SUMMER TRAINING PROJECT REPORT

On

Development of MCM test programs for Rabbit Micro-controller

Completed at

Giant Metrewave Radio Telescope

National Center for Radio Astrophysics,

Tata Institute of Fundamental Research.

Khodad, Pune – 410504
Maharashtra, India

By

Avinaba Dutta
Pre-Final Year Student
Electronics and Communication Engineering
Narula Institute of Technology,Kolkata(West Bengal)

Under the Guidance of

Mr.R.Balasubramaniam
Senior Engineer
Telemetry Lab Group
GMRT-NCRA-TIFR
Pune,India

ACKNOWLEDGEMENT

It gives me a great pleasure to express my deep sense of gratitude and indebtedness to my guides **Mr. R. Balasubramaniam** and **Mr. Charu Kanade** for their valuable support and encouraging mentality throughout the project. I am highly obliged to them for providing me this opportunity to carry out their ideas and work during my project period and helping me to gain the successful completion of my Project.

I am highly grateful to the Honorable Centre Director of GMRT-NCRA, **Mr. Rajaram Nityananda**; Chief Scientist **Mr. Yashwant Gupta**; and **Mrs. N.S.Deshmukh**, STP Coordinator, GMRT, for giving me this golden opportunity to be a part of this organization for this period.

My special thanks is going to **My Parents**; to My Grand-father, **Dr. S.R.Das;** to Head of the Department of Electronics & Communication Engineering of my college, **Dr.C.R.Datta;** and to **all of the faculties** for allowing me to come here and encouraging me constantly to work hard in this project.

I am also highly obliged to **Mr. Suresh Sabhapathi**, Engineer-E & **Dr. Suvendu Joardar**, Engineer-D; **Mr. Praveen Kumar**, Engineer-F, for their support and valuable encouragement throughout my project.

There are some people without whom the completion of this Project was not possible. I feel very happy to say my deeply thanks to **Mr. C. Satheesh**, **Mr. B. Rajendran** for their familiar behavior with me and their valuable help to my Project.

I am immeasurably thankful to **Narseh M. Sisodiya**,Engineer B; **Mahadeo B. Misal**, **Satpal Gole**, **Yogesh Gaikwad**; **Somak Bhattacharyya**, Project Student of M.Tech, CU; **Pranjal Borah**, Project student of M.Sc for their kind, friendly behaviour and support throughout my project.

Last but not the least My thanks also goes to **Raj Uprade**, Engineer B; **Pallavi Roychowdhury**; **Soumavo Mukherjee**; **Mukesh Bhongade**; **Manish Gupta**; **Ram Krishna** to make my stay at GMRT one of the most memorable period of my life.

I express my deep gratitude to all......

Avinaba Dutta

Giant Metrewave Radio Telescope (GMRT), located near Pune in Maharshtra, India is the world's largest radio telescope working at metre wavelengths. **National Center for RadioAstro Physics**, Pune (NCRA,Pune), a center of the school of natural sciences of the **Tata Institute of Fundamental Research** ,Mumbai (TIFR,Mumbai), has set up this unique facility i.e. GMRT for Radio Astronomical researches at metre wavelengths. GMRT is a very versatile instrument for investigating a variety of Radio Astrophysical problems ranging from nearby Solar System to the edge of observable Universe. GMRT is an aperture synthesis array, consists of an array of 30 fully steerable parabolic dishes of 45 m each.

The Location

The GMRT is located around 80 km north of Pune,Maharashtra at Khodad (Located 19° 5'47.46"N 74° 2'59.07"E). A nearby town is Narayangoan which is around 15 km from the telescope. The office of NCRA is located in the Pune University campus right next to IUCAA**.**

The location of GMRT has been chosen at a remote place, Khodad, away from main city by considering several important criteria such as low man-made radio noise, availability of good communication, vicinity of industrial, educational and other infrastructure and,a geographical latitude sufficiently north of the geomagnetic equator in order to have a reasonably quiet ionosphere and yet be able to observe a good part of the southern sky as well.

Why metre wavelengths **:**

The study of Astronomical Bodies, which has always been a subject full of mysteries that can not be limited to only visible wavelengths, but every wavelength within our astronomical spectrum reveals a separate truth about our Universe.

The study of Universe at high frequencies can easily be done, but the most challenging work was to develop a telescope which can work at low frequencies or more specifically Radio Frequencies because this is the range of frequencies where maximum noises lie. Since in the other countries like USA, France, the RF Noise level is too high, no other country had taken an initiative to develop such a system working at this frequency range. The RF Noise level was comparatively low in India that time. And we have excellent man-power as well as brain-power with world class Technology and Funding. Those things gave India an opportunity to pick up this challenging project of GMRT working at Radio Frequencies.

The metre wavelength part of the radio spectrum has been particularly chosen for study with GMRT because man-made radio interference is considerably lower in this part of the spectrum in India. Although there are many outstanding astrophysics problems which are best studied at metre wavelengths, there has, so far, been no large facility anywhere in the world to exploit this part of the spectrum for astrophysical research. The Astronomical Bodies those can be best studied at metre wavelength through GMRT are – Pulsars, Sun, Jupiter Radio Bursts, Hydrogen Lines, Mily way galaxy, other nearby galaxies and Supernovae.

Technical Information :

The number and configuration of the dishes was optimized to meet the principal astrophysical objectives which require sensitivity at high angular resolution as well as ability to image radio emission from diffuse extended regions. Fourteen (14) of the thirty (30) dishes are located more or less randomly in a compact Central array in a region of about 1 sq km. The remaining sixteen dishes are spread out along the 3 arms of an approximately 'Y' shaped (Similar to VLA) configuration over a much larger region, with the longest interferometric baseline of about 25 km.

The multiplication or correlation of radio signals from all the 435 possible pairs of antennas or interferometers over several hours will thus enable radio images of celestial objects to be synthesized with a resolution equivalent to that obtainable with a single gigantic dish 25 kilometre in diameter. The array will operate in six frequency bands centered around **50, 153, 233, 325, 610** and **1420 MHz**. All these feeds provide dual polarization outputs. In some configurations, dual-frequency observations are also possible.

The **highest angular resolution** achievable will range from about 60 arcsec at the lowest frequencies to about 2 arc sec at **1.4 GHz.**

GMRT SUBSYSTEMS

There are various subsystems in GMRT which are taking care of different systems to operate 30 antennas and to maintain the best output after the obseravtion. The Subsystems are briefly described below:

1. **Front End System**
2. **Feed Position System**
3. **Servo System**
4. **Telemetry System or Monitor & Control System**
5. **Local Oscillator System**
6. **Base Band System**
7. **Sentinel System**
8. **Correlator System**
9. **Fibre optics System.**

FRONT END (FE) System of GMRT supports radio-astronomical observations in 5 different 5 frequency bands centered at 50 MHz, 150 MHz, 235 MHz, 327 MHz, 610 MHz and L-Band extending from 1000 to 1450 MHz. The L-Band is split into four sub-bands centered at 1060 MHz, 1170 MHz, 1280 MHz and 1390 MHz, each with a bandwidth of 120 Mhz. The 150 MHz, 235 MHz and 327 MHz bands have about 40 MHz bandwidth and the 610 MHz band has about 60 MHz bandwidth. Lower frequency bands from 150 to 610 MHz have dual circular polarization channels (Right Hand Circular and Left Hand Circular polarization) which have been conveniently named as CH1 and CH2, respectively. The higher frequency L-Band has dual linear polarization channels (Vertical and Horizontal polarization) and they have been named CH1 and CH2 respectively. The front end system has flexibility to be configured for either dual polarization observation at a single frequency band or single polarization observation at two different frequency bands.

FEED POSITION System(FPS) , as the name suggest is used to precisely position or focus the feeds that are located on the four faces of the rotating turret. The Telescope is to be operated at 150, 233, 327, 610 and 1420 MHz. The feed can be positioned for desired frequency by rotating the feed turret .

