DESIGN AND REALIZATION OF A MICROCONTROLLER CONTROLLED LINE OVERCURRENT PROTECTION RELAY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY
ENGİN ÖZTÜRK

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 1998

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Tayfur ÖZTÜRK

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

Prof. Dr. Fatih CANATAN

Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Nevzat ÖZAY

Supervisor

Examining Comitee Members:

Prof. Dr. Nezih GÜVEN

Prof. Dr. Nevzat ÖZAY

Prof. Dr. Arif ERTAŞ

Prof.Dr. Ahmet RUMELİ

M.Sc. Abdullah NADAR

ABSTRACT

DESIGN AND REALIZATION OF A MICROCONTROLLER CONTROLLED LINE OVERCURRENT PROTECTION RELAY

Öztürk, Engin
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Nevzat ÖZAY

February 1998, 110 Pages

The overcurrent protection relays are the most frequently used type of relay in power system protection due to the fact that the major percentage of faults in the power system yield with fault currents beyond the normal operating condition current values. Because of their flexibility, ease of use and cost advantages, microcontroller based relay systems are being used more and more each day in power system protection. In this thesis study, a microcontroller controlled overcurrent protection relay, which employs several new features and also a serial communication interface is developed and tested.

Keywords: Overcurrent Protection Relay, Microcontroller Controlled Relay, Line Overcurrent Protection.

ÖZ

MİKRODENETLEYİCİ KULLANILARAK HAT AŞIRI AKIM RÖLESİ TASARIMI VE GERÇEKLEŞTİRİLMESİ

Öztürk, Engin
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Danışman : Prof. Dr. Nevzat ÖZAY

Şubat 1998, 110 Sayfa

Güç sistemlerinde meydana gelen arızaların çoğunluğu normal çalışma koşullarının üzerinde arıza akımları oluşturduğundan, bu sistemlerde en çok kullanılan koruma rölesi türü aşırı akım koruma rölesidir. Esnek olması, kullanım kolaylığı ve fiyat avantajları nedeniyle mikrodenetleyici tabanlı röle sistemleri gün geçtikçe daha yaygın kullanım alanı bulmaktadır. Bu tez çalışmasında mikrodenetleyici kullanılarak bazı yeni özelliklere ve seri iletişim arayüzüne sahip bir aşırı akım rölesi geliştirilmiş ve test edilmiştir.

Anahtar Kelimeler : Aşırı Akım Koruma Rölesi, Mikrodenetleyici Tabanlı Röle, Hat Aşırı Akım Koruması.

ACKNOWLEDGEMENTS

I would like to thank to my supervisor, Prof. Dr. Nevzat ÖZAY, for his precious guidance throughtout the preparation of this thesis.

I also would like to thank to Mr. Abdullah Nadar and Mr.Ayhan Büyüksemerci for their valuable help and supports.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Relays are compact analog, digital, and numerical devices that are connected throughout the power system to detect the intolerable or unwanted electrical conditions within an assigned area, called ***protective zone*** of the relay. These undesired conditions in the power system are called the *power system **faults***. Relays can be considered as an active assurance designed to maintain a high degree of service continuity and limit equipment damage.

To improve the security and reliability of the power system, the power system is divided into protective zones, each of which is controlled by a *switchgear* system containing of electronic and mechanic units such as protective relays, auxiliary relays, circuit breakers, and current transformers. The protective relays are the intelligent units of the power system. The principal duty of a protective relay is to gather information through the power system such as current, voltage, heat, frequency and detect for undesired conditions, and in case of a power system failure, make the decision to clear the fault or isolate the faulted section of the power system. Another function of the protective relay is to initiate signals (electrical, audible or visual) to the personal about the location and status of the faulted section. Speed of protection is an important aspect in order to limit the damage to the faulted equipment, reduce the possibility of fire and to reduce the hazards to the personnel.

The objectives of a relay design should be to find the optimum solution for fulfillment of the above considerations accurately, minimizing the cost with the usage of minimum number of components.

1.2 CLASSIFICATION OF RELAYS

Relays can be divided into six functional categories:

Protective Relays: Detect defective lines, defective apparatus, or other dangerous or intolerable conditions. These relays generally trip one or more circuit breakers, but may also be used to sound an alarm.

Monitoring Relays: Verify conditions on the power system or in the protection system. These relays include fault detectors, alarm units, channel monitoring relays, synchronism verification, and network phasing. Power system conditions that do not involve opening circuit breakers during faults can be monitored by verification relays.

Reclosing Relays: Establish a closing sequence for a circuit breaker following tripping by protective relays.

Regulating Relays: Are activated when an operating parameter deviates from the predetermined limits. Regulating relays function through supplementary equipment to restore the quantity to the prescribed limits.

Auxiliary Relays: Operate in response to the opening or closing of the operating circuit to supplement another relay or device. These include timers, contact multiplier relays, sealing units, isolation relays, lockout relays, closing relays, and trip relays.

Synchronizing Relays: Assure that proper conditions exist for interconnecting two sections of a power system [1].

In addition to the above classification, relays may also be classified by input, operating principle or structure, and performance characteristics. This classification can be seen in Table.1.1 [2].

1.3 PROTECTIVE RELAYING SYSTEMS AND THEIR DESIGN

A protective relay is a small system within itself. However, the term, *Protective Relaying System* includes not only the stand alone protective relay but also the auxiliary protective equipment used in company with the relay. Protective relays or systems are not required to function during normal power system operation, but must be immediately available to handle faulted system conditions and avoid serious outages or damage.

Table.1.1, Classification of Relays

INPUTS	OPERATING PRINCIPLE OR STRUCTURES	PERFORMANCE CHARACTERISTICS
Current	Current Balance	Distance
Voltage	Percentage	Directional Overcurrent
Power	Multirestraint	Inverse time
Pressure	Product	Definite time
Frequency	Solid State	Undervoltage
Temperature	Static	Overvoltage
Vibration	Microprocessor	
Flow	Electromechanical	
	Thermal	

In practice, a relay engineer must arrive at a compromise based on the four factors that influence any relay application:

Economics,

Available measures of fault or troubles,

Operating Practices,

Previous Experience.

Since it is simply not feasible to design a protective relaying system capable of handling any potential fault in the power system, compromises must be made. In general, only those problems that, according to past experience, are likely to occur, receive primary consideration.

A relay systems engineer must have in-depth understanding of the power system as a whole in order to propose reasonable relaying system solutions not only the by means of protection performance and quality but also ease of operation and maintenance. For this reason, the relay engineer must not only know the technology of abnormal, but also have a basic understanding of all the system components and their functions in the system. Frequent reviews of protective systems should be mandatory, since power systems grow and operating conditions change.

A complex relaying system may result from poor system design or the economic need to use fewer circuit breakers. Considerable savings may be realized by using fewer circuit breakers and a more complex relay system. Such systems usually involve design compromises requiring careful evaluation if acceptable protection is to be maintained. It should be recognized that the exercise of the very best relaying application principles can never compensate for the absence of a needed circuit breaker.

1.3.1 Design Criteria

Certain criteria must be taken into consideration when designing a protective relay. These criteria directly affect the quality of a protective relay as a final product. In all relay applications, the four design criteria explained below are common to any well designed and efficient protective system or system segment [3]. Since it is impractical to satisfy fully all these design criteria simultaneously, the necessary compromises must be evaluated on the basis of comparative risks.

Reliability: System reliability consists of two elements: *dependability* and *security*. Dependability is the certainty of correct operation in response to system trouble, whereas security is the ability of the system to avoid misoperation with or without faults. Unfortunately, these two aspects of reliability tend to counter one another; increasing security tends to decrease dependability and vice versa. In general, however, modern relaying systems are highly reliable and provide a practical compromise between security and dependability. Nowadays, digital relaying technology allows more precise and flexible relay systems design both in case of dependability and security.

Dependability can be checked easier in laboratory environment by simulation of possible faults in the power system. On the other hand, testing of security is much more difficult and complicated since the relay is exposed to almost infinite number of transients and other diffusing effects when connected to real system environment. The importance of security becomes evident when one considers the fact that a protective relay may perform its function (protective action) once in tens of years. In this case, the designer should also consider the aging of the relay equipment as one of the many number of facts. A secure system is usually the result of a good background in design, combined with extensive model power system or EMTP (electromagnetic transient program) testing, and can only be confirmed in the power system itself and its environment.

Speed: Speed of a relay is the response (operating) time of the relay to a prescribed fault condition. Although high speed is a desirable feature in most relaying applications, in some cases speed of response may become a real handicap because of the question :' Is the problem really such serious that a trip was required?'. Actually, time is an excellent criterion to distinguish between real and counterfeit trouble.

Pretransient and transient fault periods should be taken into account for timed operation in order to avoid false tripping. On the other hand, the term *high speed operation* is frequently used for *instantaneous* operation. Applied to a relay, high speed indicates that the operating time usually does not exceed three source cycles (60 ms for 50Hz base).

Performance vs. Economics: Relays having clearly defined zones of protection provide better selectivity, but generally cost more. High speed relays offer greater service continuity by reducing fault damage and hazards to personnel, but also have a higher initial cost. The higher performance and cost can not always be justified. Consequently, both low and high speed relays are used to protect power systems. Both types have high reliability records. Records on protective relay operations consistently show 99.5% and better relay performance [4] .

Simplicity: As in any other engineering discipline, simplicity of a protective relay system is always the hallmark of good design. The simplest relay system, however, is not always the most economical. As previously indicated, major economies may be possible with a complex relay system that uses a minimum number circuit breakers. Other factors being equal, simplicity of design improves system reliability-if only because there are fewer elements that can malfunction.

1.3.2 Factors Influencing Relay Performance

Relay performance is generally classed as [5]:

(1) Correct,

(2) No conclusion,

(3) Incorrect.

Incorrect operation may be either failure to trip or false tripping. The cause of incorrect operation may be :

(1) Poor application,

(2) Incorrect settings,

(3) Personnel error, or

(4) Equipment malfunction.

Equipment that can cause an incorrect operation includes current transformers, voltage transformers, breakers, cable and wiring, relays, channels, or station batteries. Incorrect tripping of CBs not associated with the trouble area is often as disastrous as a failure to trip. Hence, special care must be taken in both application and installation to ensure against this.

'No conclusion' is the last resort when no evidence is available for a correct or incorrect operation. Quite often this is a personnel involvement.

1.4 ELEMENTS OF POWER SYSTEM

A power system is an interconnected system that consists three principal subsystems: the generating subsystem, the transmission subsystem, and the distribution subsystem. The term '*interconnected*' indicates that the complete electrical power system compromises many generating subsystems, feeding dispersed loads through interconnected transmission lines. The basic elements of an electrical power system are :

- Generators,
- Power Transformers,
- Transmission lines,
- Motors and loads,
- Protective apparatus.

The electrical energy generated by generators is fed to the distribution subsystem through transmission lines. Generated voltage levels are transformed to transmission voltage levels, and then to distribution voltage levels by the use of Power Transformers.

Faults can be very destructive to power systems. A great deal of study, development of devices, and design of protection schemes have resulted in continual improvement in the prevention of damage to transmission lines and equipment and interruptions in generation following the occurence of a fault.

Faults caused by surges are usually of such short duration that any circuit breakers which may open will reclose automatically after a few cycles to restore normal operation. If arresters are not involved or faults are permanent the faulted sections of the system must be isolated to maintain normal operation of the rest of the system.

A typical protection scheme consists of three basic parts:

- Interposing current transformers,
- Relay Unit,
- Circuit Breakers,

Operation of circuit breakers is controlled by relays which sense the fault by observing the signals through the interposing current transformers. In the application of relays, *zones of protection* are specified to define the parts of the system for which various relays are responsible [1]. An important factor for the relay to make correct decisions especially in transient fault periods is to get the undistorted fault data through the interposing transformers. Some degree of saturation is introduced by the relay arising from the magnetic nature of the interposing transformers especially in assymmetrical fault conditions. A modern relay should carry out some corrective actions to compensate this saturation effects, which may drastically effect the performance of the protection scheme in severe fault conditions. Effects of current transformer saturation on relay performance and some techniques to compensate these effects will be discussed in detail in Chapter 2.

1.5 BRIEF HISTORICAL ACCOUNT OF RELAY EVOLUTION

Protective relays have been used in power systems since the earliest times the electrical power systems were established. These systems were fully electromechanical equipment and their operating principle was electromagnetic induction. They have been widely used in the power systems for many years because of their robustness, inexpensivity and immunity to harsh working environments. Electromechanical relays are still being used in power systems and are important parts of some power systems.

In the late 1950's, along with the recent advances in solid-state technology, first electronic relays were introduced. These relays used analog and digital circuits in combination to provide the required output characteristics similar to electromechanical relays. Although these relays had greater possibility of malfunction because of electromagnetic interference and harsh working conditions than electromechanical relays, because of their price advantage they have reached to extensive usage in power systems. Along with the recent advances in their design, these relays have still been providing excellent service in almost all power systems in the world.

In the last two decades, digital and microcomputer based relays have been introduced and because of their reliability, flexibility and ability to control more than one process at a time, they greatly replaced the solid-state relays. By the usage of the microcomputer-based relays, power companies had the advantage of not only protecting the system, but also getting on-line information from the power system, processing these data and logging some important conditions occurring on the power system by digital means. In interconnected power systems, which are being more popular day by day, the microcomputer-based relays became a cornerstone because of their communication capabilities which is essential in centralized control of power systems. Still one great disadvantage of these type of relays is their susceptibility to noise and environmental defects, but these disadvantages are nearly reduced to zero by the emerging technology in microcomputers. [2],[5],[6]

Evolution of relays Technology can be characterized as in Fig.1 :

Electromechanical Relays 1930s

↓ ↓

Solid-state (Electronic) Relays 1950s

↓ ↓

Microcomputer_Based (digital) relays 1980s

↓

Fig 1.1, Evolution of relays technology

1.6 SCOPE OF THE THESIS STUDY

The scope of this thesis study is to design and realize a multi purpose microcontroller based digital overcurrent protection relay. The relay should be designed such that it accommodates within itself, the features listed below with a minimum number of parts count and optimum circuit complexity. Features of the relay will be as follows:

- User selectable *Inverse Time* and *Definite Time* operation,
- User_configurable *current multiplier* and *time multiplier* settings,
- Usage of Hall Effect current transducer instead of measuring current transformer,
- RMS current measurement,
- Enhanced *User Interface* capabilities with display and keypad modules,
- Serial EEPROM for data logging,
- SCI (Serial Communication Interface)
- Compensation of the distortive effects of interposing current transformers on the current waveform, by using digital data processing techniques.

An overcurrent relay that accommodates the following properties within itself is expected to be a good tool for any relay application engineer for line and feeder protection and also the enhanced features will be useful for power system management.

Chapter 2 covers an overview of overcurrent relaying fundamentals. The effects of primary current transformers on current waveform will also be discussed in this chapter. In addition to these concepts, the usage of hall effect current transducers as measuring current transformers will be discussed.

Hardware design concept and techniques used to implement the hardware features of the relay are given in Chapter 3. In this chapter, information about the microcontroller architecture will also be given.

Software design strategy and algorithm of the software are given in Chapter 4. This chapter also includes some additional information about the software tools used.

In Chapter 5, testing strategy for the relay is given. Test results and the evaluation of the relay performance according to the operating time values measured can be seen in this chapter.

Conclusive remarks for the thesis work are discussed in conclusion part, Chapter 6. Suggestions for future development of the relay can also be found in this chapter.

CHAPTER 2

AN OVERVIEW OF LINE AND CIRCUIT PROTECTION

2.1 INTRODUCTION

Alternating current lines are commonly classified by function, which is related to voltage level. Although there are no utility wide standards, typical classifications are as follows [2]:

1. *Distribution (2.4 kV to 34.5kV) :* Circuits transmitting power to end users.
2. *Subtransmission (13..8 to 138 kV) :* Circuits transmitting power to distribution substations and bulk loads.
3. *Transmission (69 to 765 kV) :* Circuits transmitting power between major substations or interconnecting systems, and to wholesale outlets. Transmission lines are further divided into:
   (a) High-Voltage (HV) : 69 to 230kV
   (b) Extra-High-Voltage (EHV) : 345 to 765kV
   (c) Ultra-High-Voltage (UHV) : greater than 765kV

Most faults experienced in a power system occur on the lines connecting generating sources with usage points. As these circuits vary widely with their configurations, length, characteristic values and degree of importance, so their protection and techniques [3]. There are several protection techniques commonly used for line protection:

1. Instantaneous overcurrent,
2. Time overcurrent,
3. Directional instantaneous and/ or time overcurrent ,
4. Step time overcurrent,
5. Inverse time distance,
6. Zone distance,
7. Pilot relaying.

In the operation of power systems, records of faults and their causes should be kept and compiled. This is of considerable importance in the design and application of protective gear as well as being a factor in the design and arrangement of the power system itself. The results of such data are given in Tables 2.1 and 2.2 [2].

Table 2.1, Percentage of several faults occurring on transmission lines

Type of Fault	% of Total
Single phase to earth faults	83
Phase to phase faults	9
Phase to phase to earth faults	5
Three phase faults without earth	1.5
Three phase to earth faults	1.5

Table 2.2, Power system equipment failure statistics

Equipment	% of total faults
Overhead lines and cables	70
Power transformers	20
Generator/ transformers	6.5
Switchgear and busbars	3.5

From Table 2.2, it can be seen that the majority of faults occur on overhead lines and cables. Of these, the majority are due to lightning and are of transient nature [2]. The most designative occurrence of these types of faults are increase in line currents beyond allowable limits, whose impact tend to increase to disastrous values for the power system equipment as it gets closer to the faulted section of the power system.

As can be seen from Table 2.1, the most frequent types of faults throughout the line failures are the single phase to earth faults. Hence, the most common application area for overcurrent relaying are these type of faults. Most overcurrent relays are designed to operate for single phase faults, although, in some cases these may be combined to design two or three phase and earth relays.

2.2 OVERCURRENT RELAYING FUNDEMANTALS

Existence of faults can be detected by monitoring current magnitude, since,

a) Fault current is larger than load current, and

b) Zero-sequence current is nearly zero during balanced load conditions (and during 3Φ faults) and has a high value (depending on system earthing conditions and fault resistance) during faults involving ground [8].

Determination of fault location (or faulted feeder) is obtained by operating time between successive relays. Thus, an overcurrent relay should have an element which measures current magnitude, and another element which measures time. When the current in the system is above a pre-determined value (i.e., the set current of the relay 'I_{set}'), the overcurrent element initiates the timing element (of the timing circuit) which starts the time counting. When the time count reaches a pre-determined value, as given on the time-current characteristic of the relay, the relay operates. The output of the overcurrent relay is, like most other protective relays, a trip signal to control the circuit breakers, and an alarm signal to warn the operator.

Figure 2.1(b) shows the application of these principles to the system in Fig 2.1(a). Here, for a fault at F1, the fault current will start all of the relays of the system through which the fault current passes (i.e., Relays C1, B1 and A1). However, only

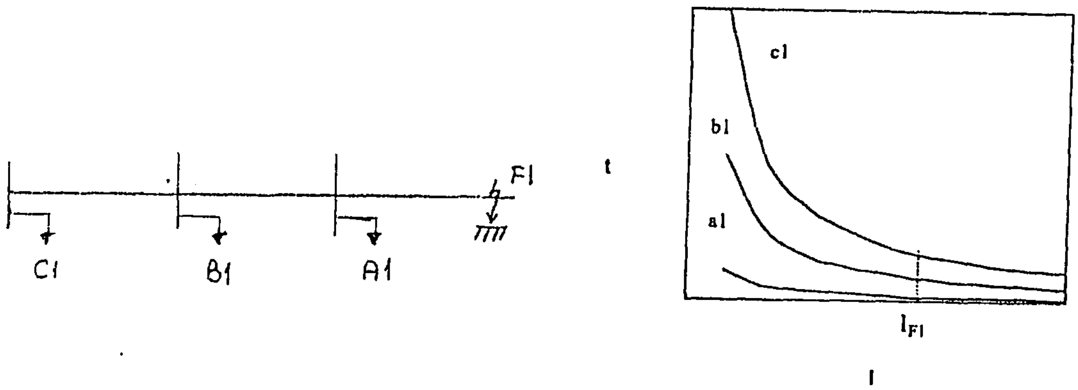


Fig 2.1, An application of overcurrent relaying on a sample system

the relay at C1 will operate since its operating time for this current value is less than that of the others.

This time discrimination between the operating times of relays connected in series on a power system is called the *selectivity of the protection system.* Relays should be well coordinated so that whenever a fault occurs anywhere in the power system, the nearest relay to the faulted area operates first and isolates the fault avoiding the other sections of the power system to be de-energized. When applying relay coordination, a relay engineer should take into account the operating time of the auxiliary circuit breaker to completely clear the fault.

2.3. PHASE AND GROUND OVERCURRENT PROTECTION

Time vs. current curve of an overcurrent relay is defined as a function defining the relationship between the operating time of the relay and relay current. Here current is the energizing quantity for the relay [9]. Each overcurrent relay has at least one time-current curve defining the operation of the relay for a specified range of fault current values.

Overcurrent relays are divided into two categories according to their time-current characteristics,

Definite-Time overcurrent relays: Operation time of these type of relays are constant for any value of fault current, which is a predefined multiple of relay set current value I_{set} .

Inverse-Time overcurrent relays: Operation time of these type of relays decay exponentially as the fault current value increases. Time-current curves of these type of relays are defined in various standards [9], [10], [11] and are called IDMT (Inverse Definite Minimum Time) curves.

2.3.1 Definite-Time Overcurrent Protection

In these type of relays, operating time is independent of the value of the fault current magnitude. The relay operates at a constant pre-determined time whenever it senses a current value exceeding the normal operating conditions. Operating times of these types of relays are usually user configurable.

2.3.2 Inverse-Time Overcurrent Protection

Time-current characteristics of inverse-time overcurrent relays are as seen in Fig 2.2, below.

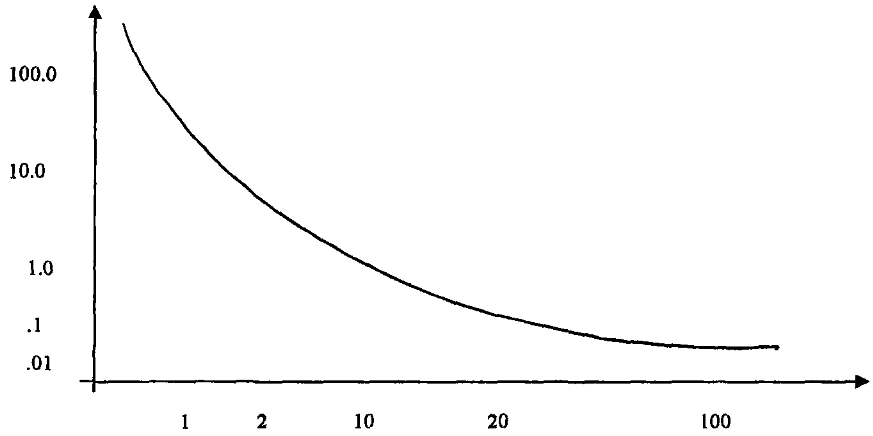


Fig 2.2, Time-current characteristic of an overcurrent relay

According to IEC standards [9], the most common formula for relays with inverse-time characteristics is defined as:

$$t = \frac{k}{\left[\frac{I}{I_{set}}\right]^{\alpha} - 1} \qquad (2.1)$$

where:

t = Theoretical operating time,

k = Constant characterizing the relay (time multiplier) ,

I = value of relay current,

I_{set} = Current set value (current tap),

α = Index characterizing the algebraic function,

These standard formulas were defined basically according to thermal withstand capabilities of the power system equipment installed on any power system, since a major duty of an overcurrent relay is to protect these equipment from thermal damage resulting from excessive fault currents. As a common practice, three different characteristics, all of which were derived from (2.1) by changing the characteristic indexes k and α were accepted as standard IDMT curves. These curves are named as:

Normal Inverse Characteristic (NI) : Characteristic equation is defined as:

$$t_{op} = \frac{0.14}{\left[\frac{I}{I_{set}}\right]^{0.02} - 1} \tag{2.2}$$

This characteristic is most suitable for systems where there is a large variation of fault current for different locations. The inverse characteristic enables improved utilization of the protected zone's overload capacity.

Very Inverse Characteristics (VI) : Characteristic equation is defined as:

$$t_{op} = \frac{13.5}{\left[\frac{I}{I_{set}}\right] - 1} \tag{2.3}$$

The operating time is strongly dependent on fault current magnitude, hence this characteristic is suitable for systems where there's a fairly large variation in fault current for different locations.

