VISVESVARAYA TECHNOLOGICALUNIVERSITY
JnanaSangama, Belagavi, Karnataka-590014

Project report on

DESIGN, DEVELOPMENT, DEMONSTRATION AND IMPLEMENTATION OF TM TC SIMULATOR

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING
in
ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

SURAJ RAO R.S	**1BI14EC167**
TEJAS.P	**1BI14EC172**
VINAY M.S	**1BI14EC180**
YASHASWINI.R	**1BI14EC186**

Carried out at
ISRO Satellite Integration and Test Establishment (ISITE)
Bangalore
Under the guidance of

Internal Guide
Mrs. RADHA B L
Associate Professor
Dept. of ECE, BIT
Bangalore

External Guide
Mrs. YOGITA M
Mrs. SREEDEVI
Scientist/Engineer- 'SD'
PSTES/TED, ISITE
Bangalore

Department of Electronics & Communication Engineering
BANGALORE INSTITUTE OFTECHNOLOGY
K .R. Road, V.V Puram, Bengaluru-560004

2017-2018

BANGALORE INSTITUTE OF TECHNOLOGY

V. V Puram, K R Road, Bengaluru-560004

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

Certified that the project report entitled **"Design, Development, Demonstration and Implementation of TM TC Simulator"** carried by **Mr. SURAJ RAO R S (USN:1BI14EC167), Mr. TEJAS P (USN:1BI14EC172), Mr. VINAY M S (USN:1BI14EC180) and Ms. YASHASWINI R (USN:1BI14EC186)**, bonafide students of **Bangalore Institute of Technology**, in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of Visvesvaraya Technological University, Belgaum, during the year 2017-2018. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

Guide: **RADHA B.L** Associate Professor, Dept. of ECE ,BIT	**Dr. K. V.PRASAD** Professor and HOD, Dept. of ECE,BIT	**Dr. A G NATARAJ** Principal, BIT

External Viva

1.

2.

भारत सरकार
अन्तरिक्ष विभाग
इसरो उपग्रह केन्द्र
पोस्ट बॉक्स नं. 1795, हवाई पत्तन मार्ग
विमानपुरा डाक घर, बेंगलूरु – 560 017. भारत
दूरभाष :
फेक्स :

सत्यमेव जयते

Government of India
Department of Space
ISRO Satellite Centre
Post Box. No. 1795, Airport Road, Vimanapura Post
Bangalore - 560 017. India
Telephone :
Fax :

CERTIFICATE

This is to certify that the project work entitled **"Design, Development, Demonstration and Implementation of Microcontroller based Telemetry Tele-command Simulator"** carried out by **Mr. Suraj Rao R S , Mr. Tejas P, Mr. Vinay M S and Ms. Yashaswini R**, bonafide students of Bangalore Institute of Technology, Bangalore in partial fulfilment for the award of **Bachelor of Engineering** in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year 2018. It is certified that they have carried out the Industrial training satisfactorily in Systems Engineering Group, ISRO Satellite Centre, Bangalore from "January 2018" to "April 2018".

27/04/18

Guide(s)

Ms. Yogita M & Ms. Sreedevi P
Scientist /Engineer-SD, PSTES
Test and Evaluation Division
Systems Engineering Group
U R Rao Satellite Centre, Bangalore

27.04.2018

Mr. Shivkumar S Patil
Head, PSTES
Test and Evaluation Division
Systems Engineering Group
U R Rao Satellite Centre, Bangalore

Ms. Selvi R
Head, TED
Systems Engineering Group
U R Rao Satellite Centre, Bangalore

Mr. A V Nirmal
Group Director
Systems Engineering Group
U R Rao Satellite Centre, Bangalore

DECLARATION

We the students of 8th Semester B.E., Electronics and Communication Engineering, Bangalore Institute of Technology, hereby declare the project entitled "DESIGN, DEVELOPMENT, DEMONSTRATION AND IMPLEMENTATION OF TM TC SIMULATOR" which is being submitted by us as partial fulfillment for the award of degree in Bachelor of Engineering in Electronics and Communication Engineering of Visvesvaraya Technological University, Belagavi, during the year 2017-2018. It is an authenticate record of our own work carried during 8th semester B.E. under the supervision of our internal guide Mrs. RADHA B.L, Associate Professor, Dept. of ECE, BIT, Bengaluru and External guide Mrs. SREEDEVI P and Mrs. YOGITHA M, Scientist/Engineer, PSTES/TED, ISITE, Bengaluru-560017.

SURAJ RAO R S	USN: 1BI14EC167
TEJAS P	USN: 1BI14EC172
VINAY M S	USN: 1BI14EC180
YASHASWINI R	USN: 1BI14EC186

ABSTRACT

The Telemetry and Tele-Command is partly on satellite and is partly at controlling Earth station. The TM system sends data from the sensors and various other packages from the satellite to the Earth station. The TC system sends commands to the satellite from the Earth station. These commands include parameters to regulate satellites altitude, azimuth angle and other orbital parameters. In order to simulate the TM TC actions during the test and Integration phase at the ground station, TM TC Simulator is used.

Previously implemented via discrete logics, now it is implemented using a microcontroller. This simulator should be able to generate clock at various frequencies and be able to send address/data (which ever applicable) in synchronization with the clock.

ACKNOWLEDGEMENT

The portion of success is brewed by the efforts put in by the individuals. It is constant support provided by people who give you the initiative, who inspire you at each step of your endeavour that eventually helps you in your goal.

We heartily extend our words of gratitude to our internal guide **Radha B.L**, Associate Professor, Dept. of ECE, BIT for the way she motivated us, for the way she directed us and for being with us all throughout the project by rendering her valuable guidance, without which we could not have been able to accomplish the project with ease.

We thank **Dr. K V Prasad**, HOD, dept. of ECE, BIT, for his overwhelming support, encouragement and valuable guidance throughout the project.

I would like to acknowledge, in particular **Dr.A.G.Nataraj**, principal, B.I.T for his invaluable support during this endeavour and the freedom he gives to think and timely support in all forms he extends.

We wish to express our deep gratitude and heartily appreciation for the invaluable guidance of **Shivakumar S Patil** throughout the span of preparing this project. Their excellent guidance made us to complete this task successfully within a short duration.

We show our immense gratitude to **P.Sreedevi** and **Yogita M** for their encouragement and timely support that they extended throughout the project course.

We thank our parents for their additional support and interest during the project which has really helped us to accomplish our destiny on time.

We thank all our friends for sharing their comments and suggestions which has really helped us to accomplish our destiny on time.

SURAJ RAO R S USN: 1BI14EC167

TEJAS P USN: 1BI14EC172

VINAY M S USN: 1BI14EC180

YASHASWINI R USN: 1BI14EC186

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

TM	Telemetry
TC	Tracking
DTH	Direct-to-home
VSAT	Very Small Aperture Terminal
DSNG	Digital Satellite News Gathering
TTC	Tracking, Telemetry and Command
AOCS	Attitude and Orbit Control System
AKM	Apogee Kick Motor
GPS	Global Positioning System
ALE	Address Latch Enable
ISP	In-System Programming
IAP	In-Application Programming
GPIO	General Purpose Input Output
DMA	Direct Memory Access
VIC	Vectored Interrupt Controller
RTC	Real Time Clock
USB	Universal Serial Bus
PLL	Phase Lock Loop
TC	Timer Counter
PR	Pre-scale Register
CTCR	Count Control Register
IDE	Integrated Development Environment
EDA	Electronic Design Automation
UART	Universal Asynchronous Receiver Transmitter

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Introduction:

Satellite Communication utilisation has now become wide spread and ubiquitous throughout the country for such diverse applications like Television, DTH Broadcasting, DSNG and VSAT to exploit the unique capabilities in terms of coverage and outreach. The technology has matured substantially over past three decades and is being used on commercial basis for a large number of applications. Most of us are touched by satellite communication in more ways than we realise. The potential of the space technology for applications of national development is enormous.

Satellites are used for many purposes. Common types include military and civilian Earth observation satellites, communications satellites, navigation satellites, weather satellites, and space telescopes. Space stations and human spacecraft in orbit are also satellites. Satellite orbits vary greatly, depending on the purpose of the satellite, and are classified in a number of ways. Well-known (overlapping) classes include low Earth orbit, polar orbit, and geostationary orbit. A launch vehicle is a rocket that throws a satellite into orbit. Usually it lifts off from a launch pad on land. Some are launched at sea from a submarine or a mobile maritime platform, or aboard a plane (see air launch to orbit).Satellites are usually semi-independent computer-controlled systems. Satellite subsystems attend many tasks, such as power generation, thermal control, telemetry, attitude control and orbit control.

1.2 Types of satellites:

Satellites are usually classified according to the type of orbit they are in. There are four types of orbits associated with satellites. And the type of orbit dictates the satellite's use.

1.2.1 Geostationary or geosynchronous earth orbit (GEO) satellite:

GEO satellites are synchronous with respect to earth. Looking from a fixed point from Earth, these satellites appear to be stationary. These satellites are placed in the space in such a way that only three satellites are sufficient to provide connection throughout the surface of the Earth (that is; their footprint is covering almost 1/3rd of the Earth). The orbit

of these satellites is circular. There are three conditions which lead to geostationary satellites. Lifetime expectancy of these satellites is 15 years.

The satellite should be placed 37,786 kms (approximated to 36,000 kms) above the surface of the earth. These satellites must travel in the rotational speed of earth, and in the direction of motion of earth, that is eastward. The inclination of satellite with respect to earth must be 0^0. Geostationary satellite in practical is termed as geosynchronous as there are multiple factors which make these satellites shift from the ideal geostationary condition.

1.2.2 Low Earth Orbit (LEO) satellites:

These satellites are placed 500-1500 kms above the surface of the earth. As LEOs circulate on a lower orbit, hence they exhibit a much shorter period that is 95 to 120 minutes. LEO systems try to ensure a high elevation for every spot on earth to provide a high quality communication link. Each LEO satellite will only be visible from the earth for around ten minutes.

1.2.3 Medium Earth Orbit (LEO) satellites:

MEOs can be positioned somewhere between LEOs and GEOs, both in terms of their orbit and due to their advantages and disadvantages. Using orbits around 10,000 km, the system only requires a dozen satellites which is more than a GEO system, but much less than a LEO system. These satellites move more slowly relative to the earth's rotation allowing a simpler system design (satellite periods are about six hours). Depending on the inclination, a MEO can cover larger populations, so requiring fewer handovers.

1.2.4 Sun- Synchronous Orbits satellites:

These satellites rise and set with the sun. Their orbit is defined in such a way that they are always facing the sun and hence they never go through an eclipse. For these satellites, the surface illumination angle will be nearly the same every time. (Surface illumination angle: The illumination angle is the angle between the inward surface normal and the direction of light. This means that the illumination angle of a certain point of the Earth's surface is zero if the Sun is precisely overhead and that it is 90 degrees at sunset and at sunrise.) Special cases of the sun-synchronous orbit are the

noon/midnight orbit, where the local mean solar time of passage for equatorial longitudes is around noon or midnight, and the dawn/dusk orbit, where the local mean solar time of passage forequatorial longitudes is around sunrise or sunset, so that the satellite rides the terminator between day and night.

1.3 Frequency Allocation:

Frequency bands are allocated to the above radio communications services to allow compatible use. The allocated bands can be either exclusive for a given service, or shared among several services. Allocations refer to the following division of the world into three regions:

- Region 1: Europe, Africa, the Middle East, the former USSR;
- Region 2: the Americas;
- Region 3: Asia Pacific, except the Middle East and the former USSR.

The bands above 30GHz will be used eventually in accordance with developing requirements and technology. Table 1.3 summarises the above discussion. The mobile satellite service makes use of the following bands:

- VHF (very high frequency, 137–138MHz downlink and 148–150MHz uplink) and UHF (ultrahigh frequency, 400–401MHz downlink and 454–460MHz uplink). These bands are for non-geostationary systems only.
- About 1.6GHz for uplinks and 1.5GHz for downlinks, mostly used by geostationary systems such as INMARSAT; and 1610–1626.5MHz for the uplink of non-geostationary systems such as GLOBALSTAR.
- About 2.2GHz for downlinks and 2GHz for uplinks for the satellite component of IMT2000 (International Mobile Telecommunications).
- About 2.6GHz for uplinks and 2.5GHz for downlinks.

Frequency bands have also been allocated at higher frequencies such as Ka band. The broadcasting satellite service makes use of downlinks at about 12 GHz. The uplink is operated in the FSS bands and is called a feeder link. Table 1.3 summarises the main frequency allocation and indicates the correspondence with some usual terminology.

Radio communications Service	Typical frequency bands for uplink/downlink	Usual terminology
Fixed satellite service	6/4GHz	C band
	8/7GHz	X band
	14/12–11GHz	Ku band
	30/20GHz	Ka band
	50/40GHz	V band
Mobile satellite service	1.6/1.5GHz	L band
	30/20GHz	Ka band
Broadcasting satellite Service	2/2.2GHz	S band
	12GHz	Ku band
	2.6/2.5GHz	S band

Table 1.1 Frequency Allocations

1.4 Configuration of a Satellite Communication System:

Figure 1.1 gives an overview of a satellite communication system and illustrates its interfacing with terrestrial entities. The satellite system is composed of a space segment, a control segment and a ground segment:

The space segment contains one or several active and spare satellites organised into a constellation.

The control segment consists of all ground facilities for the control and monitoring of the satellites, also named TTC (tracking, telemetry and command) stations, and for the management of the traffic and the associated resources on-board the satellite.

The ground segment consists of all the traffic earth stations. Depending on the type of service considered, these stations can be of different size, from a few centimetres to tens of metres.

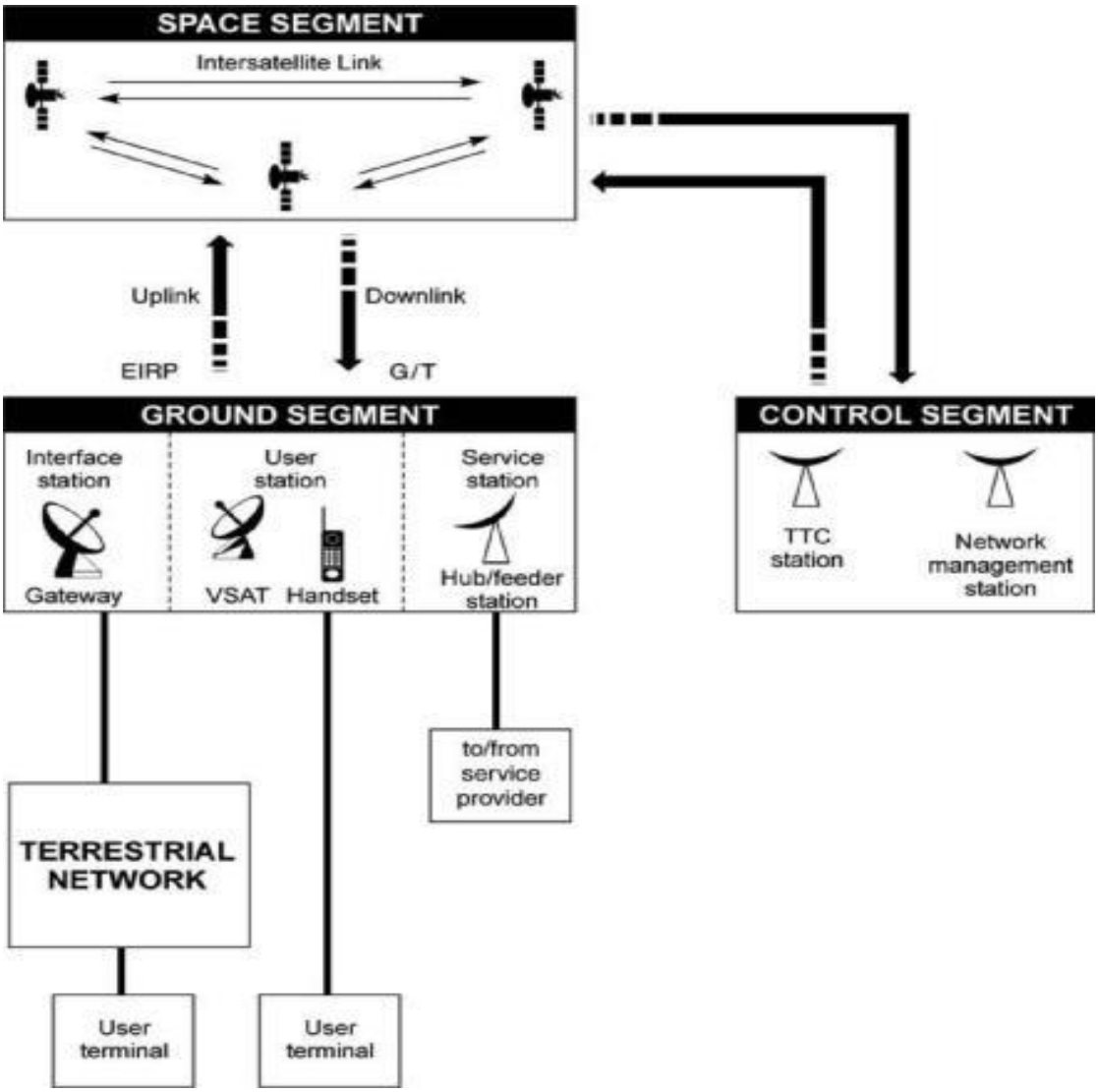


Fig. 1.1 Satellite communication system, interfacing with terrestrial entities.

1.5 Satellite Subsystem:

A satellite usually has five major subsystems: Attitude and Orbit Control System (AOCS), Telemetry, Tracking, Command, and Monitoring (TTC&M), power system, communication subsystems, and satellite antennas.

1.5.1 Attitude and Orbit Control System (AOCS):

The AOCS consists of rocket motors to move the satellite back to the correct orbit when external forces cause it to drift and gas jets or internal devices to control the attitude of the satellite.

1.5.2 Communication Subsystems:

The communication subsystem is the major component of a communication satellite, although, frequently, the communication equipment is only a small part of the weight and volume of the whole satellite. It is usually composed of one or more antennas and a set of receivers and transmitters that amplify and retransmit the incoming signal. A receiver-transmitter unit is called as a transponder.

1.5.3 Power System:

Most communication satellites derive their electric power from solar cells. The power is used by the communication system, mainly in its transmitter, and other electrical systems in the satellite. The latter use is called housekeeping.

All communications satellites obtain their electric power from solar cells which convert incident sunlight into electrical energy. Some deep space planetary research spacecraft's have used thermonuclear generators to supply electrical power. But because of the danger to people on the earth in case of launch fail and consequent nuclear spread, communications satellites have not used nuclear generators. Solar radiation falling on a satellite at geostationary orbit has an intensity of 1.39 kW/m^2. The efficiency of solar cell is typically 20 to 25 % at the beginning of life (BOL), but falls with time due to aging of cells and etching of the surface by micrometeor impacts. Since sufficient power must be available at the end of life (EOL) of the satellite to supply all the systems on board, about 15 % extra area of solar cells is usually provided as an allowance for aging. A spin-stabilized satellite has a cylindrical body covered with solar cells.

Because the cylindrical shape, half of the solar cells are not illuminated at all, and at the edges of the illuminated half, the low angle of incidence of sunlight results in little electric power being generated. Recently a large communication satellite for direct broadcasting generates up to 6 kW from solar power. A three-axis stabilized satellite can make better use of its solar cell areas than a spinner, since solar sails can be rotated to maintain normal incidence of the sunlight to the cells. Only one-third of the total area of solar cells is needed relative to a spinner, with some saving in weight. With large arrays, a three-axis stabilized satellite can have more than 10 kW of power generated. In a three-axis stabilized satellite, solar sails must be rotated by an electric motor once every 24 hours to keep the cells in full sunlight. This causes the cells to heat up, typically to 50^o to 80^oC, which

causes a drop in output voltage. In a spinner design, the cells cool down when in shadow and run at 20^o to 30^oC, with somewhat higher efficiency than a three-axis stabilized satellite.