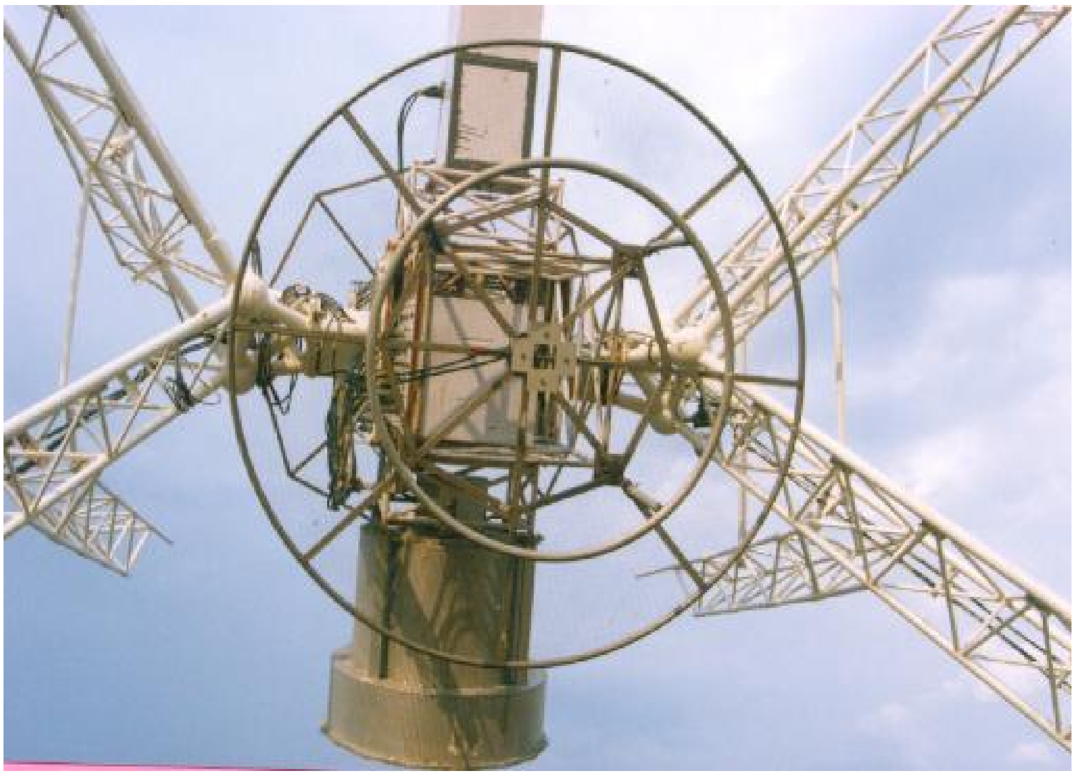
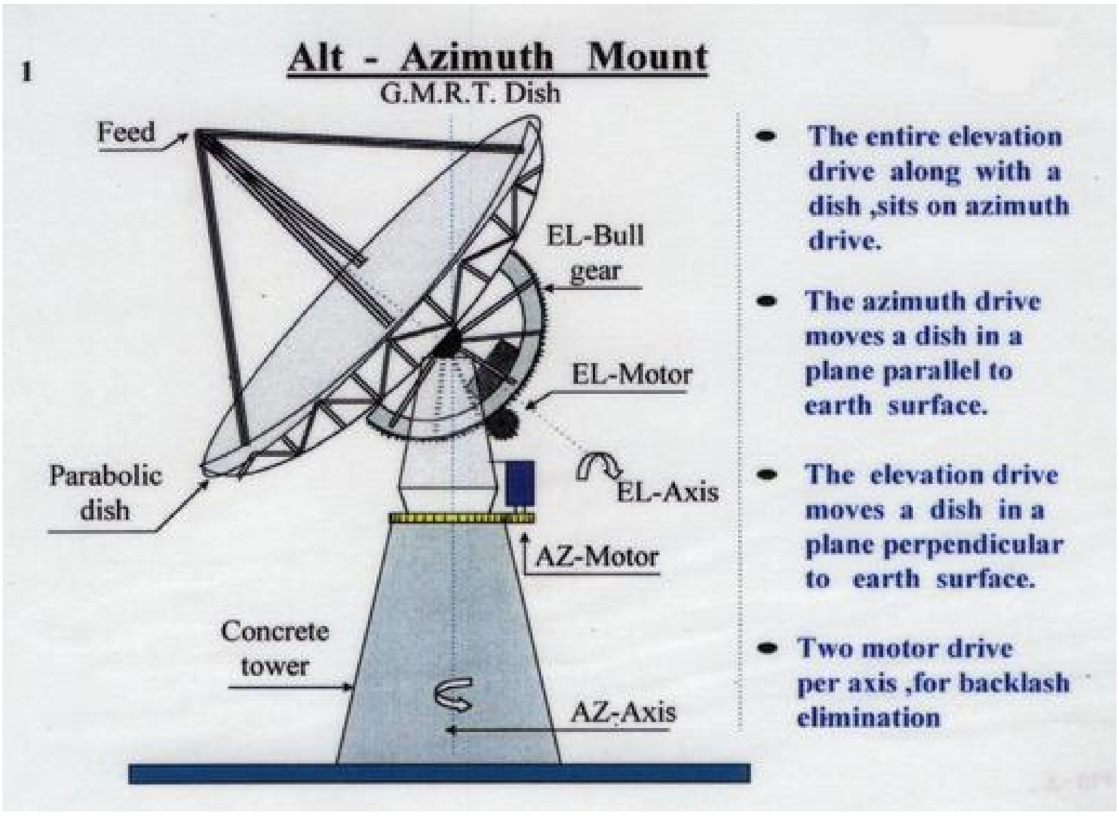


Fig:Four Feeds of one of the 30 antennas

The precise positioning is achieved by using pulse width modulation technique to vary the speed of the DC motor as per the profile shown in fig.4 .This is implemented with 8051 based micro controller and feed drive cassette with power MOSFETS in 'H' configuration. There are various specifications for the Feed Position System such as Accuracy specification, Resolution Specification, Drive Specifications, Feed Drive Controller Specifications etc. As told earlier,telescope operates at frequencies 150 MHz,233 MHz,327 MHz, 610 MHz,a420 MHz. The four feeds are mounted on four faces of the rotating turret at 90 deg apart. The feed drive circuit consists of a half HP DC servo motor with brakes. This is driven by power MOSFET-s.

SERVO System of GMRT controls and monitors the movement of the 30 antennas in Azimuth and Elevation. We all know that there 30 fully steerable parabolic dishes of 45 metre diameter each at GMRT. Motion of these gaint antennas need to be controlled by a precession control system. Pointing of the antennas should be accurate i.e. the radio source, antenna focused and the antenna center should be aligned. The GMRT servo system has designed with three nested control loops to achive the pointing accuracy of (1 or 2) arc minutes RMS for wind speed less than 20 km/ph. Because of high weight alt-azimuth mount is most favorable approach for positioning the dish antenna. Here the elevation axis sits on the azimuth drive. The elevation drive moves antenna up and down directions while azimuth drive moves antenna in clockwise & counter-clockwise direction. Hence enabling the antenna to point anywhere in the sky. There are also various specifications of Servo System at GMRT such as Dish Mount – Dish Movement – Dish speed specifications, Tracking maximum and minimum speed specification, Accuracy Specifications, Operating Voltage specification, Design and Survival wind speed specification, etc.


1
Alt - Azimuth Mount
G.M.R.T. Dish
Feed
EL-Bull gear
EL-Motor
Parabolic dish
EL-Axis
AZ-Motor
Concrete tower
AZ-Axis
The entire elevation drive along with a dish ,sits on azimuth drive.
The azimuth drive moves a dish in a plane parallel to earth surface.
The elevation drive moves a dish in a plane perpendicular to earth surface.
Two motor drive per axis ,for backlash elimination

Fia:Alt-Azimuth Mount of Antenna

TELEMETRY System or Monitor & Control System of GMRT is one of the main System to co-ordination between the various building blocks of the receiver system. This system is used for Controlling the activities of the various building blocks of GMRT like Front End, Local Oscillator, Base Band, Servo and FPS etc and Monitor the healthiness of the same in each of the antenna shells and CEB. It provides the human interface to persons like Telescope Observers,Scientists and maintenance personnel for operating all the antennas from CEB. It has to monitor all parts of the telescope system for correct operation and alert the operator in case of any anomalous behavior. And in the case of severe fault conditions,safety procedures have to be initiated locally. It has also to prevent human error from placing the telescope in a dangerous situation. This Control and Monitor System for GMRT tries to meet all the points mentioned above.

A detailed discussion about this system will be later.

LOCAL OSCILLATOR System of GMRT consists of following sub systems:

- ***Local Oscillator reference Master (LOM)System***
- ***Local Oscillator Reference Remote (LOR)System***
- ***Local Oscillator Synthesizer (LOS) System***
- ***Time and Frequency Standard (FTS)System***

Local Oscillator Reference Master (LOM) System is located at the Central Electronics Building (CEB) in the Receiver Room. High Stability and Low Phase Noise Reference as

Rubidium Oscillator of 10 MHz. A primary Reference for the GMRT Local Oscillator Frequency Standard

BASE BAND System is the final processing stage in the Analog Receiver Chain. It converts intermediate frequency signals received from Antennas to lower frequencies suitable for the Sampler in the Correlator System. The Baseband System also provides facility to vary the observation signal bandwidth over a wide range with a PC based Remote Control and Monitoring Facility. It is located in the Receiver room at the central Electronics Building (CEB) and is installed in Ten RFI shielded electronic cabinets.

The major function of the baseband system at the Central Electronics Building (CEB) is to convert the IF signals received from the antennas to frequencies suitable for the sampler units in the correlator. The two IF signals are given to the Base Band System for converting into four Base Band signals of each 16 MHz bandwidth. Baseband system also facilitates processing of baseband signals by providing switchable bandwidths for the signal to the sampler. It also provides level control circuits so that the power fed to the sampler is always within the acceptable range irrespective of gain variations in the optical fibre and baseband electronics. The output circuitry has buffers for driving the sampler units. For converting the IF signals into Base band Signals there are four types of circuits: Intermediate Frequency Circuits, Baseband Processing Circuits, Base band Local Oscillator Circuits and Base Band Control and Monitor Circuits. The block diagram of the conversion of IF signals to BB signals is given below.

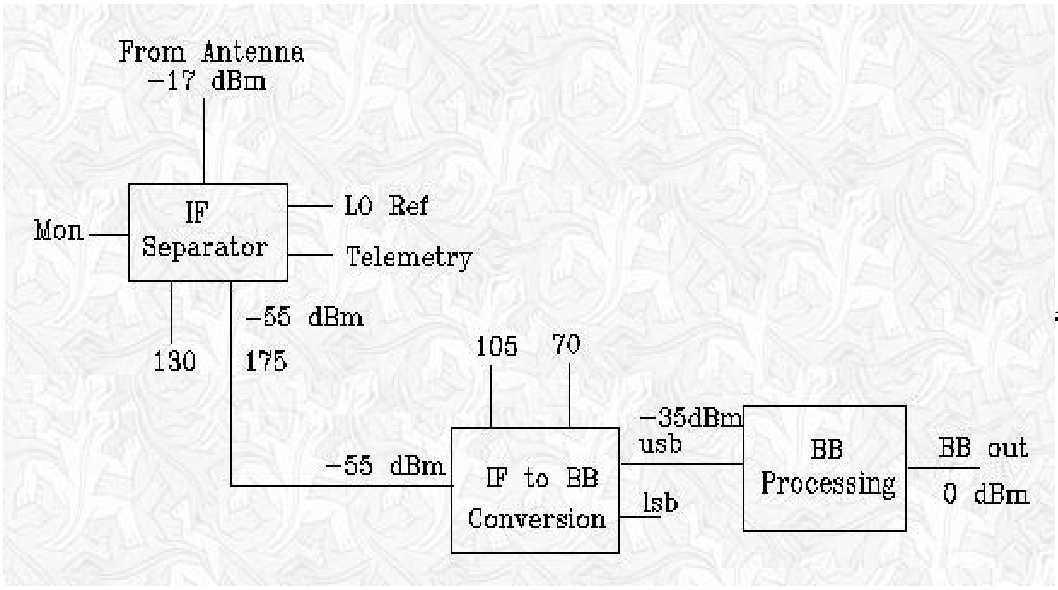


Fig:Block Diagram of Signal Conversion (From IF Signal to BB Signal)

SENTINEL System is one of the major System for different Security purposes inside of the Antennas. GMRT consists of 30 antennas spreading over an area of 50 square kilometers. All the Antenna Shells are equipped with various intelligent sub-systems for providing the best results to this Radio Observatory. Hence it is a prime responsibility for providing safety and protections to all these sub-systems especially in the remote antenna shells from intrusion, fire, smoke, over temperature etc. All these parameters are getting monitored from the remote antenna shells to the Control Room all around the clock. GMRT Sentinel System is taking the full responsibility for monitoring and controlling the above parameters for smooth operation of GMRT. A detailed discussion about that System will be held later.