Extremely Inverse Characteristics (EI) : Characteristic equation is defined as:

$$t = \frac{80}{\left[\frac{I}{I_{set}}\right]^{2} - 1} \tag{2.4}$$

For cases where generation is practically constant and discrimination with low tripping times is difficult to obtain because of the low impedance per line section, an extremely inverse relay can be very useful. In these relays, only a small difference in current is necessary to obtain adequate time difference. This relay is also very desirable for protection against overheating by short circuit currents.

IDMT curves can be considered as a combination of definite-time and inverse-time curves. A relay having an IDMT type time-current curve will operate according to a timed operation curve as the above ones up to a threshold current value, but will act as a definite-time relay as this current value exceeded. This operation is frequently called '*instantaneous tripping*' and the current threshold value is called as '*highest current value*' **(I>>) .**

In all three equations given above, the term $\left[\frac{I}{I_{set}}\right]$ is defined as n*ormalized current* and in relaying applications, known as '*current tap'* . In ANSI IEEE Standards [10], [11] , two new curves were introduced, namely 'Long-time inverse' and 'Moderately inverse' that are also derived from equation (2.1).

From the above definitions, it can be determined that all of the standard curves are derived using the formula stated in (2.1), and one has the possibility to derive almost infinite number of different curves by changing the value of indexes k and α. A relay giving the possibility of defining a number of special curves to a relay application engineer would be a good tool for overcoming discrimination problems especially for instandard cases and complicated applications. This thesis study does not concentrate on equipping the relay with the ability of recognizing a number of different time-current characteristics defined by the user by redefining the parameters characterizing the time-current curve. Indeed, the time-current curve recognized by the relay is limited with the standard NI curve defined in IEC-255. But as the relay hardware accommodates a serial EEPROM, it has the flexibility to alter the look up tables within the EEPROM according to the new curves defined by the user. This type of renovation can be proposed as a future practice.

2.4 SOME CONSIDERATIONS ON RELAY CURRENT TRANSFORMERS

The asymmetrical component of power system fault current sometimes causes some degree of saturation of current transformers supplying input current to protective relaying schemes. After saturation occurs, the CT output will show evidence of distortion and the performance of the protection scheme can be affected.

The extent to which the protection scheme is affected by distorted input current due to CT saturation is not presented quantitatively. However, the effect of distorted current is discussed generally according to relay design classes.

After the primary fault current ceases, the CT can produce unidirectional decaying output current and can have a high level of flux trapped in the core, both of which can affect the performance of protection schemes.

2.4.1 Transient Response of Interposing Current Transformers

In modern HV and EHV systems more and more emphasis is placed on the fidelity of current transformers in protective systems because of:

i- the increase in magnitude of fault current,

ii- the increase in the time constant of the asymmetrical component of fault current,

iii- the need for faster clearing of faults, and

iv- the increasingly stringent protection reliability requirements.

These factors make it essential to reduce the operating time of the protective systems, and to make relay decisions not only in the steady-state fault conditions, but also in the transient periods of the fault currents. The operating current component supplied to the relaying circuits should never be distorted sufficiently to cause slow, false, or nonoperation of the protective system. In spite of that, if there exists still some distortive effects in the current supplied to the protective relays, it is the strength of the protective relay to sense these effects and to carry out some compensating algorithms to supply correct operation despite these effects. The major cause of concern in this study is the exponentially decaying dc component of primary fault conditions [7].

Because of its ferromagnetic character, a CT core may also retain an unknown amount of flux. This residual flux may be due to various causes, for example, severe offset fault currents, geomagnetically induced dc currents in the power system, or improper use of dc continuity test methods on secondary circuits. The residual flux will either improve or worsen the transient response of the CT because it may either oppose or aid the build-up of core flux caused by the dc offset fault current .

Present standards for relaying current transformers specify accuracy only for steady-state, power-frequency currents and are not suitable for defining performance under transient conditions. As a result of this, two relays on the market, both of which suit the steady-state fault operation standards, may show completely different performance for transient fault current values.

A relay designer should be able to get some parameters defining the transient performance of a primary current transformer to implement a method of compensating these effects. This method should be as simple as possible, but still effective and applicable to various number of different CTs used in power system protection [7].

2.4.2 Current Transformer Performance During Faults

Distortion of the output current begins whenever conditions are such that the core flux density enters the region of saturation. The factors influencing the core-flux density are the physical parameters of the CT, the magnitude, duration, and waveform of the primary currents and the nature of the secondary burden. The two causes of saturation in CTs are the excessive symmetrical fault current magnitudes and lower magnitude asymmetrical fault currents containing significant dc component. The latter is more significant in determining the transient performance of CTs [7].

When a fully offset fault current of the form shown in Fig 2.3 is impressed on the primary of a current transformer, the dc offset will, in general, cause a rise in flux in the core several times greater than that required to transform the 50 Hz component of current. Fig 2.4 shows the rise in flux in a CT core when a fully offset current is applied to a CT with a purely resistive burden. In this diagram the quantity ϕ_{ac} plotted on the abscissa represents the flux required to reproduce the 50 Hz component of fault current while the quantity ϕ_{tc} represents the flux required to reproduce the transient (dc) component of current. The variation in the transient component of flux will be a function of both the primary circuit dc time constant and the time constant of the CT secondary circuit.

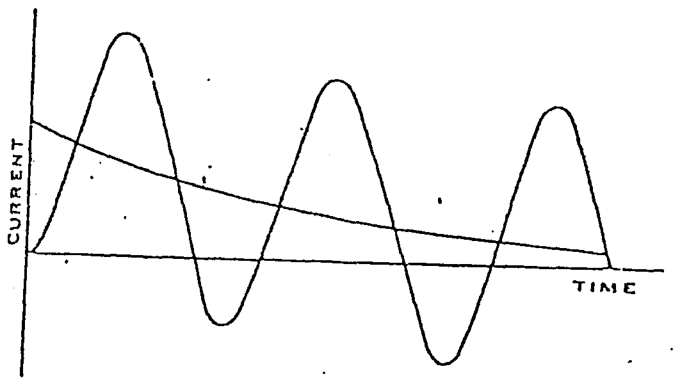


Fig 2.3, Fully offset fault current

The total flux, ϕ, required to reproduce the offset current is considerably greater than that required to reproduce a symmetrical current.

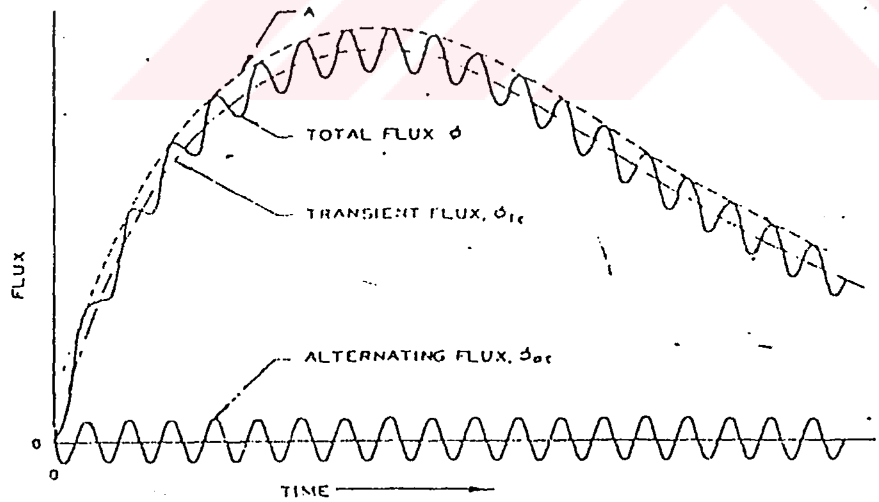


Fig 2.4, Rise of flux in the core of a current transformer

In the above illustration, the CT will reproduce the offset fault current completely as long as the induction does not reach the *saturation flux density* level. In modern CTs, this saturation flux density is between 1.9 and 2.0 teslas (20 kilogausses) [7]. If the induction greatly exceeds saturation flux density, the CT may produce a severely distorted output current similarly to that shown in Fig 2.5 (a). Note that the burden of this CT is purely resistive. As can be seen, the CT begins to saturate in the first cycle. The effects of adding inductance to the burden is seen in Fig 2.5 (b).

Time-to-saturation curve of a current transformer is the function representing the time interval that starts with the onset of primary fault current and ends when the core induction first reaches saturation flux density [7].

2.4.3 CURRENT TRANSFORMER TRANSIENT CAPABILITY

The transient performance of a CT is a function of a number of parameters which differ appreciably in each CT application. While it is possible to determine analytically the transient capability of a CT (in terms of time-to-saturation) for each operating condition, a series of generalized curves has been developed that provide this capability directly for any CT design and over a wide range of operating conditions. The derivation method for time-to-saturation curves of current transformers assumes the following conditions,

i- A current transformer with no secondary leakage reactance.

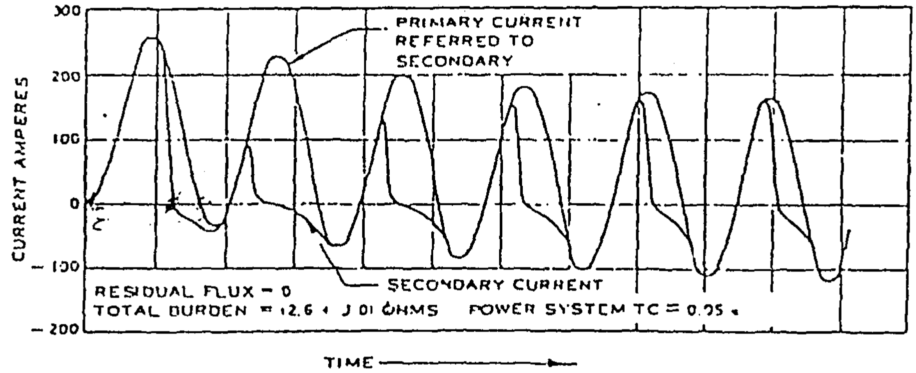
ii- A CT with a one-turn primary winding; the effects of the return conductor are ignored.

iii- Fully-offset primary current-- worst case and rare.

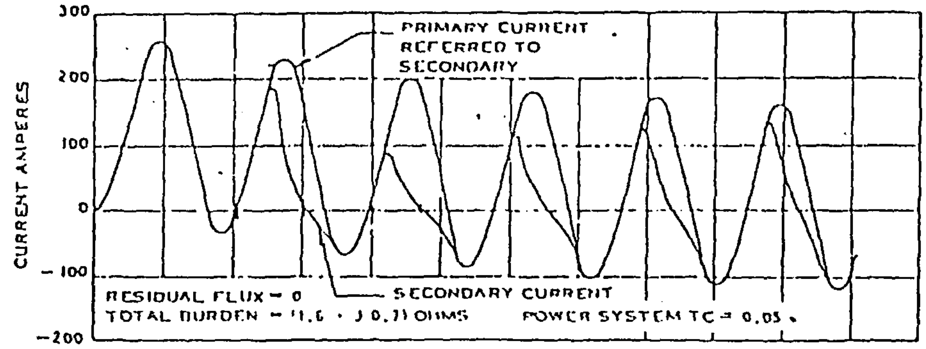
iv- Resistive burden.

The relationship between the CT and power system parameters, and the time-to-saturation is stated as the following equation:

$$K_s = \frac{V_x N_2}{I_1 R_2} = \frac{\omega T_1 T_2}{T_2 - T_1}\left(\varepsilon^{\frac{-t}{T_2}} - \varepsilon^{\frac{-t}{T_1}}\right) + 1 \qquad (2.5)$$



(a)



(b)

Fig 2.5(a),(b), Distortion in secondary current due to saturation

where,

V_x = saturation voltage-rms,

N_2 = number of turns in CT's secondary winding,

I_1 = alternating current in the CT primary and circuit,

R_2 = resistance of total CT secondary burden,

T1,T2 = Time constant of the transient component of power-system fault current, and CT secondary time constant respectively,

This basic expression is used to develop the time-to-saturation curves for CTs, an example of which is given in Fig 2.6. The ordinates of these curves are given in terms of saturation factor, K_s, while the abscissas are given as time-to-saturation in milliseconds for a 50-Hz power system. For specific values of V_x, N_2, I_1, R_2, these curves give the time to reach saturation flux density as a function of CT and system time constants. Conversely, the required CT and system parameters may be obtained for a specific time-to-saturation value.

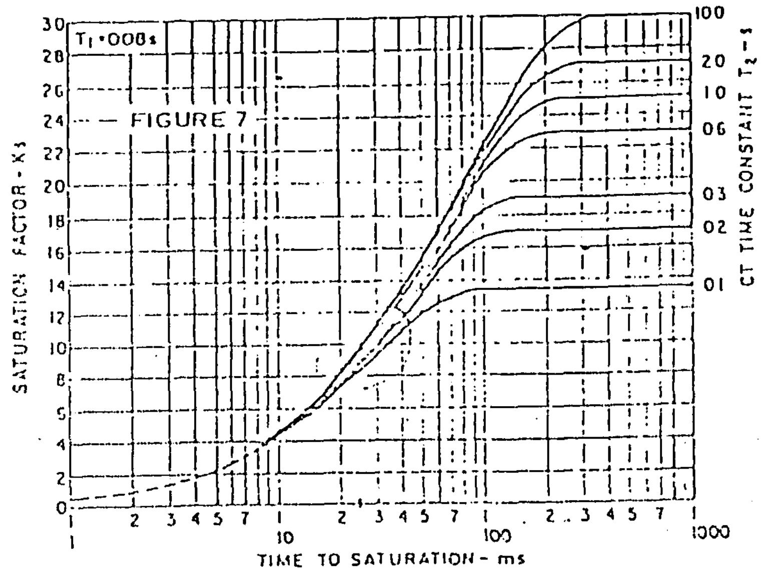


Fig 2.6, An example time-to-saturation curve

The procedure for applying time-to-saturation curves is as follows:

(1) First calculate the CT saturation factor, K_s, from (2.5). The saturation voltage, V_x is graphically determined from the standard CT excitation curve as in Fig 2.7.

(2) Determine the power system time constant, T_1 and the CT secondary time constant, T_2 . The power system time constant T1 can be calculated using the parameters, X_1, R_1 and ω:

$$T_1 = \frac{X_1}{R_1\omega} \qquad (2.6)$$

where,

X_1 = power-system short-circuit loop reactance,

R_1 = power-system short-circuit loop resistance,

The secondary time constant, T2 can be calculated as follows:

$$T_2 = \frac{V_e}{I_e R_2 \omega} \qquad (2.7)$$

where V_e and I_e are the voltage and current respectively at a suitable point on the standard secondary excitation curve in Fig 2.7. Equation (2.7) neglects the burden inductance, L_2, which is generally negligible compared to the mutual inductance, M_2.

(3) a-) Select an appropriate time-to-saturation curve from the standard curves corresponding to the power system time constant, T_1 calculated above.

b-) Time-to-saturation is obtained by locating the intersection of the ordinate corresponding to the value of K_s and the appropriate curve corresponding to the calculated value of the secondary time constant, T_2.

If the horizontal line representing K_s is above the selected curve, the CT will not saturate for this application [7]. Burden inductance, partially offset fault currents, air gaps and CT connections also effect the time-to-saturation value of the CT .

2.4.4 Effects of Distorted CT output on Protective Relay Performance

The preceding section describes the methods for designing a CT that does not saturate for a faulted condition within the power system, in order not to affect the protective equipment, and is more important for a CT designer. Possible effects of CT waveform distortion for protection schemes are:

(a) Prime aspects:

__ zero_crossing shift,

__ peak reduction,

__ rms value reduction,

__ harmonic content,

(b) Secondary aspects:

__ loss of security (undesired relay operation),

__ leak in dependability (failure to operate or operation with excessive delay),

At that point, it should be stated that the degree of the above effects varies with the design of a protective relay.

2.4.5 Discussion : Techniques for compensating the effects of distorted current in a digital overcurrent relay

The relay designed in this study is, as stated in Section 1.6, a digital overcurrent relay that measures the RMS value of line current by getting a definite number of samples from the line current signal transduced by a Hall Effect current transducer. The relay also accommodates a zero-crossing detector block assigned to the current sensing circuitry. The measurement technique for the RMS value of the current is as follows:

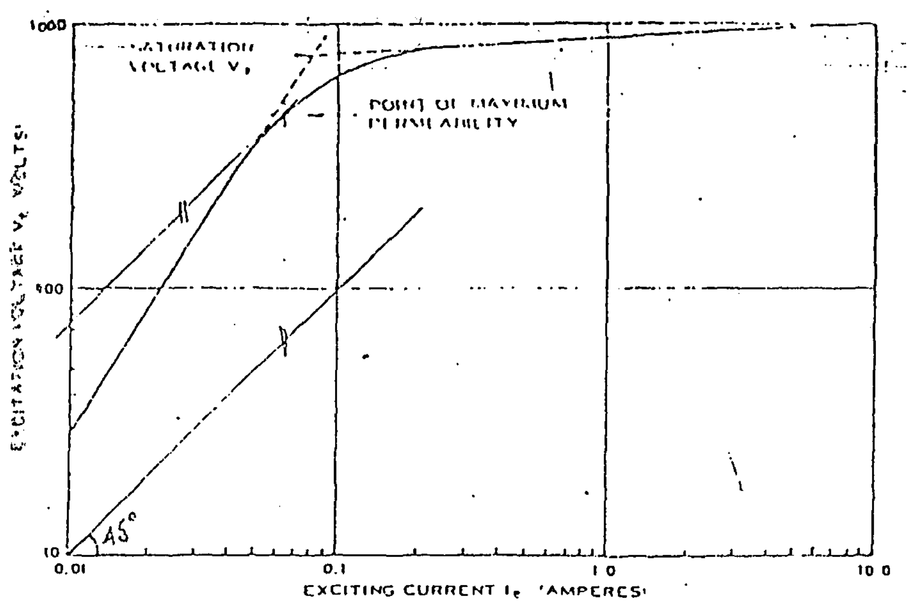


Fig 2.7 Secondary excitation curve ,(230kV current transformer,RATIO: 1200-5 Amperes, FREQUENCY: 60Hz)

(a) The relay gets 20 samples from the line current in a single supply cycle and convert the samples to digital domain by the use of an 8_bit AD converter which is embedded within the microcontroller. For a source frequency of f_s = 50 Hz (i.e., T_s = 20 msec), the sampling time of the relay will be,

$$T_{ad} = \frac{20}{20} = 1m\sec. \qquad (2.8)$$

(b) The RMS value of the line current is calculated by taking the squares of each sampled current value and getting the square-root of the mean of this value (So called **Root-Mean-Square** method). So the measured value of the fault current will be:

$$I_f = \sqrt{\frac{\sum_{n=1}^{20} i_f^2[n]}{n}} \qquad (2.9)$$

where n denotes the number of samples taken in a single supply period.

The most significant effect of a distorted primary CT output to the relay current measuring unit is the phase shift in the zero-crossing detection circuit, which effects especially the negative-cycle zero crossing instant. In Fig 2.8, the zero-crossing waveforms for the pure sinusoidal current waveform and fully-offset current waveform can be seen.

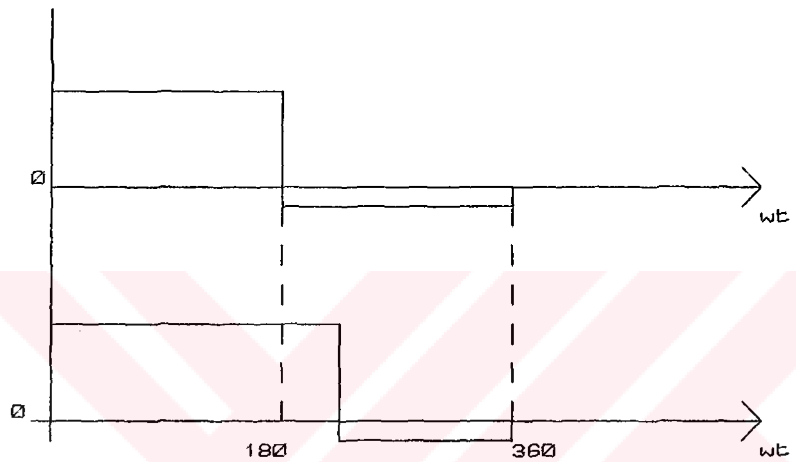


Fig 2.8, Zero_crossing waveforms

Since there's no significant change in the power frequency due to a localized fault at any point in the power system, the period of a single source cycle is more or less constant (20 ms for a 50 Hz power system). As can be seen from the above figure, the fully offset fault current does not effect the whole period of the zero crossing circuitry output, and the RMS current measurement gets the basis of a complete source cycle, the zero-crossing shift is not expected to make destructive effects on RMS current measuring block of the relay. As a precaution for zero-crossing effects due to CT distortion, the current samples can be taken using a free-running oscillator on time basis.

Peak reduction in relay current due to CT distortion, resulting in RMS value reduction, is a more harmful effect for the transient performance of the relay, since the operating time of the relay is tightly dependent on the RMS value of the fault current. To overcome this problem, some software techniques are used which will be described in detail in Chapter 4.

Several techniques can be applied for compensating the distortive effects of CTs in a protective relay. Some of these techniques will be discussed below :

a) Real time monitoring of the slope of V_x vs. I_e curve which can be seen in Fig 2.7 to determine whether the CT is operating in linear region or it entered the region of saturation. However, this seems to be almost impossible because of the fact that one can hardly be able to achieve data of the excitation current, I_e of the CT involved.

b) Calculation of time-to-saturation for a particular CT and for predetermined load conditions. This technique is also not very practical in use with a real-time operating relay system because of the difficulty of calculating the time-to-saturation of any transformer for different load and secondary time constant conditions. This technique does not seem to be feasible because of the difficulty to apply to a digitally controlled relay system since the system is expected to perform calculations compromising too much mathematical complexity.

c) Determining the instant of saturation by comparing the samples received from line voltage and current using the A/D converter with a sine wave look-up table previously loaded on the relay's EPROM memory. This could be a useful method for determining CT saturation caused by symmetrical fault currents, but for asymmetrical fault currents compromising unknown amount of dc offset fault current, it may be difficult to obtain reliable results.

d) Determining the CT saturation by observing the duty cycles of zero crossing waveforms received from fault current. Implementation of this method is the easiest among the four methods and is more applicable to offset fault currents, however the handicap is, especially for lower degrees of saturation, it may be impossible to strictly determine whether the CT has entered the region of saturation or not.

From the above arguments, it can be concluded that using the method described in (c) in conjunction with the method described in (d) increases the possibility of determining the CT's saturation for both the symmetrical and asymmetrical fault current conditions.

Since the linearity of the Hall Effect current transducer is guaranteed over the specified current range, it is not necessary to worry much about the saturation of the measuring equipment. No precautions will be taken for the harmonic content of the current waveform due to distortions from the CT, since a harmonic filter used to filter out these effects may eventually filter out the harmonic content (especially the 2^{nd} harmonic content) of the actual fault current, which must be taken into account in overcurrent protection [8].

2.5 USAGE OF HALL EFFECT CURRENT TRANSDUCERS AS MEASURING CTS

Potential-free measurement of a.c. and pulsating currents is normally accomplished by means of current transformers. These instruments are unable to detect d.c. components of the currents to be measured. In the past, so called '*Kramer Transformers*' have chiefly been employed for this purpose, but these are increasingly being superseded by current measuring modules incorporating '*Hall probes*'. These components enable currents with d.c. frequencies up to 100kHz and more with electrical isolation between the primary and secondary windings to be measured very accurately with excellent linearity over the operating range [13].

2.5.1. Operating Principles

Refer to Fig 2.9 showing a method of measuring the magnetic flux density inside a core gap, the core is a ring presenting a window, and the width of the gap is *l:*

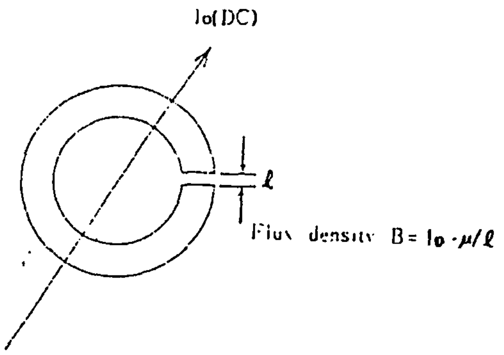


Fig 2.9, Measuring the flux density inside a core gap

The current to be measured, I_o, generates a magnetic field, H, and thus a magnetic flux, ϕ, in the gapped magnetic circuit around the live conductor. This magnetic flux flows through a Hall probe located in the air gap of the magnetic circuit. The Hall voltage, thus generated is amplified to produce an output voltage, which is directly proportional to the current to be measured.