The satellite must carry batteries to provide power during launch and eclipses. On geostationary orbit, eclipses occur during two periods per year, around the spring and fall equinoxes, with longest duration of about 70 min/day. To avoid the need for large, heavy batteries, a part of communication system load may be shut down during eclipse, but this technique is rarely used when telephony or data traffic is carried. However, TV broadcasting satellites will not carry sufficient battery capacity to supply their high-power transmitters during eclipse, and must shut down during eclipse. By locating the satellite 20^o (which is equivalent to 1 hour and 20 minutes) west of the longitude of the service area, the eclipse will occur after 1 a.m. local time for the service area, when shut down is more acceptable.

Batteries are usually of the sealed nickel hydrogen type which do not gas when charging and have good reliability and long life, and can safely discharge to 70 % of their capacity. Typical battery voltages are 20 to 50 V with capacities of 20 to 100 ampere-hours.

1.5.4 Telemetry, Tracking, Command, and Monitoring (TTC&M):

The TTC&M system is partly on the satellite and partly at the controlling earth station. The telemetry system sends data received from sensors on the satellite to monitor the satellite's health, via telemetry link to the controlling earth station. The tracking system at the earth station provides information on the range, elevation, and azimuth of the satellite needed in computing orbital elements. Based on telemetry data received from the satellite through the telemetry system and orbital data obtained from the tracking system, the control system corrects the antenna positioning and communication system configuration to suit current traffic requirements and to operate switches on the satellite.

The TT&C system is essential to the successful operation of a communication satellite. The main functions of satellite management are to control the orbit and attitude of the satellite, monitor the status of all sensors and subsystems on the satellite, and switch on or off sections of the communication system. On large geostationary satellites, some re-pointing of individual antennas is also possible, under the command of the TT&C system. Tracking is performed primarily by the earth station. Figure 1.1 shows the functions of a controlling earth station.

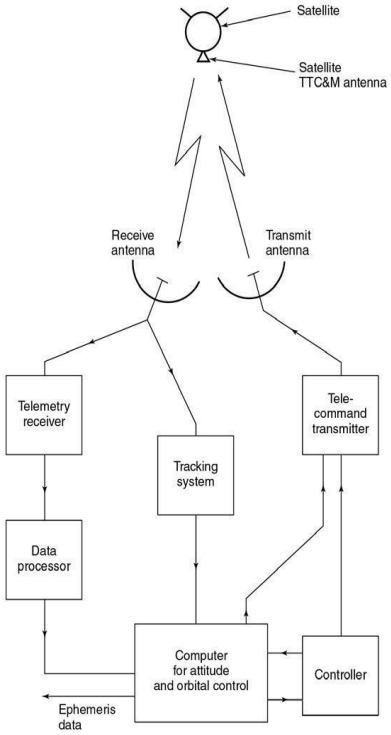


Fig. 1.2 Typical tracking, telemetry, tele-command and monitoring system.

1.5.4.1 Telemetry and Monitoring System:

The telemetry system at the satellite collects data from many sensors within satellite and sends them to the controlling earth station. As many as 100 or more sensors monitor pressure on fuel tanks, voltage and current in the power conditioning unit, critical voltage and current in communication electronics, temperature of subsystems, status of subsystems, positions of switches, and sighting device for altitude control. Telemetry data is usually transmitted as FSK or PSK of low-power telemetry carrier using time division techniques. Low data rate is normally used to allow the earth station receiver to have a narrowbandwidth and thus maintain high carrier-to-noise ratio (CNR). An entire TDM frame may contain thousands of bits of data and take several seconds to transmit.

1.5.4.2 Tracking:

The tracking system at the control earth station provides information about the range, elevation, and azimuth for a satellite. Data from velocity and acceleration sensors on the satellite can be used to establish the change in the orbit from the last known position by integrating data. Doppler shift observation at the control earth station provides the rate of range change. Active determination of range can be achieved by transmitting pulses to the satellite from the control earth station and measuring its round-trip delay. The position of a satellite can be measured by triangulation from multiple earth stations. The position of a satellite can be determined within 100 m.

1.5.4.3 Command:

A secure and effective command structure is vital to the successful launch and operation of any communication satellite. The command system makes changes the position and attitude of the satellite, controls antenna positioning and communication system configuration, and operates switches at the satellite. During launch it is used to control the firing of the apogee kick motor (AKM) and to spin up a spinner or extend the solar sails of a three-axis stabilization satellite. The command structure must have safeguards against inadvertent operation of a control due to error. The control code is converted into a command word which is sent in a TDM frame. After checking for validity in the satellite, the command word is sent back to the control earth station via the telemetry link where it is checked again. If it is received correctly, then an execute instruction is sent to the satellite so that the command is executed.

The entire process may take 5 -10 seconds, but minimizes the risk of erroneous commands causing a satellite malfunction. The command and telemetry links are usually separated from the communication system, although they may operate in the same frequency band (for example, 6 and 4 GHz). Two levels of command system are used in the Intelsat satellite: the main system operates in the 6-GHz band, in a gap between the communication channel frequencies; the main telemetry system uses a similar gap in the 4 -GHz band.

The main system can be used only after correct attitude of the satellite is achieved. During the launch phase and injection into geostationary orbit, the main TTC&M system may be inoperable because the satellite does not have correct attitude or has not extended its solar sails. Near omni-directional antenna are used at either UHF or S-band (2-4GHz). For the backup system, a great deal of redundancy and sufficient margin in SNR at the satellite receiver are allowed to guarantee control under the most adverse conditions. The backup system provides control of the apogee kick motor, the attitude control system and orbit control thrusters, the solar sail deployment mechanism (if fitted), and power conditioning unit. It is also used to eject the satellite from the geostationary orbit and to switch off all transmitters at the end of its useful life.

Chapter 2
BACKGROUND OF COMPANY

2.1 Overview:

ISRO Satellite Centre, now newly christened as the U R Rao Satellite centre (URRSC) is the lead centre of the Indian Space Research Organisation (ISRO) responsible for design, development, assembly & integration of communication, navigation, remote sensing, scientific and small satellite missions. The specialised teams of scientists, engineers and technicians of URRSC have built more than 75 complex & advanced satellites for various applications in areas of telecommunications, television broadcasting, VSAT services, tele-medicine, tele-education, navigation, weather forecasting, disaster warning, search and rescue operations, earth observations, natural resource management, scientific and space science etc. With the objective of taking the benefits of space technology to the length & breadth of the society, URRSC is actively involved in creating cost-effective space infrastructure for the country.

2.2 Evolution of ISAC:

The establishment of Thumba Equatorial Launching Station (TERLS) in 1963 and the Experimental Satellite Communication Earth Station (ESCES) in 1967 was the prodigious precursors to Space activities in the country. Activities relating to satellite technology started in the right earnest at Satellite Systems Division at Space Science & Technology Centre, Trivandrum in the late sixties. Later when a conscious decision emerged in 1972 to build the first Indian Satellite 'Aryabhata' the scene shifted to Bangalore with the formulation of the Indian Scientific Satellite Project (ISSP). The Indian Institute of Science campus initially housed the project activities until it moved to the industrial sheds at Peenya. It was here that a handful of engineers and technicians fresh from the Universities sowed the first seeds of satellite technology in the country. With practically no prior art existing within the country, and with sparse infrastructure put together from scratch, this young team developed the first Indian Satellite ARYABHATA in the make shift industrial sheds at Peenya, Bangalore. With the success of the ARYABHATA mission, the fledgling space activity soon developed into a full-fledged programme with national priorities. Thus was born the ISRO Satellite Centre (**ISAC**) in 1976. In 1984 the Centre moved to the present 32 acre campus at Old Airport Road, Vimanapura in Bangalore.In April 2018 it was renamed as the URRSC in honour of Prof. U R Rao.

To cater to the growing need of satellite for various applications, ISRO Satellite Integration&Testing Establishment (**ISITE**) was established in 2006 in a 110 acre campus which is about 8 km away from the present campus. ISITE has a large clean room and state-of-the-art electronics fabrication and test facilities under one roof for the assembly, integration and testing of communication satellites.

2.3 Vision and Mission:

Vision: To harness space technology for national development, while pursuing space science research and planetary explorations.

Mission: Design and development of satellites and related technologies for providing access to space and Research and development in satellite related technologies.

Objective: The objective of ISRO Satellite Centre is building state of the art satellites and related technologies in the area of communication, navigation, earth observation, meteorology, space science and planetary exploration etc. in a cost effective manner for the socio-economic development of the country.

2.4 Missions:

2.4.1 Communication:

Communication Satellite provides the essential satellite based services to the nation, which comprises of Telecommunication, Television and Radio broadcasting, Meteorological imaging and weather forecasting, weather data collection and data dissemination, Disaster warning etc. For providing these services the space system has two segments viz. the space segment and the user ground segment. The space segment consists of the spacecraft at different orbital slots.

2.4.2 Deep Space and Science:

A Deep Space / Interplanetary mission involves a trip to the planet/s of our solar system outside of the earth. Interplanetary missions are expedited by using any or all of the following viz., Satellites, Lander Crafts and Rovers. The primary objective is to demonstrate technologies relating to communication; survival in deep space environments and on the planets; exploration of planets physical, chemical and atmosphere systems. The result from such explorations helps in the understanding of the planet’s atmosphere with respect to possible proximity to earth’s environment.

Plan for inhabitation / Colonization of the planets

- Advance Detection of catastrophic aspects arising in the solar system such as Sun Storms, Comets and Meteorite strikes etc.

2.4.3 Earth Observation:

The main objective of Earth observation satellites is to gather information about earth - physical, chemical and biological systems.

The application of Earth observation satellites are primarily in the areas of

- Cartography, Oceanography, Meteorology, Natural Resource Management & Space Science
- Rural Development and Urban Management
- Forest Cover and Bio resources
- Weather monitoring and forecasting
- River banks erosion and fresh water mapping
- Climate impact on bio-diversity and wild life and its migration pattern
- Monitoring and forecasting natural disasters such as floods, earthquakes, cloud burst, glacier avalanches
- Town / City planning for rails, roads, flyovers and bridges etc.
- Crop estimation and disease detection and loss estimation
- Monitoring of Fish Shoals for fisheries
- Mineralogy mapping and Monitoring

2.4.4 Navigation:

Satellite-based Navigation System has emerged as a forerunner in providing the positioning, navigation and timing (PNT) services to the users across the world. India has entered into the arena of satellite navigation with its two major projects viz. **GAGAN** (GPS Aided Geo Augmented Navigation), &**IRNSS** (Indian Regional Navigation Satellite System). GAGAN is a space-based augmentation for GPS developed jointly by ISRO and Airports Authority of India (AAI) to meet the stringent Civil Aviation Requirements. IRNSS is ISRO's initiative in developing India's indigenous regional navigation system providing PNT services to users in the Indian region.

2.4.5 Student's Satellites:

ISRO has been encouraging student community to participate in ISRO missions and learn space technology as capacity building effort to prepare the future space scientists and technologists as well as to develop future vendors in this area, who can design develop, fabricate, test, space technology sub-systems and units for consumption within the country as well as to become competitors in the world market. Over the past few decades of ISRO's space programme several universities / academic institutions have participated in the space related programmes as well as technology development. As part of this, several institutions were interested in introducing education in the space technology and subsequently space sciences have taken initiative and designed student satellites as well as developed payloads. In order to coordinate all the activities related to space technology projects by students in the university, especially student satellites, ISRO has established a mechanism to streamline these activities in the form of small satellites as part of Indian remote sensing programme.

Chapter 3

SYSTEM DEVELOPMENT

The Telemetry/Telecommand Transponder is the unit on board a satellite that maintains the communication link between the satellite and the control center on the Earth. The spacecraft includes service telecommunications Telemetry and Telecommand (TMTC) and On-Board Control (OBC) subsystem to provide telemetry, telecommand and ranging functions during all phases of the mission. The function of transponder, generally, is receiving the uplink signal modulated with telecommand information, process it, and transmit the downlink signal (at different frequency) modulated with telemetry and ranging signal to the ground station.

In order to emulate the transponder related functions mainly telemetry and telecommand during the integration and testing phase at the ground station a TM TC Simulator is designed using an ARM(advanced RISC machine) based microcontroller i.e. LPC2148.

The unit basically consists of a fabricated TM and TC card which is powered by a single regulated power supply (RPS) unit which gives 5V and 3.3V DC supply. TM and TC cards are controlled by the micro-controller which is powered ON by the 5V provided by the supply. The same power supply is used to power on IC's like level shifter, Hex buffer etc. the 3.3V supply is used to feed power to the DIP switches and push buttons.

The complete unit is generated in 2 sets and is placed inside a mechanical casing. The block diagram is as follows:

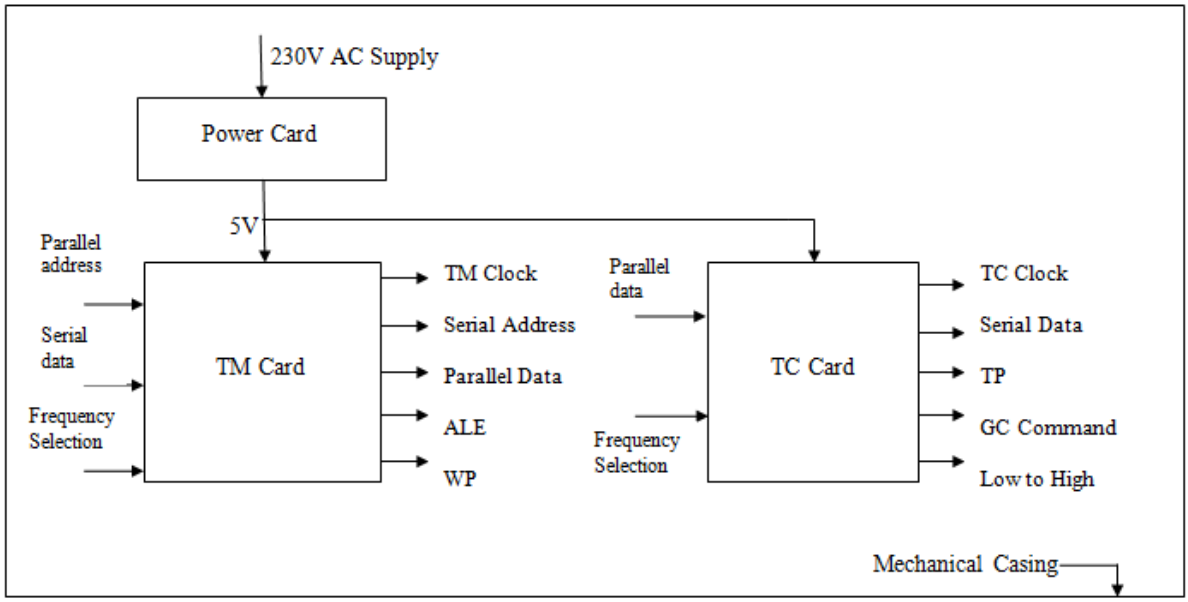


Fig3.1TM TC Simulator Block Diagram

3.1 Block Diagram

The telemetry and tele-command simulator mainly consist of a power supply unit, ARM7 microcontroller, level shifter and a hex buffer.

3.1.1 Telemetry System

The outputs obtained of telemetry are: the TM clock, TM word-pulse, TM ALE, 16Serial address and serial data of 16bits.

The block diagram of TM Simulator is as follows:

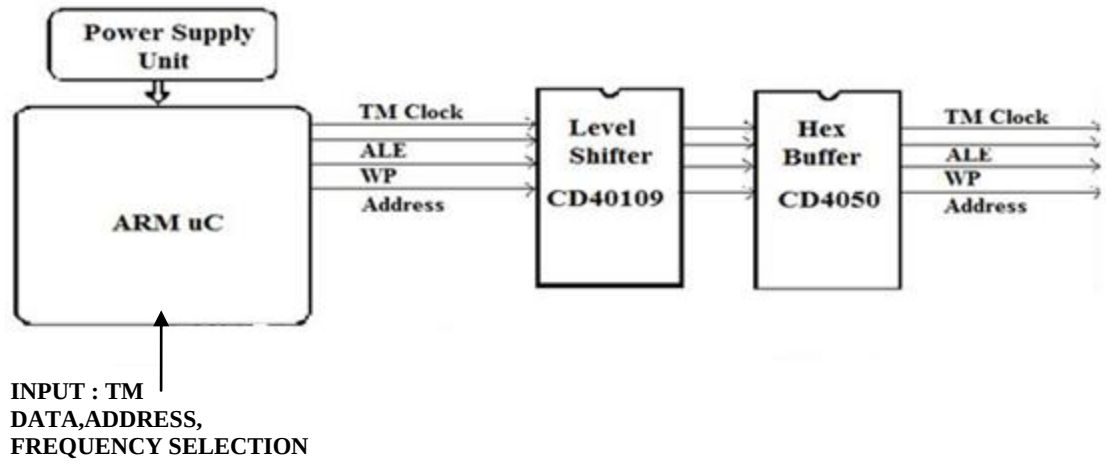


Fig 3.2 Block Diagram for Telemetry Section

The telemetry unit consists of a microcontroller which is powered by a regulated power supply card which is in turn powered by the 230v source.

The microcontroller produces free running clock signals.It gives out a 16bit parallel in (from DIP switches) serial out address.To enable the address being sent an ALE (Address latch enable) pulse is also sent.

It also takes a 16bit serial data and outputs it as a parallel data on the LED's in synchronization with the clock. A WP(Write Pulse) is also sent after successful completion of sending data. All these can be generated at desired clock of 1 kHz,2 kHz,16 kHz or 40 kHz frequency which can be selected from a frequency selection switch.

The outputs are all level shifted to 5V and read out via buffer CD4050.

The expected outputs are given by the following timing diagram:

3.1.2 Telecommand System

The outputs of Tele-command are a burst clock of 32 pulses, a transfer pulse, and a 32-bit of data. It also produces a High to low command and GC(Ground Checkout) command.

The block diagram is as follows:

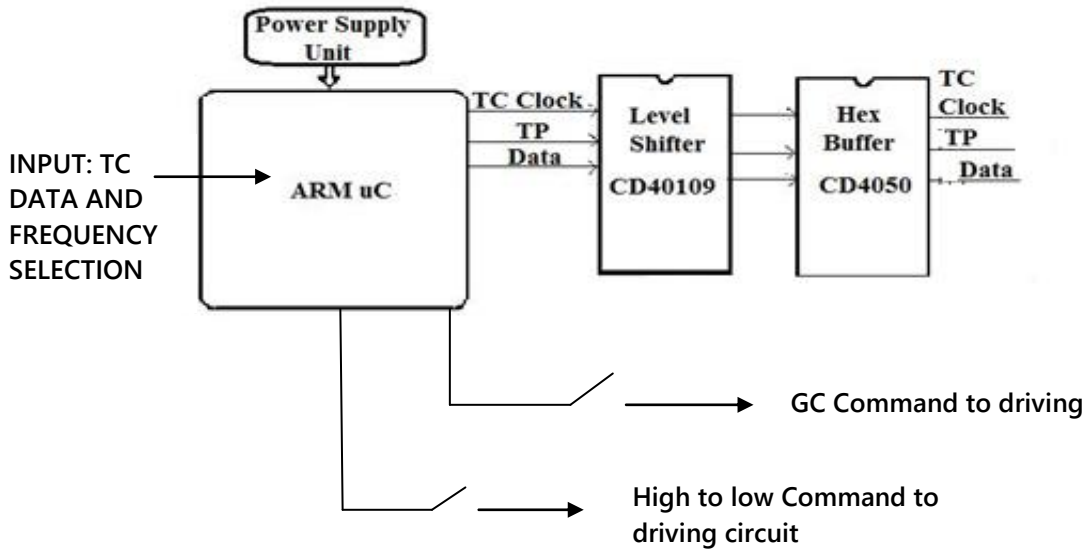


Fig 3.3 Block Diagram for Tele-command Section

The tele-command section also similarly has a microcontroller and outputs drawn via level shifter and a buffer.