Details About TELEMETRY System or MONITOR & CONTROL System

As it was said before that TELEMETRY or MONITOR & CONTROL System is one of the most important system in GMRT. It is being used for controlling the various activities of the various building blocks of GMRT like FE, IF, FPS, SERVO, BB etc.

Some key features of the Control and Monitor Systems

1) The rotation of all the thirty antennas in AZIMUTH and ELEVATION through SERVO CONTROL COMPUTER(SCC).

Azimuth--> -270 TO +270 deg
Elevation --> 17 TO 90 deg

--> Using ABC - SERVO RS422 communication link @ 9.6 Kbps.

2) Alignment of the required feed for observation to the focus of the dish thro' Feed Position System(FPS).

0 deg --> 233/610 MHz
90 deg --> 150 MHz
180 deg --> 1420 MHz
270 deg --> 325 MHz

--> Using ABC - FPS RS485 communication link @ 9.6 Kbps.

3) Selection of FRONT END SYSTEM parameters like observing freq. band,Noise Cal,Channel Swap etc through **MCM '5'** & **MCM - FE** interface card in the Common Box.

4) Sets the LO Freq. , IF bandwidth (6,16,32 MHz),IF attn.,ALC ON/OFF etc. using **MCM-s '2', '3' & '9'** located in LO/IF SYSTEM

5) Sets Baseband bandwidth (62.5 kHz to 16 MHz) using MCM-s located in BASEBAND SYSTEM through **Antcom** '0' .

6) Monitors the C & M system parameters at Antenna shell using **MCM '0'** at NRR and at CEB using 6 MCM-s through **Antcom '31'**.

7) Monitors the Temperature at various points in RFI cage in each antenna through Temperature Monitor system using **MCM '0'**.

In addition, it also provides the vital voice communication link between CEB & all the Antennas i.e. You can contact anybody in any antenna using C & M SYSTEM.

The communication medium is a single mode analog optical fibre link operating @ 1310 nm between CEB and all the antennas i.e. two fibres are available for each antenna. So,the digital command & control signals are converted to analog form using non-coherent FSK techniques and sent along with the LO and IF signals. In the forward link,Telemetry command signals are combined with the LO carriers and transmitted to all the antennas and in the return link,Telemetry monitor signals are combined with the IF signals and brought to CEB. The FORWARD link uses 18 MHz carrier & the RETURN link uses 205.5 MHz.

Description of Various Hardware used in M & C System

There are different vital Hardwares in M & C Systems. Those Hardwares are the main building block of this system and by tho help of those Hardwares in different Antennas one can monitor and control various operations in Antenna from Lab itself. Those different Hardwares are described in details below with their block diagram.

Communication Handler(COMH):

COMH is the main communication handler located at MRR C & M rack, handles all the packet communication between UNIX workstation and ANTCOM. COMH is operated in TDM mode i.e. it sends the formatted user commands to the first antenna in the subgroup & waits for the acknowledgement it receives the error free reply before the timeout period, then it accepts the data & selects the next antenna in sequence & the operation continues. But in case if it does not get any reply before the timeout period or if the reception is erroneous then it tries three times and logs the error as Timeout or Checksum error and pass on this information to the ONLINE system for further action.

COMH is basically an 80C186 -16 bit micro-controller based card working at a clock speed of 6 M Hz. This card also contains a Zilog 85C30 dual channel COMMUNICATION CONTROLLER which handles SDLC/HDLC communication through Channel A @ 125 K bits/sec,asynchronous communication @ 38.4 K bits/sec for communication with ANTCOM and PC router respectively. Intel 29C17 CODEC (VOICE CODER-DECODER) handles voice communication @ 62.5 K bits/sec , digital Phase Lock Loop and other combinational logic handles clock recovery and bit interleaving functions and FSK MODEM chips NE 5080 and NE 5081 handles FSK modulation and demodulation.

Antenna Computer (ANTCOM):

ANTCOM is the vital Antenna base computer located in each of the antenna shells. It receives various parameters sent by COMH and performs all the tasks and passes the vital information to various systems like Servo Control Computer,MCM-s and FPS through the following three communication links.

1.Main FSK communication link between COMH & ANTCOM at 250 K bits/sec using OF system.
2.Asynchronous data communication link between ANTCOM & SERVO through RS 422 link.
3.Asynchronous data communication link between ANTCOM & MCM-s

through RS 485 link.

The ANTCOM is the brain behind the operations in each and every antenna shell and services all the relevant peripherals in a cycle time of 1 second. ANTCOM has got the same circuitry as COMH and handles one more serial communication - ANTCOM - MCM communication using INTEL 82510 Communication Controller @ 9.6 K bits/sec in addition to other two serial communications(125 K bits/sec SDLC/HDLC data and 9.6 K bits/sec SERVO data).

Antcom demodulates the FSK signal into 250 K bits/sec data, regenerates 250 kHz clock using Digital Phase Lock Loop,look for sync. bits and if matches with no error or one bit error then demultiplexes the data into COMMAND, VOICE, DIAL PULSE AND AUX. DATA and passes them to respective circuits for further processing(VDM).

The STATE MACHINE shows the functions of the SYNC. DETECTOR.

The user commands are processed and sent to the various systems as follows.

- SERVO commands to SCC through link E.
- MCM and FPS commands through link D.

In the return link,the ANTCOM gets the monitoring information from SCC,MCM-s(5) and FPS and form a packet of SDLC/HDLC data and multiplexes it voice,phone Hook Status and AUX. channels into a single bit stream, converts it into FSK @ 4.5 M Hz. The UP converter up converts this IF into 205.5 MHz signal using regenerated 201 MHz as the LO carrier. This FSK signal is sent along with the IF signals (CH.1 and CH.2) of 32 MHz bandwidth each to CEB (Return Link). Thirty CEBCOM-s at CEB retrieves the monitoring information from the respective antennas and passes them to COMH through MUX 32 card. Also the voice signals from all the antennas are routed through EPABX system for TELEPHONE conversation.

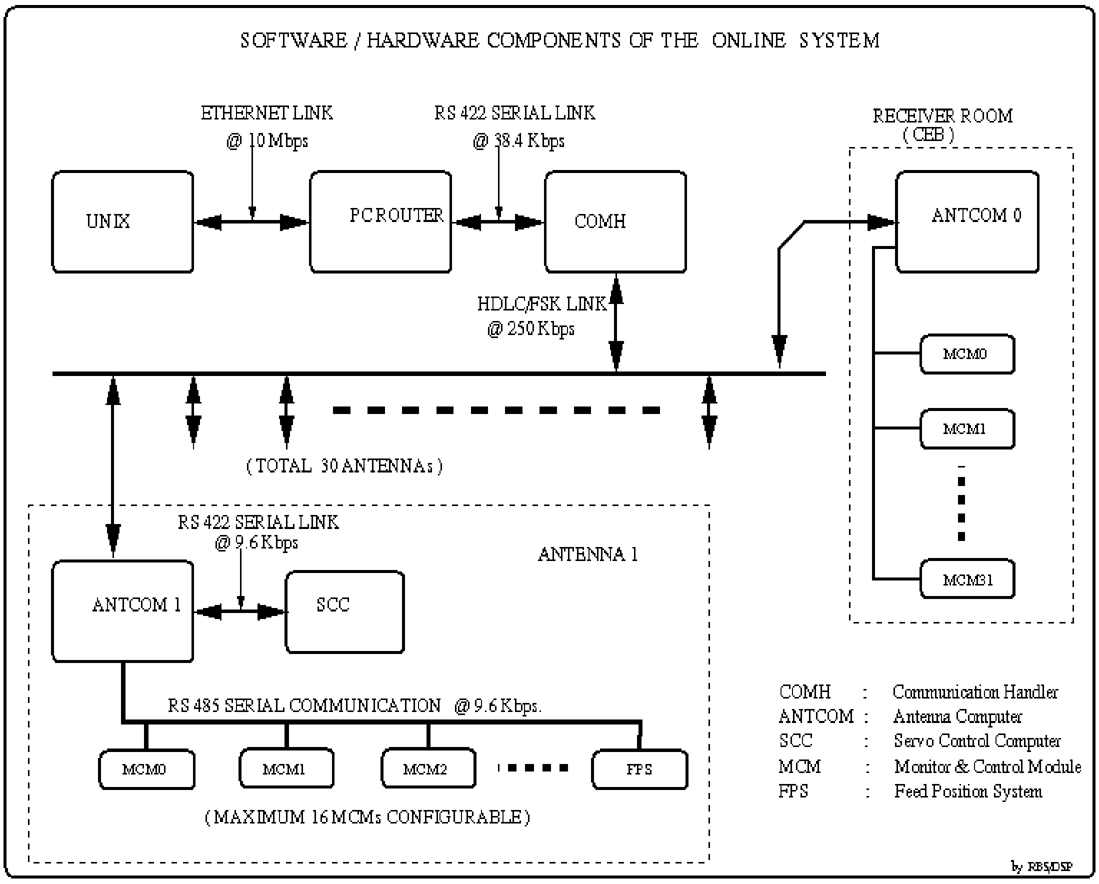
Monitor and Control Module (MCM) Card:

MCM Card is a general purpose Micro-controller based card which provides 16 TTL Control O/P-s and monitors 64 analog signals. Antcom communicates with MCM-s through RS485 communication link @ 9.6 K baud rate and sets various FE,LO and IF system parameters. The chip which is being used here is SAB80C535 which is manufactured by Philips.

Servo Control Computer: The Servo Control Computer located in each antenna shell accepts the movement command,position information etc. through Antcom-SCC link @ 9.6 K bits/sec rate,validates the data and obeys the command. It returns the antenna status information periodically through the same link and will be displayed by the ONLINE software on the monitor

System operation (ONLINE SYSTEM)

The above Block Diagram is for the Software and hardware Configuration of the Online System


SOFTWARE / HARDWARE COMPONENTS OF THE ONLINE SYSTEM
ETHERNET LINK @ 10 Mbps
RS 422 SERIAL LINK @ 38.4 Kbps
RECEIVER ROOM (CEB)
UNIX
PC ROUTER
COMH
ANTCOM 0
HDLC/FSK LINK @ 250 Kbps
MCM0
MCM1
MCM31
(TOTAL 30 ANTENNAs)
RS 422 SERIAL LINK @ 9.6 Kbps
ANTENNA 1
ANTCOM 1
SCC
RS 485 SERIAL COMMUNICATION @ 9.6 Kbps.
MCM0
MCM1
MCM2
FPS
(MAXIMUM 16 MCMs CONFIGURABLE)
COMH : Communication Handler
ANTCOM : Antenna Computer
SCC : Servo Control Computer
MCM : Monitor & Control Module
FPS : Feed Position System
by RBS/DSP

in Monitor and Control System of GMRT.