The magnetic flux density, B, in the gap is determined by three factors:

$$B = \left\langle \frac{\mu_0}{l} \right\rangle I_0 \text{ (tesla)} \qquad (2.10)$$

The value of flux density (B) is roughly proportional to the line current, I_0. This proportionality exists until the magnetic path becomes saturated. Since the saturation level of the ferromagnetic material is high compared to CTs, an almost linear operation is obtained over the full current range specified [13].

The servo-assisted Hall Effect Sensor is particularly suited to high accuracy measurement of line currents constituting a wide DC-to-AC range. Its operating principle is as illustrated in Fig 2.10.

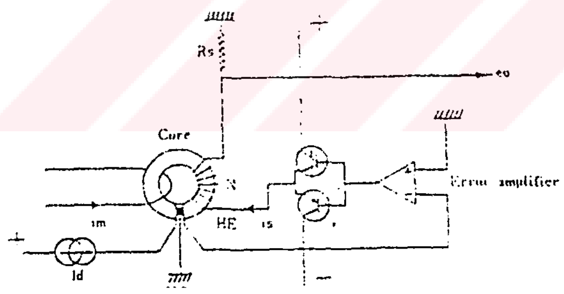


Fig 2.10, Servo-assisted Hall Current Sensor

2.5.2 Comparison with Measurement CTs

The table below shows some comparisons between measurement Cts and hall type current transducers.

Table 2.3, Hall Effect Transducers vs. Measuring CTs.

Current Transformers	Hall-type sensors
Lower levels of saturation due to magnetic properties of core material	Higher levels of saturation
Frequency dependent output current characteristics	Ability to measure currents from dc frequency level up to several kHz
Low isolation between primary and secondary	Higher isolation level
Excessive peak voltages when secondary eventually open circuited	Electronic level output voltages
Current output	Voltage output by the use of amplification circuit
No external supply needed	Auxiliary DC supply required

In this design,the LEM LA-55P module is used for current sensing. The electrical characteristics of the current sensor is given in Chapter 3.

CHAPTER 3

DESIGN FEATURES AND HARDWARE OF THE RELAY

3.1 GENERAL CONSIDERATIONS

The purpose of this thesis is to design and implement a multifunction digital overcurrent relay with programmable definite and inverse time operation and serial communication interface. The relay is implemented as a single-phase overcurrent protection relay. In designing such kind of relay, the following criteria are considered:

a) *Programmability*: A flexible multifunction relay that can be programmed for definite time and inverse time characteristics with user-selectable current multiplier and time multiplier settings.

b) *Reliability* : The ability of the relay to operate correctly when needed, and to avoid unnecessary operation should be satisfied simultaneously (i.e., dependability, security).

c) *Accuracy* : The error for current and trip time for the operative range should be less than the values specified.

d) *Economics* : The above criteria should be satisfied with the minimum cost and minimum number of parts count.

The most contemporary and practical method to satisfy the above criteria is to design an overcurrent relay controlled by an embedded microcontroller equipped with an appropriate software. Usage of a microcontroller greatly simplifies the flexibility concerns. Accuracy of the relay designed is greatly dependent on the quality of the current sensing circuitry. The transduced line current must be a true replica of the actual line current to be measured. Nonlinearity of the current transformers must be eliminated for measuring both the transient and steady-state of the fault current. To satisfy this criteria, the fault current is measured using the Hall Effect current transducer whose linearity is tested in laboratory and the measured voltage vs. input current characteristics may be seen in Fig 3.1.

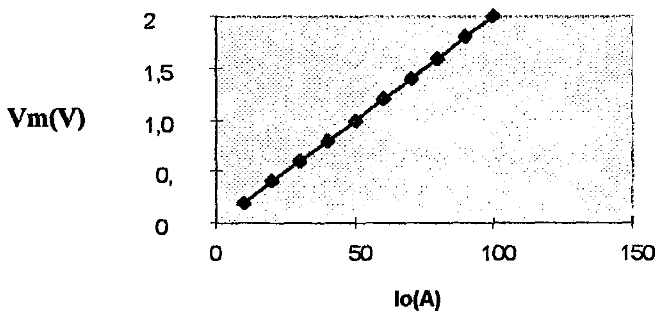


Fig 3.1, Current sensor characteristics

3.2 FEATURES OF THE OVERCURRENT RELAY

Basic features of the relay are:

- RMS current measuring,
- Selectable definite-time and inverse time operation modes,
- Increase of A/D converter accuracy by using variable gain control circuit in current sensing,
- Serial EEPROM for keeping fault data,
- Serial Communication interface to communicate with a Host PC by the use of an appropriate software program,
- Precautions for compensating the distortive effects of CT saturation.

While satisfying the above features, the main objective is to design the relay in such a way that it encounters minimum number of parts count and optimum design simplicity.

3.3 FUNCTIONAL BLOCK DIAGRAM OF THE RELAY

The relay hardware can be divided into seven main functional blocks. Each of these hardware blocks are integrated in such a way to reduce the complexity of the relay which yields in increasing the reliability of the system. These hardware blocks are:

1. *Current sensing block* : This is the input block of the relay. The current sensing unit of the relay consists of a Hall-Effect current transducer, zero-cross detecting circuit, variable gain amplification unit and a PWM unit to control the variable gain amplification circuit. These circuitry is used to transduce the current waveform to digital level within the conversion range of A/D converter.

2. *A/D converter block* : This is the block where the transduced current signals are digitized. It consists of a sample and hold circuit and 8-bit resolution A/D conversion block with a relative error of 1/2 LSb.

3. *Digital Control Block* : This block processes the digital data received from A/D conversion block, current measuring block and user interface block. Relay tripping decisions are made within this unit. This unit also initates relay tripping and signalling commands for the relay output block. These operations are implemented by the use of a single PIC16C74 8-bit microcontroller.

4. *User Interface Block* : This block is an interface between the user and the Digital Control Block. It consists of two 7-segment display modules, a 5-key keypad, test and reset buttons. Operative settings are entered into the relay by using the keypad and operative settings and status are monitored on the display module.

5. *Relay Output Block* : Consists of two auxiliary relays and two LED indicators initiated by the Digital Processing Block.

6. *Serial Communication Interface* : Is a standard RS-232 interface to communicate with a host PC by the use of an appropriate software.

7. *Power Supply Block* : This block is used to supply multi-level digital voltage to relay electronic circuits from the auxiliary d.c. supply.

The overall block diagram of the relay can be seen in Fig 3.2.

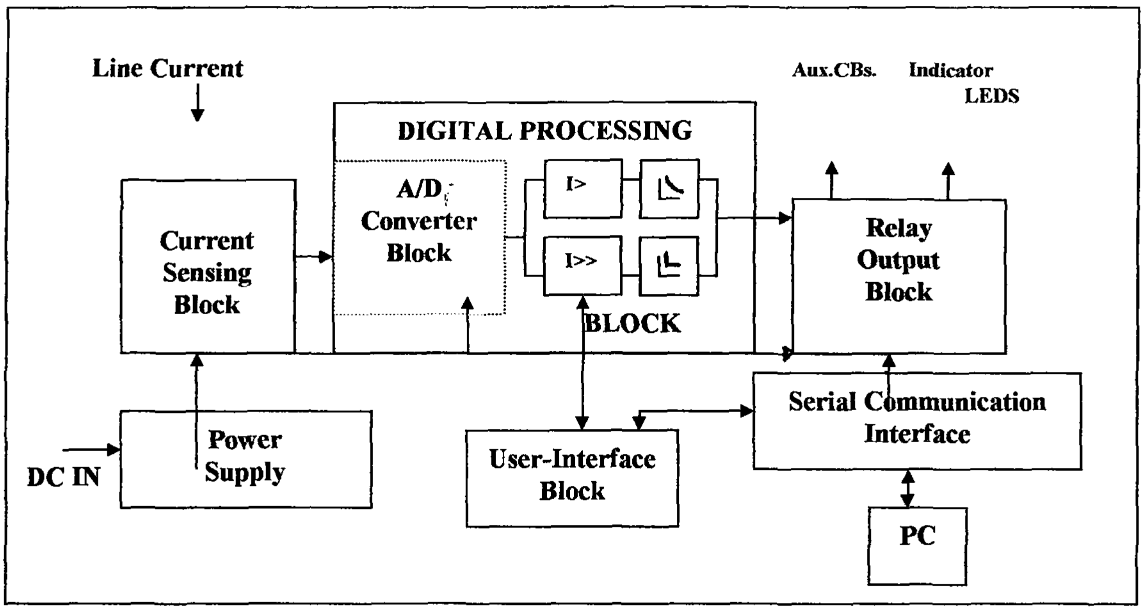


Fig 3.2, Relay Block Diagram

3.4 CURRENT SENSING BLOCK

The input circuitry of the relay designed consists of an analog current sensing block, which employs also the zero-crossing detector circuit. Line current is transduced to analog level by the use of a Hall Effect current transducer. Current data transduced is amplified by an amplifying stage, and input to the AD converter block of the relay. The current amplifying stage includes a variable gain control circuit in order to increase the accuracy of the A/D converter. Current sensing and amplification circuit can be seen in Fig 3.3.

3.4.1 Hall Effect Current Transducer

A Servo-Assisted Hall Effect current transducer module is used for current sensing instead of interposing current transformers. The electrical specifications for the Hall Effect transducer are given in Table 3.1 [14].

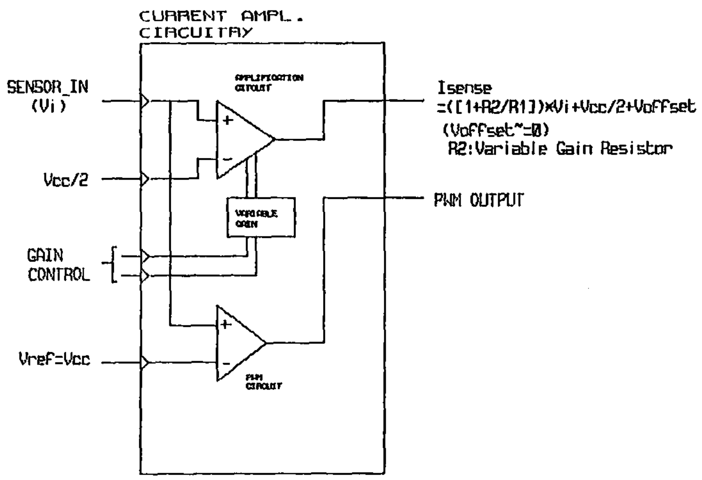


Fig 3.3, Current measurement and amplification circuit

As seen in Table 3.1, the current sensor has nearly zero offset voltage, which means the output of the sensor is an alternating current waveform. Some additional circuitry is required to settle sensed current to the conversion range of the AD converter block, which is chosen as 0 to 5V. The output voltage waveform of the current sensor for a sinusoidal input of 56.5V peak (maximum range) will be as seen in Fig 3.4.

In order to use the full range (0-5V) of the AD converter, the output waveform of the current transducer should be a sinusoid with a peak value V_p=2.5V for the full range which has an offset of 2.5V d.c.

The nominal current ,I_N , for the relay is chosen as 1Ar.m.s, which is a standard relay current value stated in IEC-255-4 [9]. Normally, the overcurrent relay settings are in the range of 0.6 to 2I_N for phase faults, with the corresponding instantaneous operating settings varying between 4 - 16I_{set}. For this relay, the setting range is declared as 0.5-2I_N with corresponding instantaneous operation settings 2-20I_{set}. So, the *accuracy limit factor*, which is defined as the ratio of the maximum primary current to the rated primary current, for this relay will be greater than 20x2=40.

Table 3.1, Current sensor electrical specifications

LEM LA-55P Current Sensing Module		Unit
Supply Volts	±12	V DC
Supply Current	≈20	MA max.
Sensed Current	55	A rms-max
Offset Current	±0.4	MA max
Sensitivity	±1%	of I_N
Response Time	< 1	μsec
Turns Ratio	N:1000	N='primary turns'
Frequency Range	$1\text{-}50*10^3$	Hz
Isolation Voltage	3kV	rms/50Hz/1min
Temperature Range	0-70	°C

The relay class index has chosen to be 5, which means that the overall error introduced by the current sensing circuitry for $20xI_N$ should be less than 5 percent [9], [10].

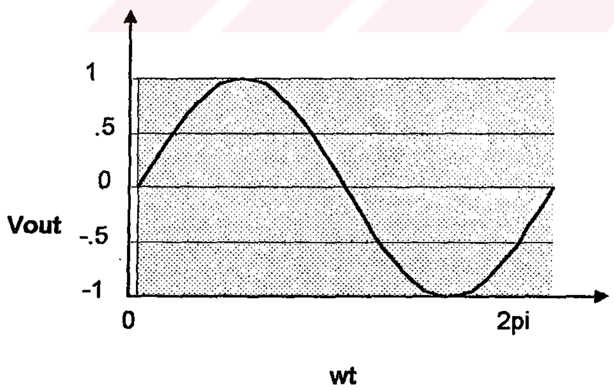


Fig 3.4, Current sensor output characteristics

3.4.2 Current Filtering and Amplification Unit

In current sensing, RMS measurement method is used. That is achieved by getting the Root Mean Square of the 20 current samples per each source cycle. Resolution of the A/D converter of the microcontroller used is 8-bits (255 counts for full scale) with 1/2 MSB relative error. The transduced current signal in Fig 3.4 is further amplified and a definite offset voltage of 2.5V dc is added on this signal by the use of the circuit in Fig 3.3.

For increasing the accuracy of the relay, a new technique is introduced in amplifying circuitry. The dynamic measurement range of the relay is $20xI_{set}$. Since an 8-bit ADC is used, the absolute error of the ADC will be equal to Least Significant Bit, which means $1/2^8$ for full scale. Assuming that we have the single measurement range which corresponds to $20xI_{set}$, then the relative reading error for this measurement range will be $100x1/2^8=0.39\%$ which seems acceptable, but for a measurement range of I_{set}, the relative reading error will be $100x20/2^8=7.84\%$, which is unacceptable.

To overcome this problem, the measurement range for this relay is divided into two regions, namely, $0\text{-}5I_{set}$, $5I_{set}\text{-}20I_{set}$. For each region, the full range of ADC reading corresponds to $5I_{set}$ and $20I_{set}$. Whenever the current value measured exceeds the threshold current value for that range, the PWM circuitry in Fig 3.3 generates an interrupt signal to the microcontroller and the software generates new control signals for increasing the gain of the op-amp amplifier circuit in Fig 3.3, by the use of an analog switch, CD 4066. When the gain applied to the current signal is increased, the measurement for the upper range is applied. The maximum reading errors for each range of signals can be calculated as follows:

$$0\text{-}5_{Iset}\text{ range}: \quad e_{max} = 100x\,\frac{5}{2^8} = 1.96\%, \qquad (3.1)$$

$$5I_{set}\text{-}20I_{set}\text{ range}: \quad e_{max} = 100x\,\frac{\frac{20}{5}}{2^8} = 1.56\%, \qquad (3.2)$$

The maximum relative error of these ranges define the maximum relative reading error of the relay and will be 1.96%, which is an acceptable range of current error for a class 5 relay. That means the reading error of the relay for any of these ranges will not exceed 5 percent. The technique applied for achieving this requirement can be called as *variable gain control*, and will be discussed in the following section.

No additional low-pass filtering is introduced within the measurement circuitry, in order to avoid the 2nd harmonic content of the fault current to be filtered out [3], which may be a designating factor especially for unbalanced fault currents.

3.4.3 Variable Gain Control Circuit

Variable gain for current measurement will be achieved by controlling the value of gain resistors in the amplifying section by the use of a digitally-controlled analog switch, CD 4066. The philosophy for the variable gain control may be understood easily by examining Fig 3.5. As seen in Fig 3.5, each time, one of the resistances that determine the gain applied to the current signal is activated. When changing the gain of the input signal, the dc bias value (2.5V dc) is not to be amplified.

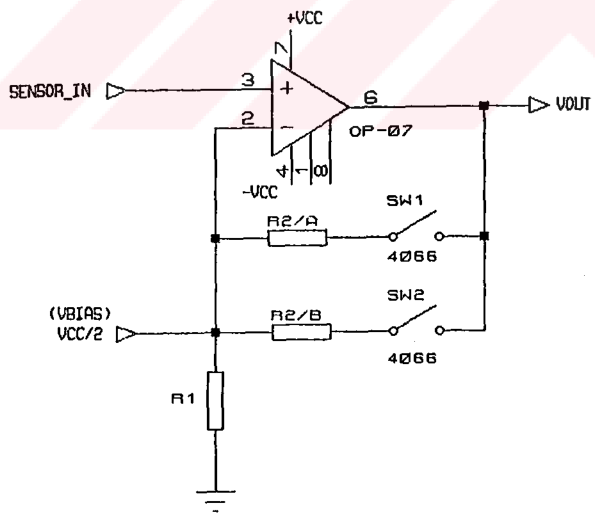


Fig 3.5, Variable Gain Control cct.

3.4.4 Zero-Crossing Detector Circuit

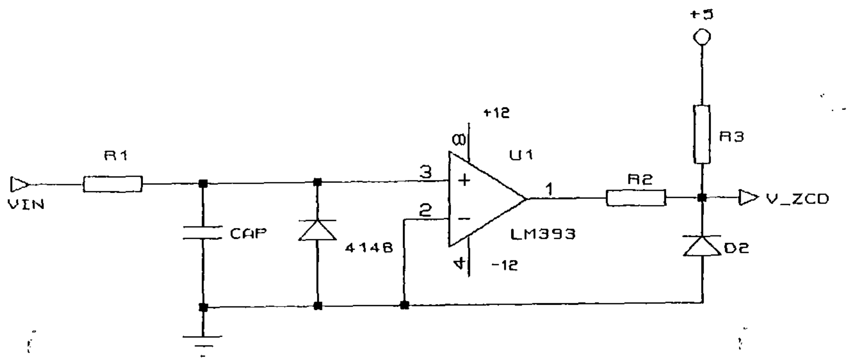
A high-speed voltage comparator, LM393, is used in the zero-crossing detector blocks of the analog circuit, whose schematic diagram and voltage vaweforms can be seen in Fig 3.6 (a), (b). The function of this circuit is to constitute a square wave output for a sinusoidal input with a logical '**1**' value corresponding to the positive half-cycles of the sinusoidal current waveform, and a logical '**0**' value corresponding to the negative half-cycles of the sinusoidal current waveform. The rising edges of the square wave will define the starting and end of one sampling period for the current. Since the current is sampled over a full supply cycle range, the distortions due to the CT saturation will not cause a significant effect on the sampling period as discussed in Section 2.4.6. Hence any precaution for sine wave distortion will not be taken into account for the zero-cross detection circuit. The rising and falling edges of the zero-crossing signals will be used as a timing interrupt for the microcontroller.

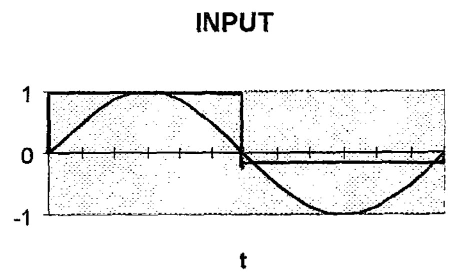
3.5 ANALOG-TO-DIGITAL CONVERSION BLOCK

Analog-to-Digital conversion block is the functional relay block where the transduced current samples are digitized. It consists of a sample and hold circuit and a 8-bit A/D conversion block. These two blocks are actually embedded within the microcontroller in the Digital Control Block. Hence, the A/D Conversion Block can be considered as a part of the Digital Control Block.

The conversion channel 'AN0' of the PIC16C74 is used for A/D conversion. The channel is programmed to employ the supply voltage, V_{cc} as external reference. Fig 3.7 shows the A/D converter block diagram. Only one of the eight conversion channels is used each time and the channel to be used is selected by the channel select bits CHS2.. CHS0 by the software. Reference voltage V_{ref} is selected as V_{cc} .

Analog input model for each conversion channel is seen in Fig 3.8. Prior to every conversion, the sampling switch 'SS' is closed and the charge hold capacitor 'C_{HOLD}' is charged through the internal limiting resistance 'R_{IC}' up to the analog input voltage level.





(b)

Fig 3.6, Schematic diagram and waveforms of zero-crossing detector cct.

Then the sampling switch 'SS' is opened by setting the 'GO/DONE' bit in the software and the conversion starts. End of A/D conversion can be realized by the automatic releasing of this bit by the hardware, either by software polling of this bit or the A/D Conversion Complete 'ADIF' interrupt. In this design software polling of the 'GO/DONE' bit is used to sense the A/D conversion complete. When the 'GO/DONE' bit is automatically cleared, the 8-bit digital data is present in the 'ADRES' register [15].

Current measurements are performed by applying RMS (Root Mean Square) technique. When the first rising edge of phase zero-crossing signal is detected, the microcontroller begins digitizing analog current signal to 8-bit digital data. Twenty samples are taken from line current signal in each cycle, relinquishing appropriate time delay between each sample. Sampling is performed until the reception of next rising edge of zero-crossing signal. At that time the digitized current data are stored in microcontroller RAM. These data contain,

$$I[i], \quad i = 1 \ldots 20, \qquad (3.3)$$

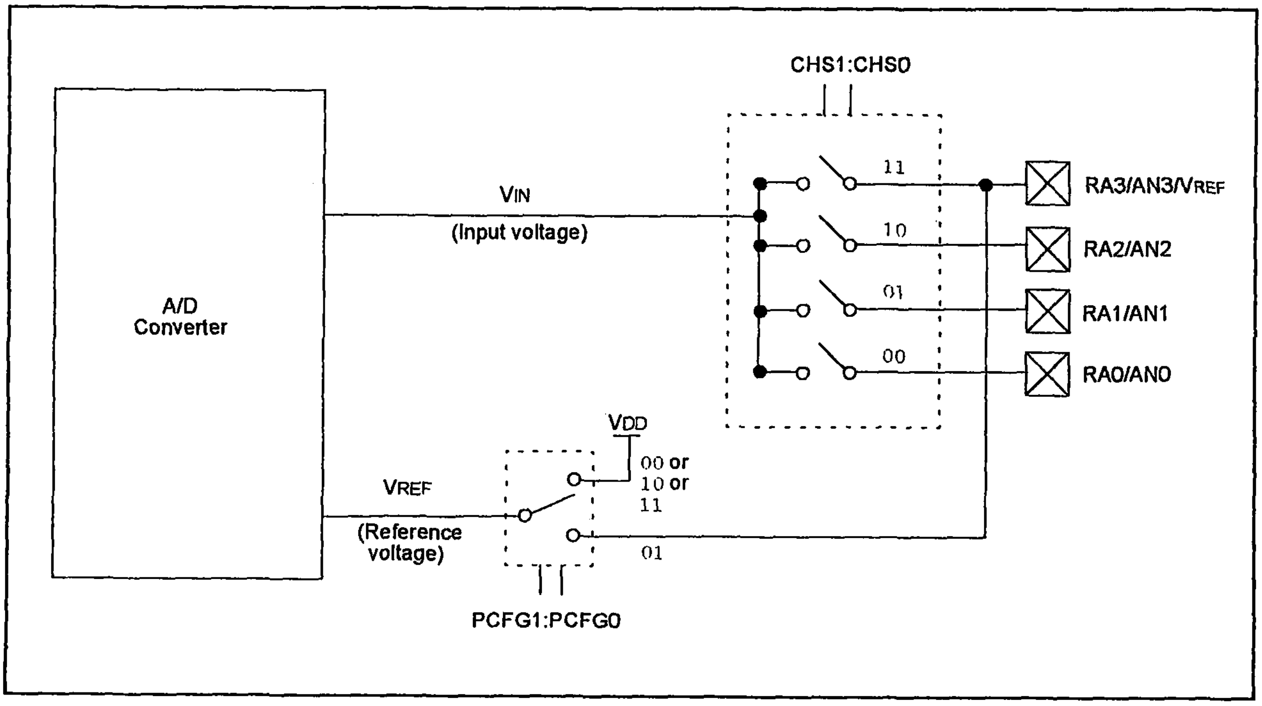


Fig 3.7, PIC16C74 A/D Converter block diagram

After the sampling period, the RMS value for current is calculated from the below equation by using the data in (3.3). The software methods to calculate the equations below will be described in Chapter 4.

$$I_{RMS} = \sqrt{\frac{\sum_{i=1}^{20} I^2[i]}{20}} \qquad (3.4)$$

The RMS value calculated is used within software algorithm to decide the relay tripping instant.

