2	-3	-2	0	0	0					

edge specification

max_dif: 19, min_dif: -17, left_edge: 91, right_edge: 180



(a) Field strength at electromagnet tip



(b) Field strength at electrode tip

Figure-8.1 Field strength against coil current

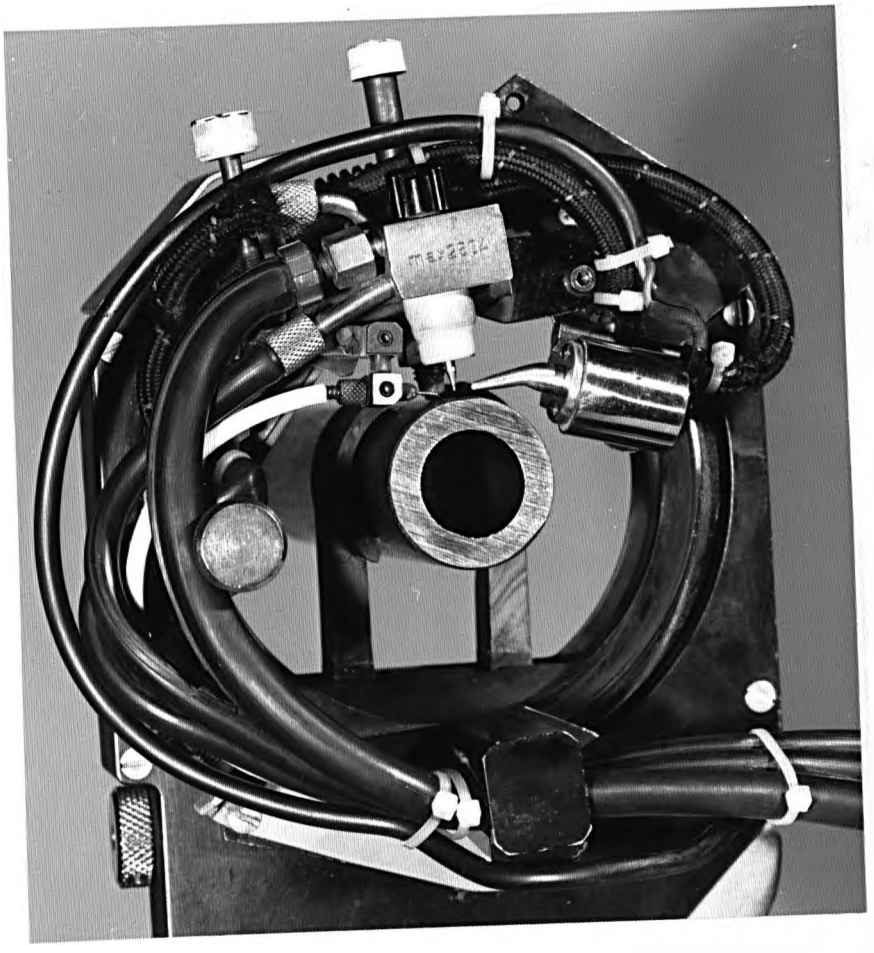
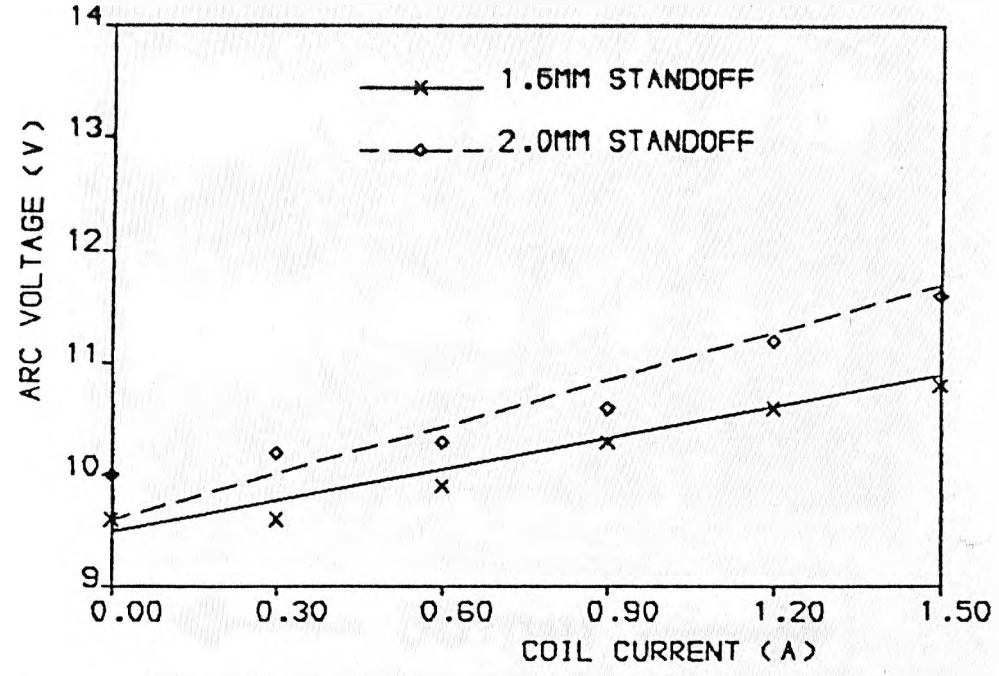
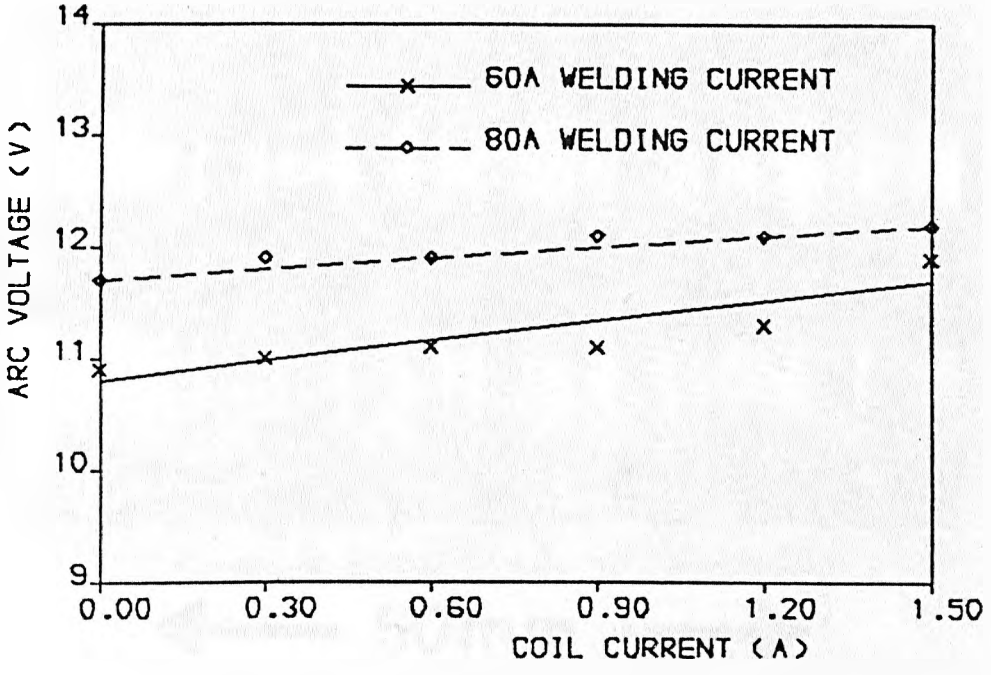


Figure-8.2 Welding rig for experimental tests



(a) Welding current: 40A, varied standoff



(b) Arc standoff: 1.6mm, varied welding current

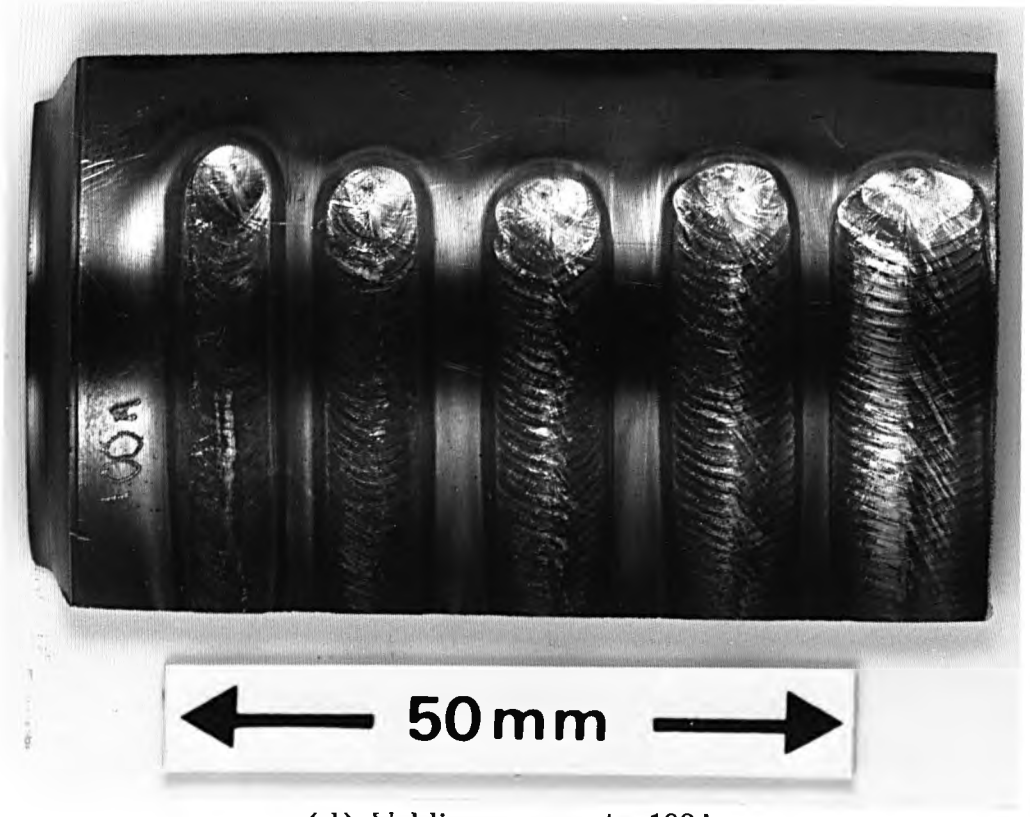
Figure-8.3 Influence of arc deflection on arc voltage

(a) Welding current: 40A

(b) Welding current: 60A

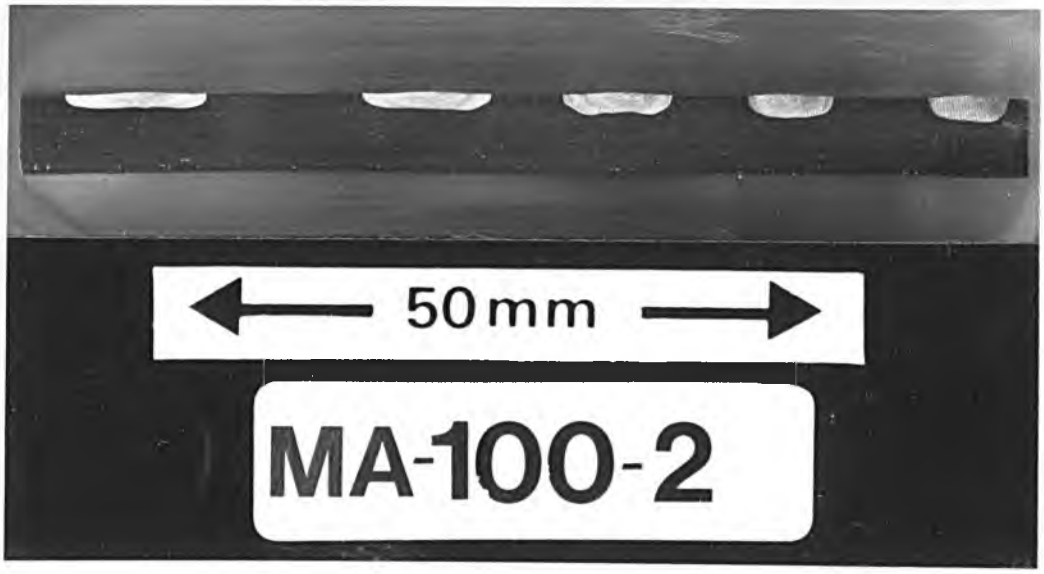
Figure-8.4 Weld beads made with 1.6mm arc standoff

(c) Welding current: 80A

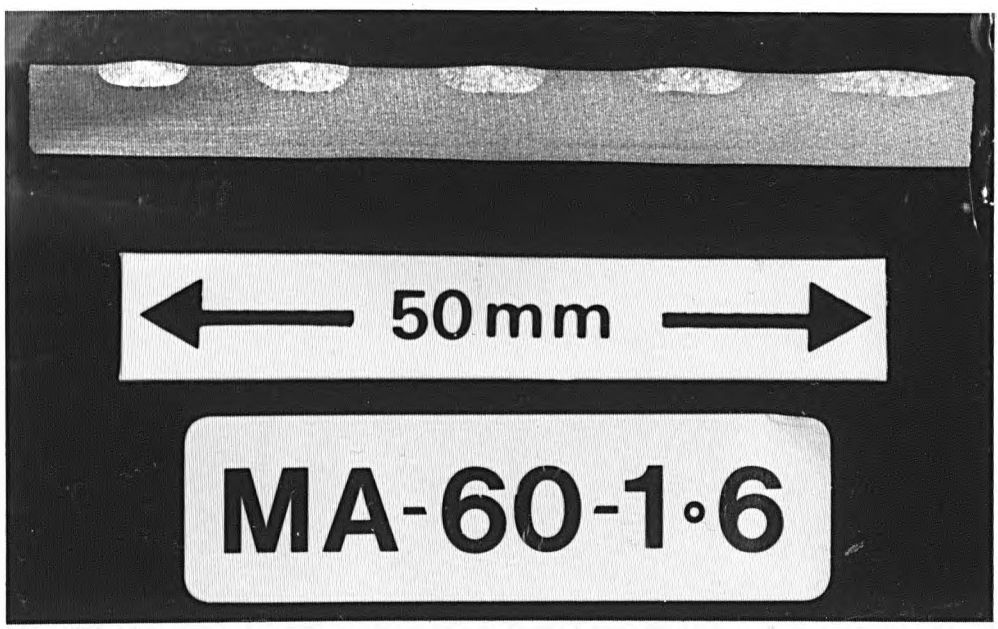


(d) Welding current: 100A

Figure-8.4 Weld beads made with 1.6mm arc standoff (cont.)



(a) Arc standoff: 2.0mm



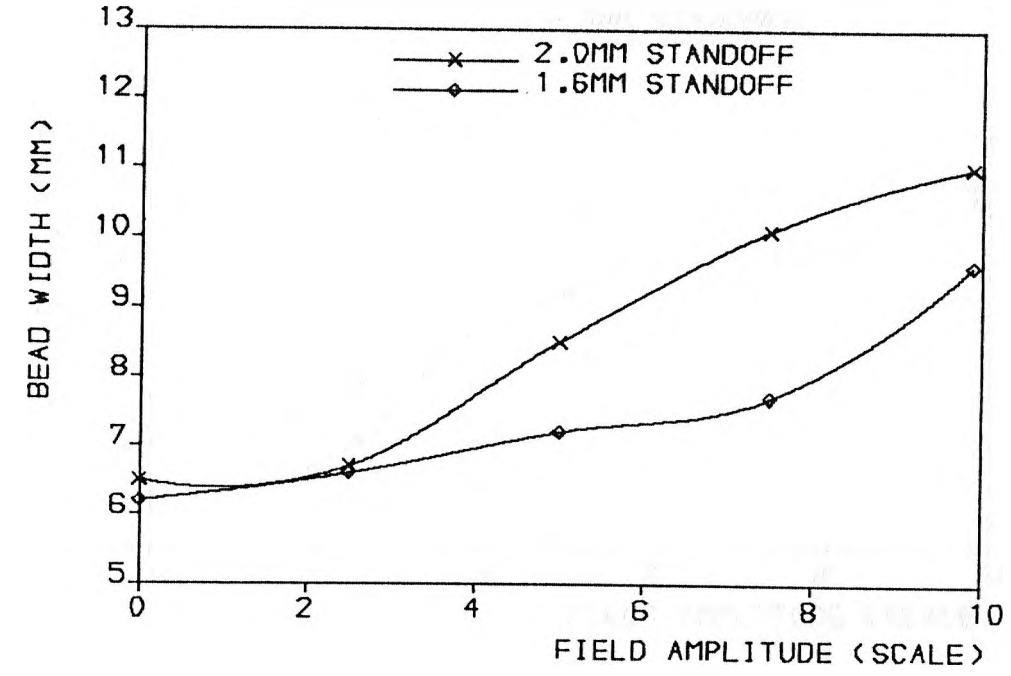
(b) Arc standoff: 1.6mm

Figure-8.5 Cross sections of beads made with 100A current

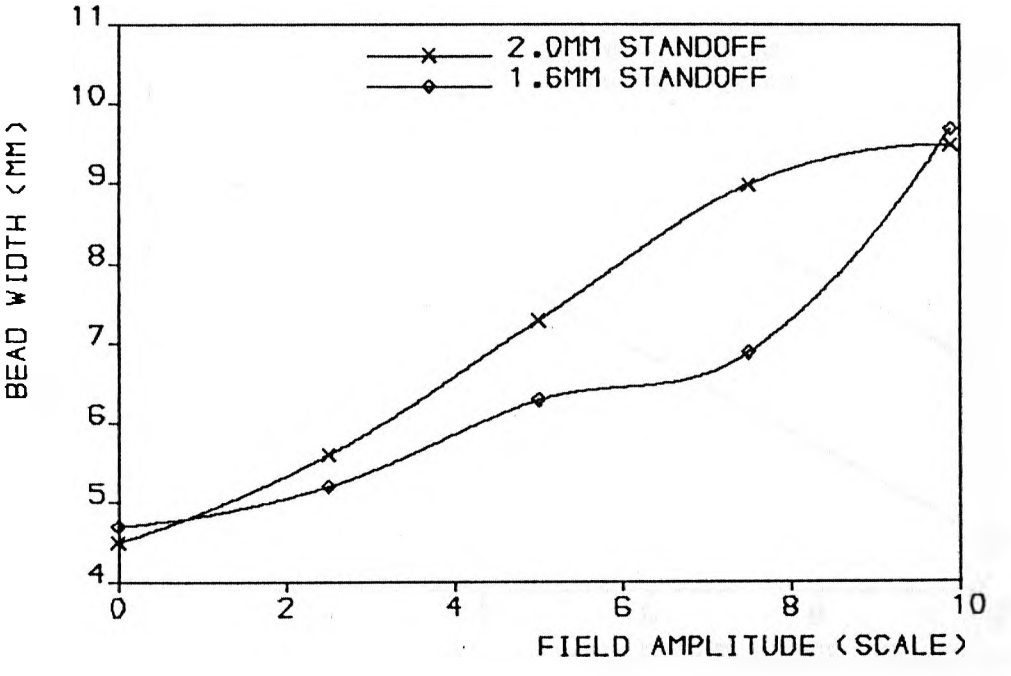
(a) Arc standoff: 2.0mm

(b) Arc standoff: 1.6mm

Figure-8.6 Cross sections of beads made with 60A current

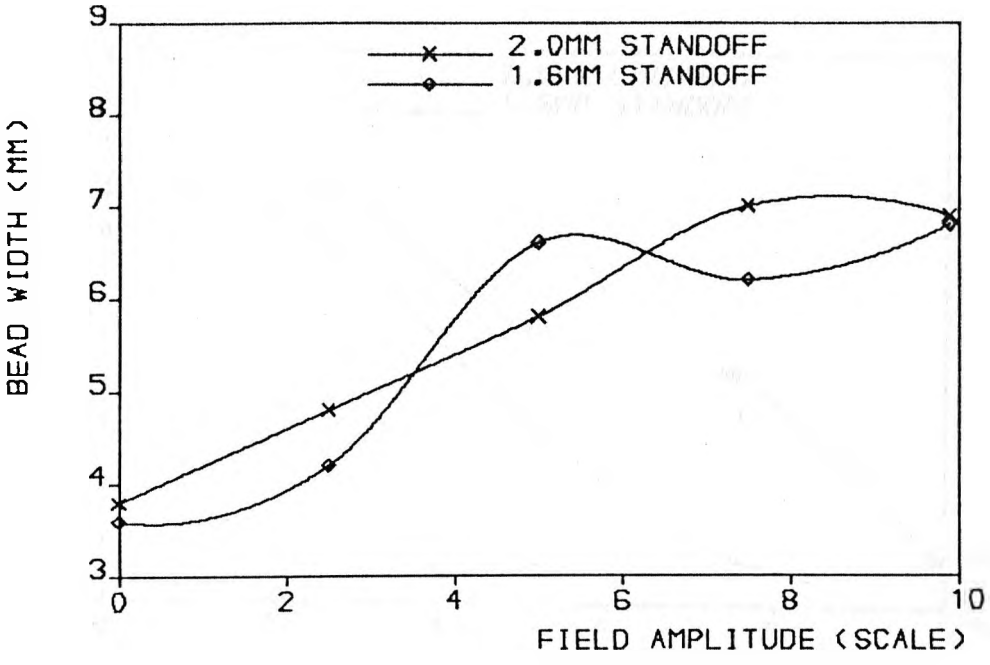


(a) Welding current: 100A



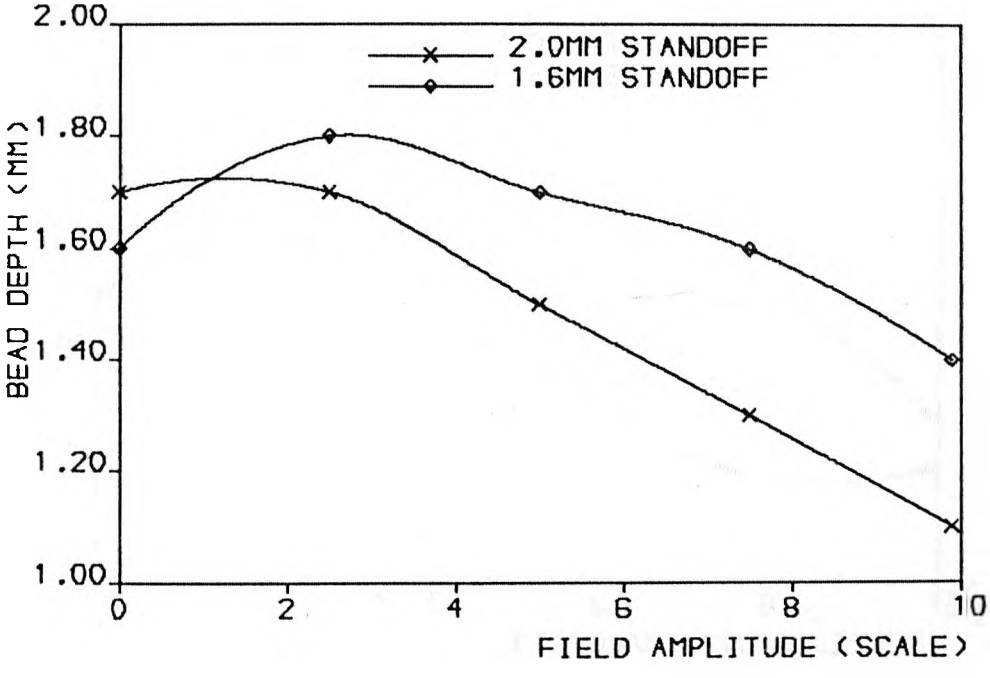
(b) Welding current: 80A

Figure-8.7 Influence of field amplitude and arc standoff on bead width



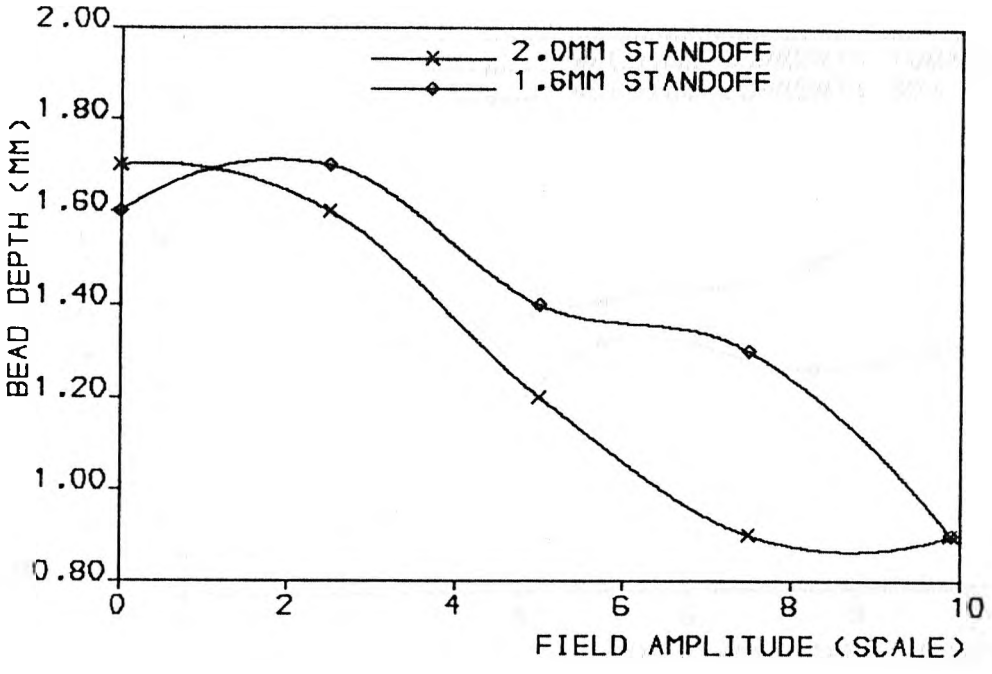
(c) Welding current: 60A

Figure-8.7 Influence of field amplitude and arc standoff on bead width (cont.)