What is the Online System of GMRT?

The On line Software running on the UNIX workstation provides the user interface for sending all the User Commands like antenna postioning,setting various receiver parameters etc to different systems in MRR & all the ANTENNAS and displays the status of the same. The user commands are formatted by UNIX workstation and sent to all the antennas through COMH at MRR & Antcom located in each of the antenna shell.

System specifications and Bit rates available for various Services of C & M system

System specifications:

1. Non-coherent FSK techniques are used for data transmission over OPTICAL FIBER links.
2. Data Baud Rate is 250 K bits/sec.
3. Bit interleaved techniques are used for multiplexing Telemetry,Voice,Sync.,Dial and Aux. Channels.
4. Polynomial and checksum ERROR DETECTION with ARQ capability.
5. The Bit Error Probability is 10^{-10} .

Available Bit Rates:

1.	Data(COMH – ANTCOMM)	--------> 125 K bits/sec
2.	Voice(Telephony Voice)	--------> 62.5 K bits/sec
3.	Dial(Telephony Signaling)	--------> 15.625 K bits/sec
4.	SYNC(Synchronization Pattern)	-------> 15.625 K bits/sec
5.	AUX1 (Auxiliary Channel1)	--------> 15.625 K bits/sec
6.	AUX2 (Auxiliary Channel 2)	--------> 15.625 K bits/sec

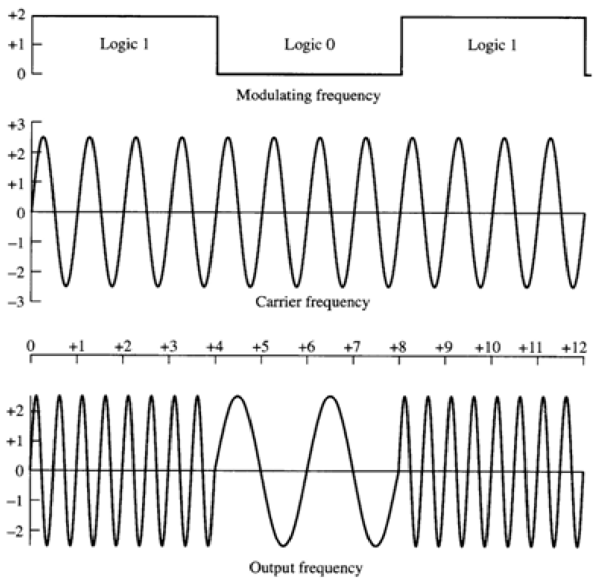
Total Bit Rate----> 250 K bits/sec

Signal Modulation in the C & M system

The C & M system at CEB consists of a digital part,analog part and OF system and the C & M system at antenna shell consists of OF system,analog part and a digital part. I have already mentioned to you that our OF link is an analog link which can carry analog signals from few MHz to about 1 G Hz .But the user commands generated in UNIX WS is a digital data and sent to the COMH as digital signals. So, the digital part multiplexes the digital command data(125 K bits/sec),digitized analog voice (62.5 K bits/sec), dialing information and aux. channels in to a single bit stream of 250 K bits/sec and sent it to the analog part. The analog part, in turn, converts it into Frequency Shift Keying (FSK)signals of appropriate power and sent it to the OF system. The OF system converts the electrical signals to the optical signal @ 1310 nm.
The systems at the antenna end reverse this process and converts it to the digital signal for further processing by the respective subsystems like COMMUNICATION CONTROLLERS and MICRO-CONTROLLERS in ANTCOM and SCC.

What is Frequency Shift Keying(FSK)?

In the method of Frequency Shift Keying or more commonly FSK, the frequency of the carrier signal is changed to two different frequencies depending on the logic state of the input bit stream. The typical output waveform of an FSK is shown below. Notice that a logic high causes the centre frequency to increase to a maximum and a logic low causes the centre frequency to decrease to a minimum.Frequency Shift Keying is a special type of modulation where the digital signals ("0" & "1") changes the frequency of the pseudo carrier to Frequency(MARK) or Frequency(SPACE). If "t" is one bit time,the bandwidth (BW)occupied by the FSK signal is calculated as:



(freq(mark)+1/t) - (freq(space) - 1\t) = freq(mark) - freq(space) + 2/t

RABBIT SEMICONDUCTOR

Rabbit Semiconductor Inc., a Digi International brand, is a global provider of high-performance 8-bit microprocessors and developed tools for Embedded Control, Communications, and Ethernet Connectivity. Rabbit Semiconductors introduced the popular Rabbit Microprocessor 2000 in 1999, the Rabbit Microprocessor 3000 Series in 2002 and Rabbit Microprocessor 4000 Series in 2006. The industry award-winning Rabbit-Core TM line of microprocessor core modules was introduced in 2001. Rabbit Semiconductor offers customers a complete embedded design system, including low-cost development kits and comprehensive technical support for both hardware and software issues. Rabbit Semiconductor is a member of the Fabless Semiconductor Association and is located in Davis, California, USA.
Establishing Embedded Solutions Rabbit's successful line of Rabbit-Core microprocessor core modules offers designers powerful solutions that are quick to develop and easy to integrate. The Rabbit product line has rapidly and continually expanded to encompass a wide variety of form factors, memory capacities, and functionalities as well as meet any other embedded development needs. Recognizing the viability and benefits of embedded networking and communications connectivity, Rabbit Semiconductor offers customers convenient networking development kits, which provide a hassle-free way to develop Ethernet and wireless applications around Rabbit microprocessors. Network-enabled Rabbit-Cores are the next logical step for reducing OEM development time, which further solidifies Rabbit's position in the embedded market.

Software that Makes a Difference Programming Rabbit products is both quick and simple with our industry-proved Dynamic C® integrated software development system, which is specifically designed for embedded control systems and is included in Rabbit development kits. Dynamic C allows embedded engineers to quickly bring robust, cost-effective, real-time communication and control products to market all with a low design risk. There is no need for expensive third-party development tools and in-circuit emulators.

Proven Track Record Rabbit products are found in a wide range of applications among virtually all major industries, including manufacturing, communications, instrumentation, industrial control, utility/energy, security/access control, and medical/health care.

Applications of Rabbit Semiconductor Products:

The various applications of Rabbit Semiconductor Products or Rabbit Core Modules include:

1. Automated meter reading devices (utility)
2. Biodefense systems
3. Wireless fleet/asset tracking and management
4. Audio/video/broadcast equipment control
5. Environmental test chambers
6. Retail point-of-sale (POS) systems
7. Traffic monitoring devices
8. Portable/hand held devices
9. Broadband applications

Rabbit Semiconductor exhibits its true commitment to customer support both pre and post-sale, providing comprehensive tech support to customers.

Different Products Lists of Rabbit Semiconductor:

Rabbit Semiconductor has different types of products which are successfully launched and successfully met the customers' satisfaction. Those products are divided into some groups. The detailed of the different products of Rabbit Semiconductor are given below:

1. **Microprocessors** which include ***Rabbit 2000*** Microprocessors, ***Rabbit 3000*** Microprocessors and ***Rabbit 4000*** Microprocessors.
2. **Rabbit-Cores** which include ***Power Core***, ***RCM2000*** Rabbit Core, ***RCM 2100*** Rabbit Core**,** ***RCM 2200*** Rabbit Core**,** ***RCM 3000*** Rabbit Core**,** ***RCM 3100*** Rabbit Core**,** ***RCM 3200*** Rabbit Core**,** ***RCM 3305*** Rabbit Core**,** ***RCM 3365*** Rabbit Core**,** ***RCM 3400*** Rabbit Core**,** ***RCM 3600*** Rabbit Core**,** ***RCM 3700*** Rabbit Core**,** ***RCM 3750*** Rabbit Core**,** ***RCM 3900*** Rabbit Core**,** ***RCM 4000***

Rabbit Core**,** ***RCM 4100*** Rabbit Core**,** ***RCM 4200*** Rabbit Core**,** ***RCM 4300*** Rabbit Core**,** ***RCM 4400W*** Rabbit Core and ***RCM 4510W*** Rabbit Core**.**

3. **Single-Board Computers** include ***LP3500*** Single-Board Computers, ***BL1800*** Single-Board Computers, ***BL2000*** Single-Board Computers, ***BL2100*** Single-Board Computers, ***BL2500*** Single-Board Computers, ***BL2600*** Single-Board Computers.

4. **Operator Interfaces** include ***OP7100 Smart Screen, OP6700 Intellicom, OP6800 Minicom, OP7200 eDisplay.***

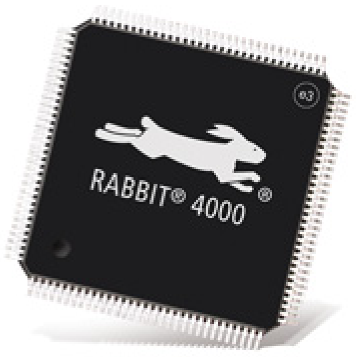
5. **Kits** which include ***Wi-Fi*** Application Kit, ***Wi-Fi Add-On*** Kit, ***Multi-Port Serial-to-Ethernet*** Application Kit, ***GPRS/GSM*** Application Kit, ***Color Touchscreen*** Application Kit, ***Camera Interface*** Application Kit, ***Embedded PLC*** Application Kit, ***Secure Embedded Web*** Application Kit ***2.0***, ***Rabbit RIO Programmable I/O*** Application Kit, ***ZigBee/802.15.A*** Application Kit, ***Wireless Control*** Application Kit.

6. **Miscellaneous** include ***EG2110 Rabbit-Link Card, SF1000 Serial Flash Expansion, Smart Star System, EM1500 Serial-Ethernet Bridge, RN1100 Rabbit-Net Digital I/O Expansion, RN1200 Rabbit-Net A/D Expansion, RN1300 Rabbit-Net D/A Expansion.***

7. **Software** which includes ***Dynamic C.***

A detailed Discussion about ***RCM4000*** Rabbit-Core and ***Dynamic C*** will be held later.