3.6 DIGITAL CONTROL BLOCK

The Digital Control Block processes the digital data received from A/D conversion block, current measuring block and user interface block. Relay tripping decisions are

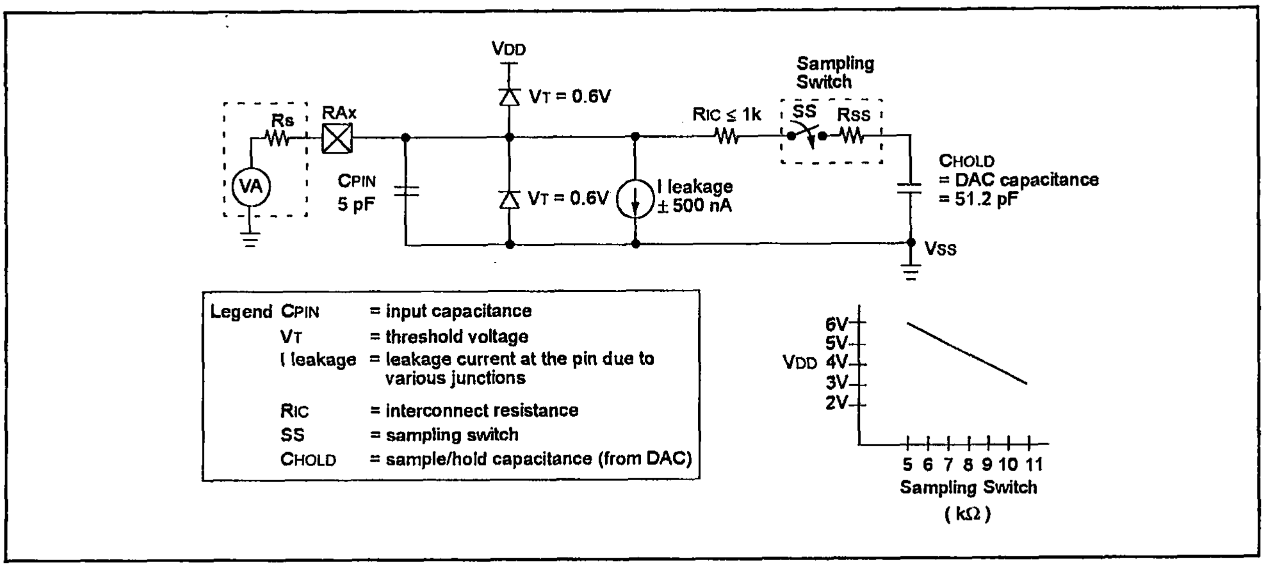


Fig 3.8, Analog Input Model

This block consists of a PIC16C74 microcontroller (high-performance 8-bit microcontroller with 8-bit A/D converter) to manage the activities described above with an appropriate software. The software performs the following functions:

a) Utilization of digitized current data to perform relay decision algorithm,
b) Carry out proofreader algorithms to compensate the distortive effects of primary current transformers,
c) Processing of relay operative settings entered by the user from the keypad module,
d) Driving the 7-segment led display and two LED indicators on the front panel,
e) Initiating relay trip signals as a result of relay tripping decision algorithm,

3.6.1 Information about the Microcontroller Architecture

The PIC16C74 is a member of the Microchip Technology Inc. PIC 16/17 microcontroller family. All PIC 16/17 microcontrollers employ an advanced RISC (Reduced Instruction Set Core) architecture. The PIC16CXX microcontroller family has enhanced core features, eight level deep stack, and multiple internal and external interrupt sources [15]. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 35 assembler instructions are available. PIC16C74 Block Diagram can be seen in Fig 3.9.

As seen in Fig 3.9, PIC16C74 has 192 bytes of RAM and 33 I/O pins. In addition, several peripheral features are available including: three timer/counters, two Capture/ Compare/ PWM modules and two serial ports. The Synchronous Serial Port can be configured as either a 3-wire Serial Peripheral Interface(SPI) or two-wire Inter-Integrated Circuit (I^2C) bus. The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as the Serial Communications Interface or SCI. An 8-bit Parallel Slave Port is provided. Also an 8-channel high-speed A/D is provided. The 8-bit resolution is ideally suited for applications requiring low-cost analog interface. The PIC16C7X family has special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption.

PIC16C74 has the capability of running with external oscillators up to 20MHz. The internal clock prescaler divides the external clock oscillator frequency by 4 and the internal instruction cycle is 5Mhz for 20MHz oscillator. In this application, the PIC16C74 is run with a high-speed crystal oscillator of 16MHz, thus the internal execution cycle clock frequency will be 4MHz. This yields with an execution time of 250ns for single cycle instructions.

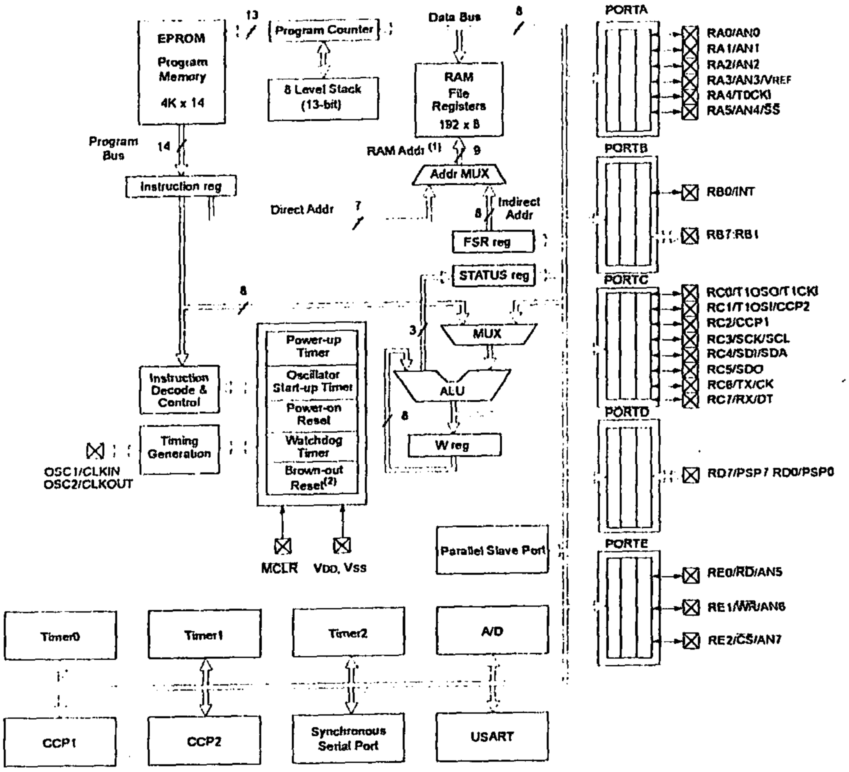


Fig 3.9, PIC16C74 Block Diagram

3.6.2 EEPROM Memory Block

The EEPROM memory block is part of the Digital Control Block and consists of a single 2K EEPROM memory chip 93LC56. The connection of EEPROM to the microcontroller is as seen in Fig 3.10.

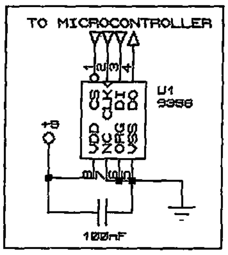


Fig 3.10, EEPROM Memory Block

The EEPROM memory block is used to keep the last 200 current value information. These values can be used to investigate the reason for relay tripping by the use of the Serial Communication Interface via a host PC with the use of an appropriate software. Data In 'DI' and Data Out 'DO' pins are used to communicate with the Serial EEPROM e.g. , read data from EEPROM and write data to EEPROM serially. Clock Input 'CLK' to the EEPROM is generated by the microcontroller.

3.6.3 Digital Processing Unit

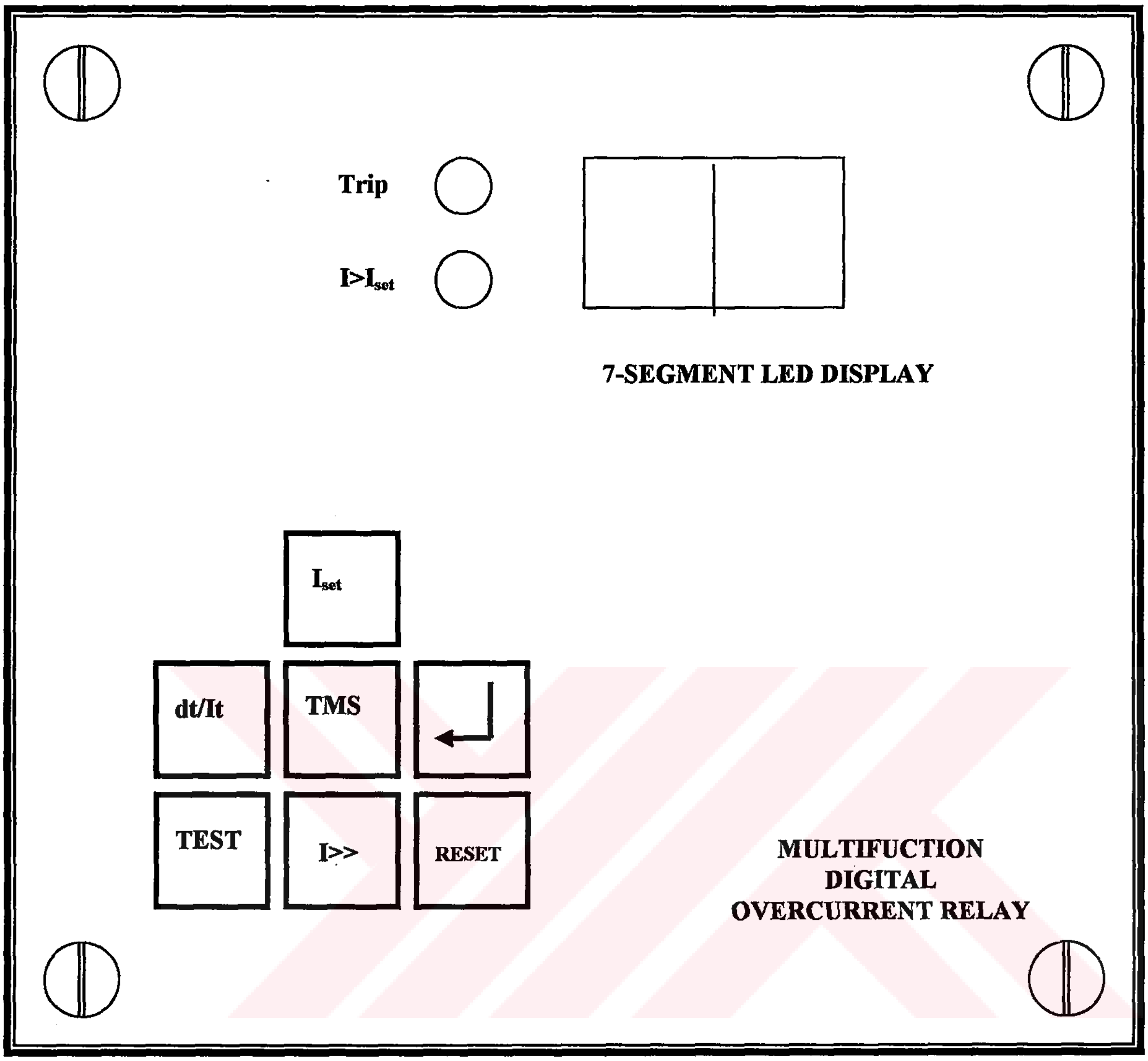
All relay decision algorithms, utilization of I/O ports and A/D conversion channels are performed by the Digital Processing Unit. This unit, with the associated software program, receives data from A/D converter, performs the arithmetical calculations to decide whether a relay tripping required based on these data, get relay operative settings and initiate control signals to peripheral units. Serial EEPROM and Serial Communication Channel procedures are also performed by this unit. All arithmetical calculations are performed by the Arithmetic Logic Unit (ALU), which can be seen in Fig 3.9.

3.7 USER INTERFACE BLOCK

The User Interface Block consists of 5 keypad buttons for entering the relay operative settings. These are:

- Definite time / Inverse time operation modes,
- Current Tap and Time Multiplier (TMS) settings for Definite time and Inverse time operations,
- Instantaneous operation settings,

Another unit within the User Interface block is the 7-segment display module. This module is used to display the operative setting values both in setting mode and operating modes of the relay. Also a software reset button is used to reset the processor



dt/It : Definite-Time/ Inverse-Time Selection

I_{set} : Current Tap Selection

TMS : Time Multiplier Selection

I>> : Hi-Set Multiplier Selection

↵ : Accept Selection

Fig 3.11

Relay front panel arrangement

in case of undesired operating conditions or for refreshing the operative settings. A test button , which is used for testing the functionality of the physical relay contacts are located on the front panel, and can be considered as part of the User Interface Block. The arrangement of the front panel of the relay is seen in Fig 3.11.

3.8 RELAY OUTPUT BLOCK

The Schematic diagram for Relay Output Block can be seen in Fig 3.12. It consists of two Finder 40.31 type single contact relays and two drivers circuit units for these relays. One of these relays is used for tripping the associated auxiliary circuit breaker, and the other one is preserved for remote signaling purposes. This block also contains two indicator LED displays, one of which is for instantaneous and other one for timed operation indication.

Whenever the relay experiences an abnormal condition in line current and decides to trip the relays, it initiates a trip signal to the relay driver circuits, hence trips both relays. Depending on the nature of the tripping action, the relay initiates visual indications (LEDs), either for instantaneous tripping, or for timed tripping.

3.9 SERIAL COMMUNICATION INTERFACE

Serial Communication Interface Block consists of a standard RS-232 interface module, a single AD232 chip connected to PIC16C74's RX and TX ports. For communicating with the host PC, the Universal Synchronous Asynchronous Receiver Transmitter (USART) feature of the microcontroller is used in Asynchronous mode. This hardware is capable of supporting data rates up to 9600 bauds (bits per second). The schematic representation of Serial Communication Interface can be seen in Fig 3.13.

Since the primary subject of interest of this thesis work is concentrated basically on the hardware and the software of the relay as a standalone unit, no additional study is carried out on the host PC interface software. In other words, the Serial Communication Interface Block exists on the relay as a hardware unit only. It can be the subject of another study to implement the Serial Communication protocols both on the relay microcontroller and host PC .

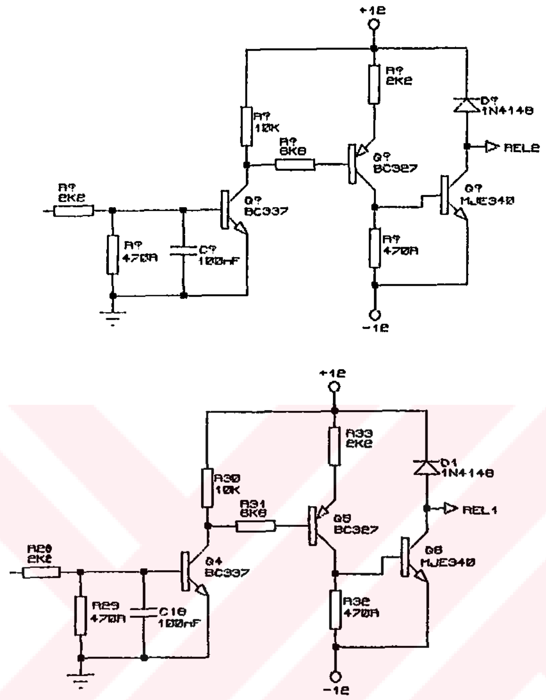


Fig 3.12, Relay Output Block

3.10 POWER SUPPLY BLOCK

Schematic representation of Power Supply Block can be seen in Fig 3.14. Function of this block is to supply multi-level digital voltage to relay electronic circuits from the Auxiliary DC supply (110V). The relay requires three supply voltage levels for operation, 5V, +12V and -12V. +12V and -12V are required for input block of the relay and the total current consumption of these circuits does not exceed 50mA. The total current required for supplying relay's digital circuits from the +5V supply is within the

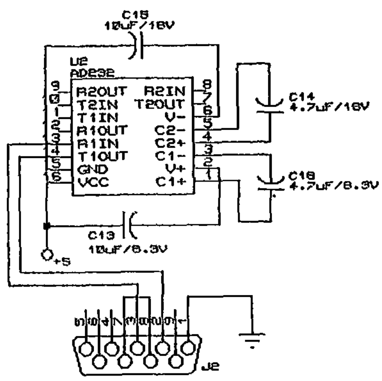


Fig 3.13, Serial Communication Interface

range of 100mA. Hence using the linear regulators 7812, 7912 and 7805 is a suitable choice for supplying voltage to the overall relay circuitry, considering that these regulators can supply up to 300mA of currents without significant degradation in their output voltage levels. Two zener diodes with forward voltages of 24 volts are used for obtaining regulator input voltages.

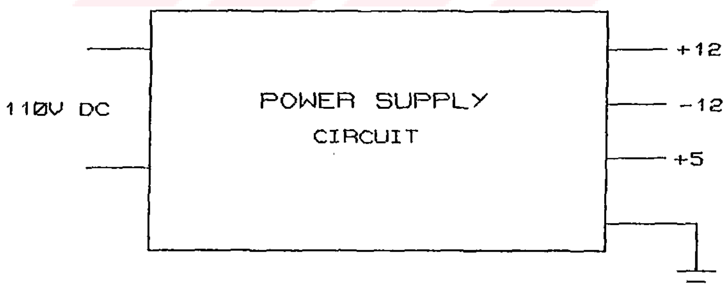


Fig 3.14, Relay Power Supply

3.11 RELAY SETTINGS

The relay's operative settings are entered by the user, --prior to putting the relay in service-- by the keypad on the front panel. These settings can be altered by the user after resetting the relay and before the operation of the relay initiates. There are three operating modes of the overcurrent relay. These are:

- Definite-time operation,
- Inverse-time operation,
- Instantaneous operation.

In definite-time operation, operation time of the overcurrent relay is constant for any value of fault current, which is a predefined multiple of relay set current value (I_{set}). The relay operates and trips in a pre-defined delay time whenever it encounters a fault current magnitude exceeding the relay set value, I_{set}. Instantaneous operating mode is associated with both of the two operating modes and will be attempted whenever the fault current value exceeds a pre-determined high set current value (I>>).

In Inverse-time operation, the relay operating time decays exponentially as the fault current value increases. Time-current curves of these type of relays are defined in various standards (IEC, ANSI IEEE) and are called IDMT (Inverse Definite Minimum Time) curves [9], [10], [11].

The following sections describe entering the relay settings and preparing the relay for operation. When entering the operative settings, the relay remains in idle mode, i.e., no current con version and protective action is carried out, only the operative settings are read through the keypad.

3.11.1 Definite Time – Inverse Time Selection

Following a power up or software reset using the '**RESET**' button, the two characters, '_ _' appears on the 7-segment displays which means that the user would press '**ENTER**' key once to start entering the relay operative settings. The procedure for Definite Time - Inverse Time selection is as follows:

1. Press '**ENTER**' key, and '`dt`' appears on 7-segment display.
2. Press '`dt/It`' button continuously, and the characters, '`dt`' and '`It`' scroll on the display, representing Definite Time and Inverse Time operating modes .
3. To select Definite Time operating mode, press '**ENTER**' key when '`dt`' prompts on the display.
4. To select Inverse Time operating mode, press '**ENTER**' key when '`It`' prompts on the display.

3.11.2 Current Multiplier Setting

Current multiplier setting range for the relay is defined as,

$$0.5*I_N < I_{set} < 2*I_N \qquad (3.5)$$

Since, the nominal relay current, $I_N = 1A$:

$$0.5A < I_{set} < 2A \qquad (3.6)$$

The relay current taps increment with 0.25 step intervals within the range described above. Hence, the relay current taps are, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0 (* I_{set}) . The current tap indexes, '`t1`'.. '`t7`' represents these current taps, accordingly. The procedure for Current Tap selection is as follows:

1. After the relay operating mode is selected, '`t1`' representing the 0.5* I_{set} tap is prompted on the display.
2. Press 'I_{set}' button continuously, and the indexes, '`t1`' to '`t7`' scroll on the screen, representing the current tap values listed above.
3. To select the desired current tap value, press '**ENTER**' when the index corresponding to that current tap prompts on the display.

3.11.3 Time Multiplier (TMS) Setting

The operating time, t_{OP} is defined by the general formula

$$t_{OP} = \frac{K}{\left[\frac{I}{I_{SET}}\right]^{\alpha} - 1} \qquad (3.7)$$

where, K denotes the time multiplier. The setting range for time multiplier for this relay is:

$$0.1 < \mathbf{K} < 1.0 \qquad (3.8)$$

Time Multiplier settings for this relay increment with 0.1 step intervals within the range specified above. Below is the procedure to be applied for entering Time Multiplier setting.

1. After the Current Tap value is selected, '`0.1`' prompts on the display, which represents the TMS value.
2. Press '**TMS**' button continuously, and the TMS setting values, '`0.1`' to '`1.0`' scroll on the display with 0.1 increments.
3. To select the desired TMS value, press '**ENTER**' when that value prompts on the display.

3.11.4 High Set (I>>) Multiplier Setting

The instantaneous current setting range for this relay is:

$$2^*I_{SET} < (I>>) < 20^*I_{SET} \qquad (3.9)$$

and the procedure to be applied for High Set Multiplier setting is as follows:

1. After the TMS setting, a '1' prompts on the display, representing the High Set current value (*I_{set}).
2. Press 'I>>' button continuously, and the Hi Set values '1' to '20' scroll on the display with '1' step increments.
3. Press 'ENTER' when the desired Hi Set value prompts on the display.

When relay settings are completed, the relay will prompt the current operative settings on the 7-segment display with 2 second intervals, and prompt 'Op.' on the display. Pressing 'ENTER' button will start relay operation.

CHAPTER 4

SOFTWARE OF THE RELAY

4.1 GENERAL DESCRIPTION

The software of the relay is the logical sequence of instructions loaded on the microcontroller's program memory. The microcontroller executes these instructions step by step in order the relay to perform its functions. The main functions performed by the microcontroller within the Digital Processing Block of the relay are:

- Receiving digitized data through A/D Conversion Block,
- Calculating RMS value of line current,
- Making relaying decisions according to measured line current value,
- Performing User Interface functions,
- Data logging to serial EEPROM,

The program instruction sequence consists of 14-bit assembler instructions specific for PIC16C74 microcontroller. These instructions are loaded onto the microcontroller's internal ROM as meaningful hexadecimal fragments called ***op-codes.*** The source code is generated by using MPC Code Development System (Version 1.02), which is a C-Code Compiler Software specific to PIC series microcontrollers. This software converts the code written in C code to a suitable hexadecimal code recognized by PIC16C74 microcontroller.

While generating the relay software, the following criteria were taken into account to generate an efficient source code:

1. *Code Efficiency,* is an important factor considering the program memory of the microcontroller is limited to 4Kbytes and there exists no external addressing capability.

2. *Time Efficiency,* is important since the execution time of software routines directly determine the speed of relay's protective actions. Since a Code Efficient

program block is generally not Time Efficient, an optimization should be made between Code Efficiency and Time Efficiency.

3. *Modularity* , is a chief aspect in software programming since it increases the strength of a software program by enhancing simplicity and rendering program modifiability. This criteria is taken into account when generating the source code and the source code is divided to independent functional software blocks each of which performs a particular function.

4.2 DATA STRUCTURES

The major data types processed by the relay software can be grouped into two categories:

- Variable data types,
- Constant data types.

Variable data types are the data types which are altered by the relay software in real-time execution of the source code. These data contain the inputs to the relay from the outside world --such as digitized current sample values, operative time and current multiplier setting values entered by the user--, data altered as result of arithmetic and logic operations performed by the relay software – such as RMS current value --, and relay output data –such as the data to be displayed on 7 segment LED displays, data to be exported to the serial EEPROM module--. Some major data types and their properties are as follows:

Digitized Current Sample Value: Is the 8-bit data obtained by digitization of line current signal through A/D Conversion Block. Twenty samples per each source cycle are used to perform RMS current calculation, so the current sample data are stored within the microcontroller RAM as an array of size 20, an array which is composed of 20 general purpose RAM registers, each containing 8-bit digitized current data.

RMS Current Value: Is the result of the arithmetic calculations performed to obtain the RMS value of line current. Stored in microcontroller RAM as 16-bit data.

Current Multiplier Setting Value: Is the scaling factor for relay set current. Entered to the relay through the keypad. Stored in microcontroller RAM as two 8-bit registers, one for dividend and other for denominator. (Example: $I_{SET}= 0.5*I_N =1/2*I_N$)

Time Multiplier Setting Value: Is the scaling factor for relay operating time. Entered to the relay through the keypad. Stored in microcontroller RAM as two 8-bit registers, one for dividend and other for denominator. (Example: K= 0.5 =1/2)

Instantaneous Current Multiplier Setting: Is the scaling factor for Instantaneous operation set current. Entered to the relay through the keypad. Stored in microcontroller RAM as 8-bit data.