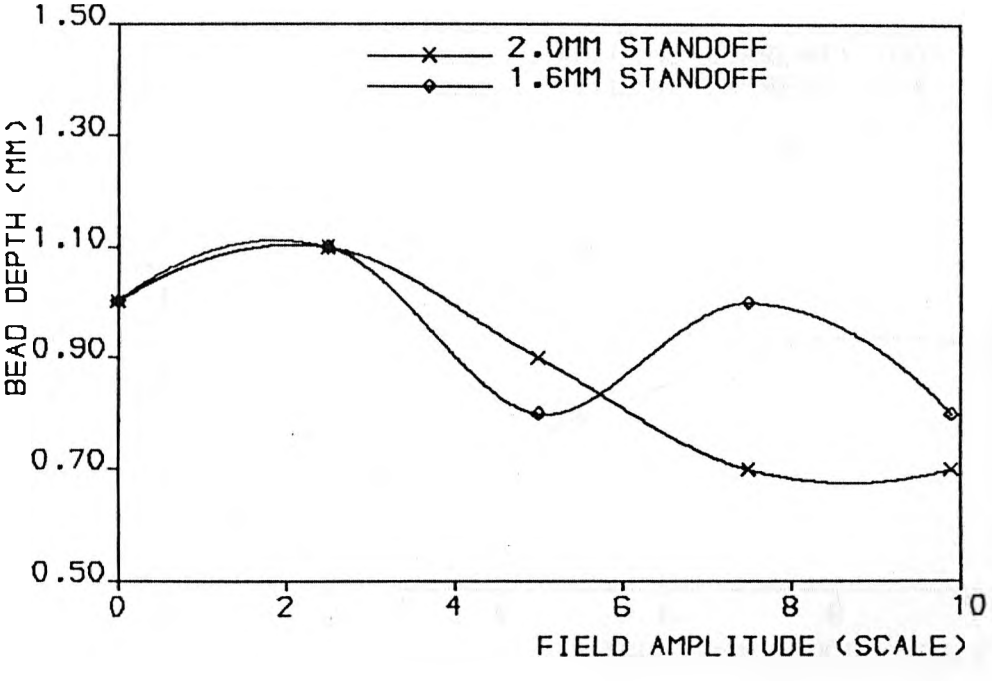


(a) Welding current: 100A

Figure-8.8 Influence of field amplitude and arc standoff on bead depth

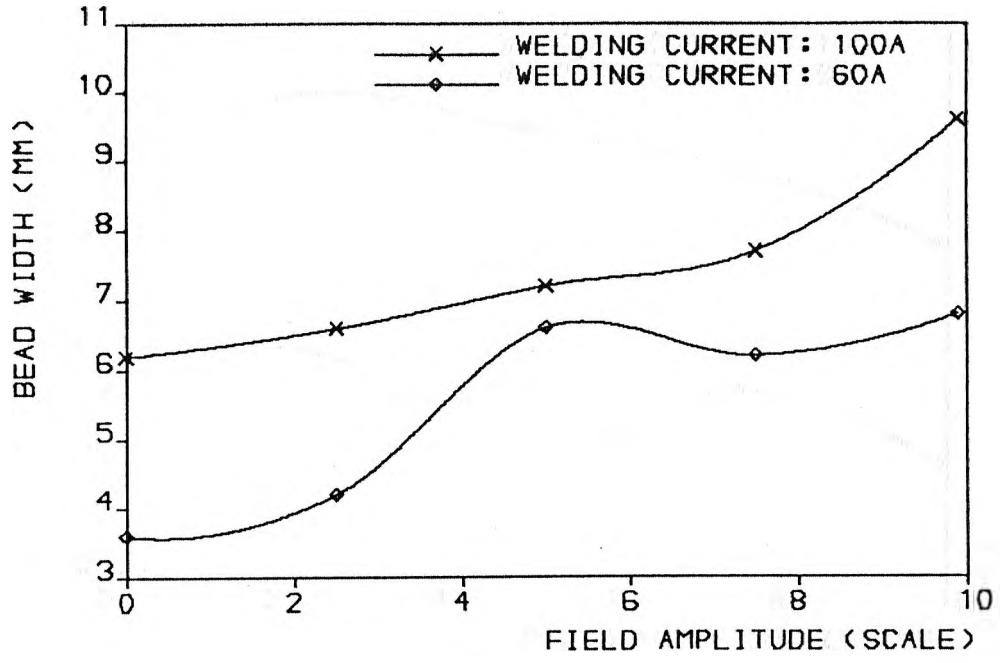


(b) Welding current: 80A

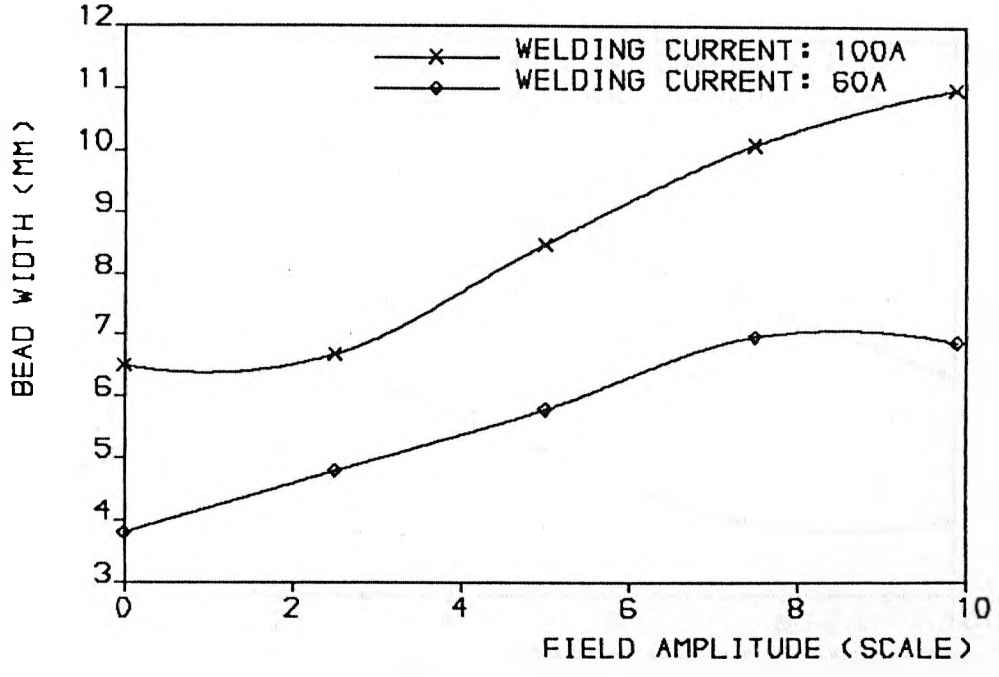


(c) Welding current: 60A

Figure-8.8 Influence of field amplitude and arc standoff on bead depth (cont.)

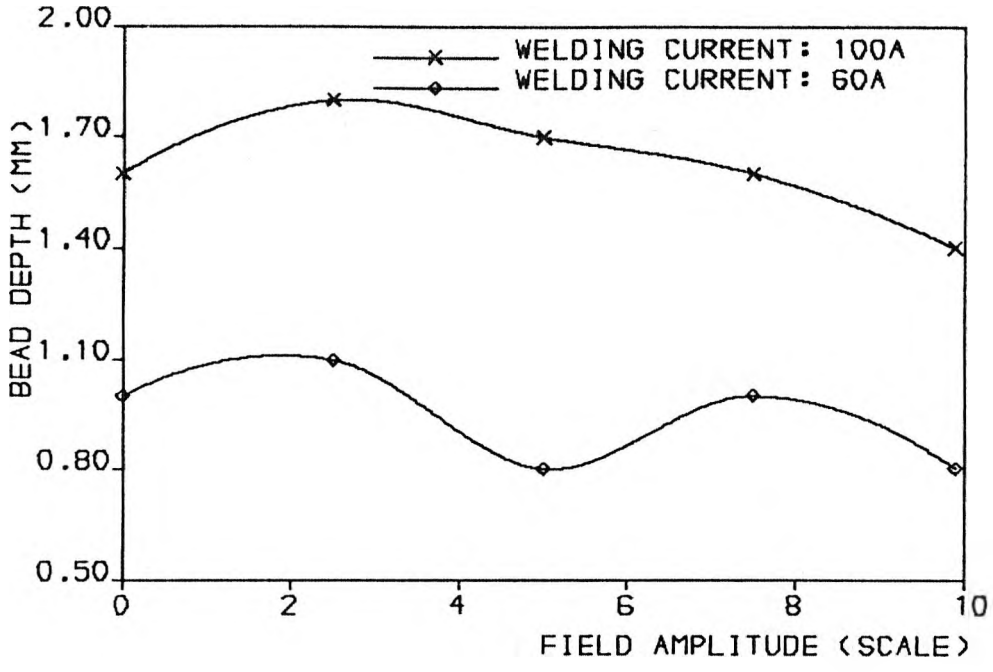


(a) Arc standoff: 1.6mm

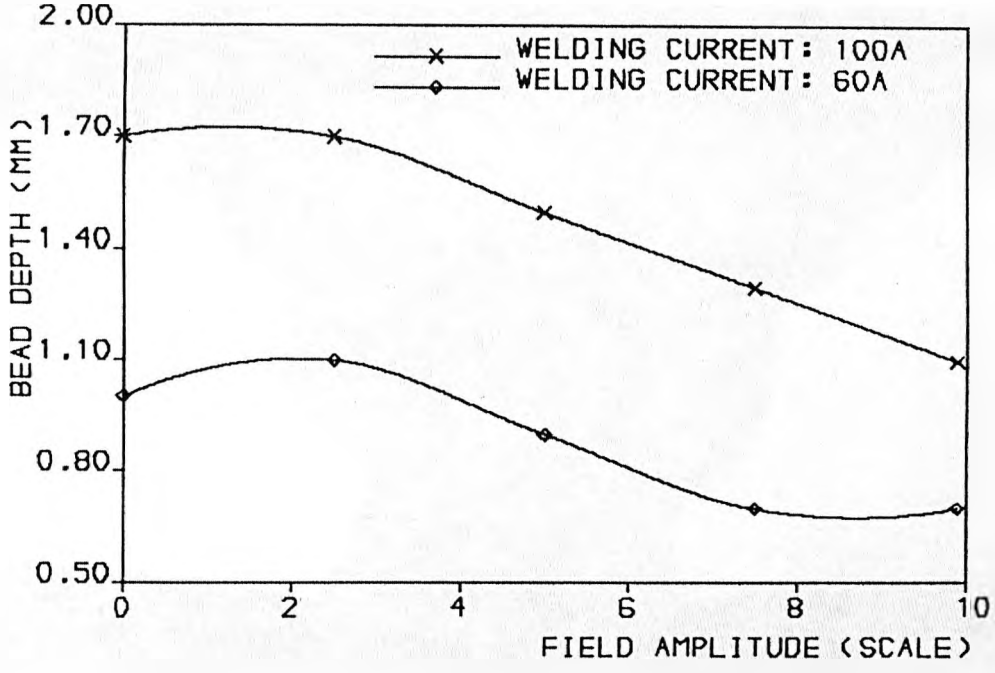


(b) Arc standoff: 2.0mm

Figure-8.9 Influence of field amplitude and welding current on bead width

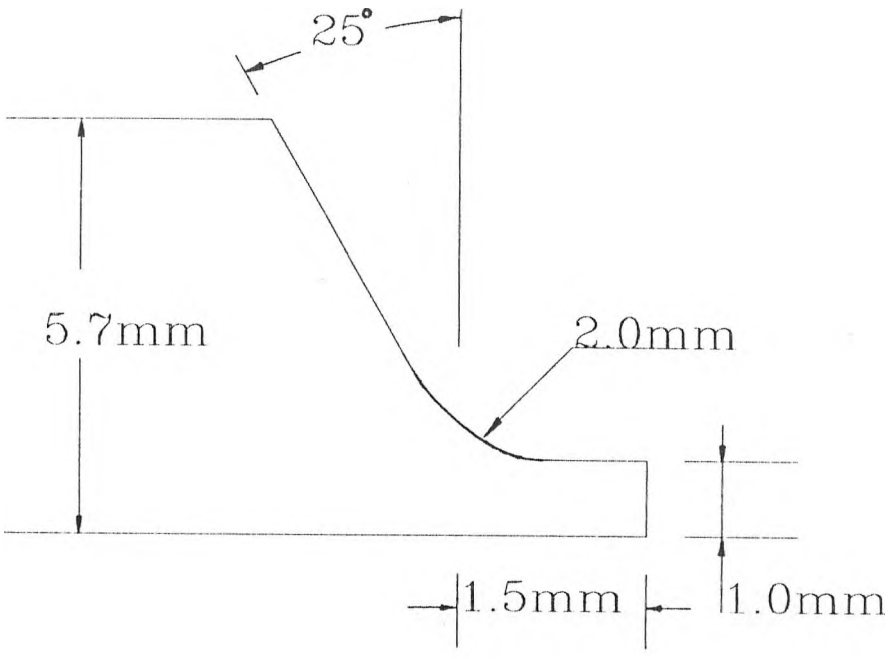


(a) Arc standoff: 1.6mm



(b) Arc standoff: 2.0mm

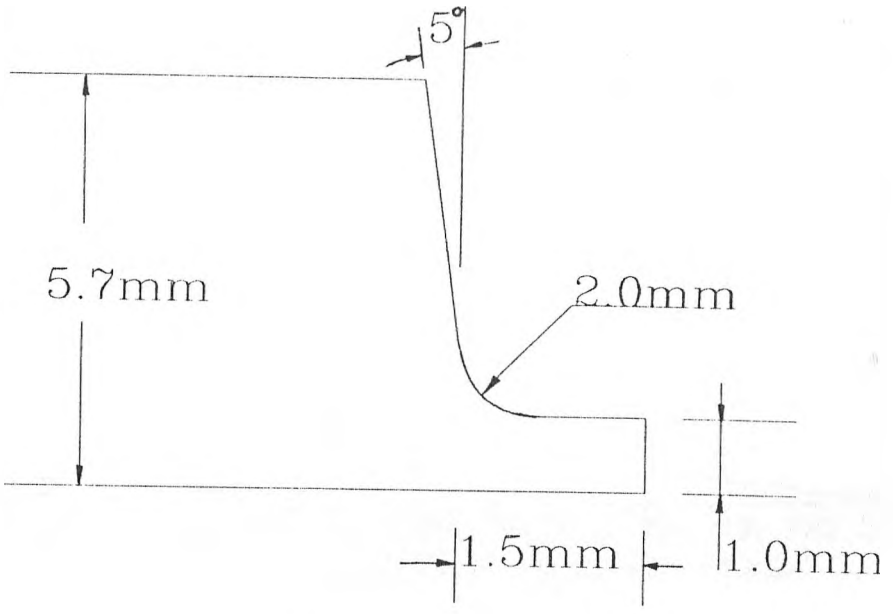
Figure-8.10 Influence of field amplitude and welding current on bead depth



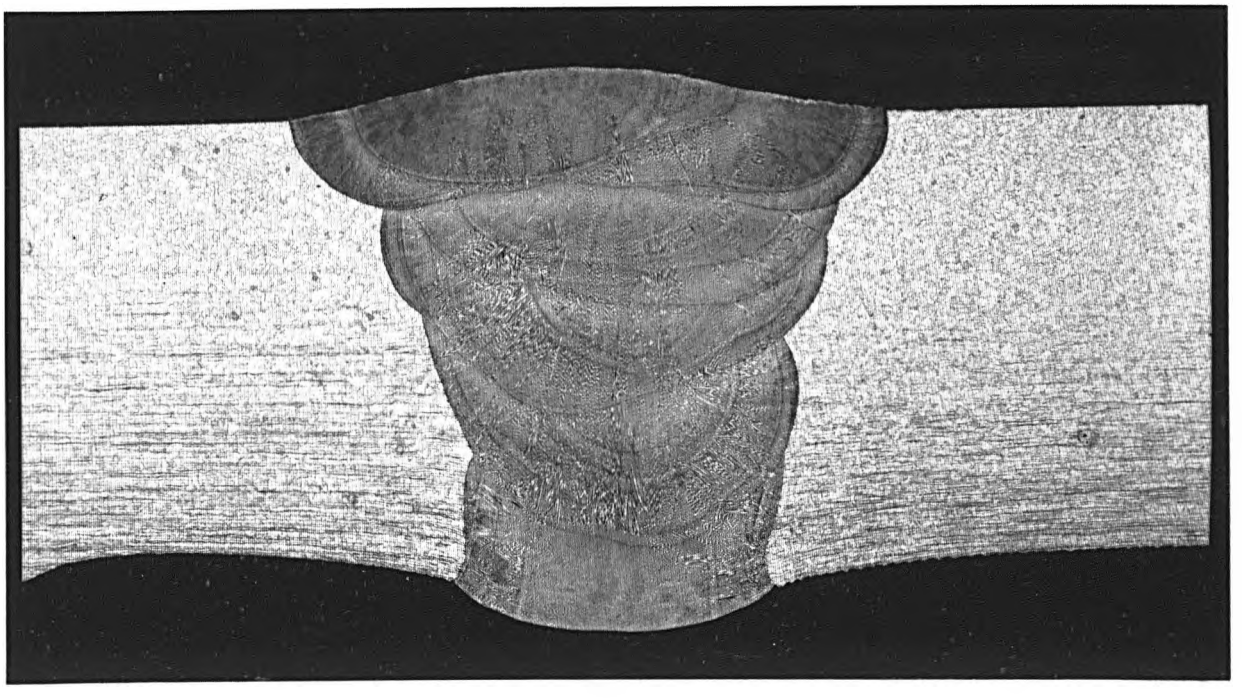
(a) "J" preparation

(b) Cross section of weld

Figure-8.11 "J" preparation and cross section of weld

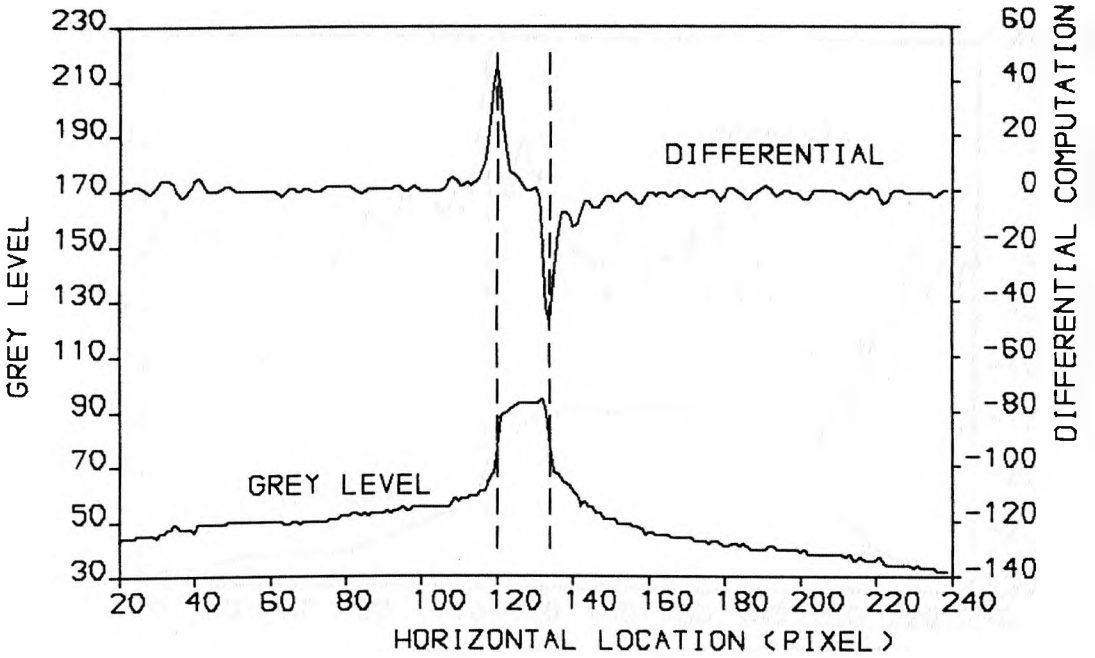


(a) "Narrow gap" preparation

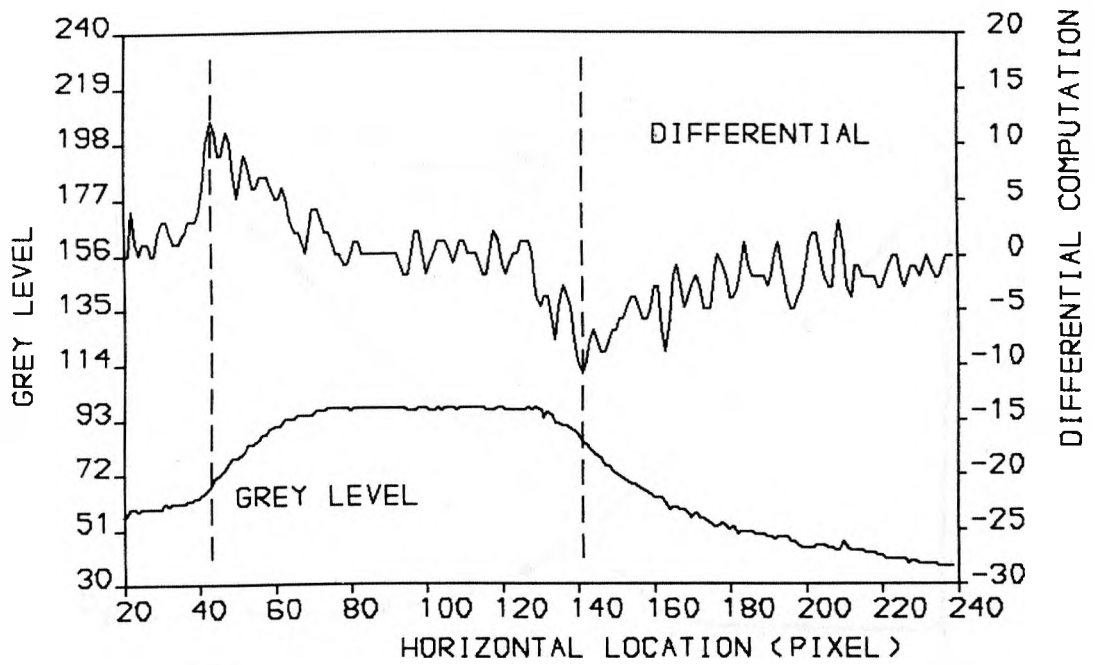


(b) Cross section of weld

Figure-8.12 "Narrow gap" preparation and cross section of weld

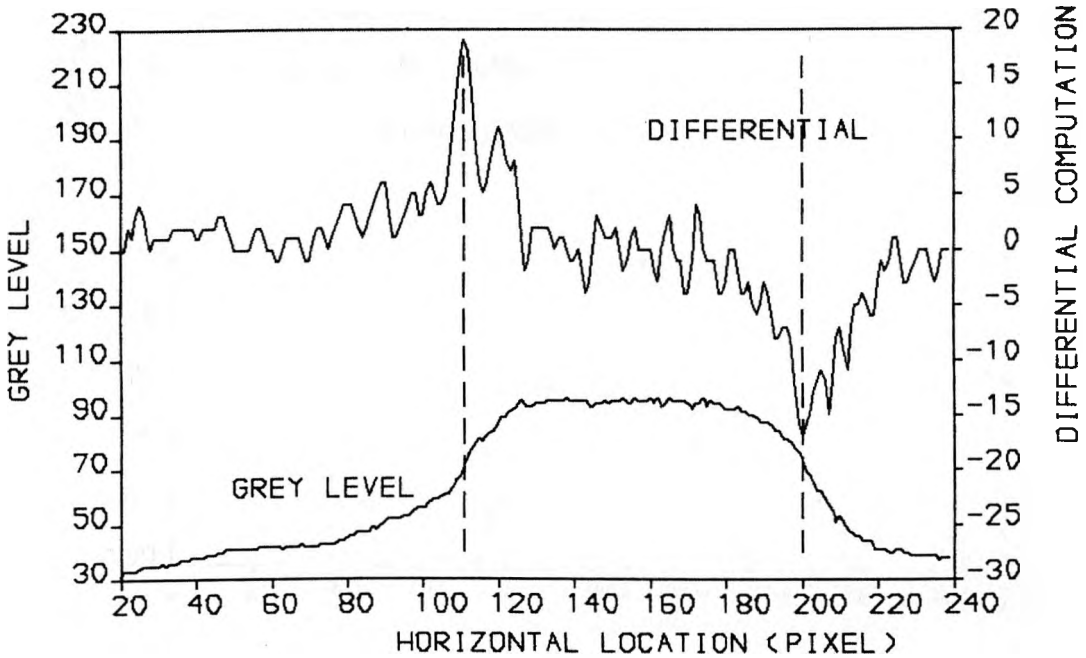


(a) tracking electrode



(b) tracking left-deflected arc

Figure-8.13 Pixel values and differentiations in tracking electrode and arc



(c) tracking right-deflected arc

Figure-8.13 Pixel values and differentiations in tracking electrode and arc (cont.)