<u>Detailed Description on RABBIT-4000 Microprocessor Chip</u>



Rabbit Semiconductor Series was formed expressly to design a better microprocessor for using in **Small and Medium Scale**

Single-Board Computers. The First Microprocessors were invented as **Rabbit-2000** and **Rabbit-3000** series. The latest Series of Rabbit Semiconductor is **Rabbit-4000 Series**. Rabbit Microprocessor Company Designers and Engineers have had experience using Z80, Z180, and HD64180 microprocessors in small single-Board computers. The Rabbit Microprocessors share a similar architecture and a **high degree of compatibility** with these microprocessors, but Rabbit is giving us with a new fresh extra facilities in the Module they have supplied.
The new Rabbit 4000 Microprocessor is a **high performance with low Electro Magnetic Interference(EMI),** and is designed specifically for **Embedded control, Communication, and Ethernet Connectivity.**
Rabbit 4000 Series Microprocessor is a **8-bit processor** but it **outperforms most 16-bit** processor without loosing the efficiency of 8-bit architecture. The most important thing in Rabbit 4000 series or any Series of Rabbit Microprocessor is that it has **C-friendly instruction set** which promotes efficient development of even the most complex application.
The Rabbit 4000 Processor is fast, running up to **60 MHZ**, with compact code and support up to 16 MB of Memory. Operating with a 1.8 V core and 3.3 V or 1.8 V I/O, the Rabbit 4000 boasts an internal 10Base-T Ethernet interface, eight channels of DMA, six Serial Ports with IrDA, 40+ digital I/O, quadrature decoder, PWM Outputs, and Pulse capture and measurement capabilities. It also features a battery -backable real time clock, glueless memory and I/O interfacing, and ultra-low power modes. Four level of interrupt priority allow fast response to real-time events. Its impact instruction set and high clock speeds five Rabbit 4000 exceptionally fast math, logic, and I/O performance.

Key Features Of Rabbit 4000 Series Microprocessors:

The Rabbit 4000 has several powerful design features that practically eliminate EMI problems, which is essential for OEM-s that need to pass CE and regulatory radio-frequency emissions tests. The amplitude of any electromagnetic radiation is reduced by the internal spectrum spreader, by gated clocks (which prevent unnecessary clocking of unused registers), and by separate power planes for the processor core and I/O pins (which reduce noise crosstalk). An auxiliary I/O bus can be used by designers to enable separate buses for I/O and memory or to limit loading the memory bus to reduce EMI and ground bounce problems when interfacing external peripherals to the processor. The auxiliary I/O bus accomplishes this by duplicating the Rabbit's data bus on Parallel Port A, and uses Parallel Port B to provide the processor's six or eight least significant address lines for interfacing with external peripherals.

The Rabbit 4000 requires no external memory driver or interface logic. Its 24-bit address bus, 8- or 16-bit data bus, 3 chip-select lines, 2 output-enable lines, and 2 write-enable lines can be directly interfaced with up to 6 Flash/SRAM devices. A built-in slave port allows the Rabbit 4000 to be used as master or slave in multi-processor systems, permitting separate tasks to be assigned to dedicated processors. An 8-line data port and 5 control signals simplify the exchange of data between devices. A remote cold boot enables startup and programming via a serial port or the slave port.

The Key Features of Rabbit 4000 Microprocessors are listed below:

1. Up to 60 MHz
2. Integrated 10Base-T Ethernet
3. Eight independent DMA channels
4. Supports 8- or 16-bit Flash and SRAM memories

5. Seven Hardware Breakpoints
6. 10x Speed Improvement in AES encryption
7. New instructions to support 32-bit values and math operations
8. On-Board slave port allows the Rabbit to be configured as an intelligent peripheral device
9. Control of clock speed by software allows dynamic trading of power vs. speed
10. 40+ digital I/O lines with up to four layers of alternate pin functions
11. Battery-backable time/date clock
12. Two Watchdog timers
13. 3.3 V I/O standard (can be set to 1.8 V for lower power)
14. RoHS Compliant

Basic Specifications of Rabbit-4000 Microprocessor:

Package	**128-Pin LQFP**	**128-ball TFBGA**
Package Size	16 mm x 16 mm x 1.5 mm	10mm x 10 mm x 1.2 mm
Operating Voltage	1.8 V DC core, 3.3 V DC I/O ring	
Operating Current	0.35 mA/MHz @ 1.8 V/3.3 V	
Operating Temp	-40°C to +85°C	
Maximum Clock Speed	60 MHz	
Digital I/O	40+(Arranged in five 8-bit ports)	
Serial Port	6 CMOS-Compatible 1. All 6 are configurable as asynchronous 2. 4 are configurable as clocked serial (SPI) 3. 2 are configurable as SDLC/HDLC 4. All 6 support IrDA transceiver	
Ethernet Port	10 Base-T	
Baud Rate	Clock Speed/8 max. Asynchronous	
Address Bus	20/24-bit	
Data Bus	8/16-bit.	
Timers	Ten 8-bit, one 10-bit with 2 match registers, and one 16-bit with 8 match registers	
Real-Time Clock	Yes, battery backable	
RTC Oscillator Circuitry	External	

Package	128-Pin LQFP	128-ball TFBGA
Watchdog Timer/Supervisor	Yes	
Clock Modes	1x, 2x, /2, /3, /4, /6, /8	

Power-Down Modes	Sleepy(32k Hz)
	Ultra-Sleepy(16, 8, 2 k Hz)	

Auxiliary I/O Bus	8 data, 8 address lines	

Comparison of Rabbit Microprocessors:

Feature	Rabbit-4000	Rabbit-3000	Rabbit-2000
Maximum Clock Speed, industrial Maximum Clock Speed, commercial	60 MHz 60 MHz	55.5 MHz 58.8 MHz	30 MHz 30 MHz
Maximum Crystal Frequency Main Oscillator	60 MHz	60 MHz	60 MHz
32.768 kHz Crystal Oscillator	External	External	Internal
Operating Voltage, core Operation Voltage, I/O	1.8 V ± 10% 3.3 V or 1.8 V ± 10%	3.3 V ± 10%	5.0 V ± 10%
Maximum I/O Input Voltage	3.6 V	5.5 V	5.5 V
Current Consumption	0.35 mA/MHz @ 3.3 V	2 mA/MHz @ 3.3 V	4 mA/MHz @ 5 V
Number of Package Pins	128	128	100
Size of Package, LQFP/PQFP Spacing Between Package Pins	16 × 16 × 1.5 mm 0.4 mm (16 mils)	16 × 16 × 1.5 mm 0.4 mm (16 mils)	24 × 18 × 3 mm 0.65 mm (26 mils)
Size of Package, TFBGA Spacing Between Package Pins	10 × 10 × 1.2 mm 0.8 mm	10 × 10 × 1.2 mm 0.8 mm	Not available
Separate Power and Ground for I/O Buffers	Yes	Yes	No
Clock Spectrum Spreader	Yes	Yes	Rabbit 2000B/C

Feature	Rabbit-4000	Rabbit-3000	Rabbit-2000
Clock Modes	1× , 2× , /2, /3, /4, /6, /8	1x, 2x, /2, /3 /4, /6, /8	1x, 2x, /4, /8
Powerdown Modes, sleepy Powerdown Modes, ultra sleepy	32 kHz 16, 8, 2 kHz	32 kHz 16, 8, 2 kHz	32 kHz
Low-Power Memory Control	Short and Self-Timed Chip Selects	Short and Self-Timed Chip Selects	None
Number of 8-bit I/O Ports	5	7	5
Auxiliary I/O Data/Address Bus	Yes	Yes	None
Extended Memory Timing for High-Frequency Operation	Yes	Yes	Rabbit 2000C
Number of Serial Ports	6	6	4
Serial Ports Capable of SPI/Clocked Serial	4 (A, B, C, D)	4 (A, B, C, D)	2 (A, B)
Serial Ports Capable of SDLC/HDLC	2 (A, B)	2 (E, F)	None
Asynch Serial Ports With Support for IrDA Communication	6	6	None
Serial Ports with Support for SDLC/HDLC IrDA Communication	2	2	None
Maximum Asynchronous Baud Rate	Clock Speed/8	Clock Speed/8	Clock Speed/32
Ethernet Port	10 Base-T	None	None
Input Capture Units	2	2	None

Design Advantages

1. High-performance 8-bit architecture with integrated peripherals permits efficient and cost-effective hardware design.
2. 10Base-T Ethernet is built right into the Rabbit 4000, leading to smaller footprint for smaller applications
3. Enhanced Instruction Set brings new power and speed to 8-bit systems with numerous one-byte opcodes and 16- and 32-bit logical, arithmetic, and data transfer instructions.
4. Exceptional performance based on highly optimized math libraries.
5. Dynamic C development software for real-time development and debugging of Rabbit-based systems using C or Assembly language.

Programming the Rabbit 4000

The Rabbit 4000 is programmed using the industry-proven Dynamic C® Software Development System— an integrated C compiler, editor, loader, and debugger created specifically for Rabbit-based systems. Developing software with Dynamic C is easy. Users can write, compile, and test both C and Assembly code without leaving the Dynamic C development environment, and no costly in-circuit emulators are required. Full TCP/IP stack with source code is provided royalty-free in Dynamic C and with our Development Kits. TCP/IP support includes PPP and SNMP, socket-level TCP and UDP, FTP, TFTP, HTTP (w/ SSI and CGI), DHCP, SMTP, POP3, and PING.

Block Diagram of Rabbit-4000 Microprocessor:

Internal Description of Rabbit 4000 Microprocessor

This part will describe various Internal things, various Registers, various Parallel ports and Serial Ports, DMA Channels, 10Base-T Ethernet , etc. of the Rabbit 4000 Microprocessor Chip.

1. **CLOCKS:**

The Rabbit 4000 supports up to three separate clocks—the **main clock, the 32 kHz clock,** and **the 20 MHz Ethernet clock**.

The Main Clock is used to derive the processor clock and the peripheral clock inside the processor.

The 32 kHz Clock is used to drive the asynchronous serial bootstrap, the real-time clock, the periodic interrupt, and the watchdog timers. If these features are not used in a design, the use of the 32 kHz clock is optional.

The Ethernet Clock can be driven by the processor clock, the processor clock divided by 2, or by the input on PE6. The Ethernet clock needs to be 20 MHz to conform to the 10Base-T specification.

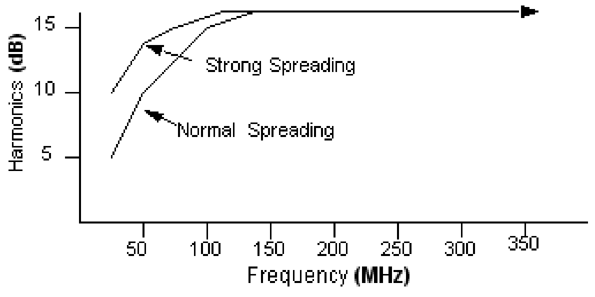
The Rabbit 4000 has a spectrum spreader on the main clock that shortens and lengthens clock cycles. This has the net effect of reducing the peak energy of clock harmonics by spreading the spectral energy into nearby frequencies, which reduces EMI and facilitates government-mandated EMI testing.