Upon press of a push button,it produces a burst of 32 clock pulses to which 32 bit data is synchronised and a transfer pulse(TP) to indicate successful transfer.

The 32 bit data is parallely taken from dip switches and converted into serial format at microcontroller output pin. And the duration for which Transfer pulse is required can be varied between 64ms or 16ms according to the requirement by having a dip switch at appropriate microcontroller pin.

All the above can be produced for a clock frequency of 1 KHz, 8 KHz or 16 KHz.

The GC command of either 64ms or 100ms duration can be generated upon press of a push button. This output is given to the base of transistor which acts as a switch /relay and produces 30V or5VGCcommand. The GC command is used to operate various relays like switching the satellite from battery simulator to battery mode during the period of satellite checkout.

A high to low pulse of 64ms duration is also used by the satellite systems to connect various points like the bus voltages by making it connect it to ground. So in order to produce this pulse a low to high pulse is generated at the microcontroller upon press of a button. This signal then drives a transistor connected to ground to generate a high to low pulse.

The expected outputs are given by the following timing diagram:

3.1.3 Power Supply unit

The power supply unit consists of the rectifier circuit, the filter circuit and the regulation circuit.The 7805 regulator block converts the rectified and filtered signal into regulated 5V DC supply to power the microcontroller and other IC's.LM117 is used to get regulated 3.3V supply to power the dip switch and push buttons. The block diagram of the power supply is as follows:

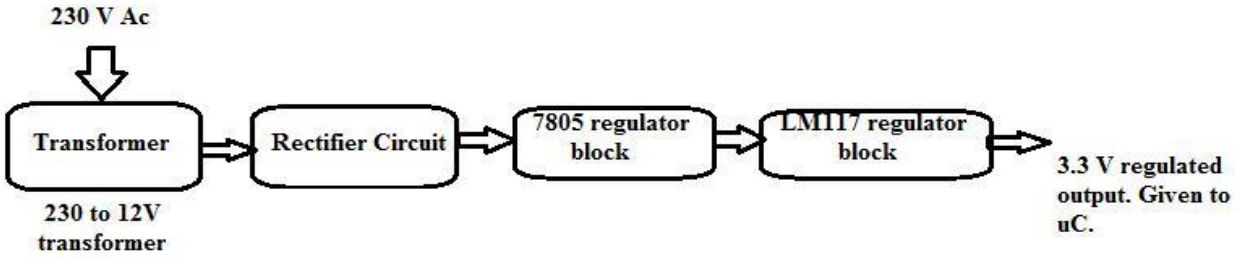


Fig 3.4 Block diagram of power supply

3.2 Circuit of Power Supply

3.2.1 Transformer

In this power supply, the transformer used is a laminated core transformer which is widely used in electric power transmission and appliances to convert mains voltage to low voltage to power electronic devices. They are available in power ratings ranging from mW to MW. This transformer converts the 230V AC supply into a 12V AC.

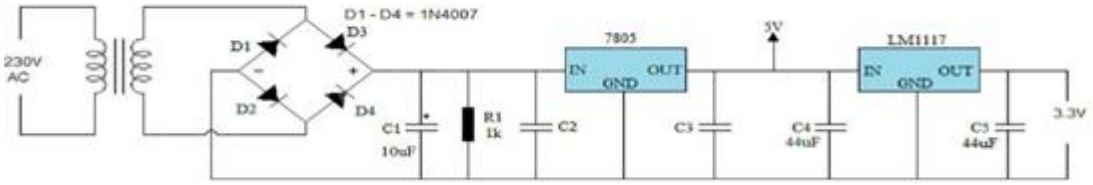
3.2.2 Rectifier Circuit

Rectifier circuit consists of a bridge rectifier formed by the four 1N4007 diodes and a filter circuit. The input to the bridge rectifier is a 12V AC output from a transformer. The output voltage of the bridge rectifier is an 8.4V DC with some AC voltage. This leads to distortion at the output. This distortion can be eliminated by using the filter circuit consisting of a resistor of 1K and a capacitor of 10uF.

3.2.3 Regulation Circuit

The output of the filter circuit that is 10.4V is given as an input to the LM7805 IC. This regulator IC converts the DC supply into the regulated DC voltage of 5V. It is a 3-Terminal Regulator with the output current from 5mA to 1A. It has internal thermal-overload protection. For pulling down the voltage from 5V to 3.3V LM1117 IC is used. LM1117 is a series of low dropout voltage regulator with a dropout of 1.2V at 800mA of load current. In this circuit a fixed

version of LM1117 is used, which eventually gives the output voltage of 3.3V.Instead LM117 can also be used with appropriate feedback resistors.



Here, C2=1000uf and C3=470uF

Fig 3.5 Power Supply Circuit Diagram

3.3 ARM 7 Microcontroller (LPC2148)

The LPC2141/42/44/46/48 microcontrollers are based on a 16-bit/32-bit ARM7TDMI-S CPU with real-time emulation and embedded trace support, that combine the microcontroller with embedded high-speed flash memory ranging from 32 kB to 512 kB. A 128-bit wide memory interface and unique accelerator architecture enables 32-bit code execution at the maximum clock rate. Due to their tiny size and low power consumption, LPC2141/42/44/46/48 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. Serial communications interfaces ranging from a USB 2.0 Full-speed device. Two 32-bit timers, single or dual 10-bit ADC(s), 10-bit DAC, PWM channels and 45 fast GPIO lines with up to nine edge or level sensitive external interrupt pins are present in LPC2148.

Technical Specifications

- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 KB to 40 KB of on-chip static RAM and 32 KB to 512 KB of on-chip flash memory.128-bit wide interface/accelerator enables high-speed 60 MHz operation.
- In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software. Single flash sector or full chip erase in 400 ms and programming of 256 B in 1 ms.
- Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high-speed tracing of instruction execution.
- USB 2.0 Full-speed compliant device controller with 2 KB of endpoint RAM. In addition, the LPC2146/48 provides 8 KB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs. LPC2144/46/48) 10-bit ADCs provide a total of 6/14 s per channel analog inputs, with conversion times as low as 2.44.

- Single 10-bit DAC provides variable analog output (LPC2142/44/46/48 only).
- Two 32-bit timers/external event counters (with four capture and four compare(channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to 21 external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 us.
- On-chip integrated oscillator operates with an external crystal from 1 MHz to 25 MHz
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.

3.4 Circuit Diagram for TM and TC

3.4.1 TM Circuit Diagram

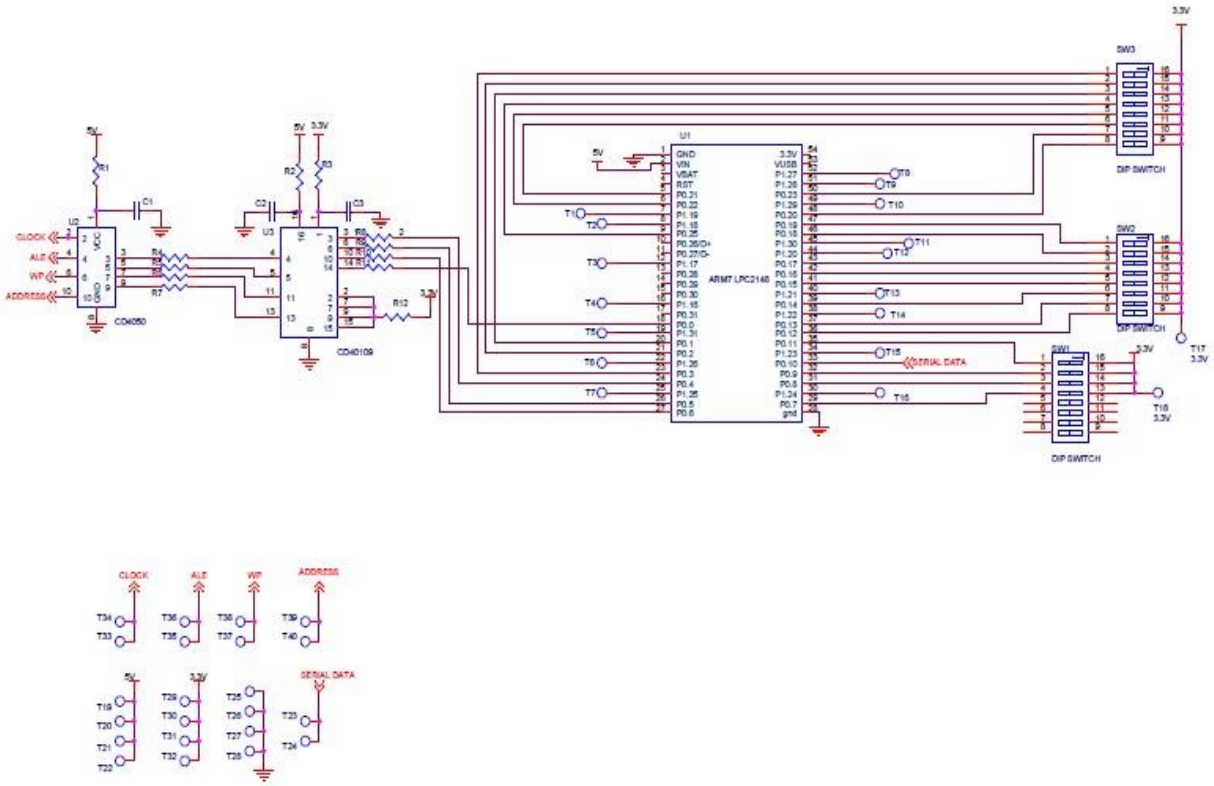


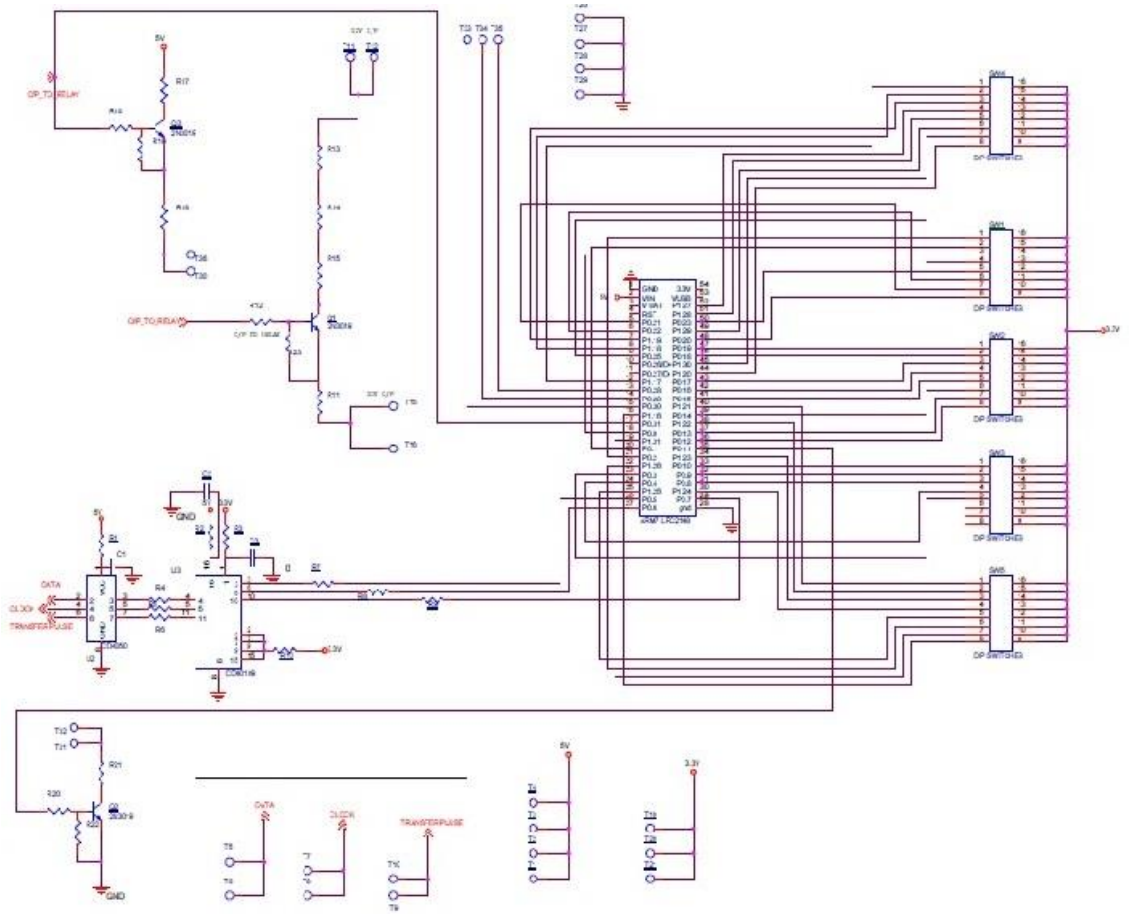
Fig 3.5.1 Circuit Diagram for TM

The above figure shows the interfacing diagram for the Telemetry Portion. It consists of the ARM board, DIP switch for Address input and frequency switching, LEDs for Data output. Level shifter CD40109 and non-inverting hex buffer CD4050 are used at the outputs. P0.0, P0.4, P0.5, P0.6 are the output pins. P0.0 and P0.10 are the serial address output and serial data input

respectively. The clock, ALE and word pulse signals are available at pins P0.4, P0.5 and P0.6 respectively. P0.7, P0.8, P0.9, P0.11 are used for frequency switching for frequencies 1 kHz, 2 kHz, 16 KHzand 40 kHz respectively. P0.1-P0.3, P0.12- P0.23, & P0.25 are used for address selection using DIP switches. P1.16 to P1.31is used for16parallel data outputs at LED's. XTAL1 and XTAL2 are the crystal inputs. Oscilloscope is used to check the waveforms and is connected to the output port pins. Vin pin is given a supply of 5V. 3.3V supply is required for triggering the port pins for frequency selection as well as DATA and ADDRESS selection through DIP switches. There are two grounds on the board and those are connected to the ground of the power supply circuit.

The outputs (P0.0, P0.4, P0.5, P0.6) of controller are given to level shifter through 1K resistor to increase voltage level to 5V. Level shifters are used to maintain the signal strength up to several kilometres. After level shifting, signals are fed to hex non inverting buffer.

3.4.2 TC Circuit Diagram



3.5.2 TC Circuit Diagram

The above figure represents the interfacing diagram for the Tele-command Portion. It consists of ARM board, DIP switches, level shifter CD40109 and non-inverting hex buffer CD4050. The port pins from P0.0 to P0.2, P0.12 to P0.23 and P0.25 are used as inputs for the 32

bit of data. The pins P0.8, P0.9, P0.10 are used for the frequency switching for frequencies 1 kHz, 8 kHz, 16 kHz respectively. The outputs clock, data and transfer pulse are obtained at port pins P0.6, P0.5, P0.7 respectively.in order to get the clock, data, TP outputs we need to press push button at P0.28. Vin pin is given a supply of 5V. 3.3V supply is connected for triggering the port pins for frequency selection as well as DATA through DIP switches. There are two grounds on the board and those are connected to the ground of the power supply circuit.

The outputs (P0.5, P0.6, P0.7) of controller are given to level shifter through 1K resistor to increase voltage level to 5V. Level shifters are used to maintain the signalstrength up to several kilometres. After level shifting, signals are fed to hex non inverting buffer. These clock, data, TP can be generated only when the push button at P0.28 is pressed.

Upon pressing push button attached to P0.29 we can generate 64ms pulse at P0.11.This is given to a base of transistor to get the high to low pulse.

Similarly the Push button at P0.30 is used to generate GC command or the low to high pulse at P0.31.the duration can be either 64ms(P0.3=HIGH) or 100ms(P0.3=LOW) according to status of P0.3.the output at P0.31 drives a transistor's base to get the required 5v or 30v GC command.

Also, P0.4 is used to switch the pulse duration of TP between 16ms (P0.4=LOW) and 64ms (P0.4=HIGH).

3.5 Calculations

A timer has a Timer Counter (TC) and Prescale Register (PR) associated with it. When timer is reset and enabled TC is set to 0 and incremented by 1 every 'PR+1' clock cycles. When it reaches its maximum value it gets reset to 0 and hence restarts counting. Prescale Register is used to define the resolution of the timer. If PR=0 then TC is incremented every 1 clock cycle of the peripheral clock. If PR=1 then TC is incremented every 2 clock cycles of peripheral clock and so on. By setting an appropriate value in PR we can make timer increment or count: every peripheral clock cycle or 1 microsecond or 1 millisecond or 1 second and so on.

To use timers we need to first configure them. We need to set appropriate values in TxCTCR, TxTR, and TxPR and reset TxPC, TxTC. Finally we assign TxTCR = 0x01 which enables the timer.

3.5.1 Steps for Configuring Timers

- Set appropriate value in TxCTCR
- Define the Prescale value in TxPR
- Set Value(s) in Match Register(s) if required

- Set appropriate value in TxMCR if using Match registers / Interrupts
- Reset Timer – Which resets PR and TC
- Set TxTCR to 0x01 to Enable the Timer when required
- Reset TxTCR to 0x00 to Disable the Timer when required

3.5.2 Calculations for Prescale

PR = (delay required* 60MHz) -1

For eg. Calculation for Delay of 1usec:

PR = (1*10^-6*60*10^6)-1

= 59

3.6 Software Requirements

The software's used in this project are the ARM Keil uVision5 and Flash Magic. Keil uVision5 is a compiler. This creates a hex file for the desired code.It can be used for debugging and troubleshooting for the 2 ports available. This hex file generated can be loaded to the simulation software called as Proteus Professional to check the working of the code with the controller virtually. For the real time working of the controller ARM7, software called Flash Magic is used. This software is used for freezing the code into the chip.

3.6.1 Compiler

The µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment. µVision is easy-to-use and accelerates your embedded software development. µVision supports multiple screens and allows you to create individual window layouts anywhere on the visual surface. The µVision Debugger provides a single environment in which you may test, verify, and optimize your application code. The debugger includes traditional features like simple and complex breakpoints, watches windows, and execution control and provides full visibility to device peripherals.

3.6.1.1 µVision Project Manager and Run-Time Environment

With the µVision Project Manager and Run-Time Environment you create software application using pre-build software components and device support from Software Packs. The software components contain libraries, source modules, configuration files, source code templates, and

documentation. Software components can be generic to support a wide range of devices and applications

The Project window shows application source files and selected software components. Below the components you will find corresponding library and configuration files.

Projects support multiple targets. They ease configuration management and may be used to generate debug and release builds or adoptions for different hardware platforms

The Manage Run-Time Environment window shows all software components that are compatible with the selected device. Inter-dependencies of software components are clearly identified with validation messages.

The Configuration Wizard is an integrated editor utility for generating GUI-like configuration controls in assembler, C/C++, or initialization files.

3.6.1.2 µVision Editor

The integrated µVision Editor includes all standard features of a modern source code editor and is also available during debugging. Color syntax highlighting, text indentation, and source outlining are optimized for C/C++.

The Functions window gives fast access to the functions in each C/C++ sourcecodemodule.
The Code Completion list and Function Parameter information helps you to keep track of symbols, functions, and parameters.

Dynamic Syntax Checking validates the program syntax while you are typing and provides real-time alerts to potential code violations before compilation.

3.6.2 Flash Magic

Flash Magic is a PC tool for programming flash based microcontrollers from NXP using a serial or Ethernet protocol while in the target hardware.

This software is used to freeze the program into the chip of the controller. You need a USB to Serial (UART) converter in order to get this working. The neat thing about the NXP controllers is they come with a UART bootloader from the factory. To program this Easy ARM development board using Flash Magic Software, a uart.fms file is to be added in the settings of the Flash Magic software. The settings are available in the file menu. Also setting

has to be done in the Keil uVision5 software so that the proper memory locations for the target device i.e. the Easy ARM board are used. The settings are to be done at the start of every new Keil uVision project.