Time Current Look-up Table: Composed of ROM arrays of size up to 256 bytes. Time-Current look up tables for the standard curve 'Normal Inverse (NI) which is defined in IEC 255-4 standards [9] were downloaded to microcontroller's program memory (ROM) with the relay source code.

There are other temporary variable data types used to store data within real time execution of the application software and for keeping these data, the general purpose registers of the microcontroller RAM are used.

4.3 SOFTWARE ALGORITHM

The relay software is composed of autonomous software blocks and a main program routine which controls data transfer and communication activities between software blocks and also controls the peripheral features of the microcontroller such as handling of timers, handling of interrupts and I/O port data transfer operations.

The relay software is such designed as particular tasks are implemented by autonomous software blocks with lowest interblock communications and data transfer to guarantee modularity of the software. All data transfer and control activities are performed by the main software loop.

The software program starts with a POWER-ON reset or a SOFTWARE reset applied on the **MCLR** pin (Pin1) of the microcontroller. After startup, the microcontroller is first initialized by the execution of ***init_init*** function. The tasks performed by the main program loop are:

- Reading operative settings from keypad module,
- Advancing 7-segment displays and indicator LEDs,
- Handling external and internal interrupt sources,
- Calling associated subroutines in sequential order which perform specific relaying functions,
- Data logging to serial EEPROM.

The main program loop performs above functions within an infinite loop using time division multiplexing. Time-current look-up tables are used for calculating the necessary trip time for any value of line current. The relay initiates current sampling when it encounters a rising edge on the PORTB.7 pin, representing a current zero-crossing signal. Current samples are digitized and stored on the microcontroller RAM until the reception of the next zero-crossing interrupt signal representing the end of the source cycle. At the beginning of the next source cycle, the RMS value is calculated using the digitized current data collected in the preceding cycle. When the RMS value for each supply cycle is calculated, the microcontroller checks whether the RMS current value exceeds the pre-determined Hi-Set current value (I>>) or not. If the RMS current value exceeds Hi-Set value, the main control loop initiates output signals to relay output block, thus tripping the output relays (Instantaneous Trip).

In the case that the RMS current value is smaller than I>> , but it still exceeds the set value, I_{set} , the relay appends a definite number, which is proportional to the calculated RMS current value, to a temporary counting register, and compares the value of this register with a constant threshold value. If the register value is greater or equal to the threshold value, the relay operates and the output relays are tripped (Timed Trip). The number to be added to the counting register is the content of the Normal Inverse look-up table register, whose index corresponds to the RMS current value calculated.

Whenever an RMS current value greater or equal to the relay pickup value (I_{set}) is encountered, the '**I> I_{set}**' led on the relay front panel lights until the reception of the first RMS current value which is less than that value. When the relay operates and initiates a trip signal, the '**Trip**' led on the front panel lights, representing a relay trip, and the cause of the relay trip is displayed on the 7-segment led display as `'tt'` (Timed Trip) or `'It'` (Instantaneous Trip). The relay must be reset in order to be re-started, following a trip.

When an RMS current value within the normal operating limits is measured after the relay counting is started, the relay stops counting, the cumulative value in the counting register is cleared and the '**I> I_{set}**' led on the front panel is turned off.

At the beginning of each source cycle, the RMS current value calculated for the preceding source cycle is logged on the serial EEPROM. Electrical Specifications of the relay and the explanation of the visual indicators on the front panel can be seen in Appendix B.

Zero-crossing of phase current signal, and PWM (current amplifier overrange indicator) signals are detected using external interrupt sources, INT and RBI. These interrupt sources are handled by the ***portb_interrupt*** service routine. When a rising edge of the PWM signal is encuontered, the microcontroller initiates the output signals to change the value of gain resistor in the variable gain circuit.

4.3.1 Program Flow

As stated in the preceding section, the software performs its functions within an infinite loop after the initialization routine is executed. The most general program flowchart can be seen in Fig 4.1.

A rising edge of the current signal indicates the zero crossings of line current waveform. A rising edge in PWM signal indicates that the current signal applied to ADC pin has exceeded the A/D Converter measuring range.

End of any A/D conversion is detected by software polling of the GO/DONE bit in the ADCON0 register. At that instant, the 8-bit digitized current data is present on the

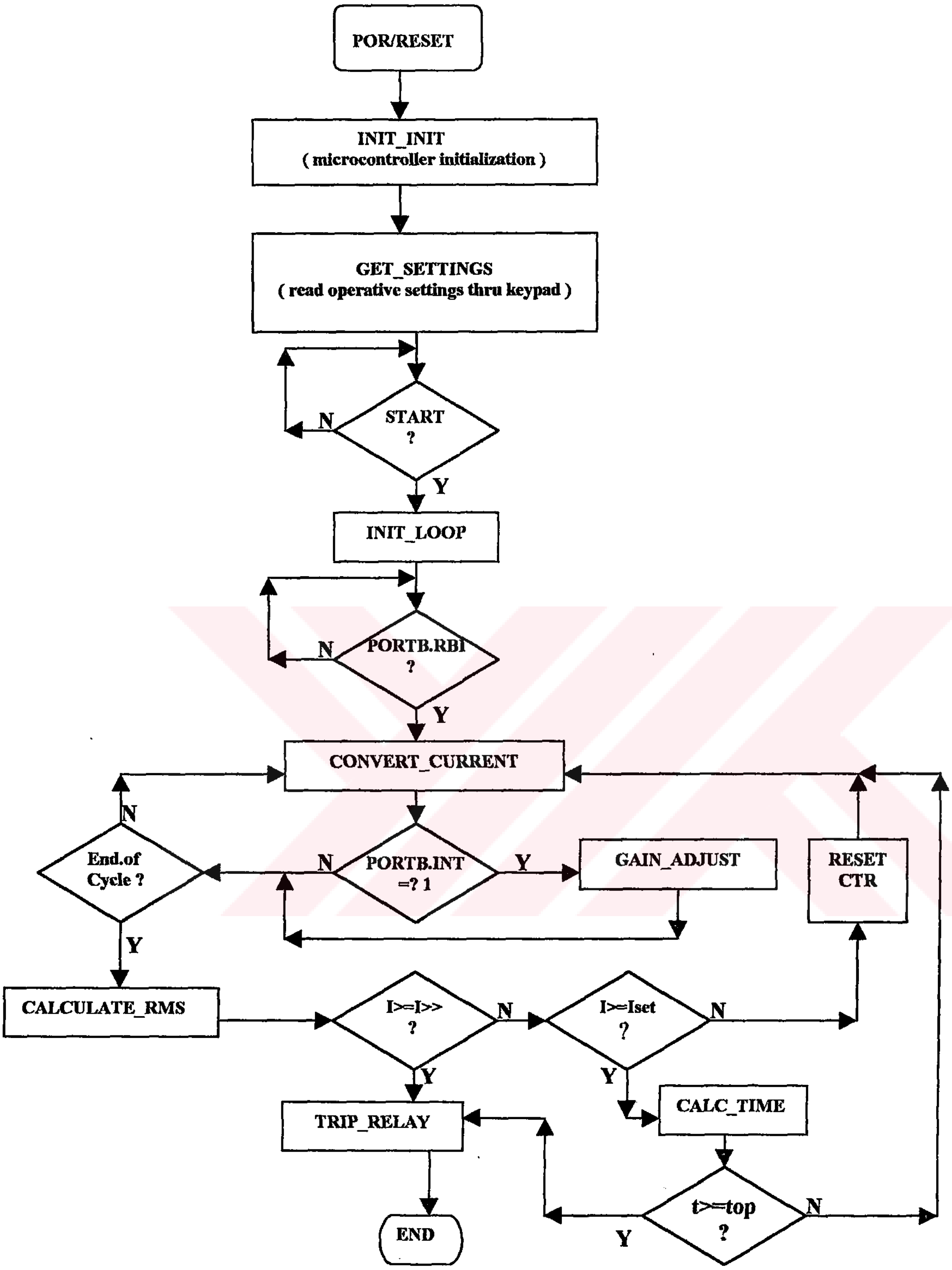


Fig 4.1, Program flowchart

ADRES register. The 8-bit digitized data is transferred to a temporary register ,the delay time is guaranteed for the next AD conversion and the AD converter is initiated for the next AD conversion. Since 20 samples are taken over a full source cycle, the necessary delay time between two successive conversions is 1msec, and is guaranteed by a software delay routine.

4.3.2 Functional Software Blocks

As stated above, the software of the relay is composed of a main operation loop and autonomous subroutines each performing a particular function. The functioning of the main operation loop was explained in the preceding section. In this section, the functioning of the subroutines constituting the software of the relay will be explained briefly. These subroutines are called by the main program loop to perform the involved function.

Init_Init : Called by the main operation loop once after the microcontroller is reset. Initializes the I/O ports, interrupt sources and interrupt flags, timer and AD Converter.

Convert_Current : Controls the AD conversion of current samples through the A/D conversion channels of the microcontroller. This routine ensures the required sampling time (100us.) for the internal charge hold capacitor, C_{HOLD}, and sets the AD conversion start switch GO/DONE to start the AD conversion of the signal. End of each A/D conversion is detected by software polling of the GO/DONE bit in the ADCON0 register.

Write_Led : Refreshes the characters displayed on 7-segment LED displays. Controls the visual indicator leds also.

Get_Settings *:* Gets operative settings through the keypad prior to relay operation.

Gain_Control : Activates-deactivates the gain resistors associated with the current amplifying circuit in order the output signal of this circuitry to be compatible with the AD conversion range.

Calculate_RMS : Performs necessary calculations to obtain the RMS values of line current using the digitized samples collected through the AD Converter Block.

Calc_Time : This subroutine ensures necessary operating time, t_{OP} , for any value of the current beyond the relay pickup current value, I_{SET} .

4.4 RMS VALUE CALCULATION

As stated in preceding sections, the line current will be calculated by the Root Mean Square (RMS) method. Digitized line current values are 8-bit hexadecimal numbers stored within the microcontroller's RAM. Twenty samples will be used for RMS calculation for each source period. The hexadecimal range for current samples is,

$$00H < I[n] < FFH, \qquad (4.1)$$

For the full relay measuring range, (i.e.,I_{SET} =2*I_N), 20* I_{SET} =40A. The sum of the squares measured for 40A will be:

$$I_{RMS} = \frac{\sum_{n=1}^{20} I^2[n]}{20} = 131044(dec) \qquad (4.2)$$

The goal is to obtain the square-root of this quantity to obtain the true RMS value of the line current in minimum time so that the calculation time does not effect the speed of protection in significant manner. The obliging criteria for any relay design is that the clearing time of any relay should not exceed three source cycles (60ms for 50 Hz base) for the worst case fault conditions. In this design, the worst case condition is approved as the fault current exceeds 20*I_{SET} (40A).

A simple but yet effective algorithm is used for calculating the RMS value of line current. Using,

$$1+3+5+\ldots\ldots+n\ (odd) = x^2 \qquad (4.3)$$

where:

x : the number of odd numbers summed.

it can be concluded that the estimated square-root of any decimal number can be calculated by extracting odd integers from this number until the result attains to zero, then the estimated square-root of this number is the total number of odd integers extracted. In the case that the number does not converge to zero, the estimated square-root is rounded to the number of odd integers which yields with the nearest value to the zero.

Numerical Examples:

1.) $16\text{-}1\text{-}3\text{-}5\text{-}7 = 0 \Rightarrow \sqrt{16} = 4;$

2.) $24\text{-}1\text{-}3\text{-}5\text{-}7 = 8$

$24\text{-}1\text{-}3\text{-}5\text{-}7\text{-}9 = \text{-}1 \Rightarrow \sqrt{24} \cong 5$

3.) $39\text{-}1\text{-}3\text{-}5\text{-}7\text{-}9\text{-}11 = 3$

$39\text{-}1\text{-}3\text{-}5\text{-}7\text{-}9\text{-}11\text{-}13 = \text{-}10 \Rightarrow \sqrt{39} \cong 6$

The rounding error introduced is less as the number increases. The maximum error introduced does not exceed ±5% and the relative error introduced will be compensated by iterative operations for calculating time-current curves. Although the method is not suitable for precise RMS measurement, it is still applicable for fixed point arithmetic and is rather practical because of its time efficiency. The maximum calculation time will exist for the number 131044, .

$$\sqrt{131044} = 362, \qquad (4.4)$$

which means that 362 subtraction operations should be performed to obtain the square-root of this number. The microcontroller can perform subtraction operation in a single instruction cycle, which is t_{CLK} =250ns for 16MHz clock. So the time required for finding the square-root of this number is, $t_{sqrt} = 362*250*10^{-9}$ =0.09ms, which is quite good for satisfying the time criteria.

4.5 CALCULATION OF TIME-CURRENT CHARACTERISTICS

The relay supports only the NI (Normal Inverse) time-current characteristics which is defined in IEC-255-4 [9] for Inverse-Time operation, and Definite-Time operation. The calculation of required operating time, t_{OP}, according to the concerned time-current characteristics requires high level arithmetic operations including the floating-point arithmetic. The PIC16C74 microcontroller is able to support floating-point arithmetic by embedding the floating-point math library which is not available in software tools of the microcontroller and must be developed.

Usage of floating-point arithmetic routines yield with long arithmetic iterations, which will effect the time-efficiency of the software negatively. There is also the risk of stack overflow for the floating-point arithmetic because of the bulkiness of the arithmetic operations to be performed.

To overcome the problems which may arise by the arithmetic calculation of time-current characteristics, time-current look-up tables are used. The nonlinearity of the time-current characteristics is implemented by tabulating the values to be loaded to the timer for each measured RMS value of the line current.

The current measurement range of the relay is divided into two regions as stated in Section 4.2 as 0-10A, and 10A-40A. The maximum possible measured RMS values for these ranges are:

- Range 1: $((255-127)^2)^{1/2} / 2^{1/2} = 90$,
- Range 2: $(4^2*(255-127)^2)^{1/2} / 2^{1/2} = 362$.

In other words, the calculated number of different current values for the ranges 1and 2 are 90 and 362 respectively. In this application, the relay pickup value is selected as 1.00, that means the relay will operate for RMS current values which exceed $1.00*I_{SET}$. The number of different RMS current values for 0-5*I_{SET} (0-10A) range is 90. Hence, the number of valid RMS values for this range is between 0.5*I_N=0.5A and 2*I_N=2A, which is equal to 9.5/10 of the full range. Thus, the number of entries in the table corresponding to this region would be 9.5/10*90 = 85.5 = 86. Identically, the number of entries of the table for the second range is (30/40)*362 = 272. For each of the seven current taps, different range-1and range-2 look up tables are prepared. The number of entries in these look-up tables are calculated as follows:

$$\#\_of\_table\_entries\_for\_range\_2 = \frac{20*I_{SET} - 10}{20*I_{SET}} \times 362, \tag{4.5}$$

$$\#\_of\_table\_entries\_for\_range\_2 = \frac{20*I_{SET} - 10}{20*I_{SET}} \times 362, \tag{4.6}$$

The number of entries in each look up table for seven different current taps will be then,

- Table_0.5_range_1 = 86,
- Table_0.75_range_1 = 84,
- Table_1.0_range_1 = 81,
- Table_1.25_range_1 = 79,
- Table_1.5_range_1 = 77,
- Table_1.75_range_1 = 75,
- Table_2.0_range_1 = 72,
- Table_0.5_range_2 = 0,
- Table_0.75_range_2 = 121,
- Table_1.0_range_2 = 181,
- Table_1.25_range_2 = 217,

- Table_1.5_range_2 = 241,
- Table_1.75_range_2 = 258,
- Table_2.0_range_2 = 272,

Thus the total ROM area to be occupied by the look up tables is 1844 bytes. The rest of the total ROM, 4096-1844=2252 bytes, will be occupied by the relay software coding.

The general equation for the standard Normal Inverse curve is:

$$t_{OP} = \frac{0.14}{\left(\frac{I}{I_{SET}}\right)^{0.02} - 1} \qquad (4.7)$$

For the worst case operating condition, $I=20*I_{SET}$, the theoretical relay operating time will be 2.27sec. The greatest entry of the look up table will exist for that current value, and is chosen as 250. Since the counting register accumulation is advanced in a 20msec basis, for the fault current value corresponding to $20*I_{SET}$, 116 accumulations are required to ensure the theoretical operating time. Accumulating the number 250, 116 times, yields with a constant number of 29000. So the operating threshold number for Inverse and Definite Time operation is 29000. When calculating this number, the assumption that the TMS setting is equal to 1.0 is taken into account. The other entries of the look up table are calculated using the following equation:

$$dt = \frac{29000}{t_{OP} \times 20.10^{-3}} \qquad (4.8)$$

where dt denotes the look up table value. The table value to be added to the cumulative sum is also scaled with the TMS value.

4.6 DEDUCTION OF CT SATURATION

In Section 2.4, the saturation phenomena in primary relaying current transformers and their effects on relay performance is examined. In Section 2.4.5, four methods to perceive that the current transformer has entered the region of saturation are proposed. Among these four methods, the latter two seems to be applicable for a microcontroller based relaying system. Since the primary case of concern in asymmetrical fault conditions is the offset fault currents, only the method of examining the duty cycles of the fault current waveform is employed within the software.

This method is to investigate the duty cycles of the zero_crossing waveforms for line current and is applicable for offset fault current conditions. This method is implemented in the software by the use of TMR2 (Timer-Counter Module) within the PIC16C74 microcontroller. The clock source for the 16-bit Timer-Counter Module, TMR2 is the internal instruction cycle ($f_{osc}/4$), prescaled by a factor 1:4. That means that the 16-bit Timer-Counter Module, TMR2 advances by one at every 4 instruction cycles (1μsec). When the rising edge of the zero_crossing waveform for the current is detected, the TMR2 registers is reset to 00H and the TMR2 module is initiated until the reception of the falling edge of the current zero_crossing signal. Then the value present in the TMR2 registers will be read to a temporary register and the TMR2 is re-initiated to 00H until the subsequent rising edge of the current zero_crossing signal. Finally, the two TMR2 values are compared. The relay comes into a decision of a CT saturation in the case that the Timer value for the positive cycle is less than the Timer value for the negative cycle. The TMR2 registers are such configured that TMR2 module will overflow in 65msec. Hence, an overflow in the TMR2 register is utilized as an error in the measuring circuitry and treated as a self-checking property.

In this design, only the action performed by the relay whenever a CT saturation detected is to prompt a 'st' message on the 7-segment led display. Determining the degree of CT saturation by examining the ratio of duty cycles of the current waveform and for certain degrees of saturation, tripping the relay can be proposed as possible future developments for the relay.

4.7 SOURCE CODE

As stated in the beginning of this chapter, the source code is generated using the MPC Ver 1.20, which is a C-Code Compiler specific for PIC 16/17 microcontroller devices. The coding of the software can be seen in Appendix D.

CHAPTER 5

RESULTS OF THE IMPLEMENTATION

5.1 TEST STRATEGY

In order to determine whether the relay performs the functions stated in the design specifications, some functional tests are carried out, results of whom can be seen in the following sections. The tests performed does not involve the standard tests stated in IEC 255 [9], which are required for a relay to be put into production, only the time values for different fault current values are taken in order to interpret whether the error criterion is satisfied or not.

The output of an overcurrent relay is the relay operating time, t_{OP}. Hence the error limits for a relay stated in various standards is based on this quantity. According to IEC and ANSI/IEEE standards [9],[10],[11], the overall time error introduced by a Class '5' relay should not exceed 5%, for any fault current value. Tests performed for this design is the verification of that criteria.

The operating time values for each current tap are measured for certain fault current values namely, $2*I_{SET}$, $5*I_{SET}$, $10*I_{SET}$ and $20*I_{SET}$, with the Time Multiplier Setting, TMS=1. For each current tap, the instantaneous operation time (t>>) is also measured. Measured values are compared with the theoretical values and percentage error is tabulated for each test. These tabulated data are seen in Section 5.3.

5.2 TEST SETUP

The experimental setup seen in Fig 5.1 is prepared for testing of the relay. Current supplied to relay measuring circuitry is simulated by connecting a variac in series with a current transformer. A precision timer with external Start/ Stop terminals is used to measure the time. Start and Stop signals to the timer are supplied by the relay contacts. When the relay encounters a current value beyond the relay pickup value, it trips one of the output relays, whose contacts starts the timer, and when the relay decides to operate, it trips the second relay, whose contacts stops the timer. When the timer stops, the operating time is latched on the display.

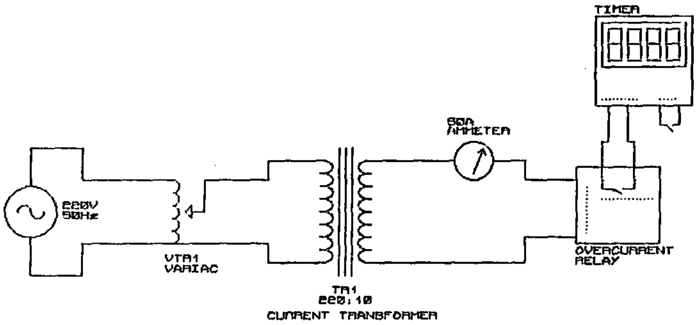


Fig 5.1, Test setup

5.3 TEST RESULTS

The tests performed and their tabulated results are as seen below:

TEST-1

$I_{SET} = 0.5^* I_N$,

TMS=1.0,

$I\gg = 20^* I_{SET}$.

Table 5.1Results of Test-1

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	10.12	0.99
5	4.28	4.35	1.63
10	2.97	2.89	2.69
20	2.27	2.33	2.64
>20	<0.04	0.03	Ok

TEST-2

$I_{SET} = 0.75^* I_N$,

TMS=1.0,

$I>> = 20^* I_{SET}$.

Table 5.2, Results of Test-2

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	9.98	0.39
5	4.28	4.36	1.87
10	2.97	2.93	1.35
20	2.27	2.32	2.20
>20	<0.04	0.03	Ok

TEST-3

$I_{SET} = 1.0^* I_N$,

TMS=1.0,

$I>> = 20^* I_{SET}$.

Table 5.3, Results of Test-3

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	10.11	0.89
5	4.28	4.36	1.87
10	2.97	2.88	3.03
20	2.27	2.34	3.08
>20	<0.04	0.03	Ok

TEST-4

I_{SET} = 1.25* I_N,

TMS=1.0,

I>> = 20* I_{SET} .

Table 5.4, Results of Test-4

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	10.12	0.99
5	4.28	4.34	1.40
10	2.97	2.90	2.35
20	2.27	2.35	3.52
>20	<0.04	0.03	Ok

TEST-5

I_{SET} = 1.5* I_N,

TMS=1.0,

I>> = 20* I_{SET} .

Table 5.5, Results of Test-5

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	10.11	0.89
5	4.28	4.29	0.23
10	2.97	2.93	1.35
20	2.27	2.30	1.32
>20	<0.04	0.03	Ok

TEST-6

$I_{SET} = 1.75^* I_N$,

TMS=1.0,

$I{>>} = 20^* I_{SET}$.

Table 5.6, Results of Test-6

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	9.97	0.5
5	4.28	4.36	1.87
10	2.97	2.85	4.04
20	2.27	2.32	2.2
>20	<0.04	0.03	Ok

TEST-7

$I_{SET} = 2.0^* I_N$,

TMS=1.0,

$I{>>} = 20^* I_{SET}$.

Table 5.7, Results of Test-7

I (*I_{SET})	t_{OP} (theoretical/sec)	t_{OP} (measured/sec)	Error %
2	10.02	10.10	0.79
5	4.28	4.34	1.40
10	2.97	2.93	1.34
20	2.27	2.30	1.32
>20	<0.04	0.03	Ok

CHAPTER 6

CONCLUSION

In this thesis study, a digital relay system proving increased reliability with minimum cost is developed. One of the most significant objectives of this study is to improve the precision of current measurement by applying new techniques for current measurement , so called 'variable gain control'. By this method, the dynamic measuring range of current is divided into two regions and the possibility to use the full resolution of the AD converter for each distinct region is supplied. This resulted with a significant increase in relay's accuracy.

RMS current measurement technique enables the relay to get the true effective value of the fault current waveform. Since the floating point arithmetic is not used for calculating the RMS value of the line current, this method is not suitable for a relay to be used for measurement purposes, but still effective for a relay used for protective purpose.

Another important goal for this study was to equip the relay with some extra properties to perceive, and to some extent, compensate the distortive effects arising from the saturation of primary current transformers. Due to the diverse nature of the transient fault currents and their dependability on power system parameters, it was impossible to fully satisfy this criteria, but to some extent is realized. Some software code is embedded to relay source code to perceive and compensate at least the high degrees of CT saturation which may cause significant effects on protection performance of the relay.

Because of memory limitations, only the inverse time characteristics for NI curve is implemented. However, other curves defined in standards such as EI, VI, LTI can also be implemented considering that an external serial EEPROM exists on the hardware.