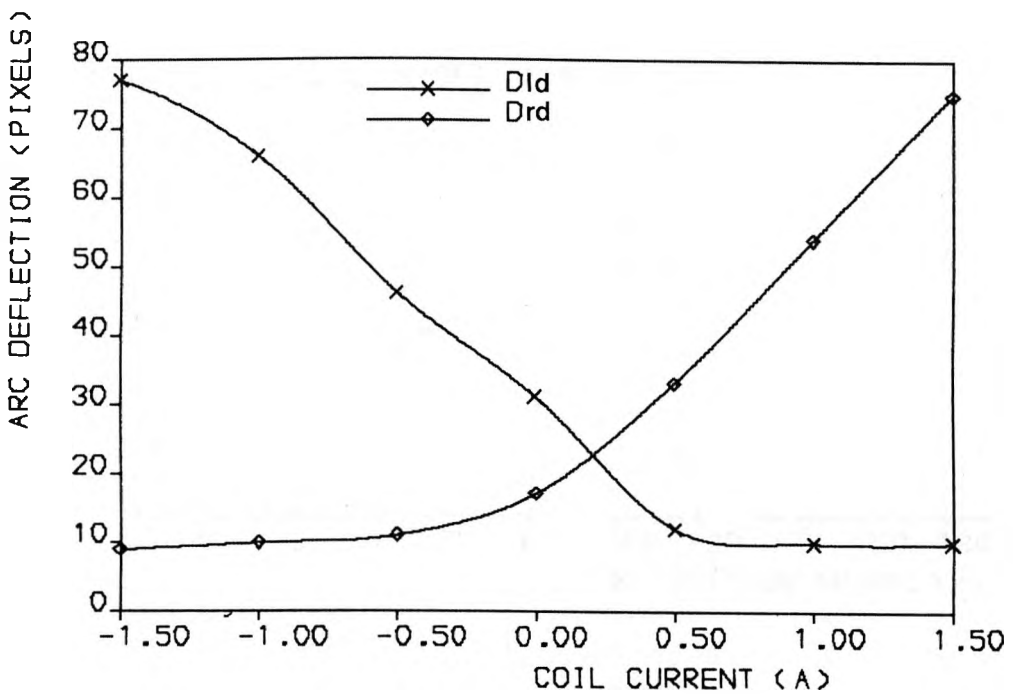
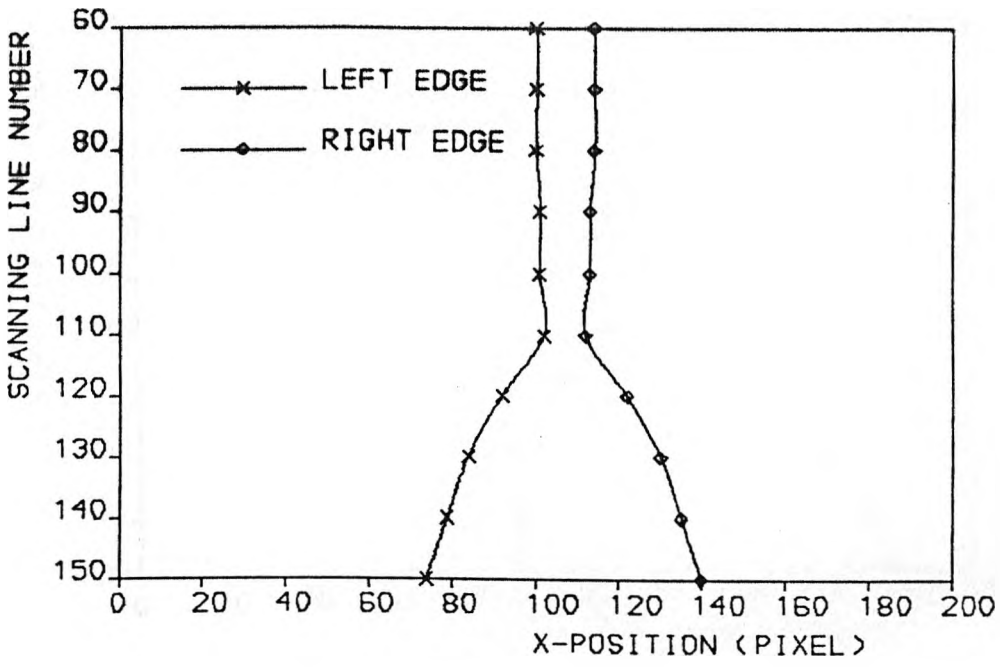
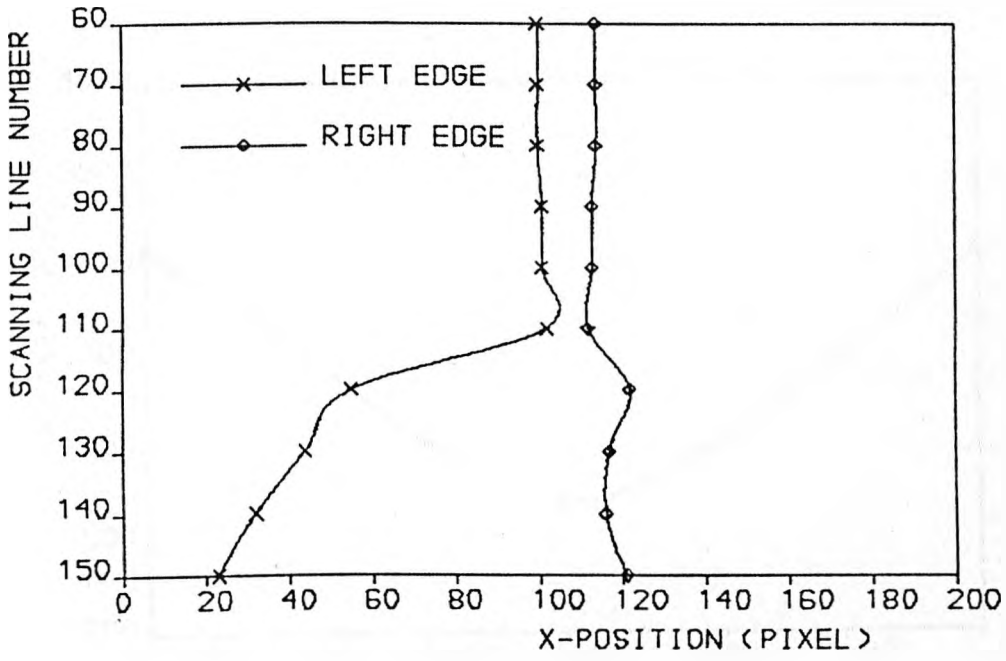


Figure-8.14 Functions of arc deflection as field amplitude (1 pixel = 0.12 mm)

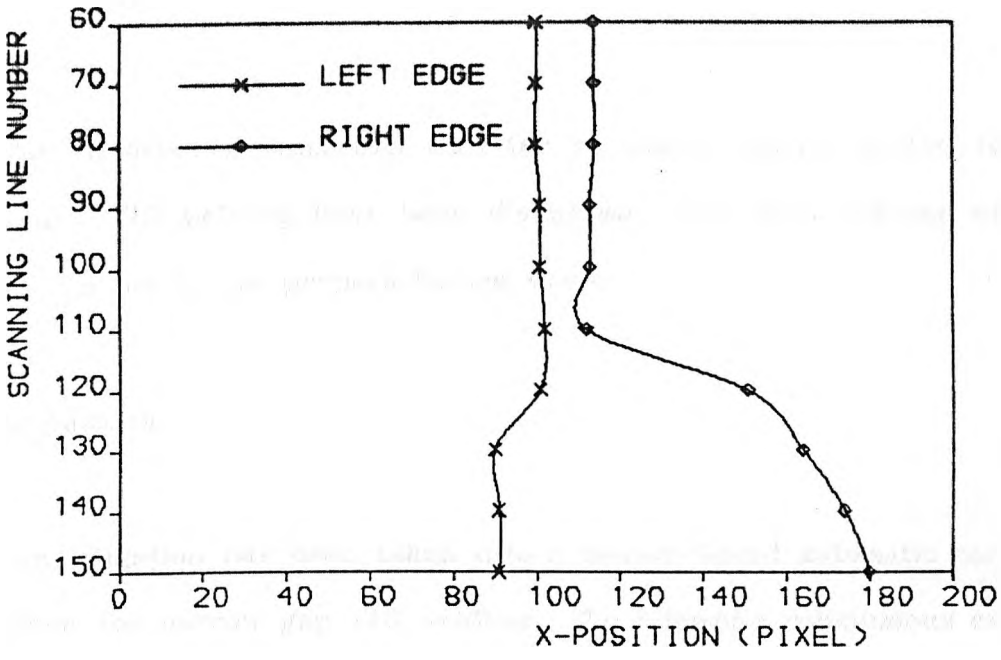


(a) Non-deflected arc



(b) Left-deflected arc

Figure-8.15 Arc shape detected by the vision system (1 pixel = 0.12 mm)



(c) Right-deflected arc

Figure-8.15 Arc shape detected by the vision system (cont.)
(1 pixel = 0.12 mm)

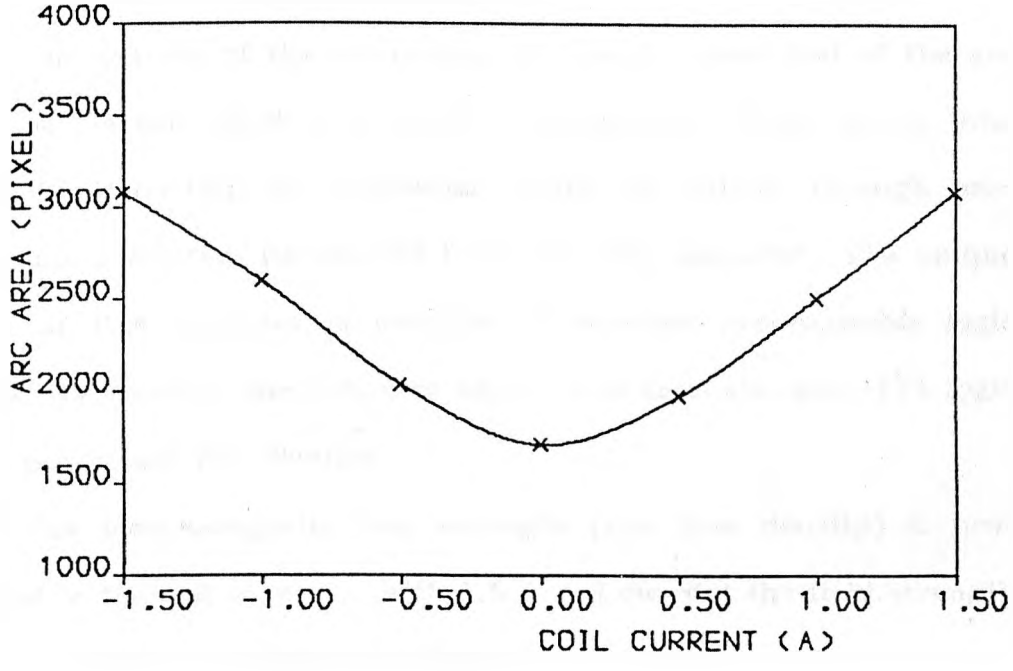


Figure-8.16 Function of arc area as field amplitude
(1 pixel = 0.014 mm^2)

9. CONCLUSIONS AND FUTURE WORK

So far all aspects concerned with the automatic control system for narrow gap TIG welding have been discussed. This final chapter will conclude the work and propose future work.

9.1 Conclusions

An investigation has been taken into a sensor-based automatic control system for narrow gap TIG welding. The following conclusions can be made:

(1) Sidewall penetration poses a difficulty in narrow gap welding. This problem can be solved by applying a transverse electromagnetic field parallel to the weld line. The welding arc is deflected towards sidewalls to improve sidewall penetration.

(2) The pattern of the electromagnetic field, hence that of the arc oscillation, is controlled by a 16-bit microcomputer based on an Intel 8088 CPU. Open-loop arc oscillation control is realised through presetting the oscillation parameters from the 8088 computer. The unique feature of this computer is adoption of erasable programmable logic devices. As a result, the computer board, free from standard TTL logic chips, is compact and flexible.

(3) The electromagnetic field strength (i.e. flux density) is proportional to the coil current. With 1.5 A coil current the field strength at the electrode tip reaches 60 Gauss.

(4) The voltage sensor resident in the 8088 computer implements real-time feedback arc length control. With an introduction of slight

mechanical weaving of the torch around the centre of the groove, the voltage signal can be used to carry out seam tracking.

(5) More advanced real-time control is implemented by a robust vision system. The welding scene is observed by a TV camera without artificial illumination. An IBM PC compatible frame store digitises the captured image into 256 × 256 pixels with 7-bit grey levels. The host IBM PC accesses the frame store by reading from or writing to the frame memory.

(6) Both edge finding and template matching methods have been developed to analyse the grey level image. The former processes a line of pixels, tracking targets at a very high speed. The latter processes an area of pixels with a relatively slow speed.

(7) Fast vision algorithms, constructed entirely in C, extracts vision information about the position of the tungsten electrode and the arc. This information is fed back to the torch manipulator for real-time seam tracking and arc length control, and to the 8088 computer for feedback control of arc oscillation parameters.

(8) Bead width increases with the field amplitude, and the reverse is applied to bead depth. However, a slight arc oscillation improves the bead penetration. A higher arc standoff results in a wider, but less deep bead. A high current improves both bead width and bead depth. An oscillated arc produces fine and even ripples, therefore improving bead appearance.

9.2 Future Work

The work discussed in this thesis involves two major facets: magnetic arc oscillation and vision system. Although narrow gap TIG welding is of primary concern throughout this project, the ideas dis-

cussed in this work can be extended to other areas of welding engineering.

As compared with mechanical weaving of welding torch, magnetic arc oscillation does not involve complex actuators and inertia of mechanical movement. An arc can be magnetically oscillated as fast as 50 Hz which can not be readily achieved in a mechanical way. The idea of magnetic arc oscillation can be applied to many welding processes to control weld bead geometry and fusion characteristic. An example is thin plate welding, the primary concern of which is distortion due to concentrated heating. A common approach to solve the problem of distortion is to use pulsed arc, allowing the weld pool to cool down with a low background current. An alternative way is to employ a transverse magnetic field perpendicular to the weld line, therefore oscillating the arc back and forth along the weld line and reducing heat concentration, hence distortion. Magnetic oscillation of an arc in a normal welding procedure can control the bead geometry and fusion characteristic.

As far as vision system is concerned, it can be used in either quality inspection and real-time control. In fact, a number of attempts have been made in the area of vision inspection. A x-ray image can be analysed by a vision system [141]. Real-time measurements of weld bead geometry is another application of vision system. For real-time application, the paramount requirement is fast image processing speed. A promising approach is to use transputer and parallel processing in image analysing.

Another encouraging trend is the widespread use of computers in welding. In particular, microprocessors and microcomputers have

found wide application. They can be used for welding power supply control [142], or for process monitoring and control [143]. In recent years, welding expert systems have drawn much attention from welding researchers. A expert system acts as a "master" consultant. It has the knowledge from references, books, and welding experts. It gives advice to a user on a specific subject, for instance welding costs and procedures. The Welding Institute [144] has designed a number of expert systems for welding engineers. These include PREHEAT which gives advice on welding carbon-manganese steels, WELDCOST which compares costs of different welding processes and procedures, and WELDSPEC which is a data base of welding procedures. Expert systems to generate arc welding procedures have emerged [145]. Some schools have proposed computer-aided design of electrodes for manual metal arc welding [146]. It is no doubt that computer application in welding engineering continues to grow.

REFERENCE

[1] Koenigsberger, F., "Welding technology", 3rd ed., 1965, St Ann's press, Altrincham, U.K

[2] Malin, V.Y., "The state-of-the-art of narrow gap welding (part 2)", Welding Journal, June 1983, pp. 37 -46

[3] Henderson, I.D., "Developments in narrow gap welding." Australian Welding Journal, May/June 1987, pp. 4-15.

[4] Malin, V., "Monograph on narrow-gap welding technology", WRC Bulletin 323, May 1987.

[5] Render, G.S., "Welding advances in power plant construction: narrow gap welding", Metal Construction, November 1984, pp. 696-700.

[6] Hutt, G.A., "Narrow gap welding", Metal Construction, June 1984, pp. 355-361.

[7] Ellis, D.J., "Mechanised narrow gap welding of ferritic steel", Joining & Materials, August 1988, pp. 80-86.

[8] Lochhead, J.C., "Welding techniques in pressure part technology", Welding Technology in Japan, The Welding Institute, Cambridge, 1984, pp. 119-129.

[9] Nakayama, H. et al., "Application of narrow-gap automatic CO_2 arc weaving process to heavy steel structures of building", Proc. of 2nd Int. Symp. on Advanced Welding Technology, Japan Welding Society, 1975, pp. 403-408.

[10] Ito, T. et al., "Development of arc oscillating narrow-gap welding process", Proc. of 2nd Int. Symp. on Advanced Welding Technology, Japan Welding Society, 1976 Vol. 2, pp. 391-395.

[11] Halmoy, E. et al., "Adaptively controlled MIG narrow gap welding", Developments and Innovations for Improved Welding Production, 1983 Vol. 1, pp. p8-1 to p8-4.

[12] Iversen, K. & Palussek, A.P., "Narrow-gap inert gas-metal-arc welding of pressure vessels made from the 2.4663 nickel alloy", Developments and Innovations for Improved Welding Production, 1983 vol. 1, pp. p26-1 to p26-8.

[13] Kennedy, N.A., "Narrow gap submerged-arc welding of steel (part1): applications", Metal Construction, November 1986, pp. 687-692.

[14] Kennedy, N.A., "Narrow gap submerged-arc welding of steel (part2): equipments, consumables and metallurgy", Metal Construction, December 1986, pp. 765-769.

[15] Timchenko, V.A., "Multipass circumferential welds made in thin-walled products using an automatic control system", Automatic Welding, 1975 Vol. 2, pp. 35-37.

[16] Grin, V.V. & Shtrikman, M.M., "Increasing the level of automation of argon-arc non-consumable electrode", Welding Production, 1982 Vol. 7, pp. 21-24.

[17] Uratani, Y. et al., "Application of narrow-gap GTA welding process to the welding of large type stainless steel nuclear pressure vessels", IIW Doc. XII-E-42-83, December 1982.

[18] Cook, G. & Levick, P.C., "Narrow gap welding with hot wire GTA process", Welding Journal, August 1985, pp. 27-31.

[19] Levick, P.C. & Cook, G. E., "Pipe welding with the HW-GTAW process", Developments and Innovations for Improved Welding Production, 1983 Vol. 1, pp. p10-1 to p10-6.

[20] Steen, W.M. & Eboo, M., "Arc augmented laser welding", Metal Construction, 1979 11(7), pp. 332-335

[21] Matsuda, J. et al., "TIG or MIG arc augmented laser welding of thick mild steel plate", Joining & Materials, July 1988, pp. 31-34.

[22] Jones, R. et al., "Tubular cored wires", Joining & Materials, Feburary 1989, pp. 55-77.

[23] Butler, C.A. et al., "Narrow-gap welding - a process for all positions", Welding Journal, February 1969, pp. 102-108.

[24] Sawada, S. et al., "Application of narrow-gap process", Welding Journal, September 1979, pp. 17-25.

[25] Richardson, R.W., "Robotic weld Joint tracking systems - theory and implementation methods", Welding Journal, November 1986, pp. 43-51.

[26] Fenn, R., "Sensors, present and future", Welding and Metal Fabrication, October 1984, pp. 313-315.

[27] Hanright, J., "Robotic arc welding under adaptive control - a survey of current technology", Welding Journal, November 1986, pp. 19-24.

[28] Munezane, Y. et al., "A sensing system for arc welding robots", 15th Int. Symp. Industrial Robots, 11-13 September 1985, Tokyo.

[29] Laing, B. et al., "Narrow gap welding of HY-100 plate using closed loop, adaptive feedback, through-the-arc tracking technology", Welding Journal, November 1985, pp. 39-42.

[30] Cook, G. et al., "Electric arc sensing for robot positioning control", Robotic Welding, by J. Lane, IFS Publication Ltd., 1987.

[31] Cook, J. et al., "Weld tracking/electronic arc sensing system", U.S. Patent 4,336,440, June 22 1982.

[32] Fujimura, H. et al., "Joint tracking control sensor of GMAW - development of method and equipment for position sensing in welding with electric arc signals (report 1)", Transactions of the Japan Welding Society, 1987 18(1), pp. 23-31.

[33] Fujimura, H. et al., "Weave amplitude control sensor of GMAW - development of method and equipment for position sensing in welding

with electric arc signals (report 1)", Transactions of the Japan Welding Society, 1987 18(1), pp. 32-36.

[34] Essers, W.G. & Gompl. M.R.M., "Arc control with pulsed GMA Welding", Welding Journal, June 1984, pp. 26-32.

[35] Eichhorn, F. et al., "GMA welding of filler passes with a new type of arc-controlled seam tracking system", Proc. 1st Int. Conf. Advanced Welding Systems, the Welding Institute, Abington, Cambridge, 1984, pp. p18-1 to p18-10.

[36] Madigan R.B. et al., "Computer-aided control of full penetration GTA welds using pool oscillation sensing", 1st Int. Conf. Computer Technology in Welding, London, 3-5 June 1986.

[37] Hollinggum, J., "Assessing weld quality with the aid of 'direct arc sensing'", The Industrial Robot, March 1987, pp. 20-24.

[38] Cook, G., "Robotic arc welding: research in sensory feedback control", IEEE Transactions on Industrial Electronics, Vol. IE-30, No. 3, August 1983, pp. 252-268.

[39] Jones, S.B. et al., "A review of optical methods for adaptive control in arc welding", Research Report, The Welding Institute, November 1982.

[40] Vroman, A.R. & Brandt, H., "Feedback control of GTA welding using puddle width measurement", Welding Journal, September 1976, pp. 742-749.

[41] Smith, C., "Simultaneous control of weld pool size and position for precision TIG welding", Welding and fabrication in the nuclear industry, BNES, London, 1979, pp. 361-367.

[42] Rider, G., "On line measurement of weld pool surface size", Welding and fabrication in the nuclear industry, BNES, London, 1979, pp. 351-359.

[43] Rider, G., "Control of weld pool size and position for automatic and robotic welding", Proc. of 3rd Int. Conf. on Robot Vision and Sensory Controls, Massachusetts U.S.A., 1983, pp. 381-389.

[44] Arata, Y. & Inoue, K., "Automatic control of arc welding by monitoring molten pool", Transactions of JWRI, 1972 1(1), pp. 99-113.

[45] Arata, Y. & Inoue, K., "Automatic control of arc welding (report 2): optical sensing of joint configuration", Transactions of JWRI, 1973 2(1), pp. 87-101.

[46] Smati. Z., "Laser guidance system for robots", Proc. of 4th Int. Conf. on Robot Vision and Sensory Controls, 9-11 October 1984, London.