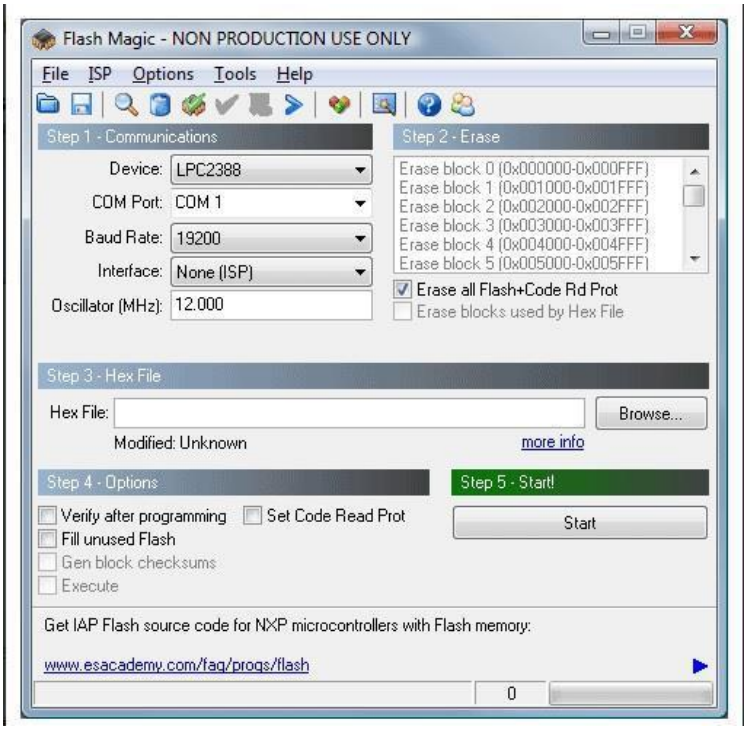


Fig 3.6 Flash Magic Main window

The settings to be done are as shown in the figure below:

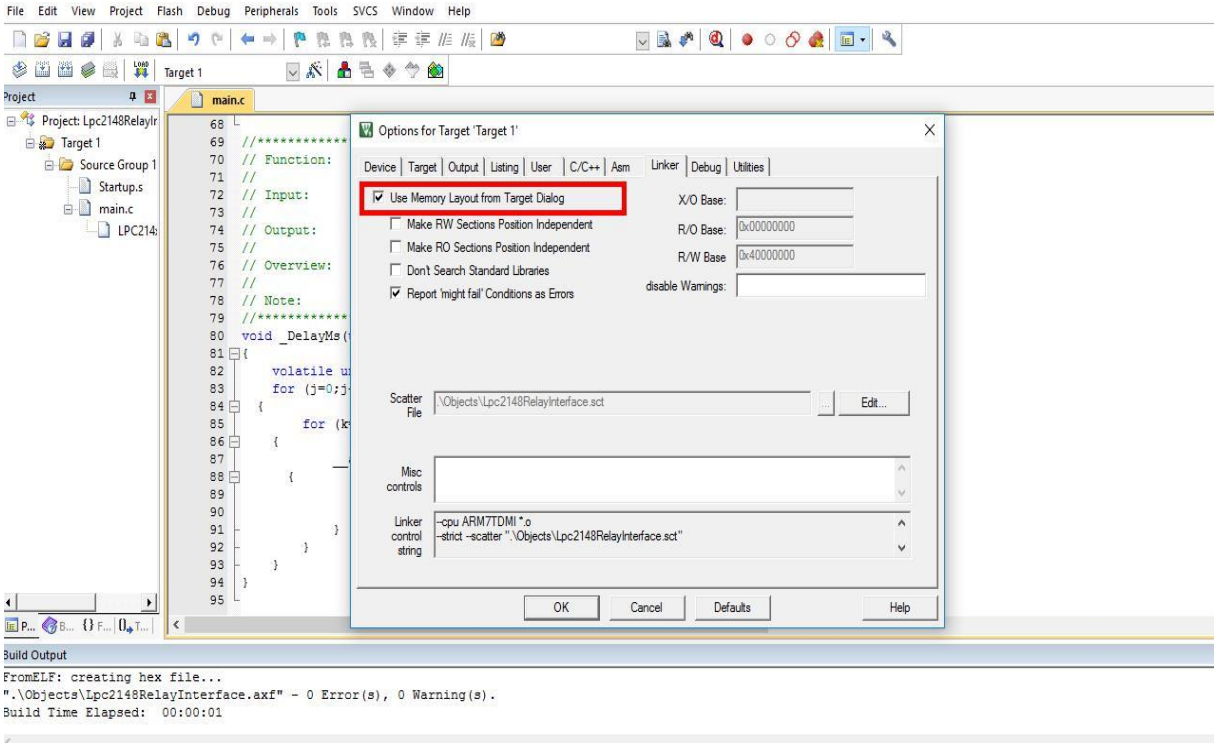


Fig 3.7 Keil Linker Settings

Chapter 4

PERFORMANCE ANALYSIS

4.1 Simulation results

4.1.1 TM Result

- This is the simulation result for 1 kHz TM. When the switch is ON at P0.7, clock of 1kHz is obtained at P0.4. The telemetry outputs are continuous. The ALE at P0.5, Word Pulse at P0.6 and serial Address at P0.0 are synchronized with the clock.

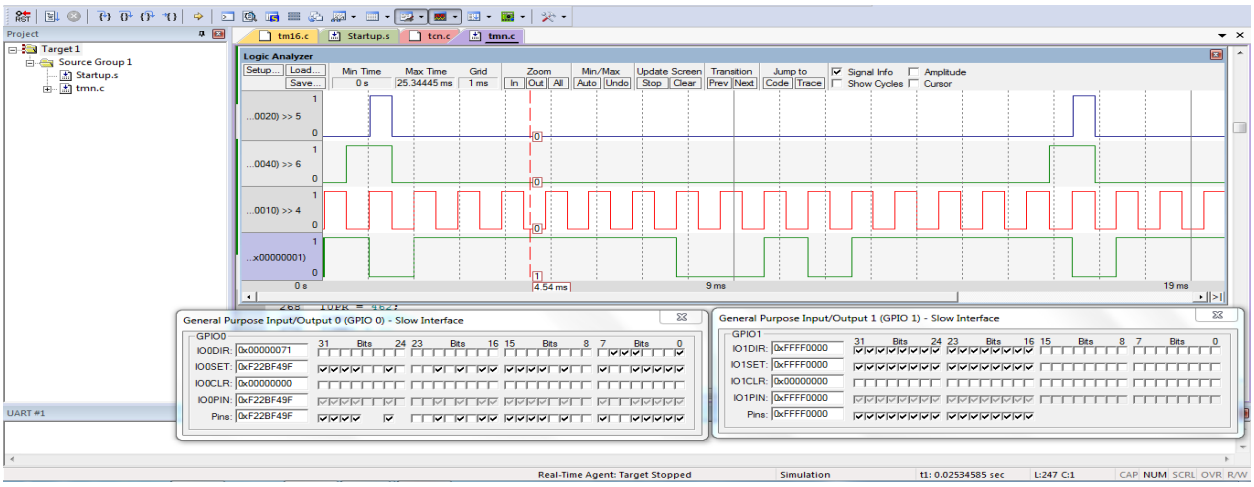


Fig 4.1: TM signal for 1 kHz

- This is the simulation result for 2 kHz TM. When the switch is ON at P0.8, clock of 2 kHz is obtained at P0.4. The telemetry outputs are continuous. The ALE at P0.5, Word Pulse at P0.6 and serial Address at P0.0 are synchronized with the clock.

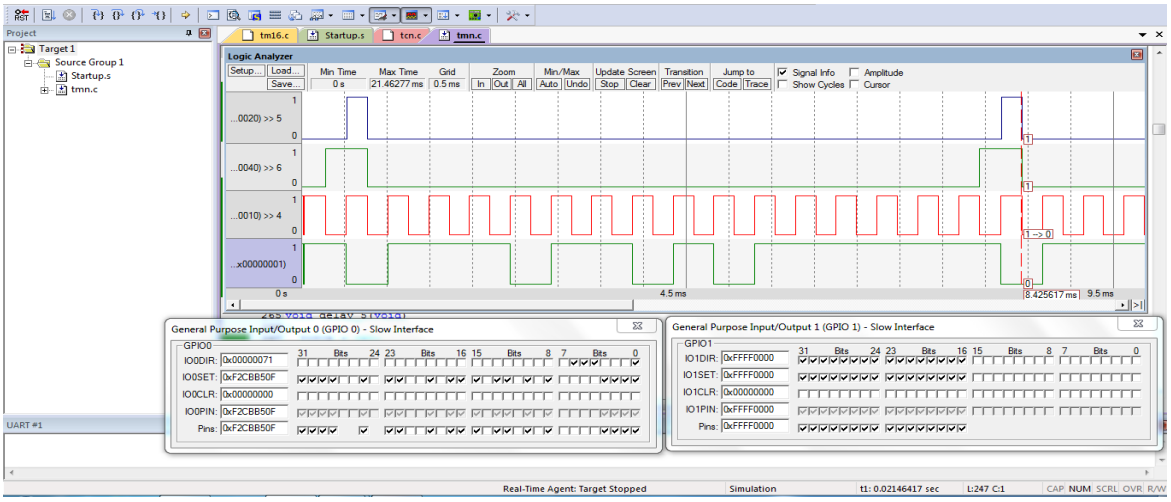


Fig 4.2: TM signal for 2 kHz

- This is the simulation result for 16 kHz TM. When the switch is ON at P0.9, clock of 16 kHz is obtained at P0.4. The telemetry outputs are continuous. The ALE at P0.5, Word Pulse at P0.6 and serial Address at P0.0 are synchronized with the clock.

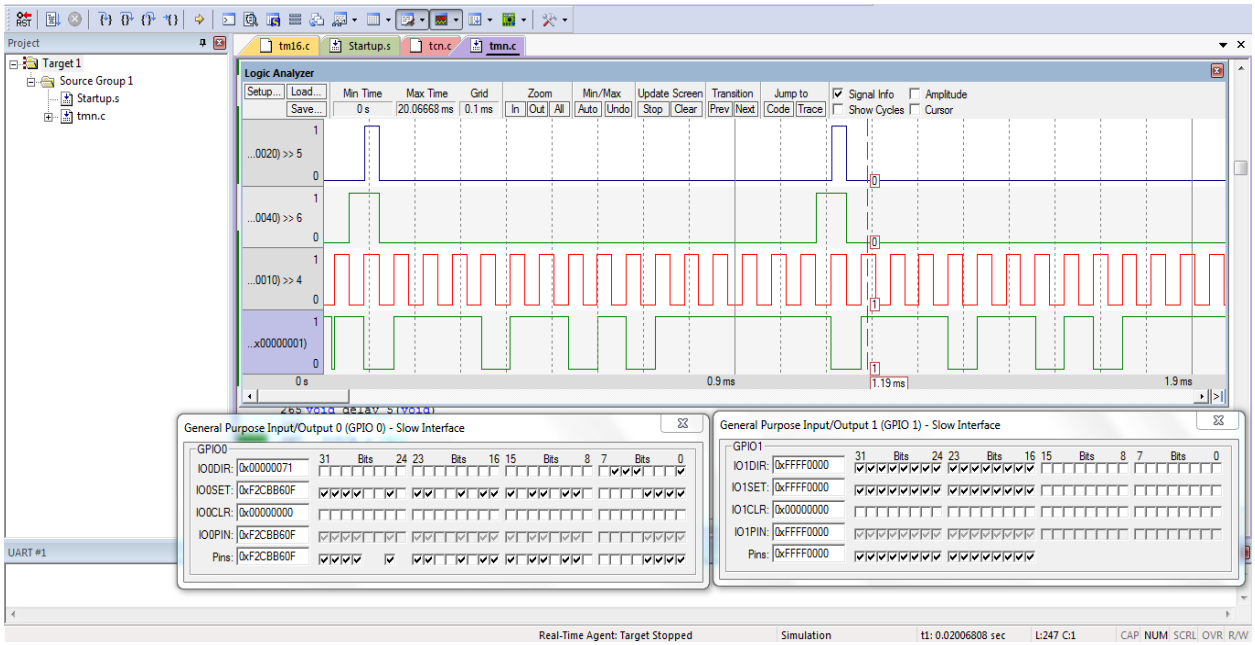


Fig 4.3: TM signal of 16 kHz

- This is the simulation result for 40 kHz TM. When the switch is ON at P0.11, clock of 40 kHz is obtained at P0.4. The telemetry outputs are continuous. The ALE at P0.5, Word Pulse at P0.6 and serial Address at P0.0 are synchronized with the clock.

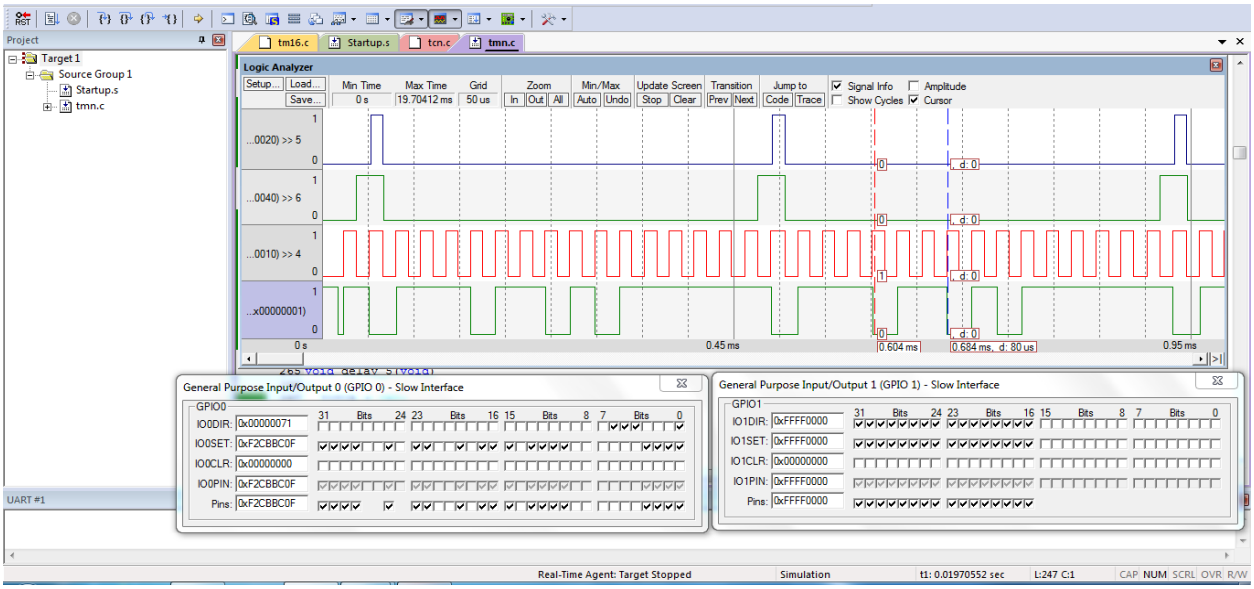


Fig 4.4: TM signal of 40 kHz

4.1.2 TC Result

- This is the simulation result for 1 kHz TC. Whenever the push button at P0.28 is pressed and the switch is ON at P0.8, clock burst of 32 pulses of 1 kHz is obtained at P0.6. Transfer pulse of selected period (64ms/16ms selectable by P0.4) is obtained at P0.7. Data is obtained at P0.5. High to Low signal is obtained at P0.11 when push button at P0.29 is pressed.

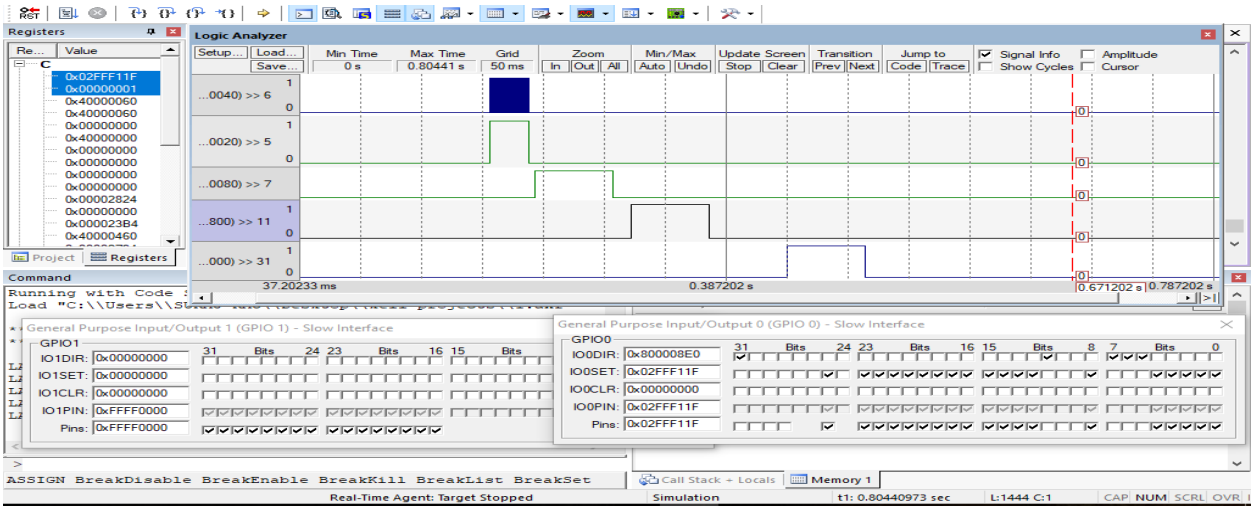


Fig 4.5: TC signal of1 kHz

- This is the simulation result for 8 kHz TC. Whenever the push button at P0.28 is pressed and the switch is ON at P0.9, clock burst of 32 pulses of 8 kHz is obtained at P0.6. Transfer pulse of selected period (64ms/16ms selectable by P0.4) is obtained at P0.7. Data is obtained at P0.5. High to Low signal is obtained at P0.11when push button at P0.29 is pressed.

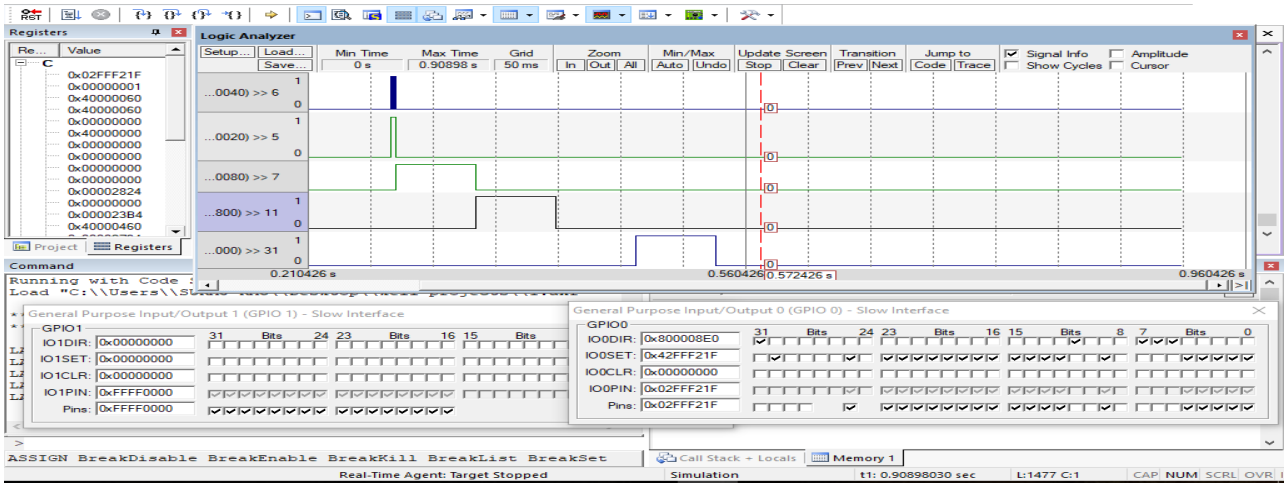


Fig 4.6: TC signal of 8 kHz

- This is the simulation result for 16 kHz TC. Whenever the push button at P0.28 is pressed and the switch is ON at P0.10, clock burst of 32 pulses of 16 kHz is obtained at P0.6. Transfer pulse of selected period (64ms/16ms selectable by P0.4) is obtained at P0.7. Data is obtained at P0.5. GC command signal of selected period (64ms/100msselectable by P0.3) is obtained at P0.31 upon press of push button at P0.30.