Test results denote that the relay can be considered as a Class '5' relay taking into account that the overall time error introduced by the relay does not exceed 5% for any fault current value. Also, a 30msec Instantaneous Trip time is accepteble for protection purposes.

Injection of a voltage measuring unit, and adding directional overcurrent functions to the relay can be proposed as a future development of the relay. For embedding more sophisticated CT saturation detection algorithms, usage of a stronger microcontroller which can support floating point algorithms is recommended. The relay can also be equipped with a serial communication software and a visual PC interface for remote monitoring and programming purposes.

As a result of this study, it can be concluded that microcontroller based systems can supply great flexibility and cost advantage in protection systems design.

REFERENCES

[1] William D. Stevenson, 'Elements of Power System Analysis', McGraw-Hill International Editions, 1982.

[2] W. A. Elmore, ''Protective Relaying Theory and Applications', ABB Power T&D Company Inc, 1994.

[3] Lewis Blackburn, 'Protective Relaying, Principles and Applications', Marcel Dekker Inc., 1987.

[4] Westinghouse Electric Corporation Relay-Instrument Division, 'Applied Protective Relaying', Newark, 1976.

[5] Stanley H. Horowitz, 'Protective Relaying for Power Systems', IEE, 1980.

[6] Hakan Çiftçi, 'Implementation of Digital Overcurrent Protection Relay', A Msc. Thesis submitted to the Graduate School of Natural and Applied Sciences, METU, May 1997.

[7] Power System Relaying Commitee of the IEEE Power Engineering Society, 'Transient Response of Current Transformers', IEEE, January 1976.

[8] A.R. Warrington, 'Protective Relays', Chapman&Hall Co., 1962.

[9] IEC Standard Publication 255-4, 'Single input energizing measuring relays with dependent specified time', International Electrotechnical Commision, 1976.

[10] 'ANSI/IEEE Standard 100-1984', IEEE,1984.

[11] 'ANSI/IEEE Standard C37.100-1981', IEEE,1981.

[12] 'Solid State Sensors', Honeywell Inc., 1995.

[13] 'AC and DC Current Sensors', URD Inc., Germany, 1996.

[14] 'Current Sensor Databook', LEM Sa , GENEVE, 1994.

[15] 'High Performance 8-bit CMOS Eprom/Rom Microcontroller Data Book', Microchip Technology Inc., 1995.

[16] A.D. Parker, 'Development of a microcomputer based overcurrent relay', GEC Rewiew, 1995.

[17] Mudhafar A. Al-Nema, Sinan M. Bashi, Abdulhadi Ubaid, 'Microprocessor-based overcurrent relay', IEEE Transactions on Industrial Electronics, Vol. IE-33, No.1, February,1986.

[18] The Kansai Electric Power Co., Inc, Osaka, Japan and Mitsubishi Electric Corp. Kobe, Japan, 'Development and field experience of digital Protection equipment in power system ', IEEE WM 173.3,1982.

[19] ABB Power T&D Co., 'Multipurpose Digital Overcurrent Relay Release Notes', 1995.

[20] 'MPC Code Development System, User's Manual', BYTE CRAFT Ltd., December,1994.

[21] James W. Nilsson, 'Electric Circuits', Addison Wesley Co., 1990.

APPENDIX A

SOME DEFINITIONS ABOUT OVERCURRENT RELAYS

Characteristic curve: The graphical representation of the relationship between time and the current.

Time Multiplier Setting (TMS): The constant that multiplies the characteristic curve's time scale.

Current Tap (I_{SET}): The constant which multiplies the Nominal current value, I_N, to obtain the set current value.

Instantaneous Current Multiplier Setting (I>>): The integer that multiples the set current value to obtain the instantaneous operation threshold current value.

Hi Set (Instantaneous) operation: Instantaneous tripping of the output relays whenever a current value that exceeds the Hi Set threshold current value is encountered.

Timed operation: Tripping of the relay after ensuring a predefined operating time, t_{OP}, for the existence of a fault current value between relay pickup value and the Hi Set value.

Operating time (t_{OP}): The time interval between the instant that the relay encounters the fault current and the relay operates.

The relative error: The percentage difference between the theoretical and actual operating times.

Class index: Measure of the relay relative error range.

Normalised current value: The ratio of measured current to set current.

Operating current value ($I_{Pick-up}$): The current value at which the relay starts.

Initial Condition: The condition which the relay leaves in order to complete its designated function.

Operated condition: The condition of a relay as long as the designated function is completed.

A relay is said to start: the instant it leaves the initial condition.

A relay is said to switch: The instant it completes the designated function.

A relay resets: When it retains an initial condition.

A relay disengages: The instant it terminates a function previously effected in a given trip element.

A relay operates: When sequentially it starts, passes from initial condition towards the prescribed operated condition, and it switches.

A relay returns: When sequentially it disengages, it passes from an operated towards the prescribed initial condition, and it resets.

APPENDIX B

RELAY ELECTRICAL SPECIFICATIONS

Below are the information about the Electrical Specifications of the overcurrent relay, the keypad configuration, and the relay visual indicators.

Electrical Specifications:

Nominal Current: $I_N = 1A$.

Current Taps: 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0.

Relay pickup value: $1.01*I_{SET}$.

Operating Modes: Definite Time, Inverse Time.

Standard Curve: NI (Normal Inverse).

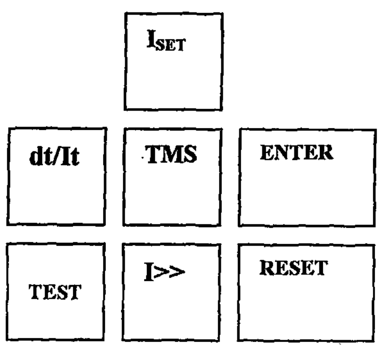
TMS range: 0.1..1.0, with 0.1 step increments.

I>> range: 2..20* I_{SET}, with 1 step increments.

Definite Time range (for TMS=1): 2sec.

Keypad Configuration:

The arrangement of the buttons on the front panel and their functions are as follows:



dt/It: Definite Time – Inverse Time selection,

TEST: Tests the functionality of relay contacts,

I_{SET}: Current Tap Setting,

TMS: Time Multiplier Setting,

I>>: Hiset Multiplier Setting,

ENTER: Accept selection,

RESET: Software reset.

Visual Indicators:

7-segment display:

'dt' : Definite Time,

'It': Inverse Time,

't1'.. 't7': Current Taps,

'_ _': Ready for getting set values,

'Op.': Ready for operation,

'tt': Timed Trip,

'It': Instantaneous trip,

'O.r.': Overrange,

'st': Current transformer saturation.

LED Indicators:

LED-1: Trip Indicator. (RED)

0: No relay trip,

1: Relay tripped.

LED-2: Overcurrent Indicator. (RED)

0: No overcurrent detected,

1: Overcurrent detected.

APPENDIXC

SCHEMATIC DIAGRAMS AND PCB LAYOUTS

This appendix contains the schematic drawings and PCB layouts of the overcurrent relay. This section includes:

- Digital Card Schematic Diagram,
- Analog Card Schematic Diagram,
- Power Supply Circuit Drawings,
- Analog Card Drill Mask,
- Analog Card Solder Side,
- Analog Card Component Side,
- Analog Card Component Layouts,
- Digital Card Drill Mask,
- Digital Card Solder Side,
- Digital Card Component Side,
- Digital Card Component Layouts,