[47] Edling, G. & Porsander, T., "Adaptive control of torch position and welding parameters in robotic arc welding - examples and practical use", Proc. of 14th Int. Symp. on Industrial Robots and 7th Int. Conf. on Industrial Robot Technology, IFS Publications, 1984, pp. 359-363.

[48] Bamba, T. et al., "A visual seam tracking for arc welding robots", Proc. of 14th Int. Symp. on Industrial Robots and 7th Int. Conf. on Industrial Robot Technology, IFS Publications, 1984, pp. 365-374.

[49] "Seampilot: a three-dimensional look at the workpiece", Oldelft Seampilot Product Description, Optische Industrie De Oude Delft, P.O. Box 72, 2600 MD, Delft, Holland.

[50] Verbeck, W.J.P.A., "Arc welding process control by preview sensor", The Industrial Robot, June 1984, pp. 86-88.

[51] Oomen, G.L. & Verbeck, W.J.P.A., "A real-time optical profile sensor for robot arc welding", Proc. of 3rd Int. Conf. on Robot Vision and Sensory Feedback Controls, 1983, pp. 62-71.

[52] Appels, J.A.C., "Application and economic aspects of a 3-dimensional HeNe laser sensor", Proc. of 1st Int. Conf. on Advanced

Welding Systems, the Welding Institute, Abington, Cambridge, 1984, pp. p47-1 to p47-10.

[53] Huber, C., "Sensor-based tracking of large quadrangular weld seam paths", 14th Int. Symp. on Industrial Robots, 2-4 October 1984, Gothenburg, Sweden.

[54] Davey, P.G. et al., "Laser sensors for arc welding robots", Manufacturing in the Automotive Industry, IFS Publications, 1986, pp. 80-89.

[55] "Advanced technology gives entry into new markets", The Industrial Robot, September 1986.

[56] Beattie, R.J. et al., "The use of vision sensors in multipass welding applications", Welding Journal, November 1988, pp. 28-33.

[57] Kawahara, M., "Tracking control system using image sensor for arc welding", Automation, 1983 19(4), pp. 357-363.

[58] Bamba, T. et al., "A visual sensor for arc-welding robot", Proc. of 11th Int. Symp. on Industrial Robots, 1981, pp. 151-158.

[59] Niepold, R. & Brummer, F., "PASS - a visual sensor for seam tracking and on-line process parameter control in arc-welding applications", Proc. 14th Int. Symp. on Industrial Robots and 7th Int. Conf. on Industrial Robot Technology", IFS Publications, 1984, pp. 375-385.

[60] Agapakis, J.E. et al., "Joint tracking and adaptive robotic welding using vision sensing of the weld joint geometry", Welding Journal, November 1986, pp. 33-41.

[61] Libby, C.J., "An approach to vision controlled arc welding", Proc. Conf. on CAD/CAM Technology in Mechanical Engineering, 1982, pp. 180-185.

[62] Wittels, N. & Libby C.J., "Vision aided arc welding", AutoWeld Conference, 16-17 November 1983, Southfield, Michigan, U.S.A., pp. MS83-825 to MS83-825.

[63] Dufour M. & Begin G., "Adaptive robotic welding using a rapid image pre-processor", 3rd Int. Conf. on Robot Vision and Sensory Controls, 6-10 November 1983, Massachusetts, U.S.A., pp. 338-345.

[64] Inoue, K., "Image processing for on-line detection of welding process (report 1): simple binary image processor and its application", Transaction of JWRI, 1979 8(2), pp. 9-14.

[65] Inoue, K. et al., "Automatic control of horizontal narrow gap welding (report 2): detection and measurement by image processing", Trasactions of JWRI, 1980 9(1), pp. 31-37.

[66] LaCoe, D., "3-d vision-guided welding robot system", The Industrial Robot, March 1984, pp. 18-20.

[67] Gordon, S.S. et al, "Development of a CCTV system for welder training and monitoring of space shuttle main engine welds", Welding Journal, March 1987, pp. 47-54.

[68] Richardson, R.W. et al., "Coaxial arc weld pool viewing for process monitoring and control", Welding Journal, March 1984, pp. 43-50.

[69] Baheti, R.S., "Vision processing and control of robotic arc welding system", IEEE Conf. on Decision and Control, 1985, pp. 1022-1024.

[70] Arata, Y., "Automatic control of arc welding (report 5): application of digital picture processing technique to automatic control", Transactions of JWRI, 1976 5(1), pp. 77-85.

[71] Inoue, K., "Image processing for on-line detection of welding process (report 2): binary processing for the image of arc welding process", Transactions of JWRI, 1980 9(1), pp. 27-30.

[72] Inoue K., "Image processing for on-line detection of welding process (report 3): improvement of image quality by incorporation of spectrum of arc", Transactions of JWRI, 1981 10(1), pp. 13-18.

[73] Inoue, K. et al., "Automatic control of horizontal narrow gap welding (report 3): the total control system and the wire position control method by arc observation", Transactions of JWRI, 1980 9(2), pp. 9-14.

[74] Pandjiris, A.K. & Weinfurt, E.J., "Tending the arc", Welding Journal, September 1972, pp. 633-637.

[75] Billinger, J.G., "The use of tactile sensing for the guidance of a robotic device for welding", 1st Int. Conf. on Robot Vision and Sensory Controls, 1-3 April 1981, Stratford-upon-Avon, U.K., pp. 193-203.

[76] Presern, S. & Gyergyek L., "An intelligent tactile sensor - an on-line hierarchical object and seam analyser", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, March 1983, pp. 217-220.

[77] Presern, S. et al., "Tactile sensing system with sensory feedback control for industrial arc welding robots", 1st Int. Conf. on Robot Vision and Sensory Controls, 1-3 April 1981, Stratford-upon-Avon, U.K., pp. 205-213.

[78] Cullen, C.P., "An adaptive robotic welding system using weld-wire touch sensing", November 1988, pp. 17-21.

[79] Lott, L.A., "Ultrasonic detection of molten/solid interface of weld pool", Materials Evaluation, March 1984, pp. 337-341.

[80] Hardt, D.E. & Katz, J.M., "Ultrasonic measurement of weld penetration", Welding Journal, September 1984, pp. 273-s to 280-s.

[81] Fenn, R. & Stroud, R.R., "Development of an ultrasonically sensed penetration controller and seam tracking system for welding robots", C471/84 IMechE, 1984, pp. 105-108.

[82] Gunnarsson, K.T. & Prinz, F.B., "Ultrasonic sensors in robotic seam tracking", American Control Conference, 1984, pp. 45-47.

[83] Tan, C. & Lucas, J., "Low cost sensors for seam tracking in arc welding", 1st Int. Conf. on Computer Technology in Welding, London, 3-5 June 1986, pp. 35-42.

[84] Chin, B.A. et al., "Infrared thermography for sensing the arc welding process", Welding Journal, September 1983, pp. 227-s to 233-s.

[85] Begin, G. & Boillot, J.P., "Welding adaptive functions using infrared vision schemes", 3rd Int. Conf on Robot Vision and Sensory Controls, 6-10 November 1983, Cambridge, Massachusetts, U.S.A.

[86] Lukins, W.E. & Morris, R.A., "Infrared temperature sensing of cooling rates for arc welding control",Welding Journal, January 1982, pp. 27-33.

[87] Kenyon, D.M. & Boyce, A.J., "Effect of cathode length and diameter on tungsten arc characteristics", British Welding Journal, February 1966, pp. 103-108.

[88] The effect of electrode geometry in gas tungsten-arc welding", Welding Journal, November 1965, pp. 489-s to 496-s.

[89] Teriwal, P. & Mazumder, J., "Effect of torch angle and shielding gas flow on TIG welding - a mathematical model", Metal Construction, 1988 20 (6), pp. 275R-279R.

[90] Schultz, J.F., "TIG process with dual shield: intermediate process between TIG and plasma arc welding", Welding in the World, 1986 24(11/12), pp. 37-44.

[91] Omar, A.A. & Lundin, C.D., "Pulsed plasma - pulsed GTA arc: a study of the process variables", Welding Journal, April 1979, pp. 97-s to 105-s.

[92] Lundin, C.D. & Ruprecht, W.J., "Pulsed current plasma arc welding", Welding Journal, January 1974, pp. 11-19.

[93] Cook, G.E. & Eassa, H.E.D.E.H., "The effect of high-frequency pulsing of a welding arc", IEEE Transactions on Industry Applications, Vol. IA-21, No. 5, September/October 1985, pp. 1294-1299.

[94] Jennings, C.H. & White, A.B., "Magnetic arc blow", Welding Journal, October 1941, pp. 427-s to 436-s.

[95] Levakov, V.S. & Lyubavskii, K.V., "Influence of a longitudinal magnetic field on an electric arc using a non-consumable tungsten electrode", Welding Production, Trans. BWRA, 1965 12(10), pp. 15-20.

[96] Kovalen, I.M. & Akulov, A.I., "Stability of the welding arc in a transverse magnetic field", Welding Production, Trans. BWRA, 1963 12(10), pp. 9-14.

[97] Hicken, G.K. & Jackson, C.E., "The effect of applied magnetic fields on welding arcs", Welding Journal, November 1966, pp. 515-s to 524-s.

[98] Kovalov, I.M., "Deflection of a welding arc in a transverse magnetic field", Welding Production, Trans. BWRA, 1965 12(10), pp. 5-9.

[99] Serdyuk, G.B., "Calculation of welding arcs in transverse magnetic field", Automatic Welding, Trans. BWRA, 1960 Vol. 11, pp. 26-34.

[100] Bachelis, I.A., "Calculation of the deflection of a welding arc in a constant transverse magnetic field", Welding Production, Trans. BWRA, 1963 Vol. 7, pp. 15-18.

[101] Serdyuk, G.B., "Aspects of the magnetic control of welding arcs: rotation of a magnetically controlled welding arc between concentric electrodes", Welding Production, Trans. BWRA, 1965 12(10), pp. 1-5.

[102] Hicken, et al., "Application of magnetically controlled welding arcs", Welding Journal, April 1976, pp. 264-267.

[103] Deminskii, Yu.A. & Dyatlov, V.A., "Magnetic control during gas shielded arc welding with a consumable electrode", Automatic Welding, Trans. BWRA, 1967 Vol. 4, pp. 67-68.

[104] Hughes, R.V. & Walduck, R.P., "Electromagnetic arc path control in robot plasma welding", International Journal of Advanced Manufacturing Technology, 1985 1(1), IFS Publications Ltd., pp. 9-26

[105] Kraus, J.D., "Electromagnetics", 3rd edition, McGraw-Hill Book Company, 1984.

[106] "Personal Computer Hardware Reference Library: Technical Reference", IBM Corporation, 1984.

[107] "IBM Software for IBM Personal Computers: DOS 3.0 Technical Reference", IBM Corporation, 1984.

[108] "IBM Software for IBM Personal Computers: DOS 3.0 Manual", IBM Corporation, 1984.

[109] "Microsystem Components Handbook Vol. 1", Intel Corporation, 1984.

[110] "The TTL Data Book", Texas Instruments, 1985.

[111] "RS Components Catalogue", RS, 1987

[112] "8080/8085 Assembly Language Programming Manual", Intel Corporation, 1987.

[113] Ralston, A. & Reilly, E.D.Jr., "Encyclopaedia of computer science and engineering", 2nd edition, Van Nostrand Reinhold Company, 1983.

[114] Liu, Y.C. & Gibson, G.A.,"Microcomputer systems: the 8086/8088 family: architecture, programming, and designs", Prentice-Hall, 1984.

[115] "Impose Electronics Catalogue", Impose Electronics, 1988.

[116] "Microsystem Components Handbook Vol. 2", Intel Corporation, 1984.

[117] "Altera Data Book", Altera Corporation, January 1988.

[118] "CUPL User's Manaul", Personal Cad Systems Technology Inc., November 1986.

[119] "Farnell Components Catalogue", FEC, 1986.

[120] "Verospeed Components Catalogue", Verospeed, 1987.

[121] Kernihham, B.W. & Rithie, D.M. "The C programming language," 2nd edition, Prentice-Hall, 1988.

[122] "Lattice C Compiler for 8086/8088 Series Microprocessor", Lattice Inc., 1985.

[123] "Intel Software Development Tools - ASM86/R&L: Language Reference, Macro Assembler", Intel Corporation, 1985.

[124] "Plink86 User's Manual", Phoenix Software Associates Ltd., 1985

[125] "Intel Software Development Tools - ASM86/R&L: Utilities, O.S. Interface", Intel Corporation, 1985.

[126] "Intel Software Development Tools: iC-86 Compiler User's Guide for DOS Systems", Intel Corporation, 1985.

[127] Critchlow, A.J."Introduction to robotics", Macmillan, New York, 1985.

[128] Eskenazi, R. "Video signal input", Robotics Age, 1981 3(2), pp. 2-11, 19.

[129] Fu, K. S., Gonzalez, R.C. & Lee, C.S.G. "Robotics: control, sensing, vision and intelligence", Mc Graw-Hill Book Company, 1987.

[130] Fu, K.S. "Digital Pattern Recognition", 2nd edition, Springer-Verlag, 1980.

[131] Dawson, B.M. "Introduction to image processing algorithms", Byte, March 1987, pp. 169-186.

[132] Fu, K.S. & Mui, J.K. "A survey on image segmentation", Pattern Recognition, 1981 Vol. 13, pp. 3-16.

[133] Rosenfeld, A, & Davis, L.S. "Image segmentation and image models", Proceedings of the IEEE, 1979 67(5), pp. 764-772.

[134] Coleman, G.B. & Andrews, H.C. "Image segmentation by clustering", Proceedings of the IEEE, 1979 67(5), PP. 773-785.

[135] Cunningham, R. "Segmenting binary images", Robotics Age, 1981 3(4), pp. 4-19.

[136] Wesska, K.S. "A survey of threshold selection techniques", Computer Graphics and Image Processing, 1978, pp. 259-265.

[137] Rosenfeld, A."Image pattern recognition", Proceedings of the IEEE, 1981 69(5), pp. 596-605.

[138] Woods, R.E. & Gonzalez, R.C. "Real-time digital image enhancement", Proceedings of the IEEE, 1981 69(5), pp. 643-654.

[139] Barrow, H.G. & Tenenbaum, J.M." Computational vision", Proceedings of the IEEE, 1981 69(5), pp. 572-595.

[140] Corby, N.R. Jr. "Machine vision for robotics", IEEE Transactions on Industrial Electronics, August 1983, pp. 282-291.

[141] "X-ray real-time imaging for weld inspection" (reported by a Welding Group of Sub-Commision VA", Welding in the World, 1986 24(1/2), pp. 38 - 41.

[142] Amin, M. "Microcomputer control of synergic pulsed MIG welding", Metal Construction, 1986 18(4), pp. 216-221.

[143] Norrish, J. "The application of microprocessors in welding", Metal Construction, 1987 19(8), pp. 440-443, 445.

[144] Lucas, W. & Brightmore, A.D. "Expert systems for welding engineers", Metal Construction, 1987 19(5), pp. 254-260.

[145] Taylor, W.A. "Expert systems to generate arc welding procedures", Metal Construction, 1986 18(7), pp. 426-431.

[146] Sevensson, L.E. et al, "Computer-aided design of electrodes for manual metal arc welding", First Int. Conf. Computer Technology in Welding, London, 3-5 June 1986, pp. 113-120

APPENDICES

Appendix-A1 Components List of Pendant Controller

Unit	Component
u01	74LS393
u02	74LS74
u03	74LS28
u04	74LS74
u05	74LS08
u06	74LS28
u07	74LS138
u08	74LS138
u09	MAX232
u10	Intel 8085
u11	8256
u12	74LS373
u13	6264
u14	2764
u15	MAX232
u16	74LS28
u17	74C922
u18	keypad
u19	μPD7002
u20	DMX402