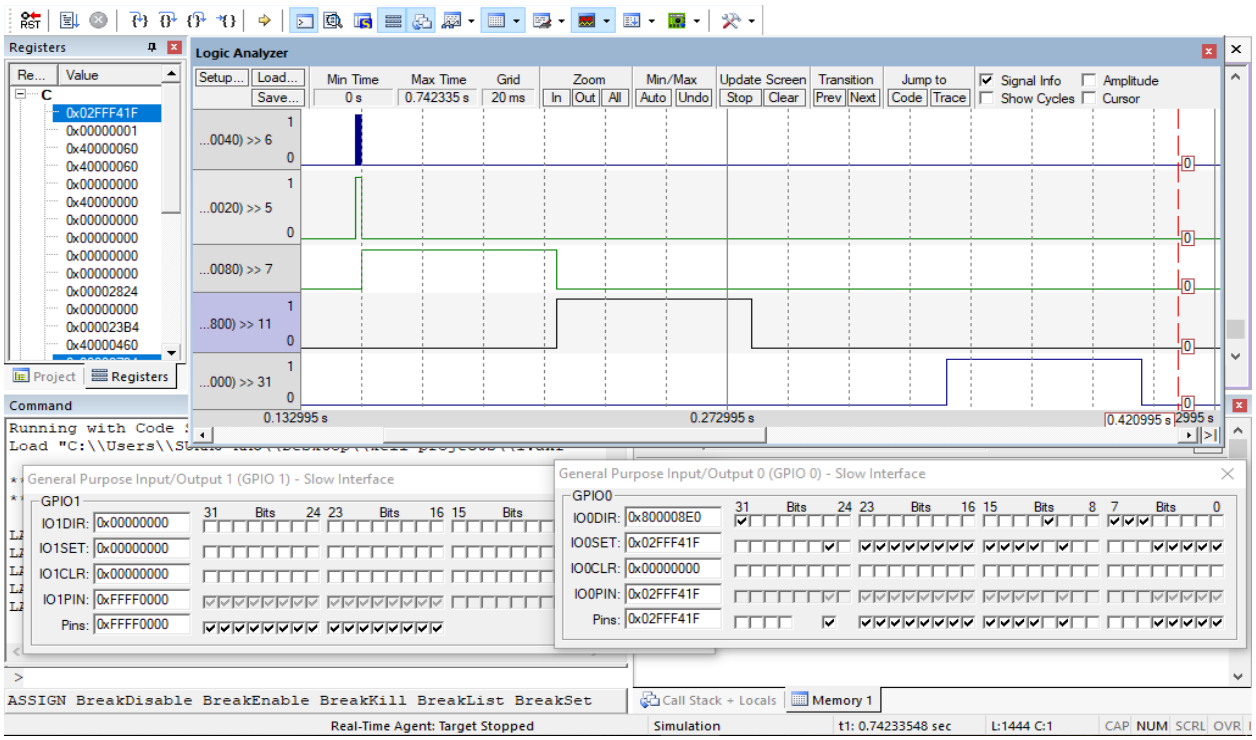


Fig 4.7: TC signal of 16 kHz

4.2 Hardware results

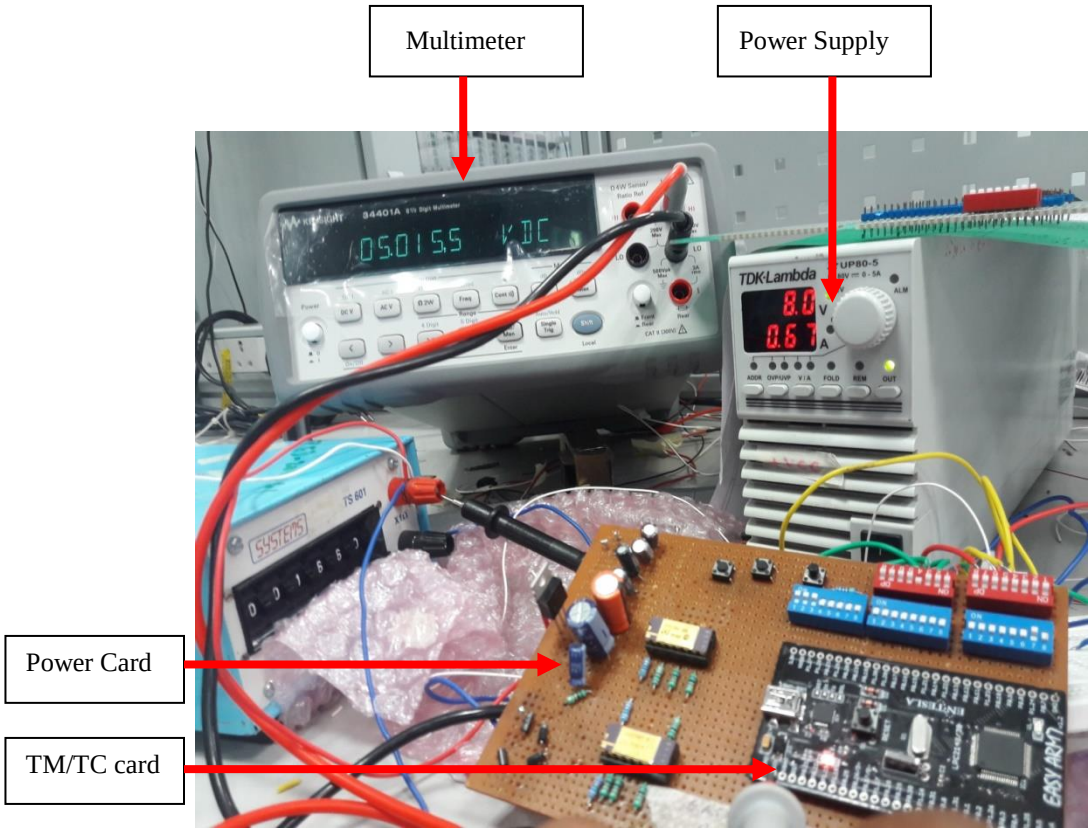


Fig 4.2.1 Hardware setup of TC or TM

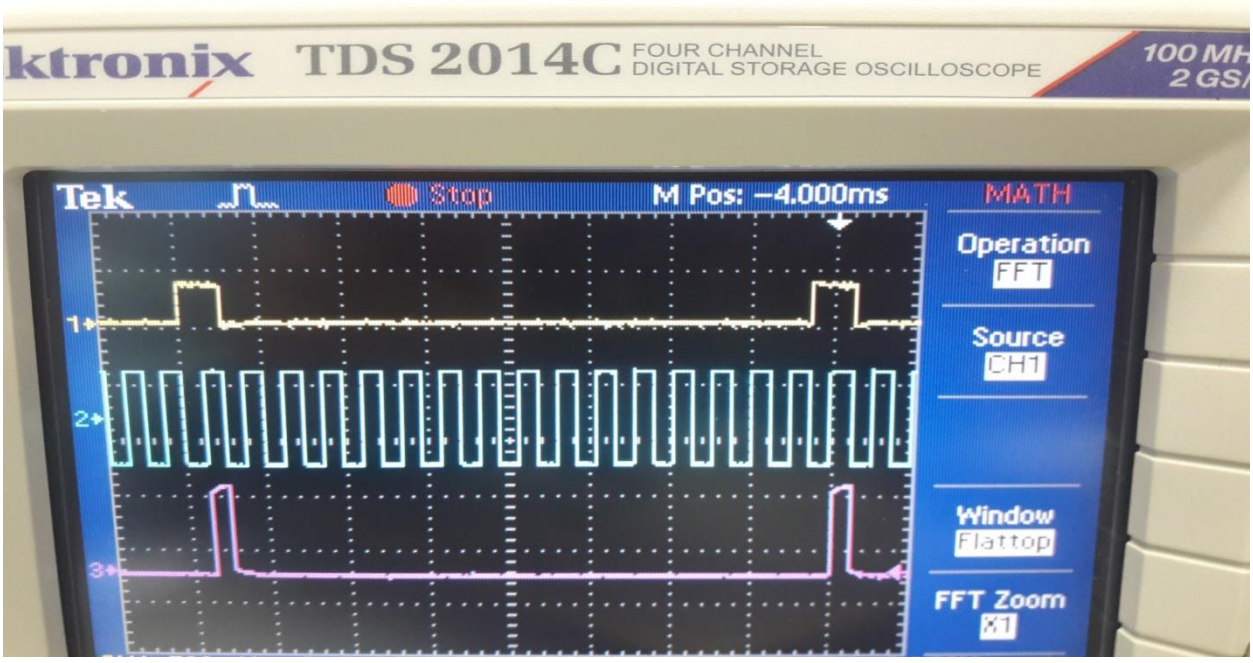


Fig 4.2.2 TM Clock, Write pulse and ALE signals

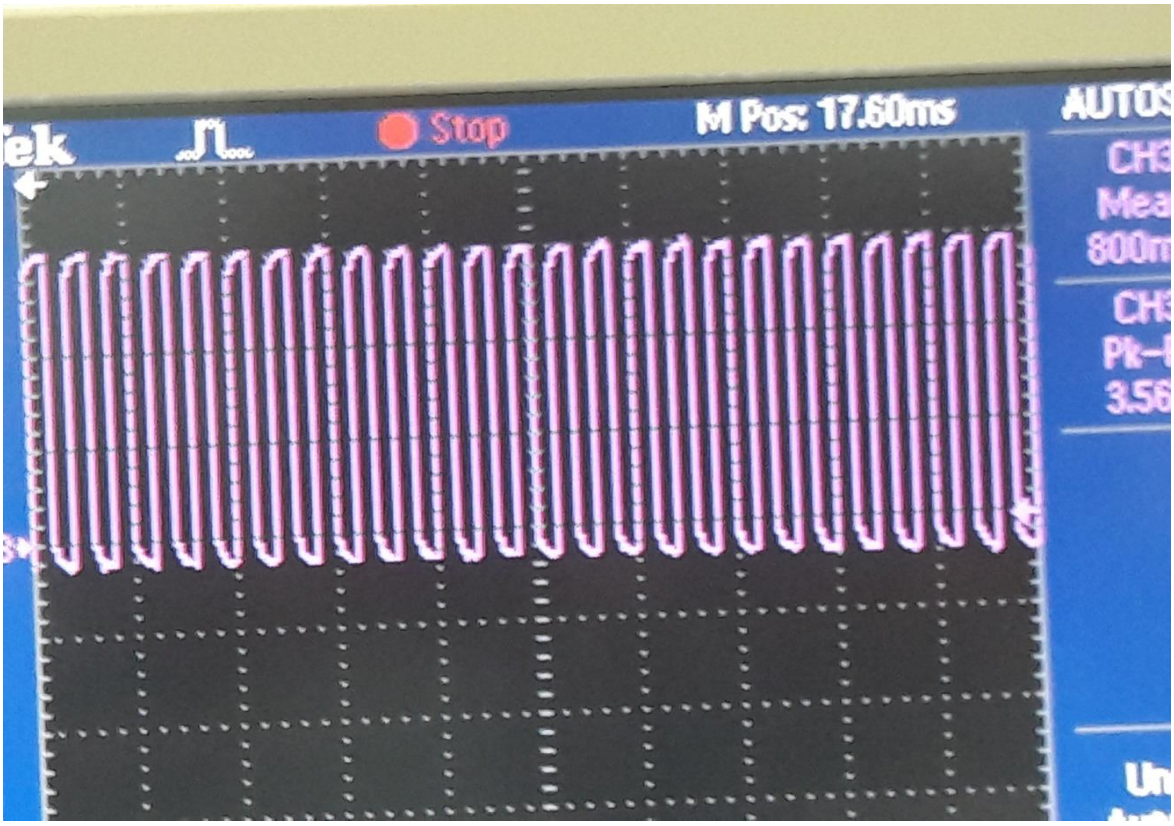


Fig 4.2.3TC clock signal

Fig 4.2.4TC clock and Transfer pulse

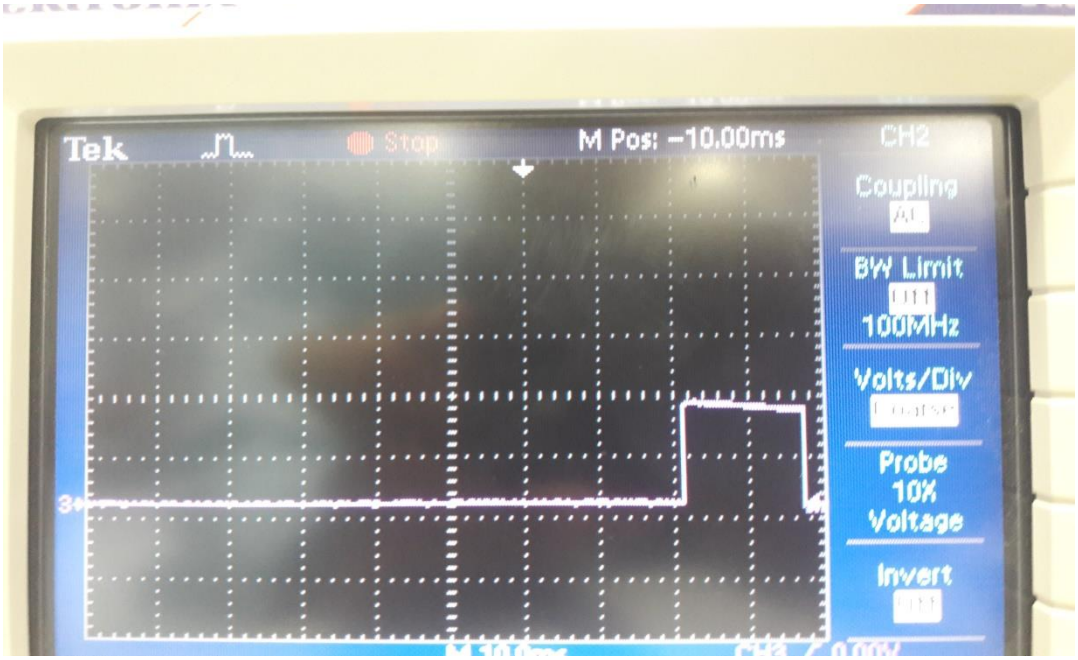


Fig 4.2.5GC Command signal (64ms)

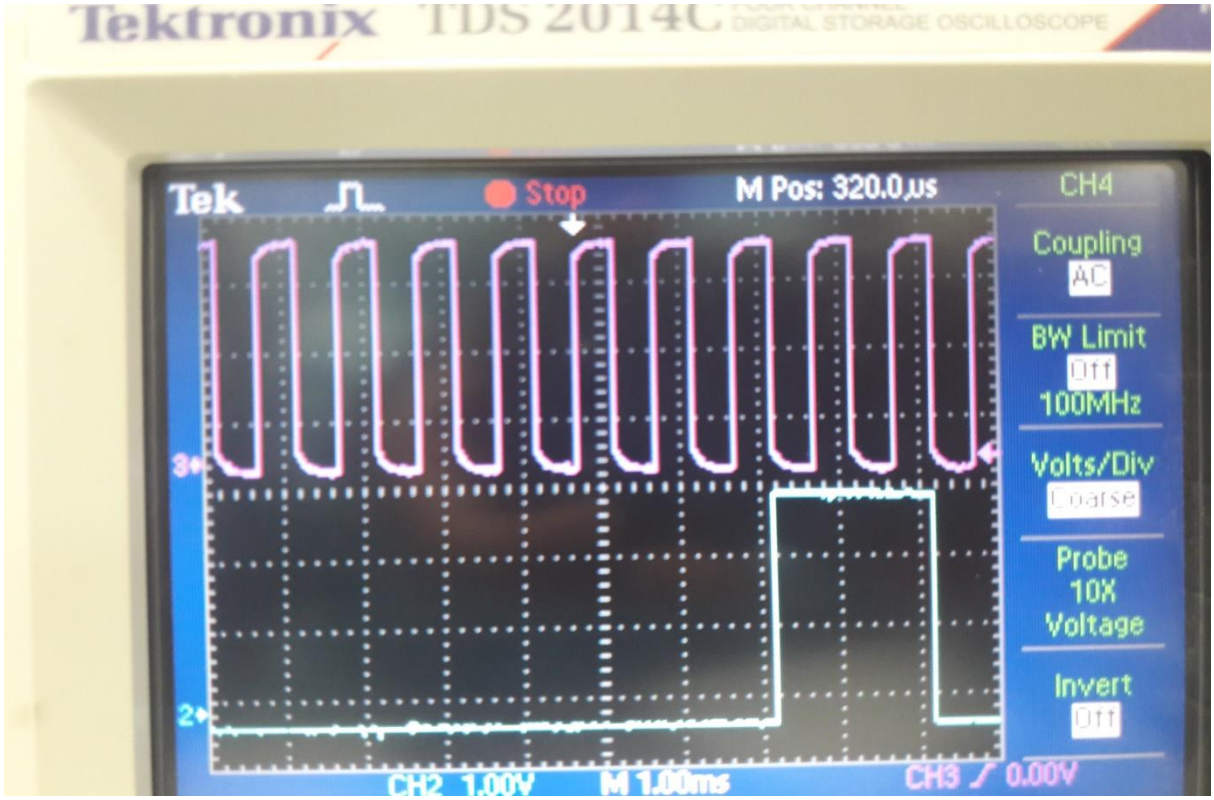


Fig 4.2.6 TC data output

Chapter 5

CONCLUSION

The project “Design, Development, Demonstration and Implementation of Microcontroller based Telemetry and Tele-command Simulator" is developed as part of Industrial Project Training Programme for Bachelor of Engineering in Electronics and Communication Engineering.

This proposed system is proved to be more efficient than the existing system which uses discrete logic. It is less bulky than the earlier system and also consumes less power. It has successfully generated the required signals with precise timings. This system is also capable of generating Tele-command signal of 2 KHz which makes it compatible for the geostationary system and TM Clock of 40 KHz for IRS satellites. It also generates Transfer pulse and GC Command signal of two different periods (64ms/16ms and 64ms/100ms respectively) which are used for switching of satellites battery from Battery Sim to onboard battery.

REFERENCES

[1] Gerald Maral and Michel Bousqet, “Satellite Communication Systems- Systems, Techniques and Technologies”.

[2] http://www.keil.com/support/man_arm.htm

[3] http://www.knowledgebase.entesla.com/category/mcu-tools/lcd-arm/

[4] http://www.entesla.com/easy-arm7-development-board

[5] NXP Semiconductors,”LPC2141/42/44/46/48-single chip 16/32 bit microcontrollers” datasheet, Rev.5-12 August 2011.

[6] Texas Instruments,”uA7800 series positive voltage regulators”, SLVS056J- MAY 1976 REVISED MAY 2003.