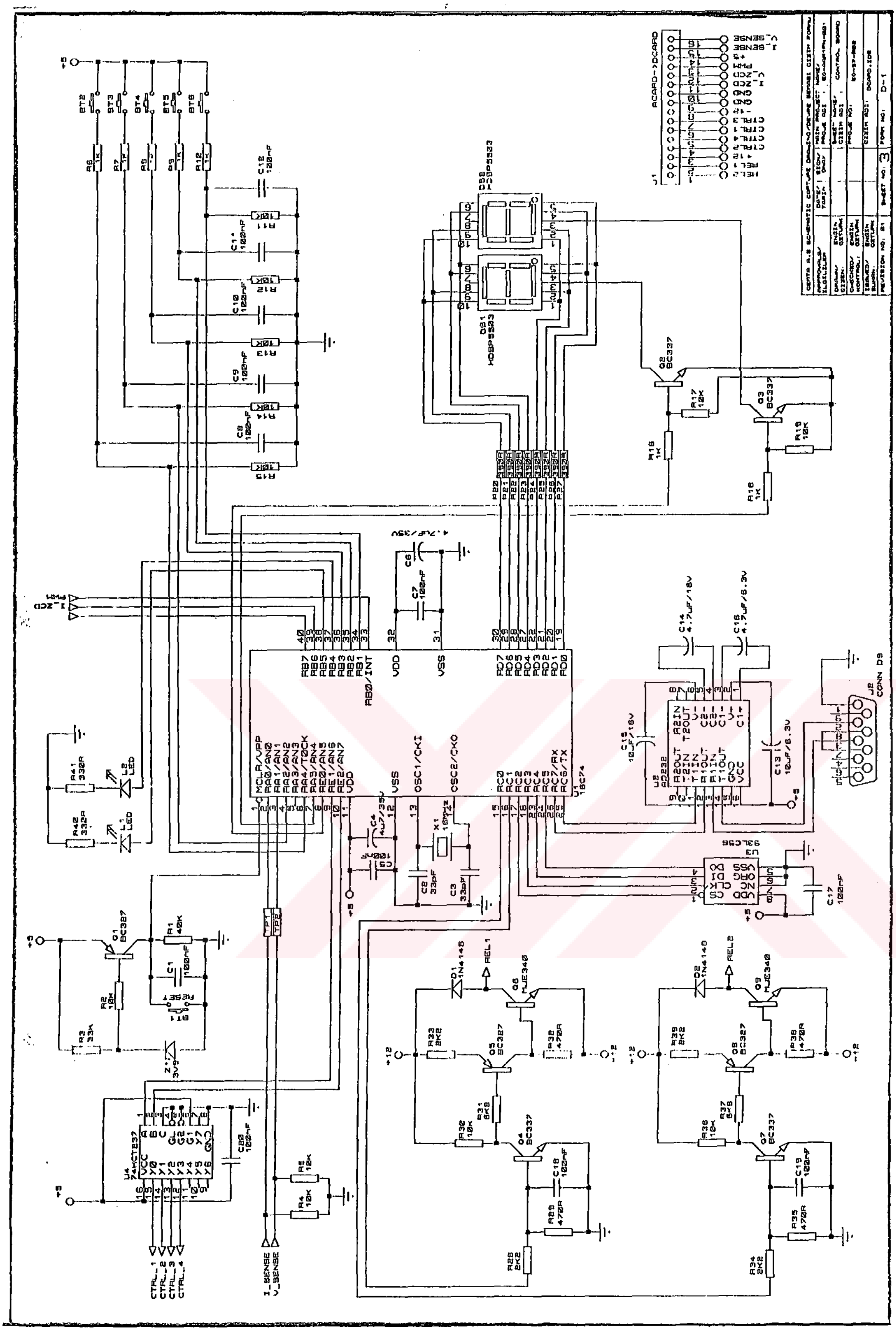


Fig A.1, Digital Card Schematic Diagram

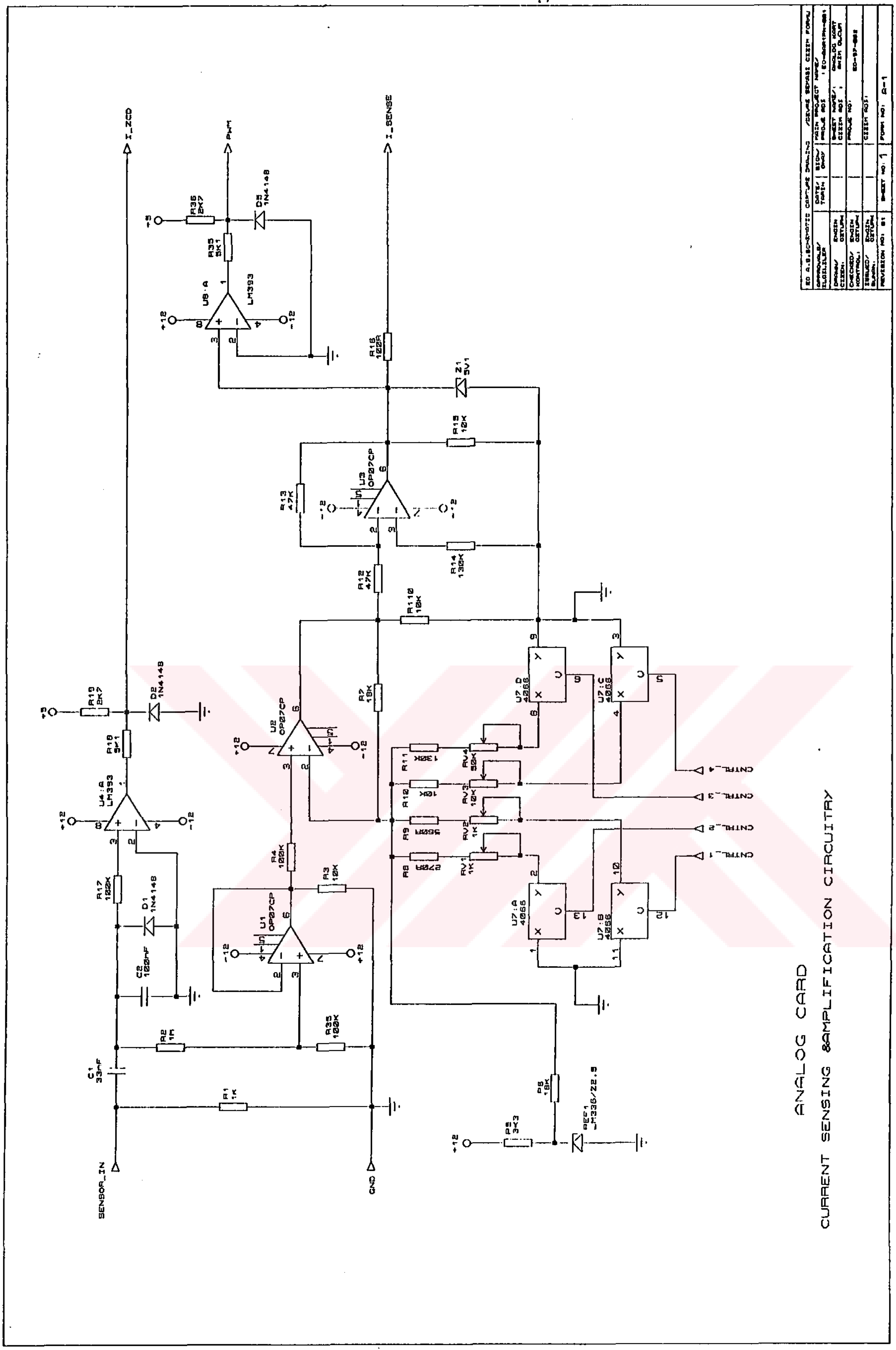


Fig A.2, Analog Card Schematic Diagram

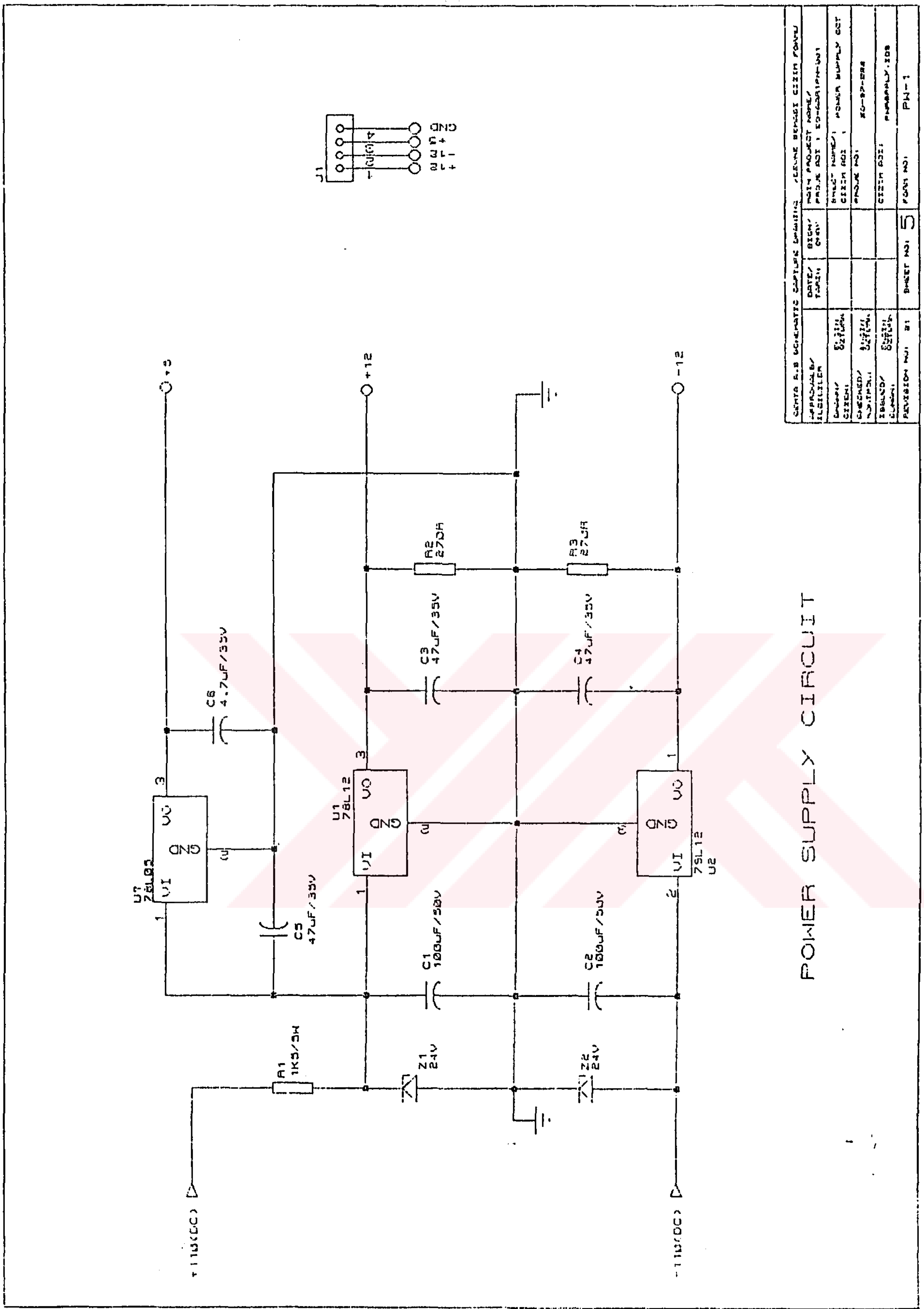


Fig A.3, Power Supply Circuit

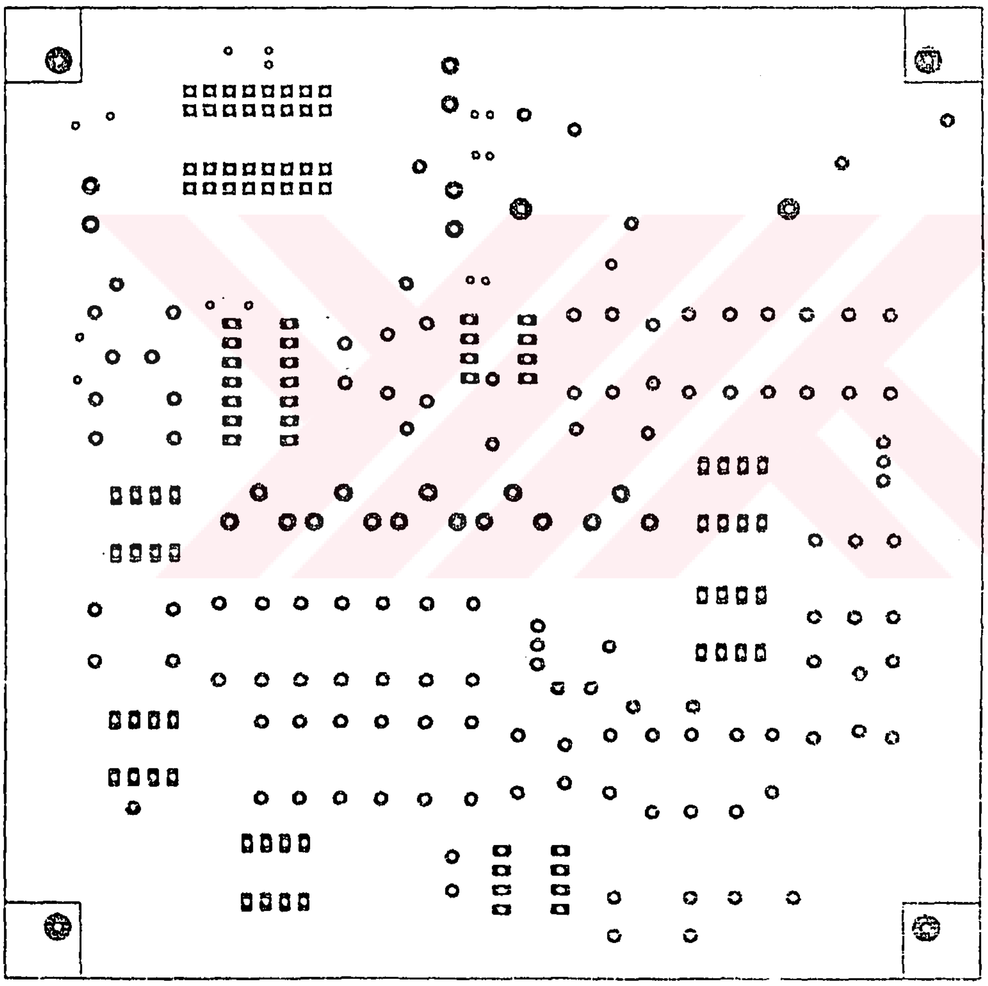


Fig A.4, Analog Card Drill Mask

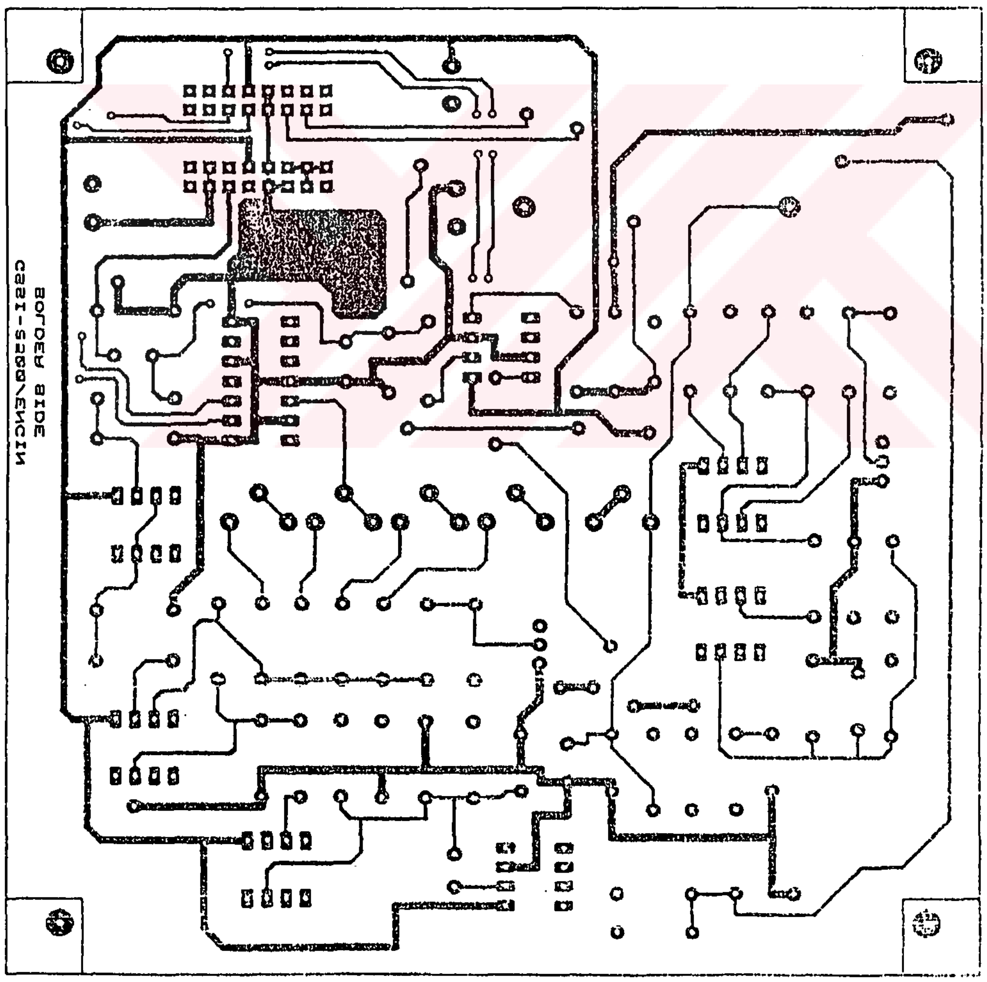


Fig A.5, Analog Card Solder Side

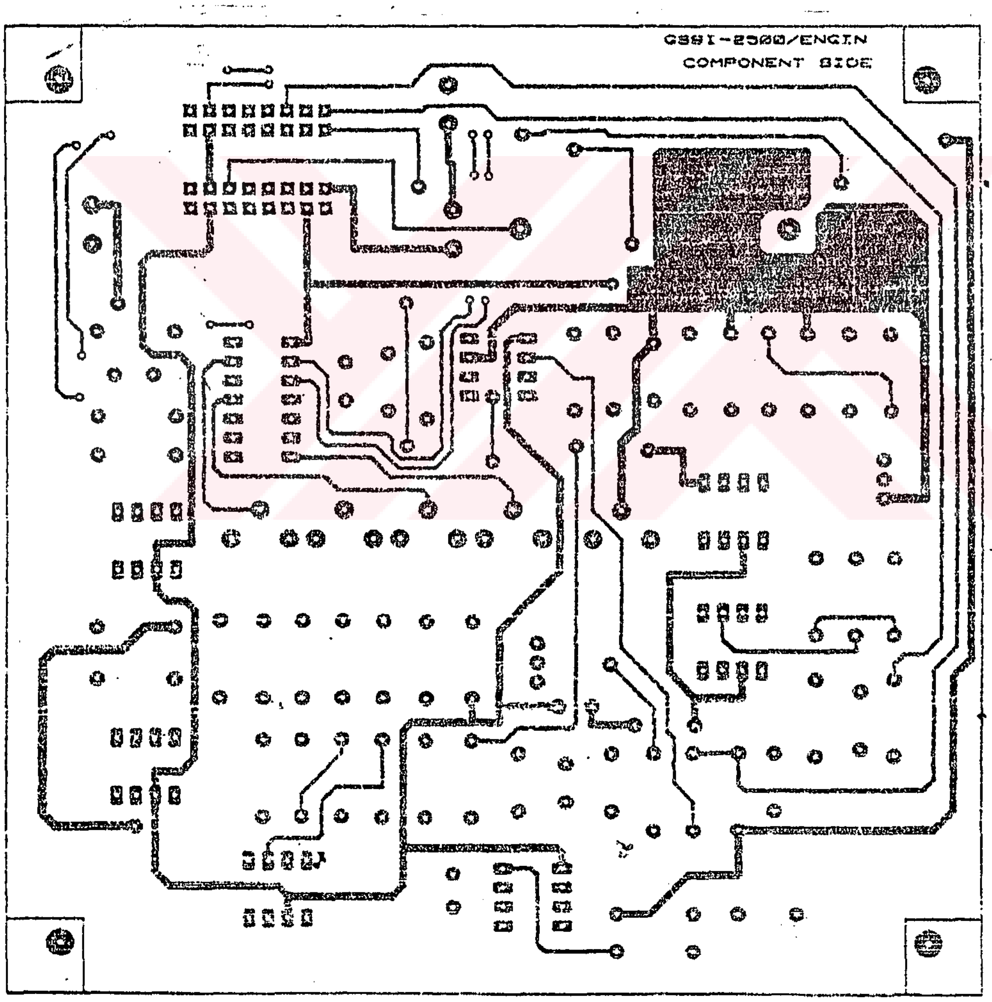


Fig A.7,Analog Card Component Side

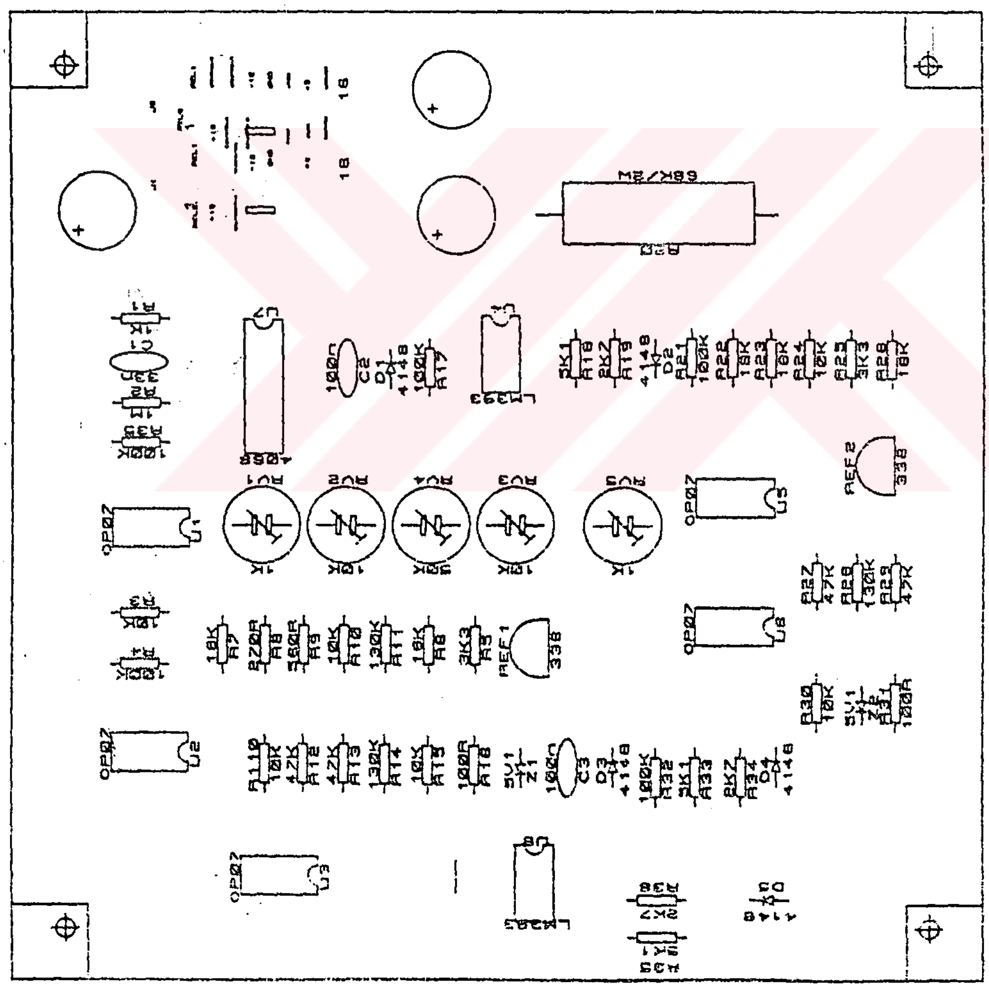


Fig A.8, Analog Card Component Layout

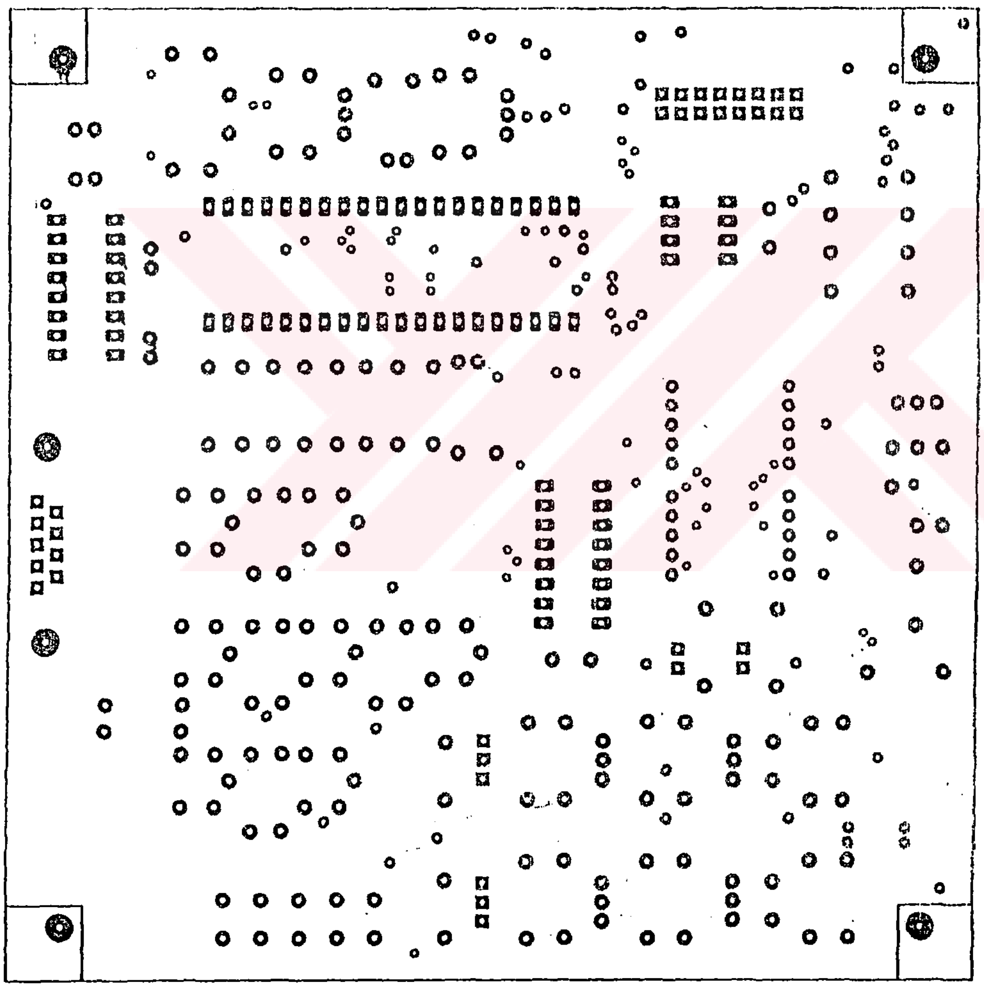


Fig A.9, Digital Card Drill Mask

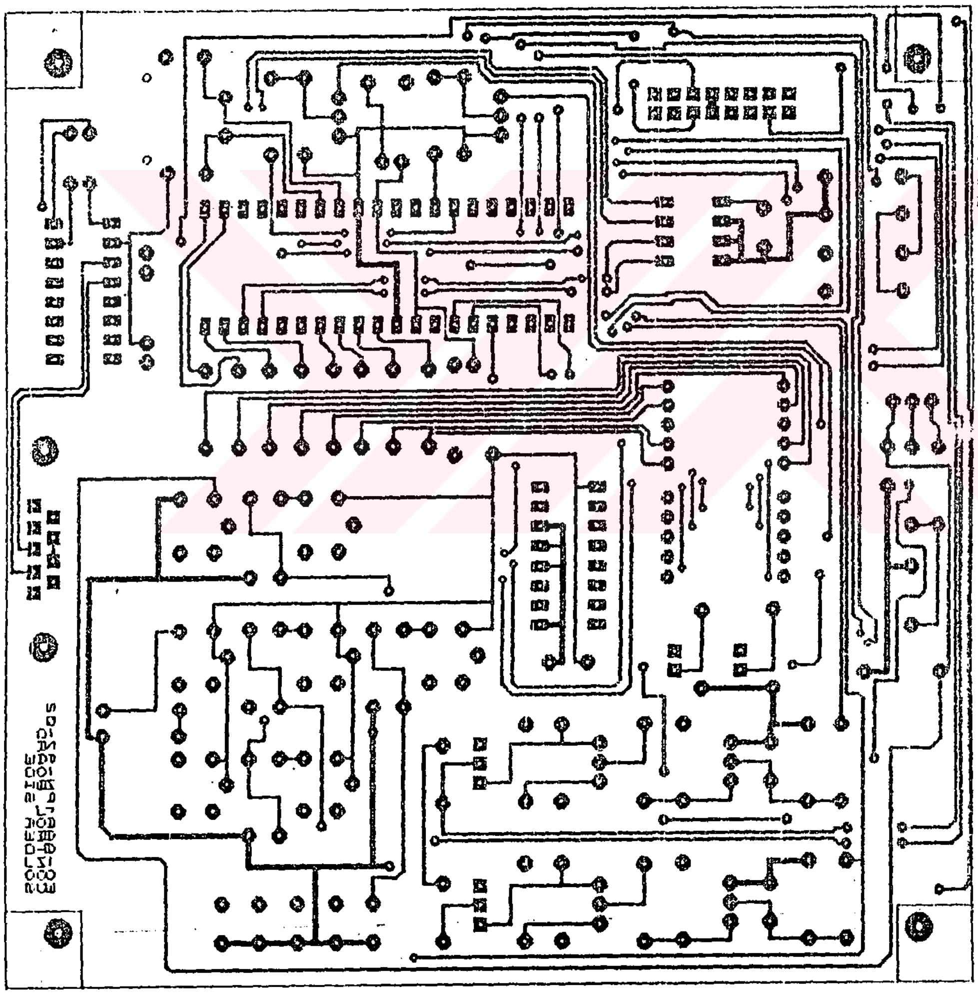


Fig A.10,Digital Card Solder Side

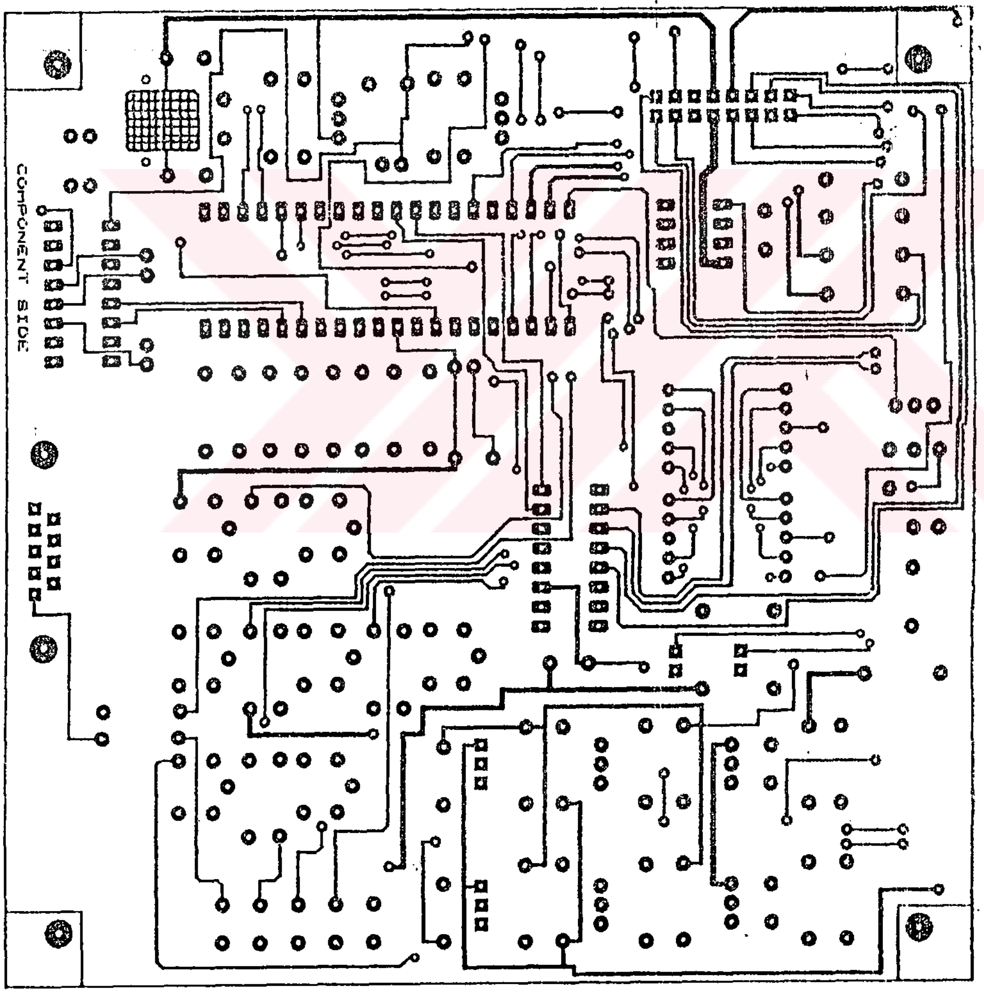


Fig A.11, Digital Card Component Side

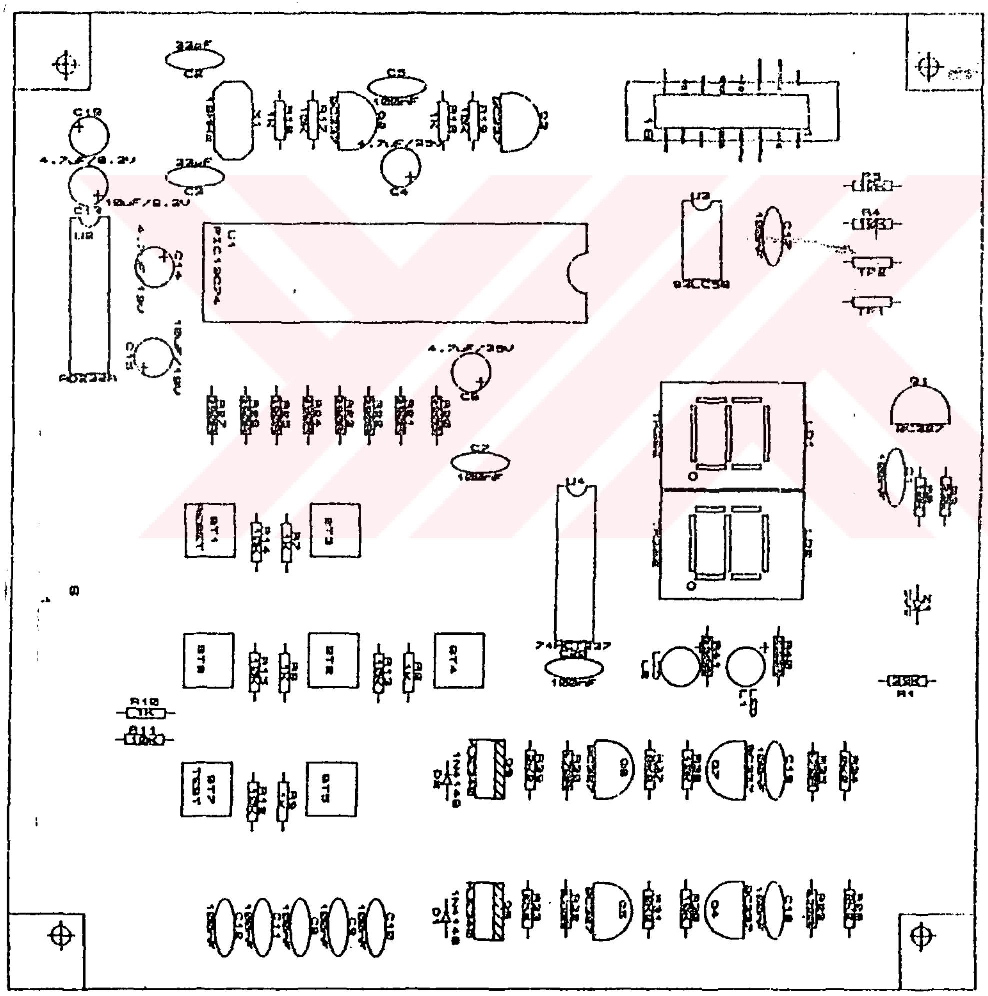


Fig A.12, Digital Card Component Layout

APPENDIX D

RELAY SOURCE CODE

```
#pragma option v;
#pragma option +i;
//#pragma memory ROM[MAXROM - 0x810] @ 0x810
#include <16c74.h>
#include <math.h>

#define ICS 0
#define IVS 1
#define ADDLY 2
#define ADSEQ 3
#define DISP 0
#define ENTER 1
#define SEL 2
#define UP 4
#define DOWN 3
#define MSI 0
#define DTI 1
#define ITI 2
#define HSI 3
#define MODES 6
#define HISET 7
#define TSI 4
#define CSI 5
#define MSC 0


// Forward Function Declarations:
// ------------------------------
void init_init(void);
void init_LED(void);
void delay(unsigned int time);
void sampling_delay(unsigned int dtime);
void write_LED(void);
void TMR0_overflow(void);
void TMR1_overflow(void);
void PORTB_interrupt(void);
void PWM_interrupt(void);
void AD_interrupt(void);
void ad_convert_current(void);
void ad_convert_voltage(void);
void get_keypad(void);
unsigned int dec_to_disp(unsigned int dec_num);
void Page_Cross(void);
void calculate_RMS_voltage(void);
void calculate_RMS_current(void);
unsigned long get_sqrt(unsigned long sum);
void calculate_curve(void);
void saturation_detection(void);
void trip_relay(void);
```

```
void load_TMR1(unsigned int value);
void scale_adjust(void);

// Variable Declarations:
// ----------------------
unsigned int led[2];
unsigned int i,j,k,l,m;
unsigned int get_portb;
bits flag_reg_1=0;
bits flag_reg_2=0;
bits flag_reg_3=0;
bits flag_reg_4=0;
unsigned int current_sample_ctr,voltage_sample_ctr;
unsigned int curr_arr[20];
unsigned int volt_arr[20];
unsigned int s_ctr=0,ud_ctr=0,ds_ctr=0;
unsigned long Time_Mult,Curr_Mult;
unsigned int HiSet_Mult;
unsigned int check1,check2,div1,div2;
unsigned int Scale_Factor;
unsigned int scaling_arr[20];
unsigned long square_volt,square_curr;
unsigned long mean_volt,mean_curr;
unsigned long RMS_voltage,RMS_current;
unsigned long RMS_value;
unsigned long odd_number;
unsigned long odd_ctr;
unsigned int converge;
int normalized_current=0;
int normalized_voltage=0;
bits Mode_Setting=0;
unsigned int TMR1_value;

// Constant ROM Array Declarations:
// --------------------------------
const unsigned int menu_array[] = {0x9E,0xCC,0x02,0xCC,0xCA,0xE6};
                                                    //Display Values
for:
                                                    // dt
                                                    // it
                                                    // hs
const unsigned int ts_cs_arr[] = {0xCC,0xE6,0x8C,0xE6};
                                                    // Display values
for:
                                                    // cs
                                                    // ts
const unsigned int cmul_arr[] =
{2,10,3,10,4,10,5,10,6,10,7,10,8,10,9,10,
                10,10,11,10,12,10,13,10,14,10,15,10,16,
                10,17,10,18,10,19,10,20,10};
                                    // Current Multiplier values for
dt/it
                                    // Range: (0.2..2.0)
const unsigned int tmul_arr[] =
{1,10,2,10,3,10,4,10,5,10,6,10,7,10,8,10,9,10,10,10};
                                    // Time Multiplier Values:
                                    // Range:0.1..1.0
/******************************/
// INTERRUPT SERVICE ROUTINE
 void __INT(void)
```

```
{
 if(INTCON.T0IF==1)
            {
              TMR0_overflow();
              }
 else if(INTCON.RBIF==1)
            {
              PORTB_interrupt();
              }
 else if(INTCON.INTF==1)
            {
              PWM_interrupt();
              }
 else if(PIR1.ADIF==1)
            {
              AD_interrupt();
              }
 else if(PIR1.TMR1IF==1)
            {
              TMR1_overflow();
              }
 else
      {
       NOP();
      }
// Restore PCLATH, W, __WImage and STATUS
                                        #asm
                                            BCF   STATUS,RP0        ;Bank 0
                                            MOVF temp_PCLATH, W
                                            MOVWF PCLATH            ;PCLATH
restored
                                            MOVF temp_WImage, W
                                            MOVWF  __WImage         ;__WImage
restored
                                            MOVF  temp_FSR, W
                                            MOVWF FSR               ;FSR
restored
                                            SWAPF temp_STATUS,W
                                            MOVWF STATUS            ;RP0
restored
                                            SWAPF temp_WREG,F
                                            SWAPF temp_WREG,W   ;W restored
                                        #endasm
 }

/*********************************/
void init_init(void)
{
 PORTA=0;
 PORTB=0;
 PORTC=0;
 PORTD=0;
 PORTE=0;              // Clear Initial Contents of Ports
 TRISA=0b00011111;
 TRISB=0b11001111;
 TRISC=0b10100000;
 TRISD=0;
 TRISE=0;              // Assign Data Directions to Ports
 OPTION=0b11000011;    // PortB pull_ups:Disable
                       // INTEDG:0to1
```

```
                         // T0CS: Internal Instruction Cycle
                         // Prescaler:TMR0
                         // TMR0 Rate:1:16
 ADCON0=0X80;
 ADCON1=0X04;            // Configure AD Converter:
                         // fosc/32,Channel0
                         // RA0 RA1 RA2 RA3 RA5 RE0 RE1 RE2
                         // ------------------------------
                         //  A   A   D   A   D   D   D   D
 T1CON=0;                // ------------------------------
                         // T1CKPS=00 / 1:16 (overflow at 65ms.)
                         // T1OSC=OFF, T1CS=OSC/4
 INTCON=0b01111000;
 PIE1=0X41;
 PIR1=0;                 // Initialize interrupts

 flag_reg_3.MODES=0;
 Curr_Mult=cmul_arr[36];
 *(Curr_Mult+1)=cmul_arr[37];
 Time_Mult=tmul_arr[18];
 *(Time_Mult+1)=tmul_arr[19];
 HiSet_Mult=20;                 //Load default operative settings:
                                //--------------------------------
                                // Mode: Definite_Time,
                                // Current_Tap: 2.0,
                                // TMS = 1.0,
                                // I>>=20*Iset
 Scale_Factor=1;                // Current Scaling Factor
 Mode_Setting.0=1;
   }
/********************************/
void init_LED(void)
{
 led[0]=0b11101100;
 led[1]=0b11111110;
 PORTA.5=0;
 PORTE.0=0;              // 'EO'on LEDs
 }
/********************************/
void delay(unsigned int time)
{
 for(i=time;i>0;i--)
   {
    for(j=0;j<100;j++)
      {
       }
    }
 }
/********************************/
void sampling_delay(unsigned int dtime)
{
 for(k=dtime;k!=0;k--);
 }
/********************************/
void write_LED(void)
{
 PORTD=led[0];
 PORTE.0=1;
 PORTA.5=0;
 delay(1);
```

```
 PORTD=led[1];
 PORTE.0=0;
 PORTA.5=1;
 delay(1);
 }
/********************************/
void TMR0_overflow(void)
{
 INTCON.T0IF=0;
 flag_reg_1.ADDLY=0;              //Delay time between two samples
ensured
 write_LED();
 }
/********************************/
void TMR1_overflow(void)
{
 trip_relay();
 write_LED();
 }
/********************************/
void PORTB_interrupt(void)      // Zero Crossing of Line voltage or
current
{
 get_portb=PORTB;
 INTCON.RBIF=0;                 // Clear interrupt Flag
 if(PORTB.7=1)                  //V_ZC
        {
         flag_reg_1.ICS=1;      // Initiate Current Sampling Routine
         current_sample_ctr=0;  // Reset current sample counter
         }
 else if(PORTB.6==1)            //I_ZC
        {
         flag_reg_1.IVS=1;      // Initiate Voltage Sampling Routine
         voltage_sample_ctr=0;  // Reset voltage sample counter
         }
 else                           //180_ZC,don't care
        {
         NOP();
         }
 write_LED();
 }
/********************************/
void PWM_interrupt(void)
{
 write_LED();

 }
/********************************/
void AD_interrupt(void)
{
 PIR1.ADIF=0;
 if(flag_reg_1.ADSEQ==0)              // Voltage Value in ADRES
       {
        volt_arr[voltage_sample_ctr]=ADRES;
        voltage_sample_ctr++;
        }
 if(flag_reg_1.ADSEQ==1)              // Current Value in ADRES
       {
        curr_arr[current_sample_ctr]=ADRES;
        scaling_arr[current_sample_ctr]=Scale_Factor;
```

```
        current_sample_ctr++;
        }
 ADCON0.0=0;                        //Turn OFF AD converter
 TMR0=0x7F;                         //???????
 flag_reg_1.ADDLY=1;                // Ensure Delay between two
samples
 write_LED();
 }
/*********************************/
void ad_convert_current(void)
{
 if(flag_reg_1.ICS==1)
    {
     if(flag_reg_1.ADSEQ==0)
        {
         if(flag_reg_1.ADDLY==0)
            {
             ADCON0.5=0;
             ADCON0.4=0;
             ADCON0.3=0;            // Channel0
             ADCON0.0=1;            // Turn ON AD module
             sampling_delay(0xA0);  // Ensure required sampling time
             ADCON0.2=1;            // Start AD conversion
             flag_reg_1.ADSEQ=(!flag_reg_1.ADSEQ);
                                    // Switch to Voltage conversion
             }
          }
       }
 }
/*********************************/
void ad_convert_voltage(void)
{
 if(flag_reg_1.IVS==1)
    {
     if(flag_reg_1.ADSEQ==1)
        {
         if(flag_reg_1.ADDLY==0)
            {
             ADCON0.5=0;
             ADCON0.4=0;
             ADCON0.3=1;            // Channel1
             ADCON0.0=1;            // Turn ON AD module
             sampling_delay(0xA0);  // Ensure required sampling time
             ADCON0.2=1;            // Start AD conversion
             flag_reg_1.ADSEQ=(!flag_reg_1.ADSEQ);
                                    // Switch to Current conversion
             }
         }
     }
 }
/*********************************/
void get_keypad(void)
{
 flag_reg_2.DISP=PORTA.2;
 flag_reg_2.ENTER=PORTB.2;
 flag_reg_2.SEL=PORTB.3;
 flag_reg_2.UP=PORTA.4;
 flag_reg_2.DOWN=PORTB.1;            // Get keypad status

 switch(flag_reg_2)
```

```
	{
	 case  2: {                    //*************** ENTER
**************
		if(flag_reg_3.MSI==0)        //Menu1-->Menu2
		  {
		   if(flag_reg_3.DTI==1)
		     {
		      flag_reg_3.DTI=0;        //Clear
Indicator
		      flag_reg_3.MODES=0;
//DefiniteTimeMode
		      led[0]=ts_cs_arr[s_ctr];
		      led[1]=ts_cs_arr[s_ctr+1];
//refresh LEDs
		      flag_reg_3.MSI=1;        //Menu1
Item Selected
		      }
		   else if(flag_reg_3.ITI==1)
		     {
		      flag_reg_3.ITI=0;        //Clear
Indicator
		      flag_reg_3.MODES=1;
//InverseTimeMode
		      led[0]=ts_cs_arr[s_ctr];
		      led[1]=ts_cs_arr[s_ctr+1];
//refresh LEDs
		      flag_reg_3.MSI=1;        //Menu2
Item Selected
		      }
		   else if(flag_reg_3.HSI==1)
		     {
		      flag_reg_3.HSI=0;        //Clear
Indicator
		      flag_reg_3.HISET=1;
		      check1=cmul_arr[ud_ctr];
		      check2=cmul_arr[ud_ctr+1];
		      if(check1<check2)        // < 1.0
		        {
		         led[0]=0x7F;
		         led[1]=dec_to_disp(check1);
		         }
		      else
		        {
// >=1.0
		         div1=cmul_arr[ud_ctr];
		         div2=cmul_arr[ud_ctr+1];
		         led[0]=dec_to_disp(div1/div2);
		         led[1]=dec_to_disp(div1%div2);
		         }
		      flag_reg_3.MSI=1;        //Menu2
Item Selected
		      }
		   else if(flag_reg_4.MSC==1)   //SettingMenu--
>Finish
		     {
		      if(flag_reg_3.TSI==1)
		        {
		         Time_Mult=tmul_arr[ud_ctr];
*(&Time_Mult+1)=tmul_arr[ud_ctr+1];
```

```
                                        }
                                   if(flag_reg_3.CSI==1)
                                         {
                                          Curr_Mult=cmul_arr[ud_ctr];

*(&Curr_Mult+1)=cmul_arr[ud_ctr+1];
                                          }
                                   if(flag_reg_3.HISET==1)
                                         {
                                          HiSet_Mult=cmul_arr[ud_ctr];
                                          }
                                   flag_reg_3.MSI=0;       //Reset
Procedure!
                                   flag_reg_4.MSC=0;
                                   }
                             else
                                   {
                                   NOP();                  //Dummy Enter
                                   }
                             }
                      else if(flag_reg_3.MSI==1)   //Menu2-->Setting_Menu
                            {
                             if(flag_reg_3.TSI==1)
                                   {
                                   check1=tmul_arr[ud_ctr];
                                   check2=tmul_arr[ud_ctr+1];
                                   if(check1<check2)       // < 1.0
                                     {
                                      led[0]=0x7F;
                                      led[1]=dec_to_disp(check1);
                                      }
                                   else
                                     {                      // >=1.0
                                      div1=tmul_arr[ud_ctr];
                                      div2=tmul_arr[ud_ctr+1];
                                      led[0]=dec_to_disp(div1/div2);
                                      led[1]=dec_to_disp(div1%div2);
                                      }
                                   flag_reg_3.MSI=0;     //?????
                                   flag_reg_4.MSC=1;     //Menu_2 Item
Selected
                                   }
                             else if(flag_reg_3.CSI==1)
                                   {
                                   check1=cmul_arr[ud_ctr];
                                   check2=cmul_arr[ud_ctr+1];
                                   if(check1<check2)       // < 1.0
                                     {
                                      led[0]=0x7F;
                                      led[1]=dec_to_disp(check1);
                                      }
                                   else
                                     {                     // >=1.0
                                      div1=cmul_arr[ud_ctr];
                                      div2=cmul_arr[ud_ctr+1];
                                      led[0]=dec_to_disp(div1/div2);
                                      led[1]=dec_to_disp(div1%div2);
                                      }
                                   flag_reg_3.MSI=0;     //?????
```

```
                                    flag_reg_4.MSC=1;     //Menu_2 Item
Selected
                                    }
                            else if(flag_reg_3.HISET==1)
                                   {
                                    check1=cmul_arr[ud_ctr];
                                    check2=cmul_arr[ud_ctr+1];
                                    if(check1<check2)       // < 1.0
                                      {
                                       led[0]=0x7E;
                                       led[1]=dec_to_disp(check1);
                                       }
                                    else
                                      {                     // >=1.0
                                       div1=cmul_arr[ud_ctr];
                                       div2=cmul_arr[ud_ctr+1];
                                       led[0]=dec_to_disp(div1/div2);
                                       led[1]=dec_to_disp(div1%div2);
                                       }
                                    flag_reg_3.MSI=0;     //?????
                                    flag_reg_4.MSC=1;     //Menu_2 Item
Selected
                                    }
                            else
                                   {
                                    NOP();
                                    }
                            }
                     else                          //Dummy Enter