Appendix-A2 Components List of Application Computer and Drive Circuit

Unit	Component
u01	8284
u02	EP600:ADDR
u03	EP600:IOLOG
u04	EP600:LOGIC
u05	Intel 8088
u06	8259A
u07	8286
u08	8282
u09	8282
u10	8282
u11	2764
u12	M5M5256AP-70
u13	8254
u14	74ALS574
u15	ZN427
u16	74C922
u17	keypad
u18	DMX402
u19	74LS06
u20	7542
u21	MAX232
u22	8251
u23	μPD7002
u24	741
u25	741
u26	165

```
;*********************************************
;* Pendant control software for TIG welding robot
;*          by Xiao Qi Chen
;*           November 1985
;*********************************************

;PROGRAM 'START'

          proc '8085'
          tab 16

;equates of display
set_402 equ &60
write_402   equ &61
busy_flag   equ &62
read_402    equ &63
clr_disp    equ &01
home        equ &02
disp_mode   equ &06         ;disable display shift
disp_on equ &0f             ;display, cursor, flash on
function    equ &38         ;two lines
forward equ &10
backward    equ &14

;equates of data display ram address
first_org   equ &80
second_org  equ &c0
manu_home   equ &86
rig_home    equ &88
drive_home  equ &85
mode1_home  equ &85
mode2_home  equ &85
preset_home equ &86
iwave_home  equ &86
para_home   equ &87
sensor_home equ &86

write_7002  equ &20
status_7002 equ &20
data_7002   equ &21
cs_key      equ &40
interface   equ &03
enable_int56 equ &11
stop_receiv equ &15
enable_key  equ &16
enable_8256 equ &29
top_stack   equ &8200
cs_8256 equ &00
inter_addr  equ &06
transmit_buf equ &07
receive_buf equ &07
status_8256 equ &0f

 ;service memory store
          org &8400
;stores of ADC
adc_reading: ds 1
scale_data: ds 1
adc_scale:  ds 1        ;channel 0
          ds 1        ;channel 1
          ds 1        ;channel 2
```

```
        ds 1        ;channel 3

;character stores
aloc_char:  ds 5        ;axis b preset
bloc_char:  ds 5        ;c axis
cloc_char:  ds 5        ;d axis
dloc_char:  ds 5        ;e axis

Ip_char:    ds 3        ;Ip setting
Ib_char:    ds 3        ;Ib
Tp_char:    ds 4        ;Tp
Tb_char:    ds 4        ;Tb

ws_char:    ds 4        ;welding speed
wf_char:    ds 4        ;wire feeding speed
wd_char:    ds 4        ;swing width
fq_char:    ds 4        ;swing frequency

;DATA STORES: the first memory of a buffer is for control code,
;the last memory of a buffer is for terminator '$'.
drive_a_code:   ds 3    ;Stores for drives
drive_b_code:   ds 3
drive_c_code:   ds 3
drive_d_code:   ds 3
drive_e_code:   ds 3
drive_x_code:   ds 3

aloc_code:  ds 7        ;store for axes  preset
bloc_code:  ds 7
cloc_code:  ds 7
dloc_code:  ds 7

Ip_code:    ds 5
Ib_code:    ds 5
Tp_code:    ds 6
Tb_code:    ds 6

ws_code:    ds 5
wf_code:    ds 5
wd_code:    ds 5
fq_code:    ds 5

key_data_buf:   ds 1
prev_key:   ds 1
pres_flag:  ds 1
disp_ram:   ds 1
char_length: ds 1
data_length: ds 1
decim_posit: ds 1       ;position of decimal point
conv_ptr:   ds 2
mode_ptr:   ds 2
data_buf_ptr: ds 2
char_buf_ptr: ds 2
hl_temp:    ds 2
drive_loc:  ds 5

        org &1000
        db '(C)W',0     ;header
        db 'JOYSTICK CONTROLLER',0  ;title
        jmp serv_start

;Interrupt table
int_table:  org &10+int_table
        jmp drive_isr
```

```
;service vector table
manu_vector: dw test_rig
        dw i_wave
        dw weld_pa
        dw sensor
rig_vector: dw drive
        dw preset
        dw datum
        dw go
drive_vector:   dw mode1
        dw mode2
        dw mode3
        dw mode4
drive_data: dw mode1_data
        dw mode2_data
        dw mode2_data
        dw mode2_data
mode1_data: dw drive_c_code
        dw drive_d_code
        dw drive_a_code
        dw drive_b_code
mode2_data: dw drive_c_code
        dw drive_d_code
        dw drive_e_code
        dw drive_x_code
preset_char:    dw aloc_char
        dw bloc_char
        dw cloc_char
        dw dloc_char
preset_data:    dw aloc_code
        dw bloc_code
        dw cloc_code
        dw dloc_code
cur_char:   dw Ip_char
        dw Ib_char
        dw Tp_char
        dw Tb_char
cur_data:   dw Ip_code
        dw Ib_code
        dw Tp_code
        dw Tb_code
para_char:  dw ws_char
        dw wf_char
        dw wd_char
        dw fq_char
para_data:  dw ws_code
        dw wf_code
        dw wd_code
        dw fq_code
sensor_data: dw off_code
        dw vol_code
        dw vis_code
        dw vv_code
off_code: db '1$'
vol_code: db '2$'
vis_code: db '3$'
vv_code:  db '4$'

;Initial code and data for data stores
initdata: DB 'A5$B5$C5$D5$E5$X5$'
        DB 'A00000$B00000$C00000$D00000$'
        DB 'Q000$R000$S0000$T0000$'
        DB 'q000$r000$s000$t000$'
```

```
;Keypad code table
key_hex_code: db &03    ;selction key #3
        db &02          ;#2
        db &01          ;#1
        db &00          ;#0
        db &30
        db &39
        db &38
        db &37
        db &2e          ;decimal point
        db &36
        db &35
        db &34
        db &7e          ;ENT
        db &33
        db &32
        db &31

;Table of control codes
reset_code: db 'J$'
drive_code: db 'K$'
preset_code: db 'L$'
datum_code: db 'M$'
go_code:    db 'G$'
wave_code:  db 'I$'
para_code:  db 'P$'
sensor_code: db 'V$'
exit_code: db &04,'$'
enq_code: db &05,'$'


;Data diplay ram address for key input
loc_ddr: db &c2         ;Location settig
        db &cc
        db &d6
        db &e0
drive_ddr:  db &d6      ;Drive A
        db &e0          ;B
        db &c2          ;C
        db &cc          ;D
        db &d6          ;E
iwave_ddr:  db &c3      ;I_wave
        db &cd
        db &d7
        db &e1
para_ddr:   db &c3      ;Welding parameters
        db &ce
        db &d7
        db &e2
sensor_ddr: db &c3      ;Sensor
        db &d1
        db &da
        db &e2

;ADC low and up limits
;ADC reading between the two limits
;is regarded as zero drive speed, that is scale 5
adc_low: db &68         ;Low limit for channel 0
        db &8f          ;Channel 1
        db &5c          ;Channel 2
        db &96          ;Channel 3
adc_up: db &70          ;Up limit for channel 0
        db &98          ;Channel 1
```

```
        db &64          ;Channel 2
        db &9e          ;Channel 3


;*********************
;Start of main program
;*********************

        org &1200
serv_start: lxi sp,top_stack
        rst interface   ;Initialise 8256
        db enable_8256
        lxi h,reset_code ;Send reset code
        call string_86
        call init_disp  ;Initialise display

;Initialise data buffer
        lxi d,drive_a_code  ;Starting buffer RAM
        lxi h,initdata  ;Starting data ROM
        mvi c,88        ;Number of bytes
byte_copy:  mov a,m
        stax d
        inx h           ;next byte
        inx d           ;next memory
        dcr c
        jnz byte_copy

;Initialise char buffer of preset, drive locations
;and i_wave to zero
        lxi h,aloc_char
        mvi c,34        ;Number of memories
        mvi a,&30       ;Zero
        call init_value

;Initialise weld_pa buffer in the form of 00.0
        mvi c,4
        lxi h,ws_char
        call init_char
        lxi h,wf_char
        call init_char
        lxi h,wd_char
        call init_char
        lxi h,fq_char
        call init_char

        lxi h,key_read    ;Load vector point after
        rst interface     ;interrupt,
        db enable_key     ;enable key reading
manu_start: call manu_disp ;Display manual.
manu_wait: call key_action ;Read key
        lda key_data_buf
        cpi &04           ;If not selection key
        jnc manu_wait     ;wait
        lxi h, manu_start ;Get ret address
        push h            ;Store in the stack
        lxi h,manu_vector
        call vect_addr  ;Calculate jmp address
        pchl            ;Put the addr to PC

;This routine initialise character buffers,
;char length in C
init_char:  push b      ;save C for next setting
        lxi d,char_zero
char_lp: ldax d
```

```
        mov m,a
        dcr c
        jz char_ret
        inx h
        inx d
        jmp char_lp
char_ret: pop b
        ret
char_zero:  db '00.0'

;The following routine sets memory locations to
;the value held by A, with C holding number of locations
init_value: mov m,a
        inx h
        dcr c
        jnz init_value
        ret

vect_addr:  push d      ;Save registers
        push psw
        add a           ;Double A
        mov e,a
        mvi d, 0        ;Calculate vector ptr
        dad d           ;with the offset
        mov e, m        ;Get call adrress
        inx h
        mov d,m
        xchg            ;Put in HL
        pop psw         ;Get registers
        pop d
        ret

;This routine loops round untill a key is pressed.
key_action: mvi a,&ff
        sta key_data_buf
key_wait:   lda key_data_buf
        cpi &ff
        jz key_wait
        ret

;The following routine is exceuted after interrupt RST7.5
key_read: push h        ;save registers
        push psw
        push d
        in cs_key       ;read key
        ani &0f         ;get rid of 4 MS bits
        lxi h,key_hex_code  ;convert to hex code
        mvi d,0
        mov e,a
        dad d
        mov a,m
        sta key_data_buf
        ei              ;reenable interrupts
        pop d           ;get registers
        pop psw
        pop h
        ret

trans_8256: push psw    ;Save data to be sent
trans_wait: in status_8256
        ani &20         ;check TBE
        jz trans_wait   ;wait untill TB empty
        pop psw
        out transmit_buf
```

```
        ret

receiv_8256:    in status_8256
        ani &40         ;check RBF
        jz receiv_8256  ;wait if RB empty
        in receive_buf
        ret

receiv_ack: call receiv_8256
        cpi &06
        jnz receiv_ack
        ret

;This routine sends a string to 8086. Data location in HL.
;It requires ACK after sending terminator '$'.
string_86:  mov a,m     ;get byte
        call trans_8256 ;send it
        cpi '$'
        jz string_ok
        inx h           ;next byte
        jmp string_86
string_ok: call receiv_ack
        ret

;Signal to 8086 that a fault data string is received
error_sig:  lxi h,enq_code
        call string_86
        ret

        extend 'displ'


;**********************
;Subroutines of display
;**********************

;PROGRAM 'DISPL'

init_disp: call fg_ck
        mvi a,function
        out set_402
        call fg_ck
        mvi a, disp_mode
        out set_402
        call fg_ck
        mvi a, disp_on
        out set_402
clear_disp: call fg_ck
        mvi a,clr_disp
        out set_402
        ret

fg_ck:  push psw        ;Save registers
        push b          ;
        mvi a, 10       ;
        mov b,a
ck_lp:  in busy_flag
        ani &80         ;check flag
        jz ck_ret
        dcr b
        jnz ck_lp
ck_ret: pop b
        pop psw
        ret
```

```
;To display windows call the following subroutine, with HL
;holding the starting location of char buf.
;To display a string, call disp_ddram
;with A holding the starting display location,
;B holding the numbe of charaters to be displayed
;and HL holding the character memory location.

disp_window: mvi b,80   ;80 characters
        call fg_ck      ;
        mvi a, first_org ;starting display location
        out set_402
disp_loop:  call fg_ck  ;Get char
        mov a,m         ;display it
        out write_402
        inx h           ;next char
        dcr b           ;decrement counter
        jnz disp_loop   ;Loop, if not finished.
        ret


manu_disp: lxi h, manu_messg ;Display manual window
        call disp_window
        call fg_ck
        mvi a, manu_home  ;return cursor to
        out set_402     ;home position
        ret
manu_messg: db 'MANUAL    '
        db '            '
        db '            '
        db '            '
        db 'TEST_RIG  '
        db 'I_WAVE    '
        db 'WELD_PA   '
        db 'SENSOR    '

rig_disp: lxi h, rig_messg
        call disp_window
        call fg_ck
        mvi a, rig_home
        out set_402
        ret
rig_messg:  db 'TEST_RIG  '
        db '            '
        db '            '
        db '            '
        db 'DRIVE     '
        db 'PRESET    '
        db 'DATUM     '
        db 'GO        '

drive_disp: lxi h, drive_messg
        call disp_window
        call fg_ck
        mvi a, drive_home
        out set_402
        ret
drive_messg: db 'DRIVE    '
        db '            '
        db '            '
        db '            '
        db 'MODE1     '
        db 'MODE2     '
        db '            '
```

```
        db '          '

 mode1_disp: lxi h, mode1_messg
        call disp_window
        call fg_ck
        mvi a, mode1_home
        out set_402
        ret
 mode1_messg: db 'MODE1      '
        db '(step)    '
        db '          '
        db '          '
        db 'C=        '
        db 'D=        '
        db 'A=        '
        db 'B=        '

 mode2_disp: lxi h, mode2_messg
        call disp_window
        call fg_ck
        mvi a, mode2_home
        out set_402
        ret
 mode2_messg: db 'MODE2      '
        db '(step)    '
        db '          '
        db '          '
        db 'C=        '
        db 'D=        '
        db 'E=        '
        db '          '

 preset_disp: lxi h, preset_messg
        call disp_window
        call fg_ck
        mvi a,preset_home ;Ret cursor to home
        out set_402       ;position
        ret
preset_messg: db 'PRESET    '
        db '(step)    '
        db '          '
        db '          '
        db 'A=        '
        db 'B=        '
        db 'C=        '
        db 'D=        '

;To display strings of characters in buffer,
;load C with number of strings,
;B with nomber of characters in a string,
;DE with starting address of ddram,
;and HL with starting location of char buf

string_disp: call fg_ck
        ldax d          ;Load ddram
        out set_402     ;Move cursor
        lda char_length ;Get char length
        mov b,a         ;an put in b
char_displ: call fg_ck
        mov a,m         ;Get char
        out write_402   ;and display it
        inx h           ;Next char address
        dcr b           ;Decrement char counter
        jnz char_displ  ;More char
```

```
        inx d           ;Next ddram
        dcr c           ;Decrment string counter
        jnz string_disp ;More string
        ret

iwave_disp: lxi h, wave_messg
        call disp_window
        lxi d,iwave_ddr ;Display previous curreent
        lxi h,Ip_char   ;setting
        mvi c,2         ;Ip and Ib
        mvi a,3
        sta char_length
        call string_disp
        mvi c,2         ;Tp and Tb
        mvi a,4
        sta char_length
        call string_disp
        call fg_ck
        mvi a, iwave_home
        out set_402
        ret
wave_messg: db 'I_WAVE    '
        db '(amp,amp,m'
        db 's,ms)       '
        db '            '
        db 'Ip=         '
        db 'Ib=         '
        db 'Tp=         '
        db 'Tb=         '

para_disp:  lxi h, para_messg
        call disp_window
        lxi d, para_ddr ;Display previous
        lxi h,ws_char   ;welding parameters setting
        mvi c,4
        mvi a,4
        sta char_length
        call string_disp
        call fg_ck
        mvi a, para_home
        out set_402
        ret
para_messg: db 'WELD_PA   '
        db '(mm/s,m/mi'
        db 'n,mm,Hz)    '
        db '            '
        db 'WS=         '
        db 'WFS=        '
        db 'WD=         '
        db 'FRQ=        '

sensor_disp: lxi h, sensor_messg
        call disp_window
        call fg_ck
        mvi a, sensor_home
        out set_402
        ret
sensor_messg: db 'SENSOR    '
        db '            '
        db '            '
        db '            '
        db 'OFF         '
        db 'VOLTAGE     '
        db 'VISION      '
```

```
        db 'BOTH    '

        extend 'drive'


;****************************************
;Subroutine "DRIVE": drive test rig manually
;using joysticks
;****************************************

;PROGRAM 'DRIVE'

test_rig: call rig_disp     ;Display test_rig window
rig_wait: call key_action   ;Read key
        lda key_data_buf
        cpi &7e     ;If ENT
        rz                  ;back to manu_start
        cpi &04             ;Wait for
        jnc rig_wait        ;selection key
        lxi h,test_rig      ;Get ret addr to test_rig
        push h      ;Save in stack
        lxi h,rig_vector    ;Get call address
        call vect_addr
        pchl        ;and jmp to it

go: call clear_disp
        lxi h,go_code       ;Send code
        call string_86      ;to 8086
        ret

datum:  call clear_disp
        lxi h,datum_code    ;Send code to 8086
        call string_86
        ret

;In DRIVE, joysticks control axis movement directly.
;In mode1, left:C,D  right:A,B.
;In mode2, left:C,D  right:E,_

drive:  call clear_disp
        lxi h,drive_code    ;Send code
        call string_86
        mvi a,5
        sta char_length     ;Char length is 5
        sta adc_scale       ;Scale 5 for channel 0
        sta adc_scale+1     ;channel 1
        sta adc_scale+2     ;channel 2
        sta adc_scale+3     ;channel 3
        lxi h, inta_8256    ;Enable receiver int
        rst interface       ;of 8256
        db enable_int56
drive_wait: call drive_disp ;Display drive window
        call key_action     ;Read key
        lda key_data_buf
        cpi &7e             ;If ENT,
        jz drive_exit       ;exit from drive
        cpi &04             ;If not selection key
        jnc drive_wait      ;wait
        lxi h,drive_wait    ;Get mode ret address
        push h              ;Save in stack
        lxi h,drive_data    ;Get mode data store
        call vect_addr
        shld mode_ptr
        lxi h,drive_vector  ;Calculate
```

```
        call vect_addr      ;call address
        pchl                ;and jmup to it

drive_exit: lxi h,exit_code ;Send exit code
        call string_86
        rst interface       ;Disable RST6.5
        db stop_receiv
        ret                 ;Return to test_rig

inta_8256: push psw         ;Save status
        push b              ;and registers
        push d
        push h
        in inter_addr       ;Read interrupt address
        mov l,a             ;put it into HL
        xra a
        mov h,a
        lxi b,int_table     ;load BC with iner table
        dad b               ;added to inter addr
        pchl                ;jmp to the inter table

mode1: call mode1_disp      ;Display mode1 window
        jmp drive_lp
mode2:  call mode2_disp
drive_lp: lda key_data_buf  ;Read key buffer
        cpi &7e             ;If ENT has been pressed
        jz mode_exit        ;exit from mode1
        call start_7002
        jmp drive_lp

mode3:  ret                 ;Not used

mode4:  ret                 ;Not used

drive_isr: xra a            ;Clear carry
        in receive_buf      ;Get char
        cpi 'A'             ;If A to E
        jc non_A_E
        stc
        cpi 'E'
        jnc non_A_E
        sui &41
        mov c,a             ;get ddram
        mvi b,0
        lxi h,drive_ddr
        dad b
        mov a,m
        sta disp_ram
        lxi h,drive_loc ;set char_buf_ptr
        shld char_buf_ptr
isr_ret: pop h
        pop d
        pop b
        pop psw
        ei                  ;Reenable RST6.5
        ret

non_A_E: cpi '$'
        jnz non_term        ;If '$'
        call fg_ck
        lda disp_ram        ;display
        out set_402
        mvi b,5
        lhld char_buf_ptr
```

```
        dcx h           ;point to the last char
dr_disp: call fg_ck
        mov a,m
        out write_402
        dcx h
        dcr b
        jnz dr_disp
        call fg_ck
        mvi a,drive_home
        out set_402
        jmp isr_ret

non_term: lhld char_buf_ptr
        mov m,a         ;Store into buffer
        inx h
        shld char_buf_ptr ;increment buf ptr
        jmp isr_ret

start_7002: mvi b, 0    ;B holds channel number
        mvi c, 4        ;Counter, 4 channels
next_chann: mov a,b     ;Initialise the channel
        out write_7002  ;pointed by B
        lhld mode_ptr   ;Get data string buf
        call vect_addr
        shld data_buf_ptr
        mov e,b         ;Calculate adc_scale ptr
        mvi d,0
        lxi h,adc_scale
        dad d           ;
        shld hl_temp    ;store in hl_temp
wait_7002: in status_7002   ;Polling adc status
        ani &80             ;If data not availble
        jnz wait_7002       ;wait.
        in data_7002        ;Read data
        sta adc_reading     ;and store in the buffer.
        call data_scaling   ;Scale adc reading.
        call data_diff      ;Adjust present data
        inr b               ;Next channel
        dcr c               ;Decrement counter
        jnz next_chann
        ret

data_scaling: push b        ;Save regiaters
        mvi c,5             ;Scaling reference is zero
        lda adc_reading     ;Get adc reading
        call low_limit      ;Get low limit of zero
        cmp m               ;If adc reading is below
        jc back_drive       ;the limit, decremnt by 4
        call up_limit       ;Get up limit of zero
        cmp m
        jz save_adc     ;If adc reading is
        jc save_adc     ;above the limit
        mov a, m        ;Increment scale
forw_drive: inr c
        adi 4           ;Increment by 4
        cmp d           ;Compare with ADC reading
        jc forw_drive
        mov a,c         ;Limit the maximun right
        cpi 9           ;speed scale to 9
        jz save_adc
        jc save_adc
        mvi c,9
```

```
        jmp save_adc
back_drive: mov a, m
back_dr_lp: dcr c
        jz min_scale    ;Limit the max. reverse
        sbi 4           ;Decrement by 4
        cmp d
        jz save_adc
        jnc back_dr_lp
        jmp save_adc
min_scale:  inr c       ;Limit the min scale to 1
save_adc:   mov a,c
        sta scale_data
        pop b           ;Pop registers
        ret             ;Ret with scaled data

low_limit:  lxi h,adc_low
        jmp limit_locat
up_limit:   lxi h,adc_up
limit_locat:    mov e,b
        mvi d,0
        dad d
        mov d,a         ;Put ADC reading into d
        ret

data_diff:  lda scale_data ;Compare scaled data
        lhld hl_temp
        cmp m           ;with current data
        jz no_diff
        jc dcr_drive
        inr m
        call drive_conv
        jmp data_diff
dcr_drive:  dcr m
        call drive_conv
        jmp data_diff
no_diff:    ret

;This routine converts speed scale (from 0 to A) into
;ASCII code, and send to 8086
drive_conv: mov a,m     ;Get data
        adi &30         ;Convert to ASCII code
        lhld data_buf_ptr
        inx h           ;Skip over code byte
        mov m,a         ;Save
        dcx h           ;Starting string
        call string_86
        ret

mode_exit:  mvi c, 4    ;Counter
        mvi b,0
        lxi d,adc_scale ;adc scale loc in DE
exit_loop:  lxi h,mode_ptr
        mov a,b
        call vect_addr
        shld data_buf_ptr
        ldax d
        cpi &05         ;If zero speed
        jz next_exit    ;exit
        jc inr_to_exit
        dcr a           ;Decrement if > 5
        jmp vary_exit
inr_to_exit: inr a      ;Increment if < 5
vary_exit: call drive_conv
next_exit: inx d        ;Next data location
```

```
        inr b           ;Next string location
        dcr c           ;Check if all 4 axes
        jnz exit_loop
        ret

        extend 'locat'


;***********************************************
;Subroutine "PRESET": presets axis locations.
;Presetting range is from 0 to 660.0 .
;A decimal point is needed among five key characters.
;***********************************************

;PROGRAM 'LOCAT'

preset: call clear_disp
        lxi h,preset_code
        call string_86
        call preset_disp    ;Display preset window
        mvi a,0
        sta pres_flag       ;Clear on/off flags
preset_wait: call key_action ;Read key
        lda key_data_buf
        cpi &7e
        jz preset_exit      ;If ENT, exit
        cpi &04             ;Wait for
        jnc preset_wait      ;selection key
        sta prev_key         ;save the key value
        lxi h, loc_ddr      ;Calculate
        mov e,a             ;ddram
        mvi d,0
        dad d
        mov a,m              ;and store it
        sta disp_ram
        lda prev_key
        lxi h, preset_char  ;Calculate char buff ptr
        call vect_addr
        shld char_buf_ptr
        lxi h, preset_data  ;Starting location of
        call vect_addr      ;string buf
        shld data_buf_ptr
        mvi a,5
        sta char_length     ;Char length is 5
        sta data_length     ;Data length is 5 byte
        call tog_reg
        mov b,a
        lda pres_flag
        xra b              ;Toggle preset flag
        sta pres_flag       ;and save
        call tog_flag
        jnc off_pres
        call loc_string
on_pres: call key_action
        lda key_data_buf
        cpi &7e             ;If ENT,
        jz on_old           ;on with old setting
        lxi h,prev_key
        cmp m               ;If the same ctr key,
        jnz on_pres
loc_input:  call erase_data ;ready for renew setting
        call key_number
        call loc_range
        jnc loc_input
```

```
        call data_conv
        call fg_ck
        mvi a,preset_home
        out set_402
        jmp preset_wait

off_pres: call erase_data
on_old: call fg_ck
        mvi a,preset_home
        out set_402
        jmp preset_wait

loc_string: lxi d,disp_ram
        lhld char_buf_ptr
        mvi c,1 ;display only one string
        call string_disp
        ret

preset_exit: mvi c,4        ;Counter