The ***Various Registers*** which are being used here are :

Global Control/Status Register (GCSR), Global Clock Modulator 0 Register (GCM0R), Global Clock Modulator 1 Register (GCM1R),Global Clock Double Register (GCDR), Global Output Control Register (GOCR),etc.

What is Spectrum Spreader:

When enabled, the spectrum spreader stretches and compresses the main clock in a complex pattern that spreads the energy of the clock harmonics over a wider range of frequencies. The spectrum spreader either stretches or shrinks the low plateau of the clock by a maximum of **3 ns** for the normal spreading and up to **5 ns** for the strong spreading. If the clock doubler is used, this will cause an additional asymmetry between alternate clock cycles. Both normal and strong modes reduce clock harmonics by approximately 15 dB for frequencies above 100 MHz; for lower frequencies the strong setting has a greater effect in reducing the peak spectral strength as shown in the figure.



What is Clock Doubler:

The clock doubler allows a lower frequency crystal to be used for the main oscillator and to provide an added range over which the clock frequency can be adjusted. The clock doubler is controlled via the Global Clock Double Register (GCDR). The clock doubler uses an on-chip delay circuit that must be programmed by the user at startup if there is a need to double the clock.

2. **Reset and Bootstrap:**

The Rabbit 4000's /RESET pin initializes everything in the processor except for the real-time clock registers and the contents of the battery-backed onchip-encryption RAM. If a write cycle is in progress, it waits until the write cycle is completed to avoid potential memory corruption.

After reset, the Rabbit 4000 checks the state of the SMODE pins. Depending on their state, it either begins normal operation by fetching instruction bytes from /CS0 and /OE0, or it enters a special bootstrap mode where it fetches bytes from either Serial Port A or the slave port. In this mode, bytes can be written to internal registers to set up the Rabbit 4000 for a particular configuration, or to memory to load a program. The processor can begin normal operation once the bootstrap operation is completed.

The ***Register*** which is being used here is:

Slave Port Control Register (SPCR) (Address = 0x0024) is for R/W. The RESET value of this register is 0xx00000. The Function of this register is that it enables/disables processor monitoring of SMODE pins; read current state of SMODE pins.

3. System Management:

There are a number of basic system peripherals in the Rabbit 4000 processor. The peripherals which will be covered briefly in this part are the ***periodic interrupt, the real-time clock, the watchdog timers, the battery-backed onchip-encryption RAM,*** and some of the miscellaneous output pins and their control and processor registers that provide the processor ID and revision numbers.

The Periodic Interrupt, when enabled, is generated every 16 clocks of the 32 kHz clock (every 488 µs, or 2.048 kHz). This interrupt can be used to perform periodic tasks.

The Real-Time Clock (RTC) consists of a 48-bit counter that is clocked by the 32 kHz clock. It is powered by the VBAT pin, and so can be battery-backed.

The Watchdog Timers are clocked by the 32 kHz clock. There are two Watchdog Timers in Rabbit 4000 Microprocessor Chip. The main watchdog timer can be set to time out from 250 ms to 2 seconds, while the secondary watchdog timer can time out from 30.5 µs up to 7.8 ms. The Main Watchdog Timer can reset the processor if not reload within the time, while the secondary watchdog timer generates a Priority 3 secondary watchdog interrupt when it is not reset within the time.

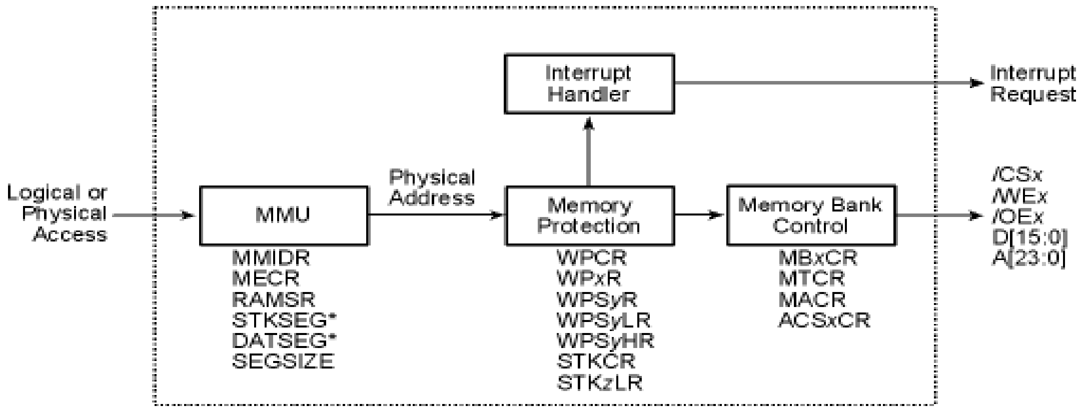
The battery-backed onchip-encryption RAM consists of 32 bytes of memory that are powered by the VBAT pin. Their values are not affected by reset, but are erased if the state of the SMODE pins changes.

4. Memory Management:

The Rabbit 4000 supports both 8-bit and 16-bit external flash and SRAM devices; three chip selects and two read/write-enable strobes allow up to six external devices to be attached at once. The 8-bit mode allows 0, 1, 2, or 4 wait states to be specified for each device, and the 16-bit mode allows 0 to 9 wait states depending on the settings. Both 8-bit and 16-bit page-mode devices are also supported.

The Rabbit 4000's physical memory space contains four consecutive banks, each of which can be mapped to an individual chip-select/enable strobe pair. The banks can be set for equal sizes ranging from 128KB up to 4MB, providing a total physical memory range from 512KB up to 16MB.

The ***Block Diagram*** for Memory Management is shown below:



The ***Various Registers*** which are being used here are described below:

MMU Instruction/Data Register (MMIDR) (Address = 0x0010) is being used for mainly select the Data Line and Address Line by selecting the higher eight bits of the internal I/O address bus.

Stack Segment Register (STKSEG) (Address = 0x0011) is being used to read the current contents of this register or to write eight LSB-s of physical address offset to use if SEGSIZ[7:4] ≤ Addr[15:12] < 0xE.

Stack Segment Low Register (STKSEGL) (Address = 0x001A) is doing the same operation like previous. The difference is that STKSEGL is doing the operation on the four LSB-s, while the previous one is doing the operation on eight LSB-s.

Some other important registers are ***Stack Segment High Register (STKSEGH), Data Segment Register (DATSEG),*** etc. There are more Registers to control the operation in Memory Management. If any one is interested about those things please consult the Users' Manual of Rabbit4000 Microprocessor in the site of Rabbit Semiconductor.

5. Interrupts;

The Rabbit 4000 can operate at one of four priority levels, 0–3, with Priority 0 being the expected standard operating level. The current priority and up to three previous priority levels are kept in the processor's 8-bit IP register, where bits 0–1 contain the current priority. Every time an interrupt is handled or an IPSET instruction occurs, the value in the register is shifted left by two bits, and the new priority placed in bits 0–1. When an IPRES or IRET instruction occurs, the value in IP is shifted right by two bits (bits 0–1 are shifted into bits 6–7). On reset, the processor starts at Priority 3.

The ***Operations*** of these interrupts are given below:

1. Push all registers to be used by the routine onto the stack before use, and pop them off the stack before returning from the ISR.
2. Keep the ISR as short and fast as possible.
3. If the ISR will run for some time, lower the interrupt priority as soon as possible within the ISR to allow other interrupts to occur.
4. A number of special rules apply to interrupts when operating in the system/user mode.

6. External Interrupts:

The Rabbit 4000 has six external interrupts available, and they share two interrupt vectors. In the case of multiple interrupts sharing an interrupt vector, the data register corresponding to the parallel port(s) being used can be read. Each interrupt vector can be set to trigger on a rising edge, a falling edge, or either edge.
The signal on the external interrupt pin must be present for at least three peripheral clock cycles to be detected. In addition, the Rabbit 4000 has a minimum latency of 10 clocks to respond to an interrupt, so the minimum external interrupt response time is three peripheral clock cycles plus 10 processor clock cycles. An external interrupt is generated whenever the selected edge occurs on an enabled pin. The interrupt request is automatically cleared when the interrupt is handled.

The ***Register*** of this External interrupts is described below:

Interrupt x Control Register (IxCR) (Address = 0x0098/0x0099) which enables or disable the interrupts on Parallel Port D and parallel Port E (Low nibble, Falling edge, rising edge) according to the data given as an Input.

7. Various parallel Ports:

There are five parallel Ports in Rabbit4000 Microprocessor Chips. Those Parallel Ports are:
Parallel Port A, Parallel Port B, Parallel Port C, Parallel Port D, Parallel Port E.
These parallel ports have some registers to handle those ports for various purpose (to blink LEC, to display some messages by LCD Display, etc.) by writing the perfect Software in the environment of Dynamic C. Some of the parallel ports have some dual characteristics i.e. beside their own job they can be used for other purposes also. Each Parallel port have 8 pins.
A detailed and in-depth description about those five parallel Ports will be held later while describing our main project since in our main project we have used those parallel Ports in a wide range.

8. Various Timers:

There are four types of Timers in Rabbit4000 Microprocessor. They are **Timer A**, **Timer B** and **Timer C**. The three Timers also used various Registers. But we are not going into the details of those registers since for our Project it is not needed.

TIMER A:

The Timer A peripheral consists of ten separate eight-bit countdown timers, A1–A10. Each counter counts down from a programmed time constant, which is automatically reloaded into the respective counter when the count reaches zero.

TIMER B:

The Timer B peripheral consists of a ten-bit free running up-counter, two match registers, and two step registers. Timer B is driven by perclk/2, by per clk/16, or by the output of timer A1.

TIMER C:

The Timer C peripheral is a 16-bit up-counter clocked by the peripheral clock divided by 2, by the peripheral clock divided by 16, or by the output of countdown timer A1.

9. Various Serial Ports:

There are 6 Serial Ports in the Rabbit 4000 Microprocessor Chip. The six Serial Ports are; **Serial Port A, Serial Port B, Serial Port C, Serial Port D, Serial Port E, Serial Port F.** Among those six serial Ports Serial Port A, Serial Port B, Serial Port C and Serial Port D are identical, except for the source of data clock and the transmit, receive, and clock pins. The rest two Serial Ports i.e. Serial port E and Serial Port F are identical to each other, and their asynchronous operation is identical to that of Serial Ports A-D except for the source of the Data clock, the buffer sizes, and the transmit, receive, and the clock pins.
In our Project we have used only Serial Ports C and D. A detailed description about those Serial ports will be later.