                           {
                            NOP();
                            }
                     break;
                     }

       case 4: {                             //************** SEL
*****************
                    if(flag_reg_3.MSI==0)        //Menu1
                          {
                           led[0]=menu_array[s_ctr];
                           led[1]=menu_array[s_ctr+1];
                           if(s_ctr==0)
                              {
                               flag_reg_3.DTI=1;    //Definite_time
                               flag_reg_3.ITI=0;
                               flag_reg_3.HSI=0;
                              }
                           if(s_ctr==2)
                              {
                               flag_reg_3.DTI=0;    //Inverse_time
                               flag_reg_3.ITI=1;
                               flag_reg_3.HSI=0;
                              }
                           if(s_ctr==4)
                              {
                               flag_reg_3.DTI=0;    //High_set
                               flag_reg_3.ITI=0;
                               flag_reg_3.HSI=1;
                              }
                           s_ctr++;
```

```
                        s_ctr++;
                        if(s_ctr>=6)
                            s_ctr=0;          //!!!!!!!!!!!!!!
                        }
                else                          //Menu2
                      {
                       led[0]=ts_cs_arr[s_ctr];
                       led[1]=ts_cs_arr[s_ctr+1];
                       if(s_ctr==0)
                           {
                            flag_reg_3.TSI=1;     //Time Multiplier
Selection
                            flag_reg_3.CSI=0;
                            }
                       if(s_ctr==2)
                           {
                            flag_reg_3.TSI=0;    //Current Mult.
Selection
                            flag_reg_3.CSI=1;
                            }
                       s_ctr++;
                       s_ctr++;
                       if(s_ctr>=4)
                           s_ctr=0;
                        }
                 break;
                 }

      case 8:  {
//*************DOWN******************
                 if(flag_reg_3.TSI==1)        //Time Multiplier Values
                        {
                         check1=tmul_arr[ud_ctr];
                         check2=tmul_arr[ud_ctr+1];
                         if(check1<check2)       // < 1.0
                                {
                                 led[0]=0x7F;
                                 led[1]=dec_to_disp(check1);
                                 }
                         else
                                {
// >=1.0
                                 div1=tmul_arr[ud_ctr];
                                 div2=tmul_arr[ud_ctr+1];
                                 led[0]=dec_to_disp(div1/div2);
                                 led[1]=dec_to_disp(div1%div2);
                                 }
                         if(ud_ctr==0)
                              ud_ctr=18;
                         ud_ctr--;
                         ud_ctr--;
                         }
                 else if(flag_reg_3.CSI=1)    //Current Multiplier
Values
                        {
                         check1=cmul_arr[ud_ctr];
                         check2=cmul_arr[ud_ctr+1];
                         if(check1<check2)       // < 1.0
                                {
                                 led[0]=0x7F;
```

```
                                        led[1]=dec_to_disp(check1);
                                        }
                            else
                                        {
// >=1.0
                                        div1=cmul_arr[ud_ctr];
                                        div2=cmul_arr[ud_ctr+1];
                                        led[0]=dec_to_disp(div1/div2);
                                        led[1]=dec_to_disp(div1%div2);
                                        }
                            if(ud_ctr==0)
                                  ud_ctr=36;
                            ud_ctr--;
                            ud_ctr--;
                            }
                  else if(flag_reg_3.HISET==1)
                           {
                            check1=cmul_arr[ud_ctr];
                            check2=cmul_arr[ud_ctr+1];
                            if(check1<check2)       // < 1.0
                                        {
                                        led[0]=0x7F;
                                        led[1]=dec_to_disp(check1);
                                        }
                            else
                                        {
// >=1.0
                                        div1=cmul_arr[ud_ctr];
                                        div2=cmul_arr[ud_ctr+1];
                                        led[0]=dec_to_disp(div1/div2);
                                        led[1]=dec_to_disp(div1%div2);
                                        }
                            if(ud_ctr==0)
                                  ud_ctr=36;
                            ud_ctr--;
                            ud_ctr--;
                            }
                  else
                            {
                             NOP();
                             }
                   break;
                   }

       case 16:  {
//*************Up*********************
                   if(flag_reg_3.TSI==1)     //Time Multiplier Values
                            {
                             check1=tmul_arr[ud_ctr];
                             check2=tmul_arr[ud_ctr+1];
                             if(check1<check2)       // < 1.0
                                        {
                                        led[0]=0x7F;
                                        led[1]=dec_to_disp(check1);
                                        }
                             else
                                        {
// >=1.0
                                        div1=tmul_arr[ud_ctr];
                                        div2=tmul_arr[ud_ctr+1];
```

```
                         led[0]=dec_to_disp(div1/div2);
                         led[1]=dec_to_disp(div1%div2);
                         }
                  if(ud_ctr==18)
                      ud_ctr=0;
                  ud_ctr++;
                  ud_ctr++;
                  }
           else if(flag_reg_3.CSI==1)  //Curr. Multiplier
Values
                 {
                  check1=cmul_arr[ud_ctr];
                  check2=cmul_arr[ud_ctr+1];
                  if(check1<check2)        // < 1.0
                        {
                         led[0]=0x7F;
                         led[1]=dec_to_disp(check1);
                         }
                  else
                        {
// >=1.0
                         div1=cmul_arr[ud_ctr];
                         div2=cmul_arr[ud_ctr+1];
                         led[0]=dec_to_disp(div1/div2);
                         led[1]=dec_to_disp(div1%div2);
                         }
                  if(ud_ctr==36)
                      ud_ctr=0;
                  ud_ctr++;
                  ud_ctr++;
                  }
           else if(flag_reg_3.HISET==1)
                 {
                  check1=cmul_arr[ud_ctr];
                  check2=cmul_arr[ud_ctr+1];
                  if(check1<check2)        // < 1.0
                        {
                         led[0]=0x7F;
                         led[1]=dec_to_disp(check1);
                         }
                  else{                                          //
>=1.0
                         div1=cmul_arr[ud_ctr];
                         div2=cmul_arr[ud_ctr+1];
                         led[0]=dec_to_disp(div1/div2);
                         led[1]=dec_to_disp(div1%div2);
                         }
                  if(ud_ctr==36)
                      ud_ctr=0;
                  ud_ctr++;
                  ud_ctr++;
                  }
           else
                 {
                  NOP();
                  }
           break;
           }
```

```
        case 1:    {
//************DISP*******************
                     break;
                      }

        }
}
/*********************************/
unsigned int dec_to_disp(unsigned int dec_num)
{                                // Converts decimal Numbers to 7_segment
Code
 unsigned int ret_val;
 switch(dec_num)
              {
              case 0:{ret_val=0b01111110;break;}
              case 1:{ret_val=0b00010010;break;}
              case 2:{ret_val=0b10111100;break;}
              case 3:{ret_val=0b10110110;break;}
              case 4:{ret_val=0b11010010;break;}
              case 5:{ret_val=0b11100110;break;}
              case 6:{ret_val=0b11101110;break;}
              case 7:{ret_val=0b00110010;break;}
              case 8:{ret_val=0b11111110;break;}
              case 9:{ret_val=0b11110110;break;}
              }

 return(ret_val);
 }
/*********************************/
void Page_Cross(void)
{
 #asm
 org 0x7F5
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 nop
 #endasm
  }
/*********************************/
void calculate_RMS_voltage(void)
{                                      //Calculate the RMS value of Line
voltage
 normalized_voltage=volt_arr[voltage_sample_ctr]-127;
 square_volt=square_volt+(normalized_voltage)^2;
 if(voltage_sample_ctr>=20)
        {
         mean_volt=square_volt/voltage_sample_ctr;
```

```
          RMS_voltage=get_sqrt(mean_volt);
          }
 }
/********************************/
void calculate_RMS_current(void)
{                                    //Calculate the RMS value of Line
current
 normalized_current=curr_arr[current_sample_ctr]-127;

square_curr=square_curr+(normalized_current*scaling_arr[current_sample
_ctr])^2;
 if(voltage_sample_ctr>=20)
         {
          mean_curr=square_curr/current_sample_ctr;
          RMS_current=get_sqrt(mean_curr);
          }

 }
/********************************/
unsigned long get_sqrt(unsigned long sum)
{
 odd_number=1;
 odd_ctr=1;
 do
   {
    sum=sum-odd_number;
    odd_number+=2;
    odd_ctr++;
    }
 while(sum=0||sum<odd_number)
 if(sum!=0)
        {
         converge=odd_number-sum;
            if(converge<sum)
                  RMS_value=odd_ctr+1;
         }
 else
        {
         RMS_value=odd_ctr;
         }
 return(RMS_value);
  }
/********************************/
void saturation_detection(void)
{
 }
/********************************/
void trip_relay(void)
{
 PORTC.0=1;
 PORTC.1=1;
 led[0]=0b10000000;
 led[1]=0b10000000;
     do
      {
       write_LED();
       }
     while(1)

 }
```

```
/********************************/
void load_TMR1(unsigned int value)
{
 }
/********************************/
void main()
{
// delay(5);                                        //(+)
 init_init();                                     //(+)
 init_LED();                                      //(+)
 NOP();
// INTCON.GIE=1;   // Activate interrupts
 do
  {
   write_LED();                                   //(+)
 //  ad_convert_current();
 //  ad_convert_voltage();
   get_keypad();
 //  calculate_RMS_voltage();
 //  calculate_RMS_current();
 //  calculate_curve();
 //  scale_adjust();
 //  saturation_detection();
   write_LED();
   }
 while(1)
}
/********************************/
const unsigned int NI_curve1[136]={0,0,0,0,0,0,0,13,
                24,34,43,50,58,65,71,77,82,87,92,97,101,105,109,

113,116,120,123,126,129,132,135,138,141,144,146,149,151,153,

156,158,160,162,165,167,169,171,173,174,176,178,179,181,183,

185,186,188,190,191,193,194,195,197,198,200,201,202,204,205,

206,207,209,210,212,212,214,215,216,217,218,219,221,221,223,

224,225,226,227,227,229,230,231,232,233,234,235,236,237,238,

239,240,240,242,242,243,244,245,246,246,247,248,249,250,250,

251,252,252,253,254,254,255,255,255,255,255,255,255,255,255,
                255,255,255,255,255,255,255,255};
 const unsigned int
NI_curve2[136]={0,0,0,0,0,0,0,13,24,34,43,50,58,65,71,

77,82,87,92,97,101,105,109,113,116,120,123,127,128,133,136,

138,141,144,146,149,151,153,156,158,160,162,164,167,169,171,

172,174,176,178,180,181,183,185,186,188,190,191,192,194,195,

197,198,199,201,202,204,205,206,208,209,210,211,212,213,215,

216,217,218,219,221,221,222,224,225,226,227,227,228,229,230,

232,233,234,235,236,237,237,238,239,240,241,242,243,244,244,
```

```
245,246,247,248,248,249,250,250,251,252,253,253,254,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255};

/*********************************/
void calculate_curve(void)
{
 if(flag_reg_3.MODES==0)                          //Definite_time
        {
         if(RMS_current>=90*Curr_Mult*HiSet_Mult) //!!!!!!
//Inst.Trip
                   {                              //I>>
                    PORTB.4=1;                    // LED Indicator
                    trip_relay();
                    }
         else
if((RMS_current>18)&&(RMS_current<90*Curr_Mult*HiSet_Mult))   //Timed
Trip    //!!!!
                   {
                    PORTB.5=1;
                    load_TMR1(0xFF*Time_Mult);     //!!!!!
                    }
         else
                   {
                    NOP();               //
                    }
        }
 else if(flag_reg_3.MODES==1)                     //Inverse_time
        {
         if(RMS_current>=90*Curr_Mult*HiSet_Mult) //!!!!!!
//Inst.Trip
                   {                              //I>>
                    PORTB.4=1;                    // LED Indicator
                    trip_relay();
                    }
         else
if((RMS_current>18)&&(RMS_current<90*Curr_Mult*HiSet_Mult))   //Timed
Trip    //!!!!
                   {
                    PORTB.5=1;
                       if(RMS_current*99/10<=255)
                              {
                               TMR1_value=NI_curve1[RMS_current*99/10];
                               load_TMR1(TMR1_value);
                               }
                       if((Mode_Setting.0==1)&&(RMS_current*99/10>255))
//NI
                              {
                               TMR1_value=NI_curve2[RMS_current*99/10-
255];

                               load_TMR1(TMR1_value);
                               }
                   }
        }
 else
        {
         NOP();                                    //No operation
         }
 }
```

```
/********************************/

// flag_reg_1:
// -----------
// ICS    (BIT0) : Initiate_Current_Sampling
//                 1:First I_ZCD detected,get current samples
//                 0:First I_ZCD not detected do not get current
samples
// IVS    (BIT1) : Initiate_Voltage_Sampling
//                 1:First V_ZCD detected,get voltage samples
//                 0:First V_ZCD not detected do not get voltage
samples
// ADDLY  (BIT2) : Delay Between two Samples
//                 1:CPU waiting for the delay between samples
//                 0:CPU ready for next sampling
// ADSEQ  (BIT3) : AD sequence indicator
//                 1: Next Conversion for Voltage
//                 0: Next Conversion for Current

// flag_reg_2:
// -----------
// DISP   (BIT0) : 1:'Display' button pressed
// ENTER  (BIT1) : 1:'ENTER' button pressed
// SEL    (BIT2) : 1:'SEL' button pressed
// DOWN   (BIT3) : 1:'DOWN' button pressed
// UP     (BIT4) : 1:'UP' button pressed

// flag_reg_3:
// -----------
// MSI    (BIT0) : Menu Selection Indicator
//                 1: 1st menu item selected
//                 0: 1st menu item not selected
// DTI    (BIT1) : Definite Time Indicator
//                 1: Valid settings for definite_time operation
// ITI    (BIT2) : Inverse Time Indicator
//                 1: Valid settings for inverse_time operation
// HSI    (BIT3) : High Set Indicator
//                 1: Valid settings for high set operation
// TSI    (BIT4) : Time Setting Indicator
//                 1:Get time setting values
// CSI    (BIT5) : Current Setting Indicator
//                 1: Get Current Setting Values
// MODES  (BIT6) : Mode Select Indicator
//                 0: definite-time(default)
//                 1: inverse-time
// HISET  (BIT7) : High Set Indicator
//                 1: Get High Set Settings

// flag_reg_4:
// -----------
// MSC    (BIT0) : Menu Selection Complete
//                 1:Menu_selection Complete, get setting.
//                 0:Menu_selection not Complete.
//

// Mode_Setting:
// -------------
// NI    (BIT0)   : 1: Normal Inverse Operation
// VI    (BIT1)   : 1: Very Inverse Operation
```

```
// EI   (BIT2)   : 1: Extremely Inverse Operation   //To be modified
later!!!!!
```