        mvi b,0
loc_exit: mov a,b
        sta prev_key        ;axis number
        call tog_flag
        jc send_on
        jmp send_off
send_on: lxi h,preset_data  ;Send axis location
        mov a,b
        call vect_addr
        call string_86      ;to 8086
send_off: inr b
        dcr c
        jnz loc_exit
        lxi h,exit_code     ;Send exit code
        call string_86
        ret

tog_reg: lda prev_key
        mov c,a
        inr c
        xra a               ;Clear A and CY
        stc                 ;set carry flag
more_ral: ral
        dcr c
        jnz more_ral
not_ral: ret

tog_flag: push b            ;Save registers
        lda prev_key
        mov c,a
        inr c
        lda pres_flag
flag_rrc: rrc               ;rotate flag to carry bit
        dcr c
        jnz flag_rrc
        pop b               ;Get back registers
        ret                 ;Ret with the flag in carry

key_number: lda data_length ;number of chars
        mov c, a            ;in C
        lhld char_buf_ptr   ;Char buf location
        call fg_ck
        lda disp_ram        ;Disp location
        out set_402
more_number: call key_action
```

```
        lda key_data_buf
        cpi &7e             ;exclude ENT
        jz more_number
        cpi &2e             ;and decimal point
        jz more_number
        cpi &04             ;and selection keys
        jc more_number
        call fg_ck
        out write_402   ;Display and store
        mov m,a
        inx h
        dcr c
        jnz more_number
        ret

;Check if the location presetting is < 65500.
;Return with carry if valid, or with no carry.
loc_range:  lhld char_buf_ptr ;Compare the leftmost digit
        mov a, m        ;with 6
        cpi '6'
        jz loc_sec
        jc valid_loc
        jmp bad_loc
loc_sec: inx h          ;Compare the second one
        mov a, m        ;from left with 6
        cpi '5'
        jz loc_last
        jc valid_loc
        jmp bad_loc
loc_last: inx h         ;Check if the last two
        mov a,m         ;are zero
        cpi '5'
        jc valid_loc
bad_loc: ora a          ;Clear carry flag
valid_loc: ret

erase_data: call erase_del
        lda char_length ;Get char length
        mov c, a
        call fg_ck
        lda disp_ram    ;Display DDRAM
        out set_402
        push psw        ;save a
        mvi a, &20      ;Erase with blank
erase_loop: call fg_ck
        out write_402
        dcr c
        jnz erase_loop
        pop psw         ;Return cursor
        call fg_ck
        out set_402     ;to starting position
        ret

erase_del:  mvi c,&ff
del_loop:   inr a
        dcr a
        dcr c
        jnz del_loop
        ret

;The following conversion copies char in the char buf
;(from low to high addr) to the data buf (from high to low addr)
data_conv:  lda data_length
        mov c,a         ;Counter
```

```
        mov e,a         ;Point HL to the last
        mvi d,0         ;data byte
        lhld data_buf_ptr
        dad d           ;
        push h
        lhld char_buf_ptr ;Point DE to the first
        xchg            ;char byte
        pop h
conv_loop:  ldax d
        cpi &2e
        jz neg_point
        mov m,a
        dcx h           ;Next data byte
        dcr c           ;Decrement counter
neg_point:  inx d       ;Next char
        mov a,c
        cpi 0
        jnz conv_loop
        ret

        extend 'paras'


;*****************************************
;Subroutine "I_WAVE": preset current parameters
;*****************************************

;PROGRAM 'PARAS'

i_wave: call clear_disp
        lxi h,wave_code
        call string_86
        call iwave_disp     ;Display iwave window
iwave_wait: call key_action ;Read key
        lda key_data_buf
        cpi &7e             ;If ENT,
        jz iwave_exit       ;exit
        cpi &04             ;If not selection key
        jnc iwave_wait      ;wait
        lxi h, iwave_ddr
        mov e,a
        mvi d,0
        dad d
        push psw            ;Save A
        mov a,m             ;Get ddram
        sta disp_ram        ;and store it
        pop psw             ;Get key data
        lxi h,cur_char      ;Get char buf
        call vect_addr
        shld char_buf_ptr
        lxi h,cur_data      ;Get data buf
        call vect_addr
        shld data_buf_ptr
        cpi &02
        jc i_length
        mvi a, 4            ;Time char length
        sta char_length
        sta data_length
        jmp iwave_input
i_length:   mvi a,3         ;Curr char length
        sta char_length
        sta data_length     ;Data length
iwave_input: call key_number ;Key in number
        call fg_ck
```

```
        mvi a,iwave_home
        out set_402
        call data_conv
        jmp iwave_wait

iwave_exit: lxi h,Ip_code   ;Send data to 8086
        call string_86      ;Ret manu_start
        lxi h,Ib_code
        call string_86
        lxi h,Tp_code
        call string_86
        lxi h,Tb_code
        call string_86
        ret


;*********************************************
;Subroutine "weld_pa": preset welding parameters
;*********************************************

weld_pa: call clear_disp
        lxi h,para_code
        call string_86
        call para_disp      ;Display window
para_wait: call key_action  ;Read key
        lda key_data_buf
        cpi &7e             ;If ENT
        jz para_exit        ;exit
        cpi &04             ;Waiting for
        jnc para_wait        ;selection key
        lxi h,para_ddr      ;Cursor position
        mov e,a
        mvi d,0
        dad d
        push psw
        mov a,m
        sta disp_ram
        pop psw
        lxi h,para_char     ;Get char buf
        call vect_addr
        shld char_buf_ptr
        lxi h,para_data     ;Get data buf
        call vect_addr
        shld data_buf_ptr
        mvi a,4             ;Char length
        sta char_length
        mvi a,3             ;data length
        sta data_length

;Input welding parameters in the form of **.*,
;where the point is fixed.
para_input: lhld char_buf_ptr
        mvi c,2
        call fg_ck
        lda disp_ram
        out set_402
        call more_number
        call fg_ck
        mvi a, &2e          ;Display the fixed point
        out write_402
        mov m,a
        inx h               ;next char
        mvi c,1
        call more_number
```

```
        call fg_ck
        mvi a,para_home
        out set_402
        call data_conv
        jmp para_wait

para_exit:  lxi h,ws_code   ;Send data
        call string_86      ;to 8086
        lxi h,wf_code
        call string_86
        lxi h,wd_code
        call string_86
        lxi h,fq_code
        call string_86
        ret


;*********************************
;Subroutine "SENSOR": select sensors
;*********************************

sensor: call clear_disp
        lxi h,sensor_code
        call string_86
        call sensor_disp    ;Display sensor window
sensor_mode: call key_action ;Read key
        lda key_data_buf
        cpi &7e             ;If ENT
        jz sensor_exit      ;exit
        cpi &04             ;Wait for
        jnc sensor_mode     ;selection key
        lxi h,sensor_ddr    ;Cursor position
        mov e,a
        mvi d,0
        dad d
        call fg_ck
        mov a,m
        out set_402
        lxi h,sensor_data   ;Get data string
        mov a,e
        call vect_addr
        mov a,m
        call string_86      ;and send
        jmp sensor_mode
sensor_exit: lxi h,exit_code
        call string_86
        ret
        end
```

```
/*****************************************
* File transfer program "TRANS.C" for transferring
* a file from IBM PC to application computer
*          by Mr. Xiao Qi Chen
*             July 19888
*****************************************/

#include "stdio.h"
#include "string.h"

#define TRUE    1
#define FALSE   0
#define BUFSIZE 128
#define ACK     6
#define CR      0X0D
#define PERIOD  10000
#define PIC0 0X021          /* 8259 IMR */
#define CRBR 0X03F8          /* Rx/Tx/DL_lsb */
#define CIER 0X03F9          /* DL_msb/IER   */
#define CIIR 0X03FA
#define CLCR 0X03FB
#define CMCR 0X03FC
#define CLSR 0X03FD
#define CMSR 0X03FE
#define NOC 0X0FF

int key_x = 128, key_y = 128, key, box_mode = 0;

char buf[BUFSIZE];

FILE *fp;
char *mode = "r";


main()
{
   char *fn, *p, *bp;
   char file[40];
   char c;

   prints("\r\nenter name  of hexfile: $");
   p = gets(buf);
   fn = file;
   if (p != NULL)
   {
      while(*fn++ =  *p++);
      if(fp  = fopen(file,mode))
      {
         startcoms();
         while(p = fgets(buf,BUFSIZE,fp))
         {
            bp = buf;
            reswrp();
            while((c = *bp++))
            {
               printc(c);
               wrtbuf0(c);
            }
            printc(CR);
            wrtbuf0(CR);
            trnbuf0();
            waitack();
```

```
            }
            stopcoms();
            if(fclose(fp))
                prints("\r\nfile close error$");
        }
        else
            prints("\r\nfile    open error$");
    }
    else
        prints("\r\nbad filename$");
}

waitack()
{
    char c;

    while((c =  rdcom1()) != ACK)
    {
        printc(c);
        if(c == '&')
            trnbuf0();
    }
}

readcom1()
{
    while((io_read(CLSR,'w') &  0X01) == 0);
    io_read(CRBR,'b');
}

delay()
{
    int i;
    for(i = 0;  i < 2000; i++);
}

startcoms()
{
    seti4();
    io_write(CLCR,0X083,'b');
    io_write(CRBR,0X0C,'b');     /* DL_lsb */
    io_write(CIER,0,'b');        /* DL_msb: BR = 9600 */
    io_write(CLCR,0X03,'b');     /* 8_bit chr,  1 stop bit, no parity */
    io_write(CMCR,0X03,'b');     /* force DTR & RTS active  */
    io_read(CMSR,'b');           /* clear all pending interrupts */
    io_read(CRBR,'b');
    io_read(CLSR,'b');
    io_read(CIIR,'b');
    io_write(CIER,0X01,'b');     /* disable all interrupts */
    di_int(16,PIC0);             /* enable irq4 */
}

stopcoms()
{
    di_int(16,PIC0);
    io_write(CIER,0,'b');        /* disable all comm ints */
    io_write(CMCR,0X03,'b');     /* enable  DTR & RTS */
    reseti4();
}
```

```
;************************************************
; IBM PC assembly routines "PCLIB.A86" for small model
; interface from Intel C to MS-DOS
;            by Mr. Xiao Qi Chen
;               May 1988
;************************************************

    NAME    PCLIB

    PGROUP  GROUP   CODE

    ASSUME  CS:PGROUP

    true    equ 0ffh
    false   equ 0h

    com1rx  equ 03f8h
    com1tx  equ 03f8h
    com1ier equ 03f9h
    com1iir equ 03fah
    com1lsr equ 03fdh
    com1msr equ 03feh
    com1int equ 0ch
    eoi     equ 020h
    pic0    equ 020h
    pic0imr equ 021h


CODE    SEGMENT PUBLIC  'CODE'

    PUBLIC  io_read, io_write
    PUBLIC  printc, prints
    PUBLIC  trans0, rdbuf0, wrtbuf0, sndtb0
    PUBLIC  reswrp, trnbuf0

;************************************
; io_read() byte or word
;************************************

io_read PROC NEAR
        push dx
        push bp
        mov bp,sp
        mov dx,[bp + 6]       ;dx = port address
        mov ax,[bp + 8]       ;al = byte or word
        cmp al,'w'
        jz i1
        cmp al,'W'            ;if 'w' or 'W' not specified
        jz i1
        in al,dx              ;perform 8-bit read
        cbw
        jmp i2                ;else
i1:     in ax,dx              ;perform 16-bit read
i2:     pop bp
        pop dx
        ret
io_read ENDP

;************************************************
;io_write(port,value,b|w) write byte or word to port
;************************************************
```

```
io_write    PROC NEAR
        push dx
        push bp
        mov bp,sp
        mov dx,[bp + 6]
        mov ax,[bp + 10]
        cmp al,'w'
        jz i3
        cmp al,'W'
        jz  i3
        mov ax,[bp + 8]
        out dx,al
        jmp i4
i3:     mov ax,[bp + 8]
        out dx,ax
        jmp i4
i4:     pop bp
        pop dx
        ret
io_write    ENDP


;*********************************************
; dos_std_out(ch byte)  renamed printc_ for mig_man
;*********************************************

printc      PROC    NEAR
    push    bp
    mov bp, sp
    mov dl, [bp + 4]  ; ch
    mov ah, 02h      ; Function: Display Output
    int 21h          ; DOS call
    pop bp
    ret
printc      ENDP


;*****************************************************
; dos_std_string_out(str addr)  renamed prints_ for mig_man
;   st$ based str (*) byte
;*****************************************************

prints  PROC    NEAR
    push    bp
    mov bp, sp
    mov dx, [bp + 4]  ;str
    mov ah, 09h      ;Function: Print String
    int 21h          ;DOS call
    pop bp
    ret
prints  ENDP

;*************************
;rdbuf0()   reads buf0
;*************************

rdbuf0  PROC NEAR
        push bx
        push bp
        mov bp, sp
        mov bx, bufrp           ;fetch read pointer
        cmp bx, bufwp           ;compare with write pointer
        jne fet                 ;if unequal continue
        mov al,0ffh             ;else return with
        jmp mtbuf               ;al = ff
```

```
fet:    mov al, cs:[bx]         ;read character
        cmp bx, offset bufend   ;check for end of buffer
        jl  bsp1                ;no..
        mov bx, offset buffer   ;end of buffer reset pointer
        jmp rdb0ex
bsp1:   inc bx                  ;not end of buffer, inc pointer
rdb0ex: mov bufrp, bx           ;and store
mtbuf:  pop bp
        pop bx
        ret
rdbuf0  ENDP


;*********************************
;wrtbuf0(c) transmitter buffer write
;*********************************

        tbuffer db  128 dup(?)  ;reserve transmit buffer
        tbufend db  0
        tbufwp  dw  tbuffer
        tbufrp  dw  tbuffer


wrtbuf0 PROC NEAR
        push bx
        push bp
        mov bp,sp
        mov bx,tbufwp
        mov al,[bp + 6]
        mov cs:[bx],al
        cmp bx,offset tbufend
        jl  tbsp
        mov bx, offset tbuffer
        jmp wbex0
tbsp:   inc bx
wbex0:  mov tbufwp,bx
        pop bp
        pop bx
        ret
wrtbuf0 ENDP

;*********************************
;sndtb0()   sends contents of tbuffer
;*********************************

        trans0  dw  false

sndtb0  PROC NEAR
        push bx
        push dx
        push bp
        mov bp,sp
        mov bx,tbufrp       ;fetch tbuf read pointer
        cmp bx,tbufwp       ;compare with write pointer
        jne send
        mov ax,false        ;if buffer empty return zero
        mov trans0,ax       ;reset transmitting flag
        jmp sndrt
send:   mov dx,com1lsr
thrs:   in al,dx            ;read status register
        test al,020h        ;test thre bit
        jz thrs
        mov dx,com1tx       ;transmitter ready, get thr address
        mov al,cs:[bx]      ;read buffer
```

```
        out dx,al               ;output char
        cmp bx,offset tbufend   ;end of buffer?
        jl tbsp1                ;no, increment pointer
        mov bx,offset tbuffer   ;yes, reset pointer
        jmp sndbe0
tbsp1:  inc bx                  ;increment pointer
sndbe0: mov tbufrp,bx           ;store pointer
        mov ax,true             ;non null return
        mov trans0,ax           ;set transmitting flag
sndrt:  pop bp
        pop dx
        pop bx
        ret
sndtb0  ENDP

;*******************************
;initialise buffer  write pointer
;*******************************

reswrp  PROC NEAR
        push bx
        mov bx, offset tbuffer
        mov tbufwp, bx
        pop bx
        ret
reswrp  ENDP

;*******************************
;transmit buf0 ended with tbufwp
;*******************************

trnbuf0 PROC NEAR
        push bx
        push dx
        mov bx, offset tbuffer
trnlp:  cmp bx, tbufwp
        jz trnret               ;end of line ?
        mov dx, com1msr
trnwt0: in al, dx
        test al, 010h           ;test CTS
        jz trnwt0
        mov dx, com1lsr
trnwt1: in al, dx
        test al, 020h           ;test TxRdy
        jz trnwt1
        mov dx, com1tx
        mov al, cs:[bx]
        out dx, al              ;output char
        inc bx                  ;next char
        jmp trnlp
trnret: pop dx
        pop bx
        ret
trnbuf0 ENDP

CODE    ENDS
        END
```

```
;***********************************************
; Booting program "BOOT.A86" for the 8088 application
; computer
;            by Mr. Xiao Qi Chen
;               July 1988
;***********************************************

$debug
$title('BOOT')

name start
;boot programme in ROM

        EXTRN main_:FAR

assume cs:code, ss:stack, es:seg main_

code segment para public 'code'

        PUBLIC boot
        PUBLIC dpchr
        PUBLIC fetchr
        PUBLIC prtchr
        PUBLIC time_
        PUBLIC tbase_
        PUBLIC mrp_,mwp_
        PUBLIC hexbuf_
        PUBLIC eaf_,saf_
        PUBLIC usba_,cs_,ip_

        PUBLIC kcode_
        PUBLIC f_amp_, f_bias_
        PUBLIC t_wv_, t_dw_
        PUBLIC fa_chr_, fb_chr_
        PUBLIC tw_chr_, td_chr_
        PUBLIC f_base_

time_   equ 00000400h
mwp_    equ 00000410h
mrp_    equ 00000414h
hexbuf_ equ 00000418h
eaf_    equ 000004a0h
saf_    equ 000004a2h
usba_   equ 000004a4h
cs_     equ 000004a6h
ip_     equ 000004a8h

kcode_  equ 000004aah
f_amp_  equ 000004ach
f_bias_ equ 000004aeh
t_wv_   equ 000004b0h
t_dw_   equ 000004b2h
fa_chr_ equ 000004b4h
fb_chr_ equ 000004bch
tw_chr_ equ 000004c4h
td_chr_ equ 000004cch
tbase_  equ 000004d4h
f_base_ equ 000004dch

boot    PROC
        cli             ;disable interrupts
        mov bx,0000h    ;set stack
```

```
        mov ss,bx
        mov bx,8000h
        mov sp,bx
        mov ax, seg main_
        mov es,ax
        call init59     ;initialize devices: PIC
        call init54     ;PIT
        call st_cm0      ;Channel 0
        call st_cm1      ;Channel 1
        call initdp     ;
        call settim      ;set time to zero
        call set0       ;set interrupt vector 0
        call set59i     ;8259 interrupt vector
        sti

call    main_

here:   call star
        mov cx,0fffh
here_1: loopnz here_1
        mov dx,6002h    ;8251 status
        in  al,dx
        and al,02h      ;test RBF
        jz  here
        call chr
        jmp here
boot    ENDP

star:   mov dx, 6002h
next:   in al, dx
        and al, 05h     ;test TBF
        jz next
        dec dx
        dec dx
        mov al, 2ah
        out dx, al
        ret

chr:    call fetchr
        cmp  al,54h     ;test for T
        jnz notime
        call titod
        jmp here
notime: cmp al,43h      ;test for C
        jnz noc
        call main_      ;Call loading module
        jmp here
noc:    call prtchr
        mov dx,6002h
nochr:  in al,dx
        and al,02h      ;test Rx Buffer
        call delay
        jz  nochr
        jmp chr

prtchr: push ax
        mov dx,6002h
trdy:   in  al,dx
        and al,05h      ;test TxRdy
        jz  trdy
        dec dx
        dec dx
        pop ax
        out dx,al
        ret
```

```
fetchr: mov dx,6000h
        in  al,dx
        push ax
        inc dx
        inc dx
        in al, dx
        and al, 20h     ;Detect framing error
        jnz fe
fer:    in  al,dx
        and al,10h      ;Detect overrun error
        jnz oe
oer:    pop ax
        ret

fe:     mov al,66h
        call prtchr
        call resf
        jmp fer

oe:     mov al,6fh
        call prtchr
        call resf
        jmp oer

resf:   mov dx, 6002h
        mov al, 37h
        out dx, al
        ret

init59: mov dx, 0f000h
        mov al,17h
        out dx,al       ;ICW 1
        inc dx
        inc dx
        mov al,20h
        out dx,al       ;IWC2
        mov al,0fh
        out dx,al       ;IWC4
        mov al,0feh
        out dx,al       ;OCW 1, mask
        dec dx
        dec dx
        mov al,0c0h
        out dx,al       ;OCW2
        mov al,08h
        out dx,al       ;OCW3
        ret

init54: mov dx,8006h
        mov al,0b6h     ;Counter 2, mode 3
        out dx,al
        mov al,76h      ;Counter 1, mode 3
        out dx,al
        mov al,36h      ;Counter 0, mode 3
        out dx,al
        dec dx
        dec dx
        mov al,1fh      ;counter 2 as USART#1 BR:9600 * 16
        out dx,al
        mov al,0        ;
        out dx,al
        dec dx
        dec dx
        mov al,1fh      ;counter 1 as USART#0 BR:9600 * 16
```

```
        out dx,al       ;
        mov al,0        ;
        out dx,al       ;
        dec dx          ;
        dec dx
        mov al,6fh      ;counter 0 as wave timeing:1000
        out dx,al       ;
        mov al,0bah     ;
        out dx,al       ;
        ret

st_cm0: mov dx, 6002h   ;
        jmp init51      ;
st_cm1: mov dx, 7002h   ;
init51: mov al, 4eh     ;
        out dx, al      ;
        call delay      ;
        mov al, 37h     ;
        out dx, al      ;
        call delay      ;
        dec dx
        dec dx
        in al, dx       ;dummy read
        call delay
        ret

delay:  mov cx, 0ffh
dell:   loopnz dell
        ret

initdp: mov dx, 0e000h
        mov al, 38h
        call fg_ck
        out dx, al
        mov al, 38h
        call fg_ck
        out dx, al
        mov al, 0dh
        call fg_ck
        out dx, al
        mov al, 06h
        call fg_ck
        out dx, al
        mov al, 01h
        call fg_ck
        out dx, al
        ret

fg_ck:  push ax
        push dx
        mov dx, 0c004h
ck_lp:  in al, dx
        and al, 80h
        jnz ck_lp
        pop dx
        pop ax
        ret

dpchr:  call fg_ck
        mov dx, 0e002h
        out dx, al
        ret
```

```
timer      dd  00400h
twork      dd  00404h
timtop     dd  00407h

settim: mov bx,040h
        mov ds,bx
        mov bx,0
        mov word ptr [bx],0
        inc bx
        inc bx
        mov word ptr [bx],0
        ret

titod:  lds si,timer    ;location of timer
        les di,twork    ;location of workspace
        mov cx,02h      ;number of words
        cli             ;disable updates during copy
        rep movsw       ;copy timer to workspace
        sti             ;enable interrupts

        mov bx,40h
        mov ds,bx
        mov ah,0
        mov bl,10       ;set base
        mov cl,10       ;number of decimal digits
nexd:   xchg bh,cl      ;store digit count
        mov cx,04       ;set number of bytes
        mov si,07h      ;set index to highest byte
nexb:   mov al,[si]     ;move byte to A
        div bl          ;divide
        mov [si],al     ;copy result back to mem
        dec si          ;select next byte
        loopnz nexb     ;next byte
        xchg al,ah      ;move remainder to al
        mov ah,0        ;clear ah
        push ax         ;store on stack
        xchg cl,bh      ;move digit count to cx
        loopnz nexd     ;next digit

        mov cx,10       ;display
nd:     pop ax
        add al,30h
        call prtchr
        loop nd
        ret


loc0    dd      0000h
vec0    dw      zeroe
        dw      seg zeroe

set0:   les di,loc0
        lea si,vec0
        mov bx, seg vec0
        mov ds,bx
        mov cx,02h
        rep movsw
        ret


loc_59  dd 0080h        ;
vect    dw realt        ;Real timing
        dw seg realt

set59i: les di, loc_59
```

```
        mov bx, seg vect
        mov ds, bx
        lea si, vect
        mov cx, 02h
        rep movsw
        ret

realt:  push bx
        push ds
        mov bx,040h
        mov ds,bx
        mov bx,0
        inc word ptr [bx]
        jnz ex
        inc bx
        inc bx
        inc word ptr [bx]
ex:     pop ds
        pop bx
        iret

err0    db  'FATAL ERROR DIVIDE BY ZERO'

zeroe:  push ds
        push cx
        push bx
        push si
        lea si,err0
        mov bx,0fc00h
        mov ds,bx
        mov cx,01ah
li0:    mov al,es:[si]
        call prtchr
        inc si
        dec cx
        jnz li0
        pop si
        pop bx
        pop cx
        pop ds
        iret
code    ends

stack   segment stack 'stack'

        db  400h dup (?)

stack ends

end
```

```
/************************************************
* Loading program "BOARD.C" for loading hex codes from
* IBM PC into the RAM of application computer
*            by Dr. B. J. Corllet
*              November 1986
************************************************/

#define MODE 0
#define BUFSIZE 128
#define ACK  06
#define BELL 07
#define CR   0X0D
#define LF   0X0A
#define SP      0X020
#define READMK 0X3A
#define DATAREC 0
#define EXTNDADDRSS 2
#define STARTREC 3
#define EOFREC 1
#define CHKSM 10
#define BADHEX 11
#define NOEA  12
#define MULTSA 13
#define SET 1
#define RESET 0

extern char *mwp, *mrp;
extern char hexbuf[BUFSIZE], eaf, saf;
extern unsigned int usba, cs, ip;

main()
{
   char c;
   int em, noendfile;

   prints("\r\n*** MIGMAN  APPLICATION LOADER V1.0 ***\n\n\r");
   prints("B.J.CORLETT - NOVEMBER  '86\n\n\r\n");
   prints("start address = ");
   printi(cs);
   printc(READMK);
   printi(ip);
   prints("\r\n\ntype S to start application\r\n\n");
   if(getc() == 'S')
      stapp(ip,cs);
   printc(BELL);
   initmigbuf();
   eaf = RESET;
   saf = RESET;
   noendfile = SET;
   while(noendfile)
   {
      while((c = rdcom1()) != CR)
         stor(c);
      if((em = rec2mem()) > 9)
         error(em);
      else if(em == EOFREC)
         noendfile = RESET;
      else
         printc(ACK);
      initmigbuf();
   }
   printc(ACK);
```

```
    prints("application loaded  OK\n\n\r");
    prints("start address = ");
    printi(cs);
    printc(READMK);
    printi(ip);
    prints("\r\n\n");
    prints("type S  to start application\n\r");
    if (getc() == 'S')