10. Slave Port:

The Slave Port is a parallel communication port that can be used to communicate with an external master device. The Slave port consists of three data input and data output registers, and a Status Register. The data Input registers are written by the master i.e. the external devices and are read by the processor. The data output registers are written by the processor and read by the master. Parallel Port A can be used as a Slave port. Since we have told in the earlier that Parallel Ports have some dual characteristics, so here among those five parallel ports Parallel Port A can be programmed as a Slave port in the case of Master-Slave communication between two Prototype Boards. But Since in our Project the Slave Port is not used that much so we are not going in the detailed description about the Slave Port.

11. DMA Channels:

There are eight independent DMA channels on the Rabbit 4000. All eight channels are identical, and are capable of transferring data to or from memory, external I/O, or internal I/O. The priority between the channels can be either fixed or rotating, and the DMA use of the bus can be limited to guarantee interrupt latency or CPU throughput. The DMA channels are capable of special handling for the last byte of data when sending data to selected internal I/O addresses (such as the HDLC serial ports or to the Ethernet peripheral), and can also transfer end-of-frame status after transferring data from selected internal I/O addresses. The DMA channels are inherently byte-oriented. There are also some registers like ***DMA Master Control/Status Register***, ***DMA master Auto-Load Register***, ***DMA Master Halt Register***, etc.
The above description is the basic things that how DMA channels work. Here also we are not going in the depth of this channels since in our project we have hardly used those channels.

12. Pulse Width Modulation:

The Pulse Width Modulator (PWM) consists of a 10-bit free running counter and four width registers. A PWM output consists of a train of periodic pulses within a 1024-count frame with a duty cycle that varies from 1/1024 to 1024/1024. Each PWM output is high for (n + 1) counts out of the 1024-clock count cycle, where n is the value held in the width register. The PWM is clocked by the output of Timer A9 which is used to set the period. Each PWM output high time can optionally be spread throughout the cycle to reduce ripple on the externally filtered PWM output. The PWM outputs can be passed through a filter and used as a 10-bit D/A converter. The outputs can also be used to directly drive devices such as motors or solenoids that have intrinsic filtering.
That is the basic description of how PWM works in Rabbit4000 Microprocessor Chip. We are skipping from it since we did not use that PWM in our project.

13. 10BASE-T Ethernet:

That is the most important feature of Rabbit4000 Microprocessor. But in our Project we didn't use this inbuilt Ethernet Facility. It is designed for using with Rabbit controllers and other controllers based on the Rabbit microprocessor chip. But after replacing the MCM card by Rabbit Card totally, GMRT is going to use this most important facility for sure. Now i am describing it in very short.
Network Port A implements all of the required digital elements of the 10Base-T standard, and is normally used with two channels of the DMA controller. The receiver provides 32 bytes of buffering, and the transmitter has 16 bytes of buffering. Network Port A connects externally through six dedicated pins. The network port can operate in either half-duplex or full-duplex mode, selected via auto-negotiation. This port requires an accurate 20 MHz clock to generate the 10 Mbits/s serial rate of 10Base-T. The network port contains synchronization circuitry to allow operation from the 20 MHz reference clock while the main system clock runs independently. The network port transmitter precedes the transmit data automatically with a preamble and start-frame-delimiter, and appends CRC and the end-frame-delimiter after the last byte. Frame transmission starts automatically once the transmit FIFO is full and any interframe gap time or back-off time has expired. Transmission is aborted if a collision is detected, and is retried up to 16 times using the standard random back-off time algorithm. Detection of a collision causes the transmitter to send a 32-bit "jam" pattern of all ones to guarantee that all receivers in the network recognize the collision.

These are the most important features of Rabbit4000 Series Microprocessors. Here we have described those features in brief. If anyone is getting interests about the Rabbit4000

Microprocessor to know in details, then please go the below url:

http://www.rabbit.com/products/rab4000/docs.shtml

But among those features for our Project Purpose we have used very few of them. Those Features will be described in details when we will describe the Hardware Part of our Project. In that part we will describe how we have implemented different External Hardware in the Rabbit Core Module 4000 and how we will run our programming which has written in the environment of **DYNAMIC C**.

In the next part we will describe the details of the Software which we have used here for developing our Software.

Dynamic C®

DETAIL DESCRIPTION of The SOFTWARE

Dynamic C is an integrated development system for writing embedded software. It is designed for use with Rabbit controllers and other controllers based on the Rabbit microprocessor. Dynamic C integrates the following development functions:

1. Editing
2. Compiling
3. Linking
4. Loading
5. Debugging.

In fact , compiling, linking and loading are one function. Dynamic C has an easy-to-use, built-in, full-featured text editor. Dynamic C programs can be executed and debugged interactively at the source-code or machine-code level. Pull-down menus and keyboard shortcuts for most commands make Dynamic C easy to use. It also support **ALP (Assembly language Programming).** But in our project we did not used ALP, rather we have used C only for developing our Project. Dynamic C provides extensions to the C language (such as shared and protected variables, costatements and cofunctions) that support real-world embedded system development. Dynamic C supports cooperative and preemptive multitasking. Dynamic C comes with many function libraries, all in source code. These libraries support real-time programming, machine level I/O, and provide standard string and math functions.

SPEED

Dynamic C compiles directly to memory. Functions and libraries are compiled and linked and downloaded on-the-fly. On a fast PC, Dynamic C might load 30,000 bytes of code in five seconds at a baud rate of 115,200 bps.

Differences between Dynamic C and Traditional C

At first we have to say that Dynamic C is being used for the Embedded System but our Traditional C can not used for that same purpose without making adaptations. Standard C makes many assumptions that do not apply to embedded system. For example Traditional C implicitly assumes that an OS is present and that a program starts with a clean state, whereas Embedded systems may have battery-backed memory and may retain data through Power cycles. So to write programs for Embedded System by C language, Rabbit Semiconductor has extended the C language in a number of Areas. Though in Dynamic C one can get the C-friendly environment but still there are some differences between those two. The main differences in Dynamic C from our known Traditional C are summarized in the list below and discussed in detail below:

1. If a variable is explicitly initialized in a declaration (e.g., int x=0), it is stored in Flash Memory (EEPROM) and can not be changed by any assignment statement. Such a declaration will generate a warning that may be suppressed using the **const** keyword.

const int x=0;

To initialize static variable in static RAM (SRAM) we will use #GLOBAL_INIT sections. But the other C compilers will automatically initialize all static variables to zero that are not explicitly initialized before entering the main function. Dynamic C programs do not do this because in an embedded system you may wish to preserve the data in battery-backed RAM on reset.

2. The numerous include files found in typical C programs are not used because Dynamic C has a library system that automatically provides function prototypes and similar header information to the compiler before the user's program is compiled. This is done by **#use** directive.

3. When declaring pointers to function, arguments should not be used in the declaration. Arguments may be used when calling functions indirectly via pointer, but compiler will not check the argument list in the call for correctness.

4. Bit fields are not supported.

5. Separate compilation of different parts of the program is not supported or needed.

Enhancements of Dynamic C

Many other important features have been added to Dynamic C for making it campatible with the Embedded System. Some of these Enhancements are:

1. Function Changing, a concept very unique to Dynamic C, allows special segments of code to be embedded within one or more functions. When a named function chain executes, all the segments belonging to that chain execute. Function chains allow software to perform initialization, data recovery, or other kinds of tasks on request.

2. Cooperative Multitasking is a brand new idea to Dynamic C. It allows c Cooperative Multitasking is a way to perform different tasks at virtually the same time. Unlike Preemptive Multitasking, in cooperative multitasking variables can be shared between different tasks without taking elaborate precautions. Cooperative multitasking also takes advantage of the natural delays that occur in most tasks to more efficiently use the available processor time.

Dynamic C has language extensions to support cooperative multitasking. To implement it they have **COSTATEMENTs** and **COFUNCTIONs**.

Costatements allow cooperative, parallel process to be simulated in a single program, while Costatements allow only cooperative process to be simulated in a single program.

An example of cooperative Multitasking by the costatement function is shown below:

```
main()
  {
```

```
int sec;
sec=0;
while(1)
   {
       costate
            {
                sec++;
                waitfor(DelayMs(1000));
                 printf("%d second\n", sec);
            }
        costate
            {
                 if(!kbhit())
                     abort;
                  printf("key pressed=%c\n",getchar());
            }
   }
}
```

Here in the above example the first cosatement will continue to print the the value of the variable sec by incrementing the value by 1 at every 1 second interval. When the Program Counter got the statement Delay then it will nump from that particular costatement and will run the 2nd costatement. In the 2nd costatement it will check whether any key is being pressed in the 1 second. If not then it will abort from that costatement and will jump to the immediate next line of delay function of 1st costatement and will print the value of sec. But if any key is being pressed in the time of checking the 2nd costatement condition then it will also print that particular character. So by using the costatement and Delay function we can run two tasks in a virtually same time. That is the one of the main advantage of Cooperative Multitasking.

3. Slice Statements allow preemptive process in a single program.

4. Dynamic C supports **Embedded Assembly Code** and **stand-alone Assembly Code.**

5. Dynamic C has keywords that help protect data shared between different context or stored in battery-backed memory.

6. Dynamic C has a set of features that allow the Programmer to make the fullest use of **xmem**. (Extended Memory). Before launching the Dynamic C 10 version 10.21 the compiler supported a 1 MB physical address space. But after launching the version 10.21 the compiler now supports up to the 16 MB of Physical memory; up to 16 MB can be used for data and up to the 1 MB can be used for codes.

Normally Dynamic C takes care of memory management, but there are instances where the programmer will want to take control of it. Dynamic C has various keywords and directives which are helping us to put code and data in proper place. Those keywords will be discussed later.

Here in time of writing my Project Report i have assumed that we all know Standard C language or Traditional C language very well. So here I am not discussing about those features which is totally similar to C like **variable declaration and variable names; else-if statement, various type of variables(int, float, char) looping, Functions, Structures, pointers** etc. I will in brief discuss about those features and keywords which is not present in Traditional C language i.e. which are brand new concept in Dynamic C.

Different keywords in Dynamic C

A keyword is a reserved word in any language like C that represents a basic C construct. It can not be used for any other purposes. If we are using those reserved words or key words for other purposes then it will show some errors.