[7] http://www.ti.com/lit/ds/symlink/lm1117.pdf

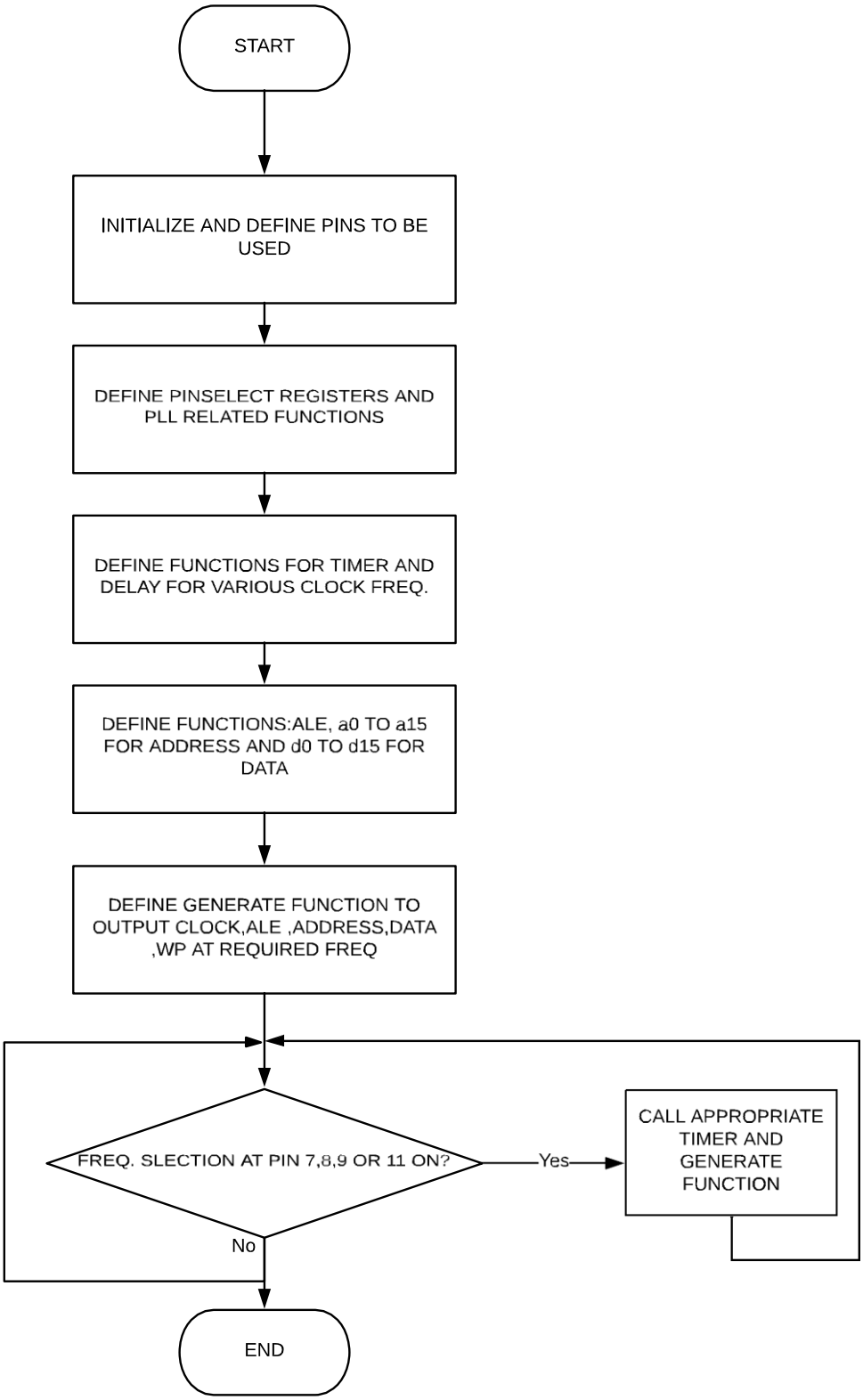
[8] http://www.ti.com/product/LM4050QML-SP

[9] http://www.ti.com/lit/ds/symlink/cd40109b.pdf

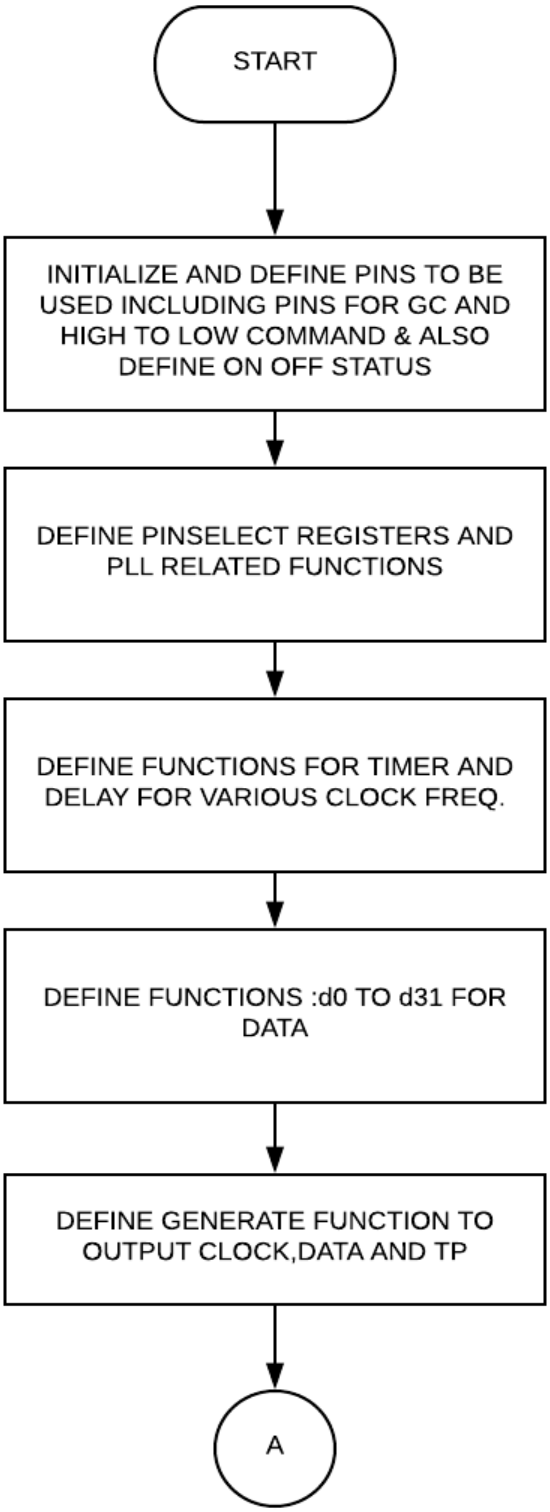
Chapter 6

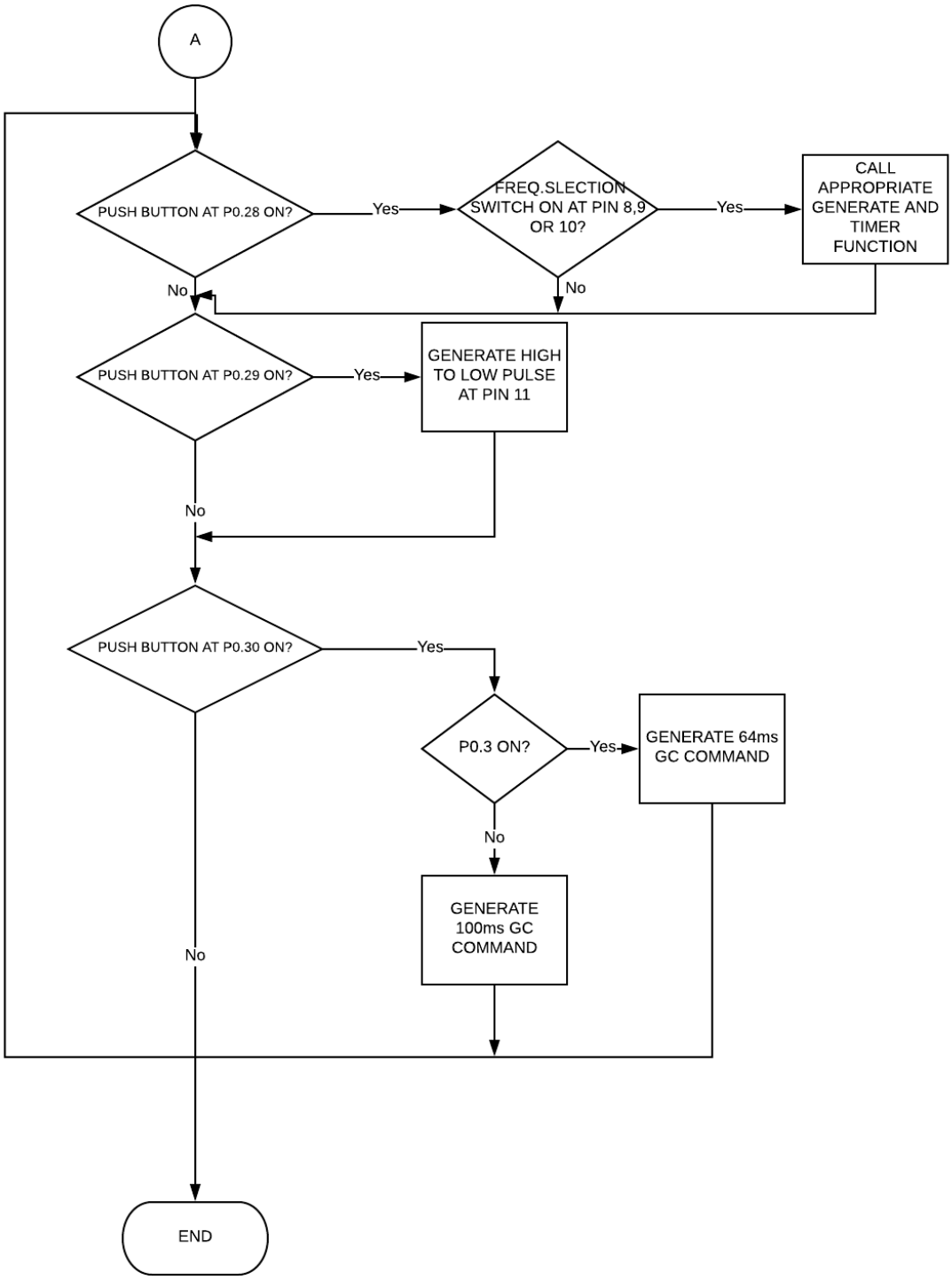
APPENDIX

TM Flowchart



TC Flowchart




A
PUSH BUTTON AT P0.28 ON?
Yes
FREQ.SLECTION SWITCH ON AT PIN 8,9 OR 10?
Yes
CALL APPROPRIATE GENERATE AND TIMER FUNCTION
No
No
PUSH BUTTON AT P0.29 ON?
Yes
GENERATE HIGH TO LOW PULSE AT PIN 11
No
PUSH BUTTON AT P0.30 ON?
Yes
P0.3 ON?
Yes
GENERATE 64ms GC COMMAND
No
GENERATE 100ms GC COMMAND
No
END