        stapp(ip,cs);
}

error(ec)           /* spouts error messages */
    int ec;
{
    switch(ec)
    {
        case CHKSM :
            prints("< CHECKSUM ERROR&\n\r");
            break;

        case BADHEX :
            prints("< BAD HEX ERROR&\n\r");
            break;

        case MULTSA :
            prints("< MULTIPLE START ADDRESS RECORDS&\n\r");
            break;

        case NOEA   :
            prints("< NO EXTENDED ADDRESS RECORD&\n\r");
            break;

        default :
            prints("< UNKNOWN ERROR&\n\r");
            break;
    }
}

initmigbuf()   /* initializes pointers to start of buffer */
{
    mrp = hexbuf;
    mwp = mrp;
}

stor(c)             /* stores charcter in buffer */
    char c;
{
    *mwp++  = c;
}

rdbuf()             /* reads character from buffer */
{
    return(*mrp++);
}

char *findstart(p)  /* finds first valid ascii char */
    char *p;
{
    while(*p++  != READMK);
    return(p);
}

rec2mem()           /* copies records to memory */
{
    char *mbstart;
```

```
    int *cwp, cbuf[BUFSIZE];
    unsigned int type,  len, datalen, drla;

    mbstart = findstart(mrp);     /* beware   change to long pointer for migman */
    mrp = mbstart;                /* set read pointer to first byte */
    cwp = cbuf;                   /* set character write pointer to start of cbuf */
    len = nextbyte(mrp);          /* read length */
    len = 10 + 2*len;             /* adjust for overhead */

    if  (convert(mrp,cwp,len) != 0)   /* convert hex file, copying to cbuf */
        return(CHKSM);
    datalen = *cwp + +;           /* data record length */
    drla = nextword(cwp + +);     /* data record load address */
    cwp + +;
    type = *cwp + +;              /* record type */
    switch  (type)
    {
        case DATAREC :
            if (eaf)
                return(datamove(drla,usba,datalen,cwp));
            else
                return(NOEA);
            break;

        case EXTNDADDRSS :
            eaf = SET;
            usba = nextword(cwp);
            break;

        case STARTREC   :
            if (saf)
                return(MULTSA);
            else
            {
                saf = SET;
                cs = nextword(cwp + +);
                cwp + +;
                ip = nextword(cwp);
            }
            break;

        case EOFREC :
            break;
    }
    return(type);
}

nextbyte(hxp)       /* returns single int from two ascii chars */
    char *hxp;
{
    char c;
    int i;

    if  ((c = *hxp) > = '0' && c < = '9')
        i   = (c-48)*16;
    else if (c  > = 'A' && c < = 'F')
        i   = (c-55)*16;
    else
        error(BADHEX);
    *hxp + +;
    if  ((c  = *hxp) > = '0' && c < = '9')
        i   + = (c-48);
    else if (c  > = 'A' && c < = 'F')
        i   + = (c-55);
    else
```

```
        error(BADHEX);
    return(i);
}

int nextword(p)     /*  combines two 8-bit values to one 16- bit word */
    unsigned int *p;
{
    unsigned int i;

    i =  *p + + ;
    i * =  256;
    i + =  *p;
    return(i);
}

checksum(p)         /* performs checksum operation on record */
    int *p;
{
    char length, i;

    i = 0;
    length = *p;
    length  + = 5;
    while (length--  >  0)
        i   + =  *p + + ;
    return(i);
}

convert(rp, wp, le) /* converts ascii chars to usable code or data */
    char *rp, le;
    int *wp;
{
    int *p;

    p = wp;
    le  / =  2;
    while (le--  > 0)
    {
        *wp + + = nextbyte(rp + + );
        rp + + ;
    }
    return(checksum(p));
}

printi(i)
    unsigned int i;
{
    char str[5];
    int b;
    int c  =  4;

    str[c--] = '\0';
    while (c  >  =  0)
    {
        if ((b = i%16)  <  =  9)
            str[c--] = b + 48;
        else
            str[c--] = b + 55;
        i / =     16;
    }
    prints(str);
}
```

```
;*****************************************************
; User assembler routines "ARCLA.A86" for booting program
; and arc oscillation control
;              by Mr. Xiao Qi Chen
;                 May 1988
;*****************************************************

name arcla.a86

    EXTRN   wavout_:FAR, keyrd_:FAR
    EXTRN   wavrd_:FAR, biasrd_:FAR
    EXTRN   volint_:FAR, optint_:FAR
    EXTRN   tx0int_:FAR, rx0int_:FAR, tx1int_:FAR

assume cs:code

code segment para public 'code'

    PUBLIC printc_
    PUBLIC getc_
    PUBLIC prints_
    PUBLIC datamove_
    PUBLIC stapp_
    PUBLIC io_read_
    PUBLIC io_write_
    PUBLIC setint_
    PUBLIC reint2_


;************************************
;character output on channel one or two
;************************************

printc_  PROC FAR
        push dx
        push bp
        mov bp,sp
        mov dx,6002h
        mov ax,[bp + 10]
        cmp al,01h
        jnz trdy
        mov dx,7002h
trdy:   in al,dx
        and al,05h          ;examine 8251 status reg
        jz trdy             ;if ready to transmit
        dec dx
        dec dx
        mov ax,[bp + 8]
        out dx,al           ;output charater
        pop bp
        pop dx
        ret
printc_  ENDP

;************************************
;character input on channel one or two
;************************************

getc_   PROC FAR
        push dx
        push bp
        mov bp,sp
```

```
        mov dx,6002h
        mov ax,[bp + 8]
        cmp al,01h
        jnz nochr
        mov dx,7002h
nochr:  in al,dx            ;test RBF
        and al,02h
        jz nochr
        dec dx
        dec dx
        in al,dx            ;input character
        cbw
        pop bp
        pop dx
        ret
getc_   ENDP

;*********************************
;string output on channel one or two
;*********************************

prints_  PROC FAR
        push si
        push di
        push bx
        push bp
        mov bp,sp
        mov bx,[bp + 16]
        p1 equ dword ptr [bp + 12]
        les si,p1
loop1:  mov al,es:[si]
        or al,al
        jz  ex1
        push bx
        push ax
        call printc_
        add sp,04h
        inc si
        jmp loop1
ex1:    mov sp,bp
        pop bp
        pop bx
        pop di
        pop si
        ret
prints_  ENDP

;************************************************
;datamove(drla,usba,len,ptr) low-level data transfer
;************************************************

datamove_   PROC FAR
        push ds
        push es
        push di
        push si
        push cx
        push bx
        push bp
        mov bp,sp
        mov di,[bp + 18]
        mov bx,[bp + 20]
        mov es,bx
        mov cx,[bp + 22]
        mov si,[bp + 24]
```

```
        mov bx,[bp + 26]
        mov ds,bx
 nb:    mov al,ds:[si]
        inc si
        inc si
        mov es:[di],al
        inc di
        loop nb
        mov ax,0h
        pop bp
        pop bx
        pop cx
        pop si
        pop di
        pop es
        pop ds
        ret
datamove_  ENDP

;*************************************
;stapp(ip,cs) - start application programm
;*************************************

stapp_  PROC FAR
        push bp
        mov bp,sp
        call dword ptr [bp + 6]
        pop bp
        ret
stapp_  ENDP

;****************************
;return current time  in dx:ax
;****************************

time_   PROC FAR
        push bx
        push ds
        push si
        mov bx,40h          ;set up ds = 0040
        mov ds,bx
        mov si,0
        cli             ;disable interrupts during transfer
        mov ax,[si]        ;move low word to ax
        inc si
        inc si
        mov dx,[si]        ;move high word to dx
        sti             ;restore interrupts
        pop si
        pop ds
        pop bx
        ret
time_   ENDP

;*****************************************
; generalised i/o read routine  read_(port,w|b)
;*****************************************

io_read_ PROC FAR
        push dx
        push bp
        mov bp,sp
        mov dx,[bp + 8]          ;dx =  port address
        mov ax,[bp + 10]         ;al =  byte or word
        cmp al,'w'
```

```
        jz j1
        cmp al,'W'              ;if word is not specified
        jz j1
        in al,dx                ;perform 8-bit io read
        mov ah, 0
        jmp j2
j1:     in ax,dx
j2:     pop bp
        pop dx
        ret
io_read_ ENDP

;*********************************************
;generalised io write routine write(port,value,b|w)
;*********************************************

io_write_ PROC FAR
        push dx
        push bp
        mov bp,sp
        mov dx,[bp + 8]
        mov ax,[bp + 12]
        cmp al,'w'
        jz j3
        cmp al,'W'
        jz j3
        mov ax,[bp + 10]
        out dx,al
        jmp j4
j3:     mov ax,[bp + 10]
        out dx,ax
j4:     pop bp
        pop dx
        ret
io_write_  ENDP

;*******************
;set interrupt vector
;*******************

loc0    dd 0080h

vector  dw ir0
        dw seg ir0
        dw ir1
        dw seg ir1
        dw ir2
        dw seg ir2
        dw ir3
        dw seg ir3
        dw ir4
        dw seg ir4
        dw ir5
        dw seg ir5
        dw ir6
        dw seg ir6
        dw ir7
        dw seg ir7

setint_ PROC FAR
        push di
        push si
        push es
        push bx
        push ds
```

```
        push cx
        les di, loc0
        lea si, vector
        mov bx, seg vector
        mov ds, bx
        mov cx, 10h
        rep movsw
        pop cx
        pop ds
        pop bx
        pop es
        pop si
        pop di
        ret
setint_ ENDP

;*************************
;redirect interrupt vector
;*************************

loc2    dd 0088h

vect2   dw ir22
        dw seg ir22

reint2_ PROC FAR
        push di
        push si
        push es
        push bx
        push ds
        push cx
        les di, loc2
        lea si, vect2
        mov bx, seg vect2
        mov ds, bx
        mov cx, 02h
        rep movsw
        pop cx
        pop ds
        pop bx
        pop es
        pop si
        pop di
        ret
reint2_ ENDP

;*****************
;interrupt handler
;*****************

ir0:    push ax
        push dx
        call wavout_
        pop dx
        pop ax
        iret

ir1:    push ax
        push dx
        call keyrd_
        pop dx
        pop ax
        iret
```

```
ir2:    push ax
        push dx
        call wavrd_
        pop dx
        pop ax
        iret

ir22:   push ax
        push dx
        call biasrd_
        pop dx
        pop ax
        iret

ir3:    push ax
        push dx
        call volint_
        pop dx
        pop ax
        iret

ir4:    push ax
        push dx
        call optint_
        pop dx
        pop ax
        iret

ir5:    push ax
        push dx
        call rx0int_
        pop dx
        pop ax
        iret

ir6:    push ax
        push dx
        call tx0int_
        pop dx
        pop ax
        iret

ir7:    push ax
        push dx
        call tx1int_
        pop dx
        pop ax
        iret


CODE ENDS
end
```

```
/*********************************************
* "C" library of user routines "ARCLC.C" for arc
* oscillation control
*            by Mr. Xiao Qi Chen
*                May 1988
*********************************************/

#include "arcio.h"
#include "const.h"

extern char kcode;
extern long time, tbase;
extern int  t_dw, t_wv;
extern int f_bias, f_amp, f_base;
extern char fb_chr[8], fa_chr[8], tw_chr[8], td_chr[8];
extern char *mwp;

pic_mask(i)          /* Mask PIC */
int i;
{
   io_write(pic_int, i ,'b');
}

pit0_rat(i)          /* Set CLK freq */
int i;
{
   io_write(pit0_reg, i, 'w');
}

pit1_rat(i)
int i;
{
   io_write(pit1_reg, i, 'w');
}

pit2_rat(i)
int i;
{
   io_write(pit2_reg, i, 'w');
}

on_wv()              /* start weaving */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0fe;
   io_write(pic_int, i, 'b');
}

off_wv()             /* stop weaving */
{
   int i;
   i = io_read(pic_int, 'b') | 0X01;
   io_write(pic_int, i, 'b');
   wv_write(f_cent);
}

on_key()             /* enable key int */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0fd;
   io_write(pic_int, i, 'b');
}
```

```
off_key()
{
   int i;
   i = io_read(pic_int, 'b') | 0X02;
   io_write(pic_int, i, 'b');
}

on_pot()          /* start pot reading */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0fb;
   io_write(pic_int, i, 'b');
   io_write(pot_stat, 0, 'b');
}

off_pot()         /* stop pot reading */
{
   int i;
   i = io_read(pic_int, 'b') | 0X04;
   io_write(pic_int, i, 'b');
}

on_vols()         /* enable voltage sensor */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0f7;
   io_write(pic_int, i, 'b');
}

off_vols()        /* disable volt sensor */
{
   int i;
   i = io_read(pic_int, 'b') | 0X08;
   io_write(pic_int, i, 'b');
}

on_opts()         /* enable opto sensor */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0ef;
   io_write(pic_int, i, 'b');
}

off_opts()        /* disable opto sensor */
{
   int i;
   i = io_read(pic_int, 'b') | 0X010;
   io_write(pic_int, i, 'b');
}

on_tx0()          /* enable Tx on com0 */
{
   int i;
   i = io_read(pic_int, 'b') & 0X0df;
   io_write(pic_int, i, 'b');
}

off_tx0()         /* disable Tx on com0 */
{
   int i;
   i = io_read(pic_int, 'b') | 0X020;
   io_write(pic_int, i, 'b');
}
```

```
on_rx0()            /* enable Rx on com0 */
{
    int i;
    i = io_read(pic_int, 'b') & 0X0bf;
    io_write(pic_int, i, 'b');
}

off_rx0()           /* disable Rx on com0 */
{
    int i;
    i = io_read(pic_int, 'b') | 0X040;
    io_write(pic_int, i, 'b');
}

on_tx1()            /* enable Tx on com1 */
{
    int i;
    i = io_read(pic_int, 'b') & 0X07f;
    io_write(pic_int, i, 'b');
}

off_tx1()           /* disable Tx on com1 */
{
    int i;
    i = io_read(pic_int, 'b') | 0X080;
    io_write(pic_int, i, 'b');
}

static key_tab[] =
{
    '7', 'A', '3', '4', '8', 'B', '2', '5',
    '9', 'C', '1', '6', '0', 'D', 14,  'P'
};

keyrd()             /* keypad int handler */
{
    kcode = io_read(key_reg,'w') & 0X0f;
    kcode = key_tab[kcode];
}

nxkey()             /* wait for new key */
{
    kcode = '@';
    while(kcode = = '@');
    return(kcode);
}

digit_k()
{
    kcode = '@';
    while(kcode = = '@');
    while(kcode < '0' || kcode > '9');
    return(kcode);
}


st_pot0()           /* initialise pot */
{
    io_write(pot_stat, 8, 'b');
}

wavrd()             /* pot ISR */
{
    int i, chan;
    chan = io_read(pot_stat, 'w') & 0X03;
```

```
    i = (io_read(pot_high, 'w') & 0X0ff);  /* 0 to 255 */
    switch(chan)
    {
       case 0:                    /* field bias */
          i *= 4;
          if(i > 999)
             i = 999;
          f_bias = i * 4;         /* f_base from downlim: 48 */
          f_bias += downlim;      /* to uplim: 4044 */
          i *= 10;                /* scale 0 to 9.990 */
          int1_asc(fb_chr, i);
          dpbuf(fb_chr, pa0_ram);
          io_write(pot_stat, 1, 'b');
          break;
       case 1:                    /* field amp */
          i *= 4;
          if(i > 999)
             i = 999;
          f_amp = i * 4;          /* f_amp from 0 to 3996 */
          i *= 10;                /* scale 0 to 9.990 */
          int1_asc(fa_chr, i);
          dpbuf(fa_chr, pa1_ram);
          io_write(pot_stat, 2, 'b');
          break;
       case 2:                    /* weave time */
          i *= 10;
          t_wv = i;               /* 0 to 2550 CLKs */
          int1_asc(tw_chr, i);    /* scale 0 to 2.550 sec */
          dpbuf(tw_chr, pa2_ram);
          io_write(pot_stat, 3, 'b');
          break;
       case 3:                    /* dwell time */
          i *= 10;
          t_dw = i;               /* 0 to 2550 CLKs */
          int1_asc(td_chr, i);    /* scale 0 to 2.550 sec */
          dpbuf(td_chr, pa3_ram);
          io_write(pot_stat, 0, 'b');
          break;
       default :
          break;
    }

}

biasrd()
{
    int i, chan;
    chan = io_read(pot_stat, 'w') & 0X03;
    i = (io_read(pot_high, 'w') & 0X0ff);  /* 0 to 255 */
    switch(chan)
    {
       case 0:                    /* field bias */
          i *= 4;
          if(i > 999)
             i = 999;
          f_bias = i * 4;         /* f_base from downlim: 48 */
          f_bias += downlim;      /* to uplim: 4044 */
          i *= 10;                /* scale 0 to 9.990 */
          int1_asc(fb_chr, i);
          dpbuf(fb_chr, pa0_ram);
          io_write(pot_stat, 0, 'b');
          break;
       default:
          break;
```

```
    }
}

para_in(bp)          /* wave para i/p from keypad: X.XXX */
char bp[8];
{
    char c;

    c = digit_k();                  /* MSB */
    bp[0] = c;
    dpchr(c);
    bp[1] = '.';                    /* decimal point */
    dpchr('.');
    c = digit_k();
    bp[2] = c;
    dpchr(c);
    c = digit_k();
    bp[3] = c;
    dpchr(c);
    c = digit_k();
    bp[4] = c;
    dpchr(c);
    bp[5] = 0;                      /* terminator */
}

wavout()             /* wave o/p */
{
    int fmin, fmax, fwav;
    int cycle, tdif;

    cycle = (t_wv + t_dw) * 2;      /* No. of CLKs in a cycle */
    tdif = + + time - tbase;

    if(tdif > = cycle)
    {
        tbase + = cycle;
        tdif = time - tbase;
    }

    fmax = f_bias + f_base;
    fmin = f_bias - f_base;

    if(tdif < = t_wv)
    {
        fwav = fmax - fmin;
        fwav *= 8;
        fwav /= t_wv;
        fwav *= tdif;
        fwav /= 8;
        fwav + = fmin;
        wv_write(fwav);
    }

    else if((tdif - t_wv) < = t_dw)
        wv_write(fmax);

    else if((tdif - t_dw) > t_wv * 2)
        wv_write(fmin);

    else
    {
        tdif -= (t_wv + t_dw);
        fwav = fmax - fmin;
        fwav *= 8;
        fwav /= t_wv;
```

```
        fwav *= tdif;
        fwav /= 8;
        fwav = fmax - fwav;
        wv_write(fwav);
    }
}

wv_write(i)
int i;
{
    int si;
    int wv_reg = 0X05000;

    si = i & 0X0f;
    io_write(wv_reg, si, 'b');
    si = (i / 16) & 0X0f;
    wv_reg += 2;
    io_write(wv_reg, si, 'b');
    si = (i / 256) & 0X0f;
    wv_reg += 2;
    io_write(wv_reg, si, 'b');
    wv_reg += 2;
    io_write(wv_reg, si, 'b');
}


dpctrl(i)           /* set display */
int i;
{
    while(io_read(dp_stat, 'w') & 0X080);
    io_write(dp_set, i, 'b');
}


dpchr(i)            /* disp a chr */
int i;
{
    while(io_read(dp_stat, 'w') & 0X80);
    io_write(dp_wr, i, 'b');
}


dpstr(c)            /* display a string */
char *c;            /* dpstr("....") */
{
    while(*c)
    {
        while(io_read(dp_stat, 'w') & 0X080);
        io_write(dp_wr, *c, 'b');
        ++c;
    }
}


dpbuf(bp, dp_loc)   /* display contents of buf */
char bp[8];
int dp_loc;
{
    dpctrl(dp_loc);             /* display start loc */
    mwp = bp;
    while(*mwp)
    {
        while(io_read(dp_stat, 'w') & 0X080);
        io_write(dp_wr, *mwp, 'b');
        ++mwp;
```

```
    }
}

volint()
{}

optint()
{}

rx0int()
{}

tx0int()
{}

tx1int()
{}

int1_asc(bp, i)
        /* convert integer to char with a integral bit */
char bp[8];
int i;
{
    bp[0] = (i / 1000) + '0';       /* MSB */
    bp[1] = '.';                    /* decimal point */
    bp[2] = (i / 100) % 10 + '0';
    bp[3] = (i / 10) % 10 + '0';
    bp[4] = i % 10 + '0';           /* LSB */
    bp[5] = 0;                      /* terminator */
}

chr_3int(bp)    /* convert chr:X.XXX to int:XXX */
char bp[8];
{
    int pa;

    pa = (bp[0] - '0') * 100;
    pa += (bp[2] - '0') * 10;
    pa += bp[3] - '0';
    return(pa);
}

chr_4int(bp)    /* convert chr:X.XXX to int:XXXX */
char bp[8];
{
    int pa;

    pa = (bp[0] - '0') * 1000;
    pa += (bp[2] - '0') * 100;
    pa += (bp[3] - '0') * 10;
    pa += bp[4] - '0';
    return(pa);
}
```

```
/**************************************
* Arc oscillation control program "WEAVE.C"
*       by Mr. Xiao Qi Chen
*           May 1988
**************************************/

#include "const.h"
#define ENT 14
extern int f_bias, f_amp, f_base;
extern int t_wv, t_dw;
extern char fb_chr[8], fa_chr[8], tw_chr[8], td_chr[8];
extern long time, tbase;
extern char kcode, *mwp;

main()
{
   off_wv();
   setint();
   pit0_rat(osc_rat);            /* select 1KHz CLK */
   st_pot0();                    /* init, pot */
   time = 0;                     /* init. timing */
   tbase = 0;
   f_base = 0;
   f_bias = f_cent;
   intl_asc(fb_chr, 5000);       /* init. parameter buffers */
   intl_asc(fa_chr, 0);
   intl_asc(tw_chr, 0);
   intl_asc(td_chr, 0);
   on_key();                     /* enable key interrupt */
   for(;;)
   {
      off_pot();
      off_wv();
      dpmanu();
      switch(nxkey())
      {
         case 'A':
            preset();
            break;
         case 'B':
            weave();
            break;
         case 'C':
            remote();
            break;
         case 'D':
            fdback();
            break;
         default:
            break;
      }
   }
}

preset()
{
   dpctrl(SET);
   dpstr("KEY_PRESET");
   wav_pa();
   while(nxkey() != ENT)
   switch(kcode)
   {
```

```
        case 'A':
            dpctrl(pa0_ram);
            para_in(fb_chr);
            f_bias = chr_3int(fb_chr);
            f_bias *= 4;
            f_bias += downlim;
            break;
        case 'B':
            dpctrl(pa1_ram);
            para_in(fa_chr);
            f_amp = chr_3int(fa_chr);
            f_amp *= 4;
            break;
        case 'C':
            dpctrl(pa2_ram);
            para_in(tw_chr);
            t_wv = chr_4int(tw_chr);
            if(t_wv > 5000)
            {
                t_wv = 5000;        /* limit t_wv to 5 sec */
                int1_asc(tw_chr, t_wv);
                dpbuf(tw_chr, pa2_ram);
            }
            break;
        case 'D':
            dpctrl(pa3_ram);
            para_in(td_chr);
            t_dw = chr_4int(td_chr);
            break;
        default:
            break;
    }
    opt_amp();
}

opt_amp()           /* optimise field amplitude */
{
    if((f_bias + f_amp / 2) > uplim)
        f_base = uplim - f_bias;
    else if((downlim + f_amp / 2) > f_bias)
        f_base = f_bias - downlim;
    else
        f_base = f_amp / 2;
}

weave()
{
    dpctrl(SET);
    dpstr("PRESET_WEAVE");
    wav_pa();
    on_wv();
    while(nxkey() != ENT);
    off_wv();
}

remote()            /* remote weaving mode */
{
    dpctrl(SET);
    dpstr("REMOTE_WEAVE");
    wav_pa();
    on_pot();
    on_wv();
    while(nxkey() != ENT)
        opt_amp();
    off_wv();
```

```
    off_pot();
}

fdback()            /*** feedback loop control ***/
{
        dpfdbk();
        while(nxkey() != ENT)
        {
            switch(kcode)
            {
                case 'A':
                    voltage();
                    break;
                case 'B':
                    vision();
                    break;
                case 'C':
                    volvis();
                    break;
                default:
                    break;
            }
    }
}

voltage()           /*** arc weaving with voltage sensor ***/
{
    dpctrl(SET);
    dpstr("VOLTAGE SENSOR");
    wav_pa();
    on_volt();
    on_wv();
    while(kcode != ENT)
        volt_larc();
    off_wv();
    off_volt();
}

vision()            /*** arc weaving with vision sensor ***/
{
    dpctrl(SET);
    dpstr("VISION SENSOR");
    wav_pa();
    on_visn();
    on_wv();
    while(kcode != ENT)
        visn_track();
    off_wv();
    off_visn();
}

volvis()            /*** arc weaving with both sensor ***/
{
    dpctrl(SET);
    dpstr("VOLT_VISN SENSOR");
    wav_pa();
    on_volt();
    on_visn();
    on_wv();
    while(kcode != ENT)
    {
        visn_track();
        volt_larc();
    }
    off_wv();
```

```
    off_visn();
    off_volt();
}

dpmanu()            /*** display top manu. ***/
{
    dpctrl(SET);
    dpstr("MANUAL");
    dpctrl(line2);
    dpstr("preset    weave     ");
    dpstr("remote    close");
}

wav_pa()            /*** display weaving para ***/
{
    dpctrl(line2);
    dpstr("fb:       fa:       ");
    dpstr("tw:       td:");
    dpbuf(fb_chr, pa0_ram);
    dpbuf(fa_chr, pa1_ram);
    dpbuf(tw_chr, pa2_ram);
    dpbuf(td_chr, pa3_ram);
}

dpfdbk()            /*** display sensor options ***/
{
    dpctrl(SET);
    dpstr("CLOOSE-LOOP");
    dpctrl(line2);
    dpstr("voltage   vision    ");
    dpstr("volt-visn");
}

;************************
; Heading file "const.h"
;************************

#define COM0        0
#define COM1        1
#define RESET       0
#define SET         1
#define EOT         04
#define ACK         06
#define BELL        07
#define LF          10
#define CR          13

#define line1       0X080
#define pend        0X0a0
#define line2       0X0c0
#define pa0_ram     0X0c3
#define pa1_ram     0X0cd
#define pa2_ram     0X0d7
#define pa3_ram     0X0e1

#define f_cent      2048
#define uplim       4048
#define downlim     48
#define wv_clk      100
#define wv_rat      0X0ba6f
#define osc_clk     1000
#define osc_rat     0X012a5
```

```
/************************************************
* C library of user routines "SYSLC.C" for vision-based
* control of narrow gap TIG welding
*            by Mr. Xiao Qi Chen
*               August 1988
************************************************/

#include "stdio.h"
#include "string.h"
#include "dos.h"

#define TRUE 1
#define FALSE 0

#define up_arr 'H'
#define dn_arr 'P'
#define lf_arr 'K'
#define rg_arr 'M'
#define ul_k 'G'
#define ur_k 'I'
#define dl_k 'O'
#define dr_k 'Q'

#define frame1 0x0c000
#define frame2 0x0d000
#define f1_adr 0x0c0000
#define f2_adr 0x0d0000