In Dynamic C one can find that it is enriched by various key words for various functions. But here we will not discuss about the all. We will only discuss in brief about those key words which we have used in writing our Program for our project.

1. abort is used to jump out of a costatement. in time of running a program if the compiler gets any abort keyword in a costatement then it will forcefully jumps out of that costatement and will go to the other costatements or simple statements.

2. auto is used before declaring a variable. A function's local variable is located on the system stack and exists as long as the function call does.

3. break is used to jump out of a loop, if, or case statement.

4. case identifies the next case in a switch statement.

5. char used to declare a variable or array element as an unsigned 8-bit character.

6. const is used to declare that a value will be stored in Flash, thus making it unavailable for modification. It is a type qualifier and may be used with any static or global type specifier. (char, int, float, etc)

7. continue is used to skip to the next iteration of a loop.

8. costate is indicating the beginning of a costtaement. Name can be preseor absent.

9. do indicates the beginning of a do-while loop. The statement must have semicolon at the end.

10. else indicates the false or Zero branch of an if statement.

11. extern indicates that a variable is defined in the BIOS, later in a library file, or in another library file. Its main use is in module headers.

12. float declares variables, function return values, or arrays, as 32-bit floating point.

13. for indicates the beginning of a for loop. A for loop has an initializing expression, a expession and an increment or decrement expression.

14. if indicates the beginning of an if statement.

15. int declares variables, function return values, or array elements to be 16-bit integers. If nothing else specified, int implies a 16-bit signed integer.

16. long declares variables, function return values, or array elements to be 32-bit integers. If nothing else specified, long implies a signed integer.

17. main indicates the main() function. All programs start at the beginning of main() function. It is actually not a key word. It is a function name.

18. sizeof is a built-in function that returns the size in bytes of a variable, array, structure, union, or a data type.

19. static declares a local variable to have a permanent fixed location in memory.

20. typedef is proving a way to create new names for existing data types.

21. unsigned declares a variable or array to be unsigned. if nothing is specified in declaration, unsigned means 16-bit unsigned integer.

22. waitfor used in a costatement or cofunction. This keyword identifies a point of suspension pending the outcome of a condition, completion or an event, or some other delay.

23. while identifies the beginning of a while loop. A while loop tests at the beginning and may execute zero or more times.

24. xmem indicates that a function is to be placed in extended memory. This keyword is semantically meaningful in function prototypes.

25. void keyward conforms to Ansi C. Thus, it can be used in three ways: parameter List, Pointer to void and return type.

Different Compiler Directives

Compiler Directives are special keywords prefixed with a symbol #. They tell the compiler how to proceed. Only one directive per line is allowed, but a directive may span more than one line if backslash ('\') is placed at the end of the line. Here I am discussing about those directives which we have used in our Program for our Project. The detailed description of different Directives and how they are working will be found in the Rabbit 4000 Designer's Handbook and in the Datasheet of Dynamic C which are available in the net.

1. **#asm** begins a block of assembly code. The available options are: const, debug, nodebug, xmem.
2. **#class** controls the storage class for local variable. The available options are: auto and static. The default storage class is auto.
3. **#define** defines a macro or without parameters according to ANSI standard. A macro without parameters may be considered a symbolic constant.
4. **#memmap** controls the default memory area for functions. The available options are: anymem NNNN, root, xmem.
5. **#use** activates a library named in LIB.DIR so modules in the library can be linked with the application program. This directive immediately reads in all the headers in the library unless they have already been read.

Those are the different compiler directives and keywords in Dynamic C which we have used in our developed Software for our Project.
There are different operators in Dynamic C also. Those operators are almost similar to ANSI C. There are different Embedded features in Dynamic C which we have used in different part in our developed Software. Those will be discussed when we will discuss about our Project in the next topic in details.

For more details of Dynamic C, please go the below url:

h**ttp://www.rabbit.com/documentation/docs/manuals/DC/DCUserManual10/**

In the next topic we will discuss about our Project. Our Project has two parts: Hardware parts and Software parts.

PROJECT DETAILS

In this topic we have discussed in details about our project. In the 2 months of time what I have done and developed for GMRT. As told earlier, my project has two parts. One part is related to Hardware, where i have used various external components with RCM4000 Prototype board; and another part is totally based on Software, where i have developed a Demo Software in the environment of Dynamic C for the future replacement of MCM Card by Rabbit Card.

Hardware Description of our Project:

In our Project we have used various types of Hardwares with the RCM4000 Prototype board. Here I have described in details of those hardwares and how they have interfaced with the prototype board. Also i have faced various real-time problems and errors in time of interfacing those various external Hardwares with RCM4000 Prototype board. I have made an Handshaking environment between the RCM4000 Problems Board and various External Hardwares through my Developed Software. At first i will discuss about RCM4000 Prototype Board. Then I will discuss about the Thermistor, A/D Converter, Parallel Ports, Serial Ports, LCD Display and how they are being interfaced with the RCM4000 Prototyping Board through our Developed Software.

RCM4000 Prototype Board:

The RCM4000 series of Rabbit Core modules is one of the next generation of core modules that take advantage of new Rabbit 4000 features such as Hardware DMA, clock speed up to 60 MHz, I/O lines shared with up to five serial ports and four levels of alternate pin functions that include variable-phase PWM, auxiliary I/O, quadrature decoder, and input capture. The RCM4000 also features an integrated 1Base-T Ethernet port. The RCM4000 Development Kit includes a complete Dynamic C software development System. The Development Kit also contains a Prototyping Board that allow user to evaluate the specific RCM4000 module and to prototype circuits that interface to the module. One can also be to write the S/W for RCM4000 modules.

The RCM4000 has a Rabbit 4000 microprocessor operating at up to 58.98 MHz, Static RAM, Flash Memory, NAND flash mass-storage option, an 8-channel 12-bit A/D Converter, two clocks, and the circuitry necessary for reset and management of battery backup of the Rabbit 4000's internal real-time clock and the Static RAM. One 50-pin header brings out the Rabbit 4000 I/O Bus lines, Parallel Ports, A/D converter channels, and Serial Ports. The RCM4000 receives its +3.3 Volt power from the customer supplied motherboard on which it is mounted.

Key Features and various specifications of RCM4000 Prototyping Board:

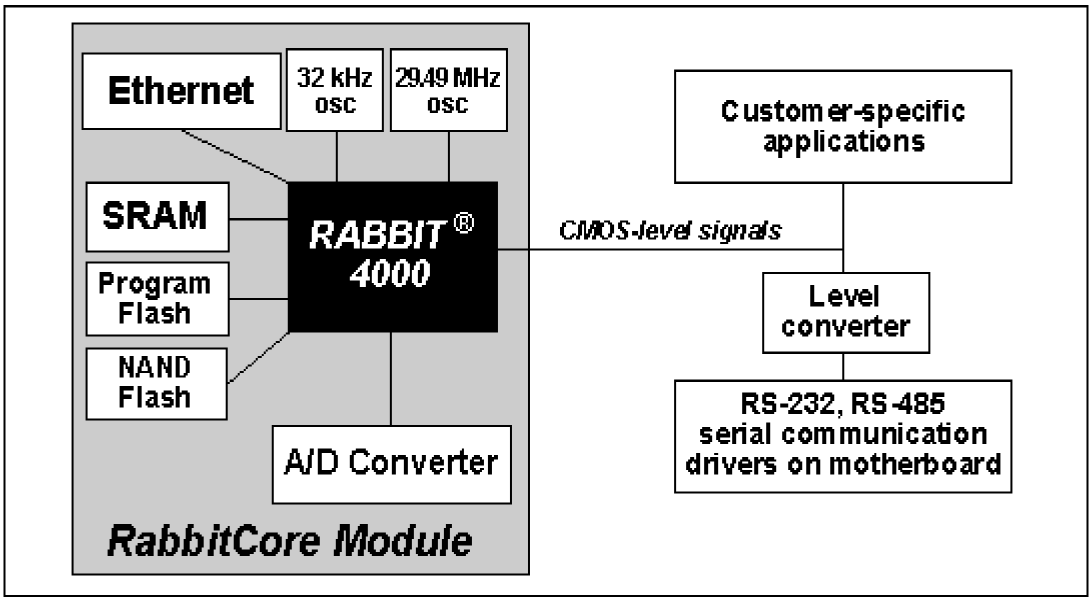
1. **Small Size:** 1.84” x 2.42” x 0.77” (47 mm x 61 mm x 20mm)
2. **Microprocessor:** Rabbit 4000 running at 58.98 Mhz
3. **EMI Reduction:** Spectrum Spreader for reduced EMI.
4. **Ethernet Port:** 10BSE-T Ethernet.
5. **General Purpose I/O:** Up to 29 general-purpose I/O lines configurable with up to four alternate functions
6. **Power:** 3.3 V I/O lines with low-power modes down to 2 kHz
7. **Serial Ports:** Up to five CMOS-compatible serial ports — four ports are configurable as a clocked serial ports (SPI), and one port is configurable as an SDLC/HDLC serial port.
8. **Analog Input:** Combinations of up to eight single-ended or four differential 12-bit analog inputs (RCM4000 only)
9. Alternate I/O bus can be configured for 8 data lines and 6 address lines (shared with parallel I/O lines), I/O read/write
10. **Memory(SRAM and Flash Memory):** 512KB or 1MB flash memory, 512KB or 1 MB SRAM, with a fixed mass-storage flash-memory option that may be used with the standardized directory structure supported by the Dynamic C FAT File System module
11. **Clock:** Real-time clock

12. Watchdog supervisor

Advantages of RCM4000:

1. Fast time to market using a fully engineered, "ready-to-run/ready-to-program" microprocessor core.
2. Competitive pricing when compared with the alternative of purchasing and assembling individual components.
3. Easy C-language program development and debugging
4. Rabbit Field Utility to download compiled Dynamic C .bin files, and cloning board options for rapid production loading of programs.
5. Generous memory size allows large programs with tens of thousands of lines of code, and substantial data storage.

The below figure will show the Hardware Description of RCM4000 Prototyping Board. Here in the Prototyping Board Rabbit 4000 Microprocessor Chip is there and various external hardwares like A/D Converter, Memory chips, Ethernet Port were being interfaced.



Serial Communication through Serial Ports:

The RCM4000 module does not have any Serial Transceivers directly on the board. However, a serial Interface may be incorporated on the board the RCM4000 is mounted on.
As described briefly earlier the Rabbit 4000 Microprocessor has 6 serial ports designated as Serial Port A, Serial Port B, Serial Port Cm, Serial Port D, Serial Port E and Serial Port F. But in RCM4000 Prototype Board there are