TM CODE

```
#include<LPC214x.h>
#define PLOCK 0X0000400
void PLL0(void);//Function to set the parameters to PLL to set the clock
void init_Timer1(void);
void init_Timer2(void);
void init_Timer3(void);
void init_Timer4(void);
#define sw1 (1<<7)  // sw 1 to 4 are frequency selection switches for 1,2,16,40 KHz clocks
resp.
#define sw2 (1<<8)
#define sw3 (1<<9)
#define sw4 (1<<11)
#define clk1 (1<<4)   // output of clock at pin 4
#define ale1 (1<<5)   // output of address latch enable at pin 5
#define wp1 (1<<6)    // output of write pulse at pin 6
#define a0 (1<<0)     //a0 to a16 are initialization for serial address output at pin 0
#define a1 (1<<0)
#define a2 (1<<0)
#define a3 (1<<0)
#define a4 (1<<0)
#define a5 (1<<0)
#define a6 (1<<0)
#define a7 (1<<0)
#define a8 (1<<0)
#define a9 (1<<0)
#define a10 (1<<0)
#define a11 (1<<0)
#define a12 (1<<0)
#define a13 (1<<0)
#define a14 (1<<0)
#define a15 (1<<0)
#define d0 (1<<25)  //d0 to d16 are initialization for parallel data output
#define d1 (1<<26)
#define d2 (1<<31)
#define d3 (1<<16)
#define d4 (1<<17)
#define d5 (1<<18)
#define d6 (1<<19)
#define d7 (1<<27)
#define d8 (1<<28)
#define d9 (1<<29)
#define d10 (1<<30)
#define d11 (1<<20)
#define d12 (1<<21)
#define d13 (1<<22)
#define d14 (1<<23)
```

```
#define d15 (1<<24)
#define sw1_dir IO0DIR
#define ale1_dir IO0DIR
#define clk1_dir IO0DIR
#define wp1_dir IO0DIR
#define sw2_dir IO0DIR
#define sw3_dir IO0DIR
#define sw4_dir IO0DIR
#define a0_dir IO0DIR
#define a1_dir IO0DIR
#define a2_dir IO0DIR
#define a3_dir IO0DIR
#define a4_dir IO0DIR
#define a5_dir IO0DIR
#define a6_dir IO0DIR
#define a7_dir IO0DIR
#define a8_dir IO0DIR
#define a9_dir IO0DIR
#define a10_dir IO0DIR
#define a11_dir IO0DIR
#define a12_dir IO0DIR
#define a13_dir IO0DIR
#define a14_dir IO0DIR
#define a15_dir IO0DIR
#define d0_dir IO1DIR
#define d1_dir IO1DIR
#define d2_dir IO1DIR
#define d3_dir IO1DIR
#define d4_dir IO1DIR
#define d5_dir IO1DIR
#define d6_dir IO1DIR
#define d7_dir IO1DIR
#define d8_dir IO1DIR
#define d9_dir IO1DIR
#define d10_dir IO1DIR
#define d11_dir IO1DIR
#define d12_dir IO1DIR
#define d13_dir IO1DIR
#define d14_dir IO1DIR
#define d15_dir IO1DIR
#define sw1_port IO0PIN //initializations of port pins
#define ale1_port IO0PIN
#define clk1_port IO0PIN
#define wp1_port IO0PIN
#define sw2_port IO0PIN
#define sw3_port IO0PIN
#define sw4_port IO0PIN
#define a0_port IO0PIN
```

```
#define a1_port IO0PIN
#define a2_port IO0PIN
#define a3_port IO0PIN
#define a4_port IO0PIN
#define a5_port IO0PIN
#define a6_port IO0PIN
#define a7_port IO0PIN
#define a8_port IO0PIN
#define a9_port IO0PIN
#define a10_port IO0PIN
#define a11_port IO0PIN
#define a12_port IO0PIN
#define a13_port IO0PIN
#define a14_port IO0PIN
#define a15_port IO0PIN
#define d0_port IO1PIN
#define d1_port IO1PIN
#define d2_port IO1PIN
#define d3_port IO1PIN
#define d4_port IO1PIN
#define d5_port IO1PIN
#define d6_port IO1PIN
#define d7_port IO1PIN
#define d8_port IO1PIN
#define d9_port IO1PIN
#define d10_port IO1PIN
#define d11_port IO1PIN
#define d12_port IO1PIN
#define d13_port IO1PIN
#define d14_port IO1PIN
#define d15_port IO1PIN
#define clk1_on clk1_port|= (clk1);  //defining on and off status of  pins required
#define clk1_off clk1_port&= ~(clk1);
#define ale1_on ale1_port|= (ale1);
#define ale1_off ale1_port&= ~(ale1);
#define wp1_on wp1_port|= (wp1);
#define wp1_off wp1_port&= ~(wp1);
#define a0_on a0_port|= (a0);
#define a0_off a0_port&= ~(a0);
#define a1_on a1_port|= (a1);
#define a1_off a1_port&= ~(a1);
#define a2_on a2_port|= (a2);
#define a2_off a2_port&= ~(a2);
#define a3_on a3_port|= (a3);
#define a3_off a3_port&= ~(a3);
#define a4_on a4_port|= (a4);
#define a4_off a4_port&= ~(a4);
#define a5_on a5_port|= (a5);
```

```
#define a5_off a5_port&= ~(a5);
#define a6_on a6_port|= (a6);
#define a6_off a6_port&= ~(a6);
#define a7_on a7_port|= (a7);
#define a7_off a7_port&= ~(a7);
#define a8_on a8_port|= (a8);
#define a8_off a8_port&= ~(a8);
#define a9_on a9_port|= (a9);
#define a9_off a9_port&= ~(a9);
#define a10_on a10_port|= (a10);
#define a10_off a10_port&= ~(a10);
#define a11_on a11_port|= (a11);
#define a11_off a11_port&= ~(a11);
#define a12_on a12_port|= (a12);
#define a12_off a12_port&= ~(a12);
#define a13_on a13_port|= (a13);
#define a13_off a13_port&= ~(a13);
#define a14_on a14_port|= (a14);
#define a14_off a14_port&= ~(a14);
#define a15_on a15_port|= (a15);
#define a15_off a15_port&= ~(a15);
#define d0_on d0_port|= (d0);
#define d0_off d0_port&= ~(d0);
#define d1_on d1_port|= (d1);
#define d1_off d1_port&= ~(d1);
#define d2_on d2_port|= (d2);
#define d2_off d2_port&= ~(d2);
#define d3_on d3_port|= (d3);
#define d3_off d3_port&= ~(d3);
#define d4_on d4_port|= (d4);
#define d4_off d4_port&= ~(d4);
#define d5_on d5_port|= (d5);
#define d5_off d5_port&= ~(d5);
#define d6_on d6_port|= (d6);
#define d6_off d6_port&= ~(d6);
#define d7_on d7_port|= (d7);
#define d7_off d7_port&= ~(d7);
#define d8_on d8_port|= (d8);
#define d8_off d8_port&= ~(d8);
#define d9_on d9_port|= (d9);
#define d9_off d9_port&= ~(d9);
#define d10_on d10_port|= (d10);
#define d10_off d10_port&= ~(d10);
#define d11_on d11_port|= (d11);
#define d11_off d11_port&= ~(d11);
#define d12_on d12_port|= (d12);
#define d12_off d12_port&= ~(d12);
#define d13_on d13_port|= (d13);
```

```
#define d13_off d13_port&= ~(d13);
#define d14_on d14_port|= (d14);
#define d14_off d14_port&= ~(d14);
#define d15_on d15_port|= (d15);
#define d15_off d15_port&= ~(d15);
#define PINSEL0 (*((volatile unsigned long *) 0xE002C000))  //pinselect and registers
required for
#define PLL0CON (*((volatile unsigned char *) 0xE01FC080))     //PLL and their addresses
#define PLL0CFG (*((volatile unsigned char *) 0xE01FC084))
#define PLL0FEED (*((volatile unsigned char *) 0xE01FC08C))
#define PINSEL1 (*((volatile unsigned long *) 0xE002C004))
void PLL0(void) //to set PLL for defining the operating clock frequency of device
{
        PLL0CON = 0X01;
        PLL0CFG = 0X24;
        PLL0FEED = 0XAA;
        PLL0FEED = 0X55;
        while(!(PLL0STAT&PLOCK));
        PLL0CON = 0X03;
        PLL0FEED = 0XAA;
        PLL0FEED = 0X55;
        VPBDIV = 0X01;
}
void init_Timer1(void) // for clock of 1khz
{
        T0TCR = 0X0;
        T0PR = 29812;
        T0MCR = 0X00000001;
        T0MR0 = 1;
        T0TCR = 0X02;
}
void init_Timer2(void) // for clock of 2khz
{
        T0TCR = 0X0;
        T0PR = 14812;
        T0MCR = 0X00000001;
        T0MR0 = 1;
        T0TCR = 0X02;
}
void init_Timer3(void) // for clock of 40khz
{
        T0TCR = 0X0;
        T0PR = 749;
        T0MCR = 0X00000001;
T0MR0 = 1;
T0TCR = 0X02;
}
void init_Timer4(void) // for clock of 16khz
```

```
{
        T0TCR = 0X0;
        T0PR = 1874;
        T0MCR = 0X00000001;
 T0MR0 = 1;
 T0TCR = 0X02;
}
void delay_1(void)
{
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

void delay_4(void)
{
        T0TCR = 0X0;
        T0PR = 930;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

void delay_5(void)
{
        T0TCR = 0X0;
        T0PR = 462;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

void delay_6(void)
{
        T0TCR = 0X0;
        T0PR = 22;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
```

```
	T0TCR = 0X01;
	while(T0TC!=T0MR0);
	T0TCR = 0X02;
	T0TC = T0PC = 0;
}

void delay_7(void)
{
	T0TCR = 0X0;
	T0PR = 28619;
	T0MCR = 0X01;
	T0MR0 = 1;
	T0TCR = 0X02;
	T0TCR = 0X01;
	while(T0TC!=T0MR0);
	T0TCR = 0X02;
	T0TC = T0PC = 0;
}

void delay_8(void)
{
	T0TCR = 0X0;
	T0PR = 14219;
	T0MCR = 0X01;
	T0MR0 = 1;
	T0TCR = 0X02;
	T0TCR = 0X01;
	while(T0TC!=T0MR0);
	T0TCR = 0X02;
	T0TC = T0PC = 0;
}

void delay_9(void)
{
	T0TCR = 0X0;
	T0PR = 726;
	T0MCR = 0X01;
	T0MR0 = 1;
	T0TCR = 0X02;
	T0TCR = 0X01;
	while(T0TC!=T0MR0);
	T0TCR = 0X02;
	T0TC = T0PC = 0;
}
void delay_10(void)
{
	T0TCR = 0X0;
	T0PR = 56;
```

```
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}
void delay_11(void)
{
        T0TCR = 0X0;
        T0PR = 1815;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

void ale_1(void)   //functions to generate ALE
{
        delay_4();
        ale1_on;
        delay_7();
        clk1_off;
        ale1_off;


}

void ale_2(void)
{
        delay_5();
        ale1_on;
        delay_8();
        clk1_off;
        ale1_off;
}
void ale_3(void)
{
        delay_6();
        ale1_on;
        delay_9();
        clk1_off;
        ale1_off;
}
```

```
void ale_4(void)
{
	delay_10();
	ale1_on;
	delay_11();
	clk1_off;
	ale1_off;
}
void a0_1()  //functions to send parallel address i/p's to serial address o/p at pin0
{
	if((IO0PIN>>19)&1)
	{
		a0_on;
	}
	else
	{
		a0_off;
	}
}
void a1_1()
{
	if((IO0PIN>>18)&1)
	{
		a1_on;
	}
	else
	{
		a1_off;
	}
}
void a2_1()
{
	if((IO0PIN>>17)&1)
	{
		a2_on;
	}
	else
	{
		a2_off;
	}
}
void a3_1()
{
	if((IO0PIN>>16)&1)
	{
		a3_on;
	}
	else
```

```
	{
		a3_off;
	}
}
void a4_1()
{
	if((IO0PIN>>15)&1)
	{
		a4_on;
	}
	else
	{
		a4_off;
	}
}
void a5_1()
{
	if((IO0PIN>>14)&1)
	{
		a5_on;
	}
	else
	{
		a5_off;
	}
}
void a6_1()
{
	if((IO0PIN>>13)&1)
	{
		a6_on;
	}
	else
	{
		a6_off;
	}
}
void a7_1()
{
	if((IO0PIN>>12)&1)
	{
		a7_on;
	}
	else
	{
		a7_off;
	}
}
```

```
void a8_1()
{
	if((IO0PIN>>20)&1)
	{
		a8_on;
	}
	else
	{
		a8_off;
	}
}
void a9_1()
{
	if((IO0PIN>>23)&1)
	{
		a9_on;
	}
	else
	{
		a9_off;
	}
}
void a10_1()
{
	if((IO0PIN>>21)&1)
	{
		a10_on;
	}
	else
	{
		a10_off;
	}
}
void a11_1()
{
	if((IO0PIN>>22)&1)
	{
		a11_on;
	}
	else
	{
		a11_off;
	}
}
void a12_1()
{
	if((IO0PIN>>25)&1)
	{
```

```
                a12_on;
        }
        else
        {
                a12_off;
        }
}
void a13_1()
{
        if((IO0PIN>>28)&1)
        {
                a13_on;
        }
        else
        {
                a13_off;
        }
}
void a14_1()
{
        if((IO0PIN>>29)&1)
        {
                a14_on;
        }
        else
        {
                a14_off;
        }
}
void a15_1()
{
        if((IO0PIN>>30)&1)
        {
                a15_on;
        }
        else
        {
                a15_off;
        }
}
void d0_1()   //functions to take serial data i/p's and produce parallel data o/p's
{
        if((IO0PIN>>10)&1)
        {
                d0_on;
        }
        else
        {
```

```
                d0_off;
        }
}
void d1_1()
{
        if((IO0PIN>>10)&1)
        {
                d1_on;
        }
        else
        {
                d1_off;
        }
}
void d2_1()
{
        if((IO0PIN>>10)&1)
        {
                d2_on;
        }
        else
        {
                d2_off;
        }
}
void d3_1()
{
        if((IO0PIN>>10)&1)
        {
                d3_on;
        }
        else
        {
                d3_off;
        }
}
void d4_1()
{
        if((IO0PIN>>10)&1)
        {
                d4_on;
        }
        else
        {
                d4_off;
        }
}
void d5_1()
```

```
{
	if((IO0PIN>>10)&1)
	{
		d5_on;
	}
	else
	{
		d5_off;
	}
}
void d6_1()
{
	if((IO0PIN>>10)&1)
	{
		d6_on;
	}
	else
	{
		d6_off;
	}
}
void d7_1()
{
	if((IO0PIN>>10)&1)
	{
		d7_on;
	}
	else
	{
		d7_off;
	}
}
void d8_1()
{
	if((IO0PIN>>10)&1)
	{
		d8_on;
	}
	else
	{
		d8_off;
	}
}
void d9_1()
{
	if((IO0PIN>>10)&1)
	{
		d9_on;
```

```
	}
	else
	{
		d9_off;
	}
}
void d10_1()
{
	if((IO0PIN>>10)&1)
	{
		d10_on;
	}
	else
	{
		d10_off;
	}
}
void d11_1()
{
	if((IO0PIN>>10)&1)
	{
		d11_on;
	}
	else
	{
		d11_off;
	}
}
void d12_1()
{
	if((IO0PIN>>10)&1)
	{
		d12_on;
	}
	else
	{
		d12_off;
	}
}
void d13_1()
{
	if((IO0PIN>>10)&1)
	{
		d13_on;
	}
	else
	{
		d13_off;
```

```
        }
}
void d14_1()
{
        if((IO0PIN>>10)&1)
        {
                d14_on;
        }
        else
        {
                d14_off;
        }
}
void d15_1()
{
        if((IO0PIN>>10)&1)
        {
                d15_on;
        }
        else
        {
                d15_off;
        }
}
void generate(void)  //to generate 1Khz clock, 16 bit parallel in serial out address
{               //16 bit serial in parallel out data, write pulse & ale1
        clk1_on;
        a0_1();
        delay_1();
        clk1_off;
        wp1_on;
        d0_1();
        delay_1();
        clk1_on;
        a1_1();
        ale_1();
        wp1_off;
        delay_1();
        d1_1();
        clk1_on;
        a2_1();
         delay_1();
        clk1_off;
        d2_1();
        delay_1();
        clk1_on;
        a3_1();
        delay_1();
```

```
clk1_off;
d3_1();
delay_1();
clk1_on;
a4_1();
delay_1();
clk1_off;
d4_1();
delay_1();
clk1_on;
a5_1();
delay_1();
clk1_off;
d5_1();
delay_1();
clk1_on;
a6_1();
delay_1();
clk1_off;
d6_1();
delay_1();
clk1_on;
a7_1();
delay_1();
clk1_off;
d7_1();
delay_1();
clk1_on;
a8_1();
delay_1();
clk1_off;
d8_1();
delay_1();
clk1_on;
a9_1();
delay_1();
clk1_off;
d9_1();
delay_1();
clk1_on;
a10_1();
delay_1();
clk1_off;
d10_1();
delay_1();
clk1_on;
a11_1();
delay_1();
```

```
        clk1_off;
        d11_1();
        delay_1();
        clk1_on;
        a12_1();
        delay_1();
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;
        a13_1();
        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;
        a14_1();
        delay_1();
        clk1_off;
        d14_1();
        delay_1();
        clk1_on;
        a15_1();
        delay_1();
        clk1_off;
        d15_1();
        delay_1();
}
void generate1(void) //to generate 2Khz clock, 16 bit parallel in serial out address
{               //16 bit serial in parallel out data, write pulse & ale2
        clk1_on;
        a0_1();
        delay_1();
        clk1_off;
        wp1_on;
        d0_1();
        delay_1();
        clk1_on;
        a1_1();
        ale_2();
        wp1_off;
        delay_1();
         d1_1();
         clk1_on;
        a2_1();
        delay_1();
         clk1_off;
        d2_1();
```

```
delay_1();
clk1_on;
a3_1();
delay_1();
clk1_off;
d3_1();
delay_1();
clk1_on;
a4_1();
delay_1();
clk1_off;
d4_1();
delay_1();
clk1_on;
a5_1();
delay_1();
clk1_off;
d5_1();
delay_1();
clk1_on;
a6_1();
delay_1();
clk1_off;
d6_1();
delay_1();
clk1_on;
a7_1();
delay_1();
clk1_off;
d7_1();
delay_1();
clk1_on;
a8_1();
delay_1();
clk1_off;
d8_1();
delay_1();
clk1_on;
a9_1();
delay_1();
clk1_off;
d9_1();
delay_1();
clk1_on;
a10_1();
delay_1();
clk1_off;
d10_1();
```

```
        delay_1();
        clk1_on;
        a11_1();
        delay_1();
        clk1_off;
        d11_1();
        delay_1();
        clk1_on;
        a12_1();
        delay_1();
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;
        a13_1();
        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;
        a14_1();
        delay_1();
        clk1_off;
        d14_1();
        delay_1();
        clk1_on;
        a15_1();
        delay_1();
        clk1_off;
        d15_1();
        delay_1();
}
void generate2(void)//to generate 40Khz clock, 16 bit parallel in serial out address
{               //16 bit serial in parallel out data, write pulse & ale3
        clk1_on;
        a0_1();
        delay_1();
        clk1_off;
        wp1_on;
        d0_1();
        delay_1();
        clk1_on;
        a1_1();
        ale_3();
        wp1_off;
        delay_1();
        d1_1();
        clk1_on;
```

```
a2_1();
delay_1();
clk1_off;
d2_1();
delay_1();
clk1_on;
a3_1();
delay_1();
clk1_off;
d3_1();
delay_1();
clk1_on;
a4_1();
delay_1();
clk1_off;
d4_1();
delay_1();
clk1_on;
a5_1();
delay_1();
clk1_off;
d5_1();
delay_1();
clk1_on;
a6_1();
delay_1();
clk1_off;
d6_1();
delay_1();
clk1_on;
a7_1();
delay_1();
clk1_off;
d7_1();
delay_1();
clk1_on;
a8_1();
delay_1();
clk1_off;
d8_1();
delay_1();
clk1_on;
a9_1();
delay_1();
clk1_off;
d9_1();
delay_1();
clk1_on;
```

```
        a10_1();
        delay_1();
        clk1_off;
        d10_1();
        delay_1();
        clk1_on;
        a11_1();
        delay_1();
        clk1_off;
        d11_1();
        delay_1();
        clk1_on;
        a12_1();
        delay_1();
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;
        a13_1();
        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;
        a14_1();
        delay_1();
        clk1_off;
        d14_1();
        delay_1();
        clk1_on;
        a15_1();
        delay_1();
        clk1_off;
        d15_1();
        delay_1();
}
void generate3(void)//to generate 16 KHz clock, 16 bit parallel in serial out address
{               //16 bit serial in parallel out data, write pulse & ale4
        clk1_on;
        a0_1();
        delay_1();
        clk1_off;
        wp1_on;
        d0_1();
        delay_1();
        clk1_on;
        a1_1();
        ale_4();
```

```
wp1_off;
delay_1();
d1_1();
clk1_on;
a2_1();
delay_1();
clk1_off;
d2_1();
delay_1();
clk1_on;
a3_1();
delay_1();
clk1_off;
d3_1();
delay_1();
clk1_on;
a4_1();
delay_1();
clk1_off;
d4_1();
delay_1();
clk1_on;
a5_1();
delay_1();
clk1_off;
d5_1();
delay_1();
clk1_on;
a6_1();
delay_1();
clk1_off;
d6_1();
delay_1();
clk1_on;
a7_1();
delay_1();
clk1_off;
d7_1();
delay_1();
clk1_on;
a8_1();
delay_1();
clk1_off;
d8_1();
delay_1();
clk1_on;
a9_1();
delay_1();
```

```
	clk1_off;
	d9_1();
	delay_1();
	clk1_on;
	a10_1();
	delay_1();
	clk1_off;
	d10_1();
	delay_1();
	clk1_on;
	a11_1();
	delay_1();
	clk1_off;
	d11_1();
	delay_1();
	clk1_on;
	a12_1();
	delay_1();
	clk1_off;
	d12_1();
	delay_1();
	clk1_on;
	a13_1();
	delay_1();
	clk1_off;
	d13_1();
	delay_1();
	clk1_on;
	a14_1();
	delay_1();
	clk1_off;
	d14_1();
	delay_1();
	clk1_on;
	a15_1();
	delay_1();
	clk1_off;
	d15_1();
	delay_1();
}

int main(void)
{
	PLL0();
	PINSEL1=0x00;  //to set into GPIO Mode
	PINSEL0=0x00;
	IO0DIR=(0<<1)|(0<<2)|(0<<3)|(0<<7)|(0<<8)|(0<<9)|(0<<10)|(0<<11)|(0<<12)|(0<<13
	)|(0<<14)|
```

```
(0<<15)|(0<<16)|(0<<17)|(0<<18)|(0<<19)|(0<<20)|(0<<23)|
(0<<21)|(0<<22)|(0<<25) //configuring the pins direction as Input
|(1<<0)|(1<<4)|(1<<5)|(1<<6); //configuring the pins direction as output
IO1DIR|=(1<<25)|(1<<26)|(1<<31)|(1<<16)|(1<<17)|(1<<18)|(1<<19)|(1<<27)|
(1<<28)|(1<<29)|(1<<30)|(1<<20)|(1<<21)|(1<<22)|(1<<23)|(1<<24);
clk1_dir|=(clk1);
ale1_dir|=(ale1);
wp1_dir|=(wp1);
a0_dir|=(a0);
a1_dir|=(a1);
a2_dir|=(a2);
a3_dir|=(a3);
a4_dir|=(a4);
a5_dir|=(a5);
a6_dir|=(a6);
a7_dir|=(a7);
a8_dir|=(a8);
a9_dir|=(a9);
a10_dir|=(a10);
a11_dir|=(a11);
a12_dir|=(a12);
a13_dir|=(a13);
a14_dir|=(a14);
a15_dir|=(a15);
d0_dir|=(d0);
d1_dir|=(d1);
d2_dir|=(d2);
d3_dir|=(d3);
d4_dir|=(d4);
d5_dir|=(d5);
d6_dir|=(d6);
d7_dir|=(d7);
d8_dir|=(d8);
d9_dir|=(d9);
d10_dir|=(d10);
d11_dir|=(d11);
d12_dir|=(d12);
d13_dir|=(d13);
d14_dir|=(d14);
d15_dir|=(d15);
while(1)
{
        if((IO0PIN>>7)&1) //generate clock of 1khz, SIPO data
        {               //PISO address, wp & ale1 if dip switch is on at 7
                init_Timer1();
                generate();
        }
        if((IO0PIN>>8)&1) //generate clock of 2khz, SIPO data
```

```
                {               //PISO address, wp & ale2 if dip switch is on at 8
                        init_Timer2();
                        generate1();
                }
                if((IO0PIN>>9)&1) //generate clock of 16khz, SIPO data
                {               //PISO address, wp & ale4 if dip switch is on at 9
                        init_Timer4();
                        generate3();
                }
                if((IO0PIN>>11)&1) //generate clock of 40khz, SIPO data
                {                //PISO address, wp & ale3 if dip switch is on at 11
                        init_Timer3();
                        generate2();
                }
        }
}
```