#define port1 0x030c
#define port2 0x030e
#define f1cap 0x015
#define f1disp 0x017
#define f2cap 0x01a
#define f2disp 0x01b
#define fstop 0x0f

int frame;
char *fsp;              /* source frame pointer */
char *ftp;              /* template frame pointer */
char *sep = " ", *newl = "\n";
char st[25], pstr[8];

struct template{
            int x_size;
            int y_size;
            int x_loc;
            int y_loc;
            long corr_lim;
            char *data;
        };

set_f(f)        /*** select source and template frame***/
int f;
{
   if(f)
   {
      frame = frame2;
      fsp = f2_adr;
      ftp = f1_adr;
   }
   else
   {
```

```
        frame = frame1;
        fsp = f1_adr;
        ftp = f2_adr;
    }
}

waitst()                    /*** wait start ***/
{
    while((inp(port2) & 0x01) == 0);
    while((inp(port2) & 0x01));
}

capdisp(frame)              /*** continuous capture ***/
int frame;
{
    waitst();
    if(frame)
        outp(port1,f2cap);
    else
        outp(port1,f1cap);
}

capture(frame)          /*** capture a single frame ***/
    int frame;
{
    if(frame)
    {
        waitst();
        outp(port1,f2cap);
        waitst();
        outp(port1,f2disp);
    }
    else
    {
        waitst();
        outp(port1,f1cap);
        waitst();
        outp(port1,f1disp);
    }
}

disp(frame)             /*** display a frame ***/
int frame;
{
    waitst();
    if(frame)
        outp(port1,f2disp);
    else
        outp(port1,f1disp);
}

disable()                   /*** disable frame store ***/
{
    waitst();
    outp(port1,fstop);
}

threshold(tv)               /*** threshold source frame ***/
char tv;
{
    char *p;
    long int i;

    p = fsp;
    for(i = 0; i < 0x010000; i + + )
```

```
    {
        if((*p & 0x07f) < tv)
            *p = 0;
        else
            *p = 0x07f;
        ++p;
    }
}

thresh_area(x,y,xs,ys,tv)      /*** threshold an area ***/
unsigned int x,y,xs,ys;
char tv;
{
    char *p;
    unsigned int i, j;

    for(i=0; i<ys; i++)
    {
        p = fsp + x + 256*(y+i);
        for(j=0; j<xs; j++)
        {
            if((*p & 0x07f) < tv)
                *p = 0;
            else
                *p = 0x07f;
            ++p;
        }
        p += 256;
    }
}

df_box(xl, yl, xs, ys)         /*** define a window ***/
int *xl,*yl,*xs,*ys;
{
    unsigned char x1,y1,x2,y2,x,y;
    char c;

    do
    {
        printf("\r\ndefine window");
        printf("\r\ndefine first point");
        df_ptr(&x1, &y1);
        printf("\r\ndefine second ponit");
        df_ptr(&x2, &y2);
        cross(x1,y1);
        cross(x2,y2);
        if(x1 > x2)
        {
            x = x1;
            x1 = x2;
            x2 = x;
        }
        if(y1 > y2)
        {
            y = y1;
            y1 = y2;
            y2 = y;
        }
        *xs = x2 - x1;
        *ys = y2 - y1;
        if(*xs==0 || *ys==0)
            printf("\r\nwindow not found\r\n");
    }
    while(*xs==0 || *ys == 0);
    draw_area(x1-1,y1-1,*xs+2,*ys+2);
```

```
    printf("\r\nmove window or press c to close");
    printf("\r\nx_loc: %d, y_loc: %d, x_size: %d,
          y_size: %d",x1,y1,*xs,*ys);
    while((c=getch()) != 'c' && c != 'C')
    {
       draw_area(x1-1,y1-1,*xs+2,*ys+2);
       switch(c)
       {
          case up_arr:
             --y1;
             break;
          case dn_arr:
             ++y1;
             break;
          case lf_arr:
             --x1;
             break;
          case rg_arr:
             ++x1;
             break;
          case ul_k:
             --y1;
             --x1;
             break;
          case ur_k:
             --y1;
             ++x1;
             break;
          case dl_k:
             ++y1;
             --x1;
             break;
          case dr_k:
             ++y1;
             ++x1;
             break;
          default:
             break;
       }
       draw_area(x1-1,y1-1,*xs+2,*ys+2);
       printf("\rx_loc: %d, y_loc: %d, x_size: %d,
             y_size: %d",x1,y1,*xs,*ys);
    }
    *xl = x1;
    *yl = y1;
    printf("\r\nwindow defined");
}

df_ptr(xp, yp)                  /*** define a point ***/
unsigned char *xp, *yp;
{
    char c;
    unsigned char x=128, y=128;

    printf("\r\nmove point or press c to close");
    cross(x,y);
    printf("\r\nx = %d, y = %d",x,y);
    while((c=getch()) != 'c' && c != 'C')
    {
       cross(x,y);
       switch(c)
       {
          case up_arr:
             --y;
             break;
```

```
            case dn_arr:
                ++y;
                break;
            case lf_arr:
                --x;
                break;
            case rg_arr:
                ++x;
                break;
            case ul_k:
                --y;
                --x;
                break;
            case ur_k:
                --y;
                ++x;
                break;
            case dl_k:
                ++y;
                --x;
                break;
            case dr_k:
                ++y;
                ++x;
                break;
            default:
                break;
        }
        cross(x,y);
        printf("\rx= %d, y = %d  ",x,y);
    }
    *xp = x;
    *yp = y;
    printf("\r\npoint defined");
}

printi(i)           /*** print 16bit numbers in hex ***/
unsigned int i;
{
    char str[6];
    int b, c = 5;

    str[c--] = '\0';
    str[c--] = '$';
    while( c >= 0 )
    {
        if ((b= i % 16) <= 9)
            str[c--] = b + 48;
        else
            str[c--] = b + 55;
        i /= 16;
    }
    puts(str);
}

printd(i)           /*** print 16 bit numbers in decimal ***/
unsigned int i;
{
    char str[7];
    int c = 6;

    str[c--] = '\0';
    str[c--] = '$';
    while(c >= 0)
    {
```

```
        str[c--] = (i % 10) + '0';
        i /= 10;
    }
    puts(str);
}

printl(i)           /*** print 32bit numbers in hex ***/
unsigned long int i;
{
    char str[10];
    int b, c = 9;

    str[c--] = '\0';
    str[c--] = '$';
    while( c >= 0 )
    {
        if ((b = i % 16) <= 9)
            str[c--] = b + 48;
        else
            str[c--] = b + 55;
        i /= 16;
    }
    puts(str);
}

strip(s)                /*** strip non-char ***/
char *s;
{
    char c;
    int i = 0;

    while((c = *s) != '\n' && c != ' ' && c != '\t' && c != '\r')
    {
        ++s;
        ++i;
    }
    *s = '\0';
    return(i);
}

fgeti(fp)           /*** get a integer from a level 2 file ***/
FILE *fp;             /*** return the int. ***/
{
    int val = 0, mul = 1;
    char c, istr[8], *p = istr;

    *p++ = '$';
    while((c = fgetc(fp)) != ' ')          /* separator: ' ' */
        if(c != ',' && c != '\n' && c != '\r' && c != '\t')
            *p++ = c;
    --p;
    while((c = *p--) != '$')
    {
        val += (c - 48) * mul;
        mul *= 10;
    }
    return(val);
}

prn_buf(data, size)  /* print chars in a buffer on the screen */
long int size;
char *data;
{
    long int i;
```

```
    for(i = 0; i < size; i + + )
        printf("%d, ", *data + + );
}

read_hl(x, y, xs, db)    /*** read a horizontal line ***/
unsigned int x, y, xs;
char *db;
{
    unsigned int i;
    char *p;

    p = fsp + x + 256*y;
    for(i = 0; i < xs; i + + )
        *db + + = *p + + & 0x07f;
}

write_hl(x, y, xs, db)   /*** write to a horizontal line ***/
unsigned int x, y, xs;
char *db;
{
    unsigned int i;
    char *p;

    p = fsp + x + 256*y;
    for(i = 0; i < xs; i + + )
        *p + + = *db + + ;
}

fill_hl(x,y,xs,gl)   /* fill horizontal line with a value */
unsigned int x,y,xs;
char gl;
{
    unsigned int i;
    char *p;

    p = fsp + x + 256*y;
    for(i = 0; i < xs; i + + )
        *p + + = gl;
}

draw_hl(x,y,xs)          /*** draw a horizontal line ***/
unsigned int x, y, xs;
{
    unsigned int i;
    char *p;

    p = fsp + x + 256*y;
    for(i = 0; i < xs; i + + )
    {
        *p = *p î 0x07f;
        + + p;
    }
}

dx_ptr(x,y)          /*** draw a ptr ***/
unsigned int x,y;
{
    char *p;

    p = fsp + x + 256*y;
    *p = *p î 0x07f;
}

read_vl(x, y, ys, db)    /*** read vertical line ***/
unsigned int x, y, ys;
```

```
char *db;
{
   unsigned int i;
   char *p;

   p = fsp + x + 256*y;
   for(i=0; i<ys; i++)
   {
      *db++ = *p & 0x07f;
      y += 256;
   }
}

write_vl(x, y, ys, db)   /*** write to vertical line ***/
unsigned int x, y, ys;
char *db;
{
   unsigned int i;
   char *p;

   p = fsp + x + 256*y;
   for(i=0; i<ys; i++)
   {
      *p = *db++;
      p += 256;
   }
}

draw_vl(x,y,ys)          /*** draw vertical line ***/
unsigned int x,y,ys;
{
   unsigned int i;
   char *p;

   p = fsp + x + 256*y;
   for(i=0; i<ys; i++)
   {
      *p = *p î 0x07f;
      p += 256;
   }
}

fill_vl(x,y,ys,gl)    /*** fill vertical line with a value ***/
unsigned int x,y,ys;
char gl;
{
   unsigned int i;
   char *p;

   p = fsp + x + 256*y;
   for(i=0; i<ys; i++)
   {
      *p = gl;
      p += 256;
   }
}

cross(x,y)               /*** draw a cross at (x, y) ***/
unsigned int x,y;
{
   draw_hl(x-5,y,10);
   draw_vl(x,y-5,10);
}
```

```
draw_area(x,y,xs,ys)     /*** draw a box ***/
unsigned int x,y,xs,ys;
{
   draw_vl(x,y,ys);
   draw_hl(x,y + ys,xs);
   draw_vl(x + xs,y,ys);
   draw_hl(x,y,xs);
}

read_area(x,y,xs,ys,db)  /*** read pixels in an area ***/
unsigned int x,y,xs,ys;  /*** into buffer ***/
char *db;
{
   unsigned int i, j;
   char *p;

   for(i = 0; i < ys; i + + )
   {
      p = fsp + x + 256*(y + i);
      for(j = 0; j < xs; j + + )
         *db + + = *p + + & 0x07f;
   }
}

get_area(x,y,xs,ys,fp)  /*** read pixels in an area ***/
unsigned int x,y,xs,ys;  /*** into a file ***/
FILE *fp;
{
   unsigned int i, j;
   char *p;

   stci_d(st, xs);
   fputs(st,fp);
   fputs(newl, fp);

   stci_d(st, ys);
   fputs(st, fp);
   fputs(newl, fp);

   stci_d(st, x);
   fputs(st, fp);
   fputs(newl, fp);

   stci_d(st, y);
   fputs(st, fp);
   fputs(newl, fp);

   for(i = 0; i < ys; i + + )
   {
      p = fsp + x + 256*(y + i);
      for(j = 0; j < xs; j + + )
      {
         if(j%16 = = 0)
            fputs(newl, fp);
         stci_d(st, *p + + & 0x07f);
         fputs(st,fp);
         fputs(sep,fp);
      }
   }
}

rec_frame(fp)       /*** record image to level 2 file ***/
FILE *fp;           /*** in integral form ***/
{
   long int i;
```

```
    char *p=fsp;

    for(i=0; i<0x010000; i++)
    {
        if((i%16 == 0) && i)
            fputs(newl, fp);
        stci_d(st, *p++ & 0x07f);
        fputs(st,fp);
        fputs(sep,fp);
    }
}

play_frame(fp)      /*** copy image data stored in a file ***/
FILE *fp;           /*** to frame store ***/
{
    long int i;
    char *p=fsp;

    for(i=0; i<0x010000; i++)
        *p++ = fgeti(fp);
}

rd_frame(fp)        /*** read 256*256 7-bit frame store to ***/
FILE *fp;           /*** to a file (in ASCII) ***/
{
    unsigned int i, j;
    char *p, c, r;

    for(i=0; i<0x0100; i++)
    {
        p = fsp + 256*i;
        printf("%X \n", i);    /*** print line number ***/
        for(j=0; j<0x0100; j++)
        {
            c = *p++ & 0x07f;
            r = fputc(c, fp);
            if(r == -1)
                clrerr(fp);     /*** clear error flags ***/
        }
    }
}

wr_frame(fp)        /*** write picture data (in ASCII) stored ***/
FILE *fp;           /*** in a file to the frame store ***/
{
    unsigned int i, j;
    char *p, c;

    for(i=0; i<0x0100; i++)
    {
        p = fsp + 256*i;
        printf("%X \n", i);   /*** print line number ***/
        for(j=0; j<0x0100; j++)
        {
            c = fgetc(fp);
            if(c == -1)
                clrerr(fp);   /*** clear error flags ***/
            *p++ = c;
        }
    }
}

dig_frame(fp)       /*** read 256*256 frame store into a file ***/
FILE *fp;           /*** 512*512 full frame ***/
{
```

```
    unsigned int i, j, k;
    char *p, c, r;

    for(i = 0; i < 0x0100; i + + )
        for(j = 0; j < 2; j + + )          /*** repeat lines ***/
        {
            printf("%X \n", i);    /*** print line number ***/
            p = fsp + 256*i;
            for(k = 0; k < 0x0100; k + + )
            {
                c = (*p + + & 0x07f) * 2;
                r = fputc(c, fp);
                if(r = = -1)
                    clrerr(fp);
                fputc(c, fp);  /* repeat pixels horizontally */
                if(r = = -1)
                    clrerr(fp);
            }
        }
}

pic_frame(fp)       /*** write 512*512 image data to ***/
FILE *fp;           /*** 256*256 frame store ***/
{
    unsigned int i, k;
    char *p, c;

    for(i = 0; i < 0x0200; i + + )          /*** 512 lines ***/
    {
        if(i%2)                 /*** get even lines only ***/
        {
            printf("%X \n", i/2);   /*** print line number ***/
            p = fsp + 256*(i/2);
            for(k = 0; k < 0x0200; k + + ) /*** 512 pixels each line ***/
            {
                c = fgetc(fp);
                if(c = = -1)
                    clrerr(fp);     /* clear error if there is */
                if(k%2)             /* get even pixels only */
                    *p + + = c / 2;
            }
        }
        else                        /*** skip odd lines ***/
            for(k = 0; k < 0x0200; k + + )
              if(fgetc(fp) = = -1)
                  clrerr(fp);
    }
}

delay()                 /*** delay routine ***/
{
    int i, j;

    for(i = 0; i < 0x0100; i + + )
        for(j = 0; j < 0x0100; j + + );
}

write_area(x,y,xs,ys,db) /*** write pixels to an area ***/
unsigned int x,y,xs,ys;
char *db;
{
    unsigned int i, j;
    char *p;

    for(i = 0; i < ys; i + + )
```

```
    {
        p = fsp + x + 256*(y+i);
        for(j=0; j<xs; j++)
            *p++ = *db++;
    }
}

fill_area(x,y,xs,ys,gl)  /*** fill an area with a value ***/
unsigned int x,y,xs,ys;
char gl;
{
    unsigned int i, j;
    char *p;

    for(i=0; i<ys; i++)
    {
        p = fsp + x + 256*(y+i);
        for(j=0; j<xs; j++)
            *p++ = gl;
    }
}

prn_area(x, y, xs, ys)   /* print values of pixels in an area */
unsigned int x, y, xs, ys;
{
    unsigned int i, j;
    char *p;

    for(i=0; i<ys; i++)
    {
        p = fsp + x + 256*(y+i);
        for(j=0; j<xs; j++)
            printf("%d, ", *p++ & 0x07f);
    }
}

copy_area(x,y,xs,ys,fs,fd) /* copy an area in source frame */
unsigned int x,y,xs,ys;    /* to template frame */
char *fs, *fd;
{
    unsigned int i, j;
    char *sp, *dp;

    for(i=0; i<ys; i++)
    {
        sp = fs + x + 256*(y+i);
        dp = fd + x + 256*(y+i);
        for(j=0; j<xs; j++)
            *dp++ = *sp++;
    }
}

track_hd(x, y, xs, lloc, rloc, ldf, rdf) /* search edge horizontally */
unsigned int x, y, xs;
int *lloc, *rloc, *ldf, *rdf;
{
    char c, *p;
    int i, uploc=0, dwloc=0, updif=0, dwdif=0;

    p = fsp + x + 256*y + 2;
              /* point to the 3rd element of the line */
    for(i=2; i<xs-2; i++)
    {
        c = *(p+2) + *(p+1) - *(p-2) - *(p-1);
        if(c > updif)
```

```
        {
            updif = c;
            uploc = i;
        }
        else if(c < dwdif)
        {
            dwdif = c;
            dwloc = i;
        }
        ++p;
    }
    if(uploc > dwloc)     /*** lloc: left, rloc: right ***/
    {
        *lloc = x + dwloc;
        *rloc = x + uploc;
        *ldf = dwdif;
        *rdf = updif;
    }
    else
    {
        *lloc = x + uploc;
        *rloc = x + dwloc;
        *ldf = updif;
        *rdf = dwdif;
    }
}

diff_bf(p, dp, size)     /*** differential computation ***/
char *p, *dp;
int size;
{
    int i;

    p += 2;
    for(i=2; i<size-2; i++)
    {
        *dp++ = *(p+2) + *(p+1) - *(p-2) - *(p-1);
        ++p;
    }
}

track_bf(p,size,ul,dl,udf,ddf) /* search edge with buffered pixels */
char *p;
int size,*ul,*dl,*udf,*ddf;
{
    char c;
    int i, uploc=0, dwloc=0, updif=0, dwdif=0;

    p = p+2;
    for(i=2; i<size-2; i++)
    {
        c = *(p+2) + *(p+1) - *(p-2) - *(p-1);
        if(updif < c)
        {
            updif = c;
            uploc = i;
        }
        else if(dwdif > c)
        {
            dwdif = c;
            dwloc = i;
        }
        ++p;
    }
    *ul = uploc;
```

```
    *dl = dwloc;
    *udf = updif;
    *ddf = dwdif;
}

long int cor_tem(temp)   /*** compute correlation ***/
struct template *temp;
{
    char *pix, *ptm, p, t;
    unsigned int x,y,xs,ys;
    int i,j,k;
    long int c=0;

    x = temp->x_loc;
    y = temp->y_loc;
    xs = temp->x_size;
    ys = temp->y_size;
    ptm = temp->data;
    pix = fsp + x + 256*y;     /* first line */
    for(i=0; i<ys; i++)
    {
        for(j=0; j<xs; j++)
        {
            p = *(pix + j) & 0X07f;
            t = *ptm++;
            if(p > t)
                k = p - t;
            else
                k = t - p;
            c += k;
        }
        pix += 256;          /* next line */
    }
    return(c);
}
```

```
/*****************************************
* The following is a collection of main programs
* for the vision system
*          by Mr. Xiao Qi Chen
*               May 1988
*****************************************/


/*****************************
* Edge finding program "VISN.C"
* ***************************/

#include "stdio.h"
#include "string.h"

#define TRUE    1
#define FALSE   0
#define MAXBOX  4096
#define BUFSIZE 128

char data[MAXBOX];
char buf[BUFSIZE];
char dfbuf[256];
FILE *fp;

main()
{
    int x_size = 0,y_size = 0,x_loc = 128,y_loc = 128;
    int left = 0, right = 0;
    int mxdf, mndf, ln, f;
    char c;

    x_loc  =  20;
    x_size  =  220;
    y_size  =  1;
    printf("\rselect frame  (0, 1 or default: 0): ");
    c  =  getche();              /* select a frame */
    switch(c)
    {
        case '0':
            f  =  0;
            break;
        case '1':
            f  =  1;
            break;
        default:
            f  =  0;
            break;
    }
    set_f(f);

    do
    {
        capdisp(f);
        printf("\rpress return to   capture frame              \r");
        c    =  getche();
        capture(f);
        y_loc    =  70;
        for(ln = 0;   ln < 2; ln + + )
        {
            draw_hl(x_loc,y_loc,x_size);
            track_hd(x_loc,y_loc,x_size,&left,&right,&mxdf,&mndf);
```

```
            draw_vl(left, y_loc-5, 10);
            draw_vl(right, y_loc-5, 10);
            y_loc + =     70;
        }
        printf("\rpress return to   continue or e to end...");
        c    = getche();
    }
    while(c !=  'e' && c !=  'E');
}


/*****************************
* Edge finding program "VIEW.C"
*****************************/

#include "stdio.h"
#include "string.h"

#define TRUE    1
#define FALSE   0
#define MAXBOX  4096
#define BUFSIZE 128

char data[MAXBOX];
char buf[BUFSIZE];
char dfbuf[256];
FILE *fp;

main()
{
    int x_size = 0,y_size = 0,x_loc = 128,y_loc = 128;
    int left = 0, right = 0;
    int ln, mxdf, mndf, f;
    char c;

    printf("\rselect frame(0, 1, default: 0): ");
    c = getche();
    switch(c)
    {
        case '0':
            f = 0;
            break;
        case '1':
            f = 1;
            break;
        default:
            f = 0;
            break;
    }
    set_f(f);
    capdisp(f);
    printf("\r\npress e to  end...");
    x_loc  = 20;
    x_size  = 220;
    y_size  = 1;

    do
    {
        y_loc    = 60;
        while(kbhit())
            c = getche();
        for(ln = 0;   ln < 2; ln + + )
        {
            track_hd(x_loc,y_loc,x_size,&left,&right,&mxdf,&mndf);
            cross(left,y_loc);
```

```
            cross(right,y_loc);
            y_loc + =    70;
        }
    }
    while(c != 'e' && c != 'E');
    capdisp(f);
}


/*********************************
* Template set-up program "TEMP.C"
*********************************/

#include "stdio.h"
#include "string.h"

#define TRUE   1
#define FALSE  0
#define MAXBOX  4096
#define BUFSIZE 128

char data[MAXBOX];
char buf[BUFSIZE];
char dfbuf[256];
char *mode1 = "a", *mode2 = "r", *sep = ",", *newl = "\n";
FILE *fp;

main()
{
    int x_loc = 128,y_loc = 128,x_size = 0,y_size = 0;

    long int i, box_size, corr_lim;
    char file[25],name[25],st[25];
    char *p, *fn, c;
    int f = 1;

    set_f(f);
    printf("\renter file name : ");
    p  = gets(buf);
    fn  = file;
    if (p != NULL)
        while(*fn + + =    *p + + );
    else
        printf("\r\nbad filename");
    do
    {
        do
        {
            printf("\r\nrun video");
            capdisp(f);
            printf("\rpress return to capture frame");
            c   = getche();
               capture(f);
            printf("\r\nrepeat y/n :    ");
            c = getche();
        }  while(c = = 'y' || c = = 'Y');

        df_box(&x_loc,&y_loc,&x_size,&y_size);
        read_area(x_loc,y_loc,x_size,y_size,data);
        box_size = x_size   * y_size;
        corr_lim = box_size;
        printf("\r\naccept template y/n :   ");
        c   = getche();
        if (c   = = 'y'|| c = = 'Y')
        {
```

```
        printf("\r\nenter template name : ");
        p = gets(buf);
        fn =    name;
        if (p !=    NULL)
        {
           if (*p != NULL)
           {
              while(*fn++ = *p++);
              if((fp  = fopen(file,mode1)) != NULL)
              {
                 fputs(name,fp);
                 fputs(newl,fp);

                 stci_d(st,x_size);
                 fputs(st,fp);
                 fputs(newl,fp);

                 stci_d(st,y_size);
                 fputs(st,fp);
                 fputs(newl,fp);

                 stci_d(st,x_loc);
                 fputs(st,fp);
                 fputs(newl,fp);

                 stci_d(st,y_loc);
                 fputs(st,fp);
                 fputs(newl,fp);

                 stcl_d(st,corr_lim);
                 fputs(st,fp);
                 fputs(newl,fp);

                 for( i=0;   i<box_size; i++)
                 {
                    stci_d(st,data[i]);
                    fputs(st,fp);
                    fputs(newl,fp);
                 }
                 if(fclose(fp))
                    puts("\r\nfile close    error");
              }
              else
                 printf("\r\nfile open   error");
           }
        }
        else
           printf("\r\nbad name");
     }
     printf("\r\n\nanother   template? y/n : ");
     c   = getche();
  }while(c != 'n' &&  c != 'N');
}


/***********************************************
* Tracking arc edge using template matching "TRAC.C"
***********************************************/

#include "stdio.h"
#include "string.h"
#include "dos.h"

#define TRUE    1
#define FALSE   0
```

```
#define MAXBOX  4096
#define BUFSIZE 128
#define NO_BOXES 20
#define SCRNWDTH 256

char boxdata[MAXBOX];
char buf[BUFSIZE];
struct template {
                int x_size;
                int y_size;
                int x_loc;
                int y_loc;
                long corr_lim;
                char *data;
            };
FILE *fp;
char *mode = "r";

main()
{
   long int cor_tem(), box_size, cmin, cor;
   char *fn, *p, *boxp = boxdata;
   char key, file[40], name[40];
   struct  template box[NO_BOXES], *pbox;
   int nbox=0, i,  tm, f;

   printf("\rselect frame(0, 1, default: 0): ");
   key = getche();
   switch(key)
   {
      case '0':
         f = 0;
         break;
      case '1':
         f = 1;
         break;
      default:
         f = 0;
         break;
   }
   set_f(f);
   printf("\r\nenter file  name: ");
   p = gets(buf);
   fn  = file;
   if  (p != NULL)
      while(*fn++ =   *p++);
   else
      printf("\r\nbad filename");
   do
   {
      printf("\r\nenter template name: ");
      p   = gets(buf);
      fn = name;
      if (*p != NULL)
      {
         if (p != NULL)
         {
            while(*fn++ = *p++);
            if((fp = fopen(file,mode)) != NULL)
            {
               do
               {
                  p   = fgets(buf,BUFSIZE,fp);
                  strip(p);
               } while(p != NULL && strcmp(name,p));
```

```
            if (p != NULL)
            {
               pbox = &box[nbox++];
               p   = fgets(buf,BUFSIZE,fp);
               stcd_i(p,&pbox->   x_size);
               p   = fgets(buf,BUFSIZE,fp);
               stcd_i(p,&pbox->   y_size);
               p   = fgets(buf,BUFSIZE,fp);
               stcd_i(p,&pbox->   x_loc);
               p   = fgets(buf,BUFSIZE,fp);
               stcd_i(p,&pbox->   y_loc);
               p   = fgets(buf,BUFSIZE,fp);
               stcd_l(p,&pbox->   corr_lim);
               box_size = pbox->x_size*pbox->y_size;
               pbox->data = boxp;
               for(i=0; i < box_size; i++)
               {
                  p = fgets(buf,BUFSIZE,fp);
                  stcd_i(p,boxp++);
               }
            }
            else
               printf("\r\ntarget template not found");
            if(fclose(fp))
               printf("\r\nfile close error");
         }
         else
            printf("\r\nfile open error");
      }
      else
         printf("\r\nbad name");
   }
   else
      break;
} while(nbox <  NO_BOXES);
capdisp(f);
printf("\r\npress e to  end\r");

do
{
   cmin = cor_tem(&box[0]);
   tm = 0;
   for(i=1; i<nbox; i++)
   {
      cor = cor_tem(&box[i]);
      if(cor <   cmin)
      {
         cmin = cor;
         tm = i;
      }
   }
   pbox = &box[tm];
   draw_area(pbox->x_loc-1,pbox->y_loc-1,pbox->x_size+2,pbox->y_size+2);
   if(kbhit())
      key = getch();
}
while(key != 'e' && key !=  'E');
capdisp(f);
}
```