TC CODE

```
#include<LPC214x.h>
#define PLOCK 0X0000400
void PLL0(void);   //Function to set the parameters to PLL to set the clock
void init_Timer1(void);
void init_Timer2(void);
void init_Timer0(void);
#define sw1 (1<<8)     // sw 1 to 3 are frequency selection switches for 1,8, 16 KHz clocks resp.
#define sw2 (1<<9)
#define sw3 (1<<10)
#define sw4 (1<<3)  //sw 4 and 5 are pulse duration selection switch for TP and GC command.
#define sw5 (1<<4)
#define clk1 (1<<6)
#define relay (1<<31)
#define tp (1<<7)  //transfer pulse at pin 7
#define l_to_h (1<<11)  //low to high pulse at pin 11
#define d0 (1<<5)       //d0 to d31 are initialization for serial data output at pin 5
#define d1 (1<<5)
#define d2 (1<<5)
#define d3 (1<<5)
#define d4 (1<<5)
#define d5 (1<<5)
#define d6 (1<<5)
#define d7 (1<<5)
#define d8 (1<<5)
#define d9 (1<<5)
#define d10 (1<<5)
#define d11 (1<<5)
#define d12 (1<<5)
#define d13 (1<<5)
#define d14 (1<<5)
#define d15 (1<<5)
#define d16 (1<<5)
#define d17 (1<<5)
#define d18 (1<<5)
#define d19 (1<<5)
#define d20 (1<<5)
#define d21 (1<<5)
#define d22 (1<<5)
#define d23 (1<<5)
#define d24 (1<<5)
#define d25 (1<<5)
#define d26 (1<<5)
#define d27 (1<<5)
#define d28 (1<<5)
#define d29 (1<<5)
#define d30 (1<<5)
```

```
#define d31 (1<<5)
#define pb1 (1<<28)  //push button 1 for generating clock, serial data output and TP
#define pb2 (1<<29)  // push button 2 for generating Low to high pulse
#define pb3 (1<<30)  // push button 3 for generating  gc command
#define sw1_dir IO0DIR  //initializations as Port 0 pins
#define sw2_dir IO0DIR
#define sw3_dir IO0DIR
#define sw4_dir IO0DIR
#define sw5_dir IO0DIR

#define clk1_dir IO0DIR
#define relay_dir IO0DIR
#define tp_dir IO0DIR
#define l_to_h_dir IO0DIR

#define d0_dir IO0DIR
#define d1_dir IO0DIR
#define d2_dir IO0DIR
#define d3_dir IO0DIR
#define d4_dir IO0DIR
#define d5_dir IO0DIR
#define d6_dir IO0DIR
#define d7_dir IO0DIR
#define d8_dir IO0DIR
#define d9_dir IO0DIR
#define d10_dir IO0DIR
#define d11_dir IO0DIR
#define d12_dir IO0DIR
#define d13_dir IO0DIR
#define d14_dir IO0DIR
#define d15_dir IO0DIR


#define d16_dir IO0DIR
#define d17_dir IO0DIR
#define d18_dir IO0DIR
#define d19_dir IO0DIR
#define d20_dir IO0DIR
#define d21_dir IO0DIR
#define d22_dir IO0DIR
#define d23_dir IO0DIR
#define d24_dir IO0DIR
#define d25_dir IO0DIR
#define d26_dir IO0DIR
#define d27_dir IO0DIR
#define d28_dir IO0DIR
#define d29_dir IO0DIR
#define d30_dir IO0DIR
```

```
#define d31_dir IO0DIR

#define pb1_dir IO0DIR
#define pb2_dir IO0DIR
#define pb3_dir IO0DIR

#define sw1_port IO0PIN  //initializations of port pins
#define sw2_port IO0PIN
#define sw3_port IO0PIN
#define sw4_port IO0PIN
#define sw5_port IO0PIN

 #define clk1_port IO0PIN
 #define relay_port IO0PIN
 #define tp_port IO0PIN
#define l_to_h_port IO0PIN

 #define d0_port IO0PIN
 #define d1_port IO0PIN
 #define d2_port IO0PIN
 #define d3_port IO0PIN
 #define d4_port IO0PIN
 #define d5_port IO0PIN
 #define d6_port IO0PIN
 #define d7_port IO0PIN
 #define d8_port IO0PIN
 #define d9_port IO0PIN
 #define d10_port IO0PIN
 #define d11_port IO0PIN
 #define d12_port IO0PIN
 #define d13_port IO0PIN
 #define d14_port IO0PIN
 #define d15_port IO0PIN

 #define d16_port IO0PIN
 #define d17_port IO0PIN
 #define d18_port IO0PIN
 #define d19_port IO0PIN
 #define d20_port IO0PIN
 #define d21_port IO0PIN
 #define d22_port IO0PIN
 #define d23_port IO0PIN
 #define d24_port IO0PIN
 #define d25_port IO0PIN
 #define d26_port IO0PIN
 #define d27_port IO0PIN
 #define d28_port IO0PIN
 #define d29_port IO0PIN
```

```
 #define d30_port IO0PIN
 #define d31_port IO0PIN

 #define pb1_port IO0PIN
 #define pb2_port IO0PIN
 #define pb3_port IO0PIN

 #define clk1_on clk1_port|= (clk1);  //defining on and off status of  pins required
#define clk1_off clk1_port&= ~(clk1);

#define relay_on relay_port|= (relay);
#define relay_off relay_port&= ~(relay);

#define tp_on tp_port|= (tp);
#define tp_off tp_port&= ~(tp);

#define l_to_h_on l_to_h_port|= (l_to_h);
#define l_to_h_off l_to_h_port&= ~(l_to_h);

#define d0_on d0_port|= (d0);
#define d0_off d0_port&= ~(d0);
#define d1_on d1_port|= (d1);
#define d1_off d1_port&= ~(d1);
#define d2_on d2_port|= (d2);
#define d2_off d2_port&= ~(d2);
#define d3_on d3_port|= (d3);
#define d3_off d3_port&= ~(d3);
#define d4_on d4_port|= (d4);
#define d4_off d4_port&= ~(d4);
#define d5_on d5_port|= (d5);
#define d5_off d5_port&= ~(d5);
#define d6_on d6_port|= (d6);
#define d6_off d6_port&= ~(d6);
#define d7_on d7_port|= (d7);
#define d7_off d7_port&= ~(d7);
#define d8_on d8_port|= (d8);
#define d8_off d8_port&= ~(d8);
#define d9_on d9_port|= (d9);
#define d9_off d9_port&= ~(d9);
#define d10_on d10_port|= (d10);
#define d10_off d10_port&= ~(d10);
#define d11_on d11_port|= (d11);
#define d11_off d11_port&= ~(d11);
#define d12_on d12_port|= (d12);
#define d12_off d12_port&= ~(d12);
#define d13_on d13_port|= (d13);
#define d13_off d13_port&= ~(d13);
#define d14_on d14_port|= (d14);
```

```
#define d14_off d14_port&= ~(d14);
#define d15_on d15_port|= (d15);
#define d15_off d15_port|= (d15);
#define d16_on d16_port|= (d16);
#define d16_off d16_port&= ~(d16);
#define d17_on d17_port|= (d17);
#define d17_off d17_port&= ~(d17);
#define d18_on d18_port|= (d18);
#define d18_off d18_port&= ~(d18);
#define d19_on d19_port|= (d19);
#define d19_off d19_port&= ~(d19);
#define d20_on d20_port|= (d20);
#define d20_off d20_port&= ~(d20);
#define d21_on d21_port|= (d21);
#define d21_off d21_port&= ~(d21);
#define d22_on d22_port|= (d22);
#define d22_off d22_port&= ~(d22);
#define d23_on d23_port|= (d23);
#define d23_off d23_port&= ~(d23);
#define d24_on d24_port|= (d24);
#define d24_off d24_port&= ~(d24);
#define d25_on d25_port|= (d25);
#define d25_off d25_port&= ~(d25);
#define d26_on d26_port|= (d26);
#define d26_off d26_port&= ~(d26);
#define d27_on d27_port|= (d27);
#define d27_off d27_port&= ~(d27);
#define d28_on d28_port|= (d28);
#define d28_off d28_port&= ~(d28);
#define d29_on d29_port|= (d29);
#define d29_off d29_port&= ~(d29);
#define d30_on d30_port|= (d30);
#define d30_off d30_port&= ~(d30);
#define d31_on d31_port|= (d31);
#define d31_off d31_port&= ~(d31);

#define PINSEL0 (*((volatile unsigned long *) 0xE002C000))  //pinselect and registers required
for PLL and their addresses
#define PLL0CON (*((volatile unsigned char *) 0xE01FC080))
#define PLL0CFG (*((volatile unsigned char *) 0xE01FC084))
#define PLL0STAT (*((volatile unsigned short *) 0xE01FC088))
#define PLL0FEED (*((volatile unsigned char *) 0xE01FC08C))
#define PINSEL1 (*((volatile unsigned long *) 0xE002C004))

void PLL0(void)                         //to set PLL for defining the operating clock frequency of
device
{
        PLL0CON = 0X01;
```

```
        PLL0CFG = 0X24;
        PLL0FEED = 0XAA;
        PLL0FEED = 0X55;
        while(!(PLL0STAT&PLOCK));
        PLL0CON = 0X03;
        PLL0FEED = 0XAA;
        PLL0FEED = 0X55;
        VPBDIV = 0X01;
}


void init_Timer0(void)      // for clock of 1khz
{
        T0TCR = 0X0;
        T0PR = 29812;
        T0MCR = 0X00000001;
T0MR0 = 1;
T0TCR = 0X02;
}

void init_Timer1(void)          // for clock of 8khz
{
        T0TCR = 0X0;
        T0PR = 3749;
        T0MCR = 0X00000001;
 T0MR0 = 1;
  T0TCR = 0X02;
}

void init_Timer2(void)   // for clock of 16khz
{
        T0TCR = 0X0;
        T0PR = 1874;
        T0MCR = 0X00000001;
T0MR0 = 1;
 T0TCR = 0X02;
}

void delay_1(void)
{
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}


void delay_2(void)  //for 64 ms delay used for TP and GC Command
```

```
{
        T0TCR = 0X0;
        T0PR = 3839999;//64ms
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

        void delay_3(void)  //100 ms delay for GC Command
{
        T0TCR = 0X0;
        T0PR = 5999999;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}

 void delay_4(void)  // delay of 16ms for TP
{
        T0TCR = 0X0;
        T0PR = 959999;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
}
void delay_250(void)  //250us delay before and after 1khz clock burst
{
        T0TCR = 0X0;
        T0PR = 14999;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
```

```
}

void delay_6(void)  //31.25us of delay before and after 8khz clock burst

{
T0TCR = 0X0;
        T0PR = 1874;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }

void delay_7(void) //15.625us of delay before and after 16khz clock burst
{
T0TCR = 0X0;
        T0PR = 937;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }

void delay_8(void)  //delay (5ms -250us) for TP
{
T0TCR = 0X0;
        T0PR = 284999;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }
void delay_8_1(void)  //delay (32us -31.25us) for TP
{
T0TCR = 0X0;
        T0PR = 44;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
```

```
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }
void delay_8_2(void)  //delay (16us -15.625us) for TP
{
T0TCR = 0X0;
        T0PR = 22;
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }



void delay_1s (void)  //1s delay to ensure only 1 burst per push button press
{
T0TCR = 0X0;
        T0PR = 59999999;  //(1s)
        T0MCR = 0X01;
        T0MR0 = 1;
        T0TCR = 0X02;
        T0TCR = 0X01;
        while(T0TC!=T0MR0);
        T0TCR = 0X02;
        T0TC = T0PC = 0;
        }


void d0_1()      //functions to send parallel data i/p's to serial data o/p at pin5
{
        if((IO1PIN>>16)&1)
        {
                d0_on;
        }
        else
        {
                d0_off;
        }
}
void d1_1()
{
```

```
	if((IO1PIN>>31)&1)
	{
		d1_on;
	}
	else
	{
		d1_off;
	}
}
void d2_1()
{
	if((IO1PIN>>26)&1)
	{
		d2_on;
	}
	else
	{
		d2_off;
	}
}
void d3_1()
{
	if((IO1PIN>>25)&1)
	{
		d3_on;
	}
	else
	{
		d3_off;
	}
}
void d4_1()
{
	if((IO1PIN>>24)&1)
	{
		d4_on;
	}
	else
	{
		d4_off;
	}
}
void d5_1()
{
	if((IO1PIN>>23)&1)
	{
		d5_on;
	}
```

```
	else
	{
		d5_off;
	}
}
void d6_1()
{
	if((IO1PIN>>22)&1)
	{
		d6_on;
	}
	else
	{
		d6_off;
	}
}
void d7_1()
{
	if((IO1PIN>>21)&1)
	{
		d7_on;
	}
	else
	{
		d7_off;
	}
}
void d8_1()
{
	if((IO1PIN>>20)&1)
	{
		d8_on;
	}
	else
	{
		d8_off;
	}
}
void d9_1()
{
	if((IO1PIN>>30)&1)
	{
		d9_on;
	}
	else
	{
		d9_off;
	}
```

```
}
void d10_1()
{
	if((IO1PIN>>29)&1)
	{
		d10_on;
	}
	else
	{
		d10_off;
	}
}
void d11_1()
{
	if((IO1PIN>>28)&1)
	{
		d11_on;
	}
	else
	{
		d11_off;
	}
}
void d12_1()
{
	if((IO1PIN>>27)&1)
	{
		d12_on;
	}
	else
	{
		d12_off;
	}
}
void d13_1()
{
	if((IO1PIN>>19)&1)
	{
		d13_on;
	}
	else
	{
		d13_off;
	}
}
void d14_1()
{
	if((IO1PIN>>18)&1)
```

```
	{
		d14_on;
	}
	else
	{
		d14_off;
	}
}
void d15_1()
{
	if((IO1PIN>>17)&1)
	{
		d15_on;
	}
	else
	{
		d15_off;
	}
}
void d16_1()
{
	if((IO0PIN>>12)&1)
	{
		d16_on;
	}
	else
	{
		d16_off;
	}
}
void d17_1()
{
	if((IO0PIN>>13)&1)
	{
		d17_on;
	}
	else
	{
		d17_off;
	}
}
void d18_1()
{
	if((IO0PIN>>14)&1)
	{
		d18_on;
	}
	else
```

```
	{
		d18_off;
	}
}
void d19_1()
{
	if((IO0PIN>>15)&1)
	{
		d19_on;
	}
	else
	{
		d19_off;
	}
}
void d20_1()
{
	if((IO0PIN>>16)&1)
	{
		d20_on;
	}
	else
	{
		d20_off;
	}
}
void d21_1()
{
	if((IO0PIN>>17)&1)
	{
		d21_on;
	}
	else
	{
		d21_off;
	}
}
void d22_1()
{
	if((IO0PIN>>18)&1)
	{
		d22_on;
	}
	else
	{
		d22_off;
	}
}
```

```
void d23_1()
{
	if((IO0PIN>>19)&1)
	{
		d23_on;
	}
	else
	{
		d23_off;
	}
}
void d24_1()
{
	if((IO0PIN>>20)&1)
	{
		d24_on;
	}
	else
	{
		d24_off;
	}
}
void d25_1()
{
	if((IO0PIN>>21)&1)
	{
		d25_on;
	}
	else
	{
		d25_off;
	}
}
void d26_1()
{
	if((IO0PIN>>22)&1)
	{
		d26_on;
	}
	else
	{
		d26_off;
	}
}
void d27_1()
{
	if((IO0PIN>>23)&1)
	{
```

```
                d27_on;
        }
        else
        {
                d27_off;
        }
}
void d28_1()
{
        if((IO0PIN>>25)&1)
        {
                d28_on;
        }
        else
        {
                d28_off;
        }
}
void d29_1()
{
        if((IO0PIN>>0)&1)
        {

                d29_on;
        }
        else
        {
                d29_off;
        }
}
void d30_1()
{
        if((IO0PIN>>1)&1)
        {
                d30_on;
        }
        else
        {
                d30_off;
        }
}
void d31_1()
{
        if((IO0PIN>>2)&1)
        {
                d31_on;
        }
        else
```

```
        {
                d31_off;
        }
}


void generate(void)    //to generate 1Khz clock,32bit parallel in serial out data & transfer pulse
{
        d0_1();
delay_250();
        init_Timer0();
        clk1_on;
        delay_1();
        clk1_off;

        d1_1();
        delay_1();
        clk1_on;
delay_1();
        clk1_off;

        d2_1();
delay_1();
clk1_on;
delay_1();
clk1_off;

d3_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d4_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d5_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d6_1();
        delay_1();
        clk1_on;
```

```
        delay_1();
        clk1_off;
        d7_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d8_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d9_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d10_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d11_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d14_1();
        delay_1();
        clk1_on;
```

```
        delay_1();
        clk1_off;
        d15_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d16_1();

        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d17_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d18_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d19_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d20_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d21_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d22_1();
        delay_1();
```

```
        clk1_on;

        delay_1();
        clk1_off;
        d23_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d24_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d25_1();
        delay_1();
clk1_on;

        delay_1();
        clk1_off;
        d26_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d27_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d28_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d29_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d30_1();
        delay_1();
```

```
   clk1_on;

        delay_1();
        clk1_off;
        d31_1();
        delay_1();
        clk1_on;
        delay_1();
        clk1_off;
        delay_250();
        IO0CLR=(1<<5); //clear data o/p pin once last bit is out
        delay_8();
        tp_on;
        if((IO0PIN>>4)&1)  //check for pin 4 status to have a 16 or 64ms TP
        delay_2();
        else
        delay_4();
        tp_off;

}

 void generate1(void)    //to generate 8Khz clock,32bit parallel in serial out data & transfer pulse

{
        d0_1();
delay_6();
        init_Timer1();
        clk1_on;
        delay_1();
        clk1_off;

        d1_1();
        delay_1();
        clk1_on;
delay_1();
        clk1_off;

        d2_1();
delay_1();
clk1_on;
delay_1();
clk1_off;

d3_1();
        delay_1();
        clk1_on;

        delay_1();
```

```
clk1_off;
d4_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d5_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d6_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d7_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d8_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d9_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d10_1();
delay_1();
clk1_on;

delay_1();
clk1_off;
d11_1();
delay_1();
clk1_on;

delay_1();
```

```
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d14_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d15_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d16_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
   d17_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d18_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d19_1();
        delay_1();
        clk1_on;

        delay_1();
```

```
        clk1_off;
        d20_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d21_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d22_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d23_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d24_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d25_1();
        delay_1();
  clk1_on;

        delay_1();
        clk1_off;
        d26_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d27_1();
        delay_1();
        clk1_on;

        delay_1();
```

```
        clk1_off;
        d28_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d29_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d30_1();
        delay_1();
   clk1_on;

        delay_1();
        clk1_off;
        d31_1();
        delay_1();
        clk1_on;
        delay_1();
        clk1_off;
        delay_6();
        IO0CLR=(1<<5);
        delay_8_1();
        tp_on;
        if((IO0PIN>>4)&1)
        delay_2();
        else
        delay_4();
        tp_off;


}

void generate2(void) //to generate 16Khz clock,32bit parallel in serial out data & transfer pulse

{
        d0_1();
delay_7();
        init_Timer2();
        clk1_on;
        delay_1();
        clk1_off;

        d1_1();
```

```
        delay_1();
        clk1_on;
delay_1();
        clk1_off;

        d2_1();
delay_1();
clk1_on;
delay_1();
  clk1_off;

d3_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d4_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d5_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d6_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d7_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d8_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d9_1();
```

```
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d10_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d11_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d12_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d13_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
   d14_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d15_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d16_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
d17_1();
```

```
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
  d18_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d19_1();
        delay_1();
        clk1_on;



        delay_1();
        clk1_off;
        d20_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d21_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d22_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d23_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d24_1();
        delay_1();
        clk1_on;

        delay_1();
```

```
        clk1_off;
        d25_1();
        delay_1();
clk1_on;

        delay_1();
        clk1_off;
        d26_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d27_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d28_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d29_1();
        delay_1();
        clk1_on;

        delay_1();
        clk1_off;
        d30_1();
        delay_1();
   clk1_on;

        delay_1();
        clk1_off;
        d31_1();
        delay_1();
        clk1_on;
        delay_1();
        clk1_off;
        delay_7();
        IO0CLR=(1<<5);
        delay_8_2();
        tp_on;
        if((IO0PIN>>4)&1)
        delay_2();
```

```
        else
        delay_4();
        tp_off;
}

int main(void)
{
PLL0();
PINSEL1=0x00;  //to set into GPIO Mode
PINSEL0=0x00;
FIO0DIR=(0<0);
IO0DIR=(0<<2)|(0<<1)|(0<<25)|(0<<23)|(0<<22)|(0<<21)|(0<<20)|(0<<19)|(0<<18)|(0<<17)|(0
<<16)|(0<<15)|(0<<14)|(0<<13)|(0<<12);  //i/p pins
IO1DIR=(0<<17)|(0<<18)|(0<<19)|(0<<27)|(0<<28)|(0<<29)|(0<<30)|(0<<20)|(0<<21)|(0<<22)|
(0<<23)|(0<<24)|(0<<25)|(0<<26)|(0<<31)|(0<<16);
IO0DIR=(0<<28)|(0<<29)|(0<<30)|(0<<8)|(0<<9)|(0<<10)|(0<<3)|(0<<4);
IO0DIR=(1<<31)|(1<<11)|(1<<5)|(1<<6)|(1<<7);  //o/p pins
clk1_dir |=(clk1);
d0_dir |=(d0);
d1_dir |=(d1);
d2_dir |=(d2);
d3_dir |=(d3);
d4_dir |=(d4);
d5_dir |=(d5);
d6_dir |=(d6);
d7_dir |=(d7);
d8_dir |=(d8);
d9_dir |=(d9);
d10_dir |=(d10);
d11_dir |=(d11);
d12_dir |=(d12);
d13_dir |=(d13);
d14_dir |=(d14);
d15_dir |=(d15);
d16_dir |=(d16);
d17_dir |=(d17);
d18_dir |=(d18);
d19_dir |=(d19);
d20_dir |=(d20);
d21_dir |=(d21);
d22_dir |=(d22);
d23_dir |=(d23);
d24_dir |=(d24);
d25_dir |=(d25);
d26_dir |=(d26);
d27_dir |=(d27);
d28_dir |=(d28);
d29_dir |=(d29);
```

```
d30_dir |=(d30);
d31_dir |=(d31);
tp_dir |=(tp);
l_to_h_dir |=(l_to_h);
relay_dir |=(relay);

while(1)
{

if((IO0PIN>>28)&1)  //generate clock of required freq, data  & TP if push button at 28 is
pressed.
{
if((IO0PIN>>8)&1)
{
generate();
        delay_1s ();
}

else if((IO0PIN>>9)&1)
{

generate1();
        delay_1s ();
}

else if((IO0PIN>>10)&1)
{
generate2();
        delay_1s ();
}
}
else{ IO0CLR=(1<<5);}

if((IO0PIN>>29)&1) //generate Low to High pulse if push button at 29 pressed
{
        l_to_h_on;
        delay_2();
        l_to_h_off;
        delay_2();
        delay_1s ();
}
if((IO0PIN>>30)&1)  //to generate GC command when push button at 30 is pressed
{
        if((IO0PIN>>3)&1)//check pin 3 for 64 or 100ms GC command duration
        {
relay_on;
                delay_2();
                relay_off;
```

```
                delay_2();
                delay_1s ();
        }
        else
        {
 relay_on;
                delay_3();
                relay_off;
                delay_3();
delay_1s ();
        }
}
}
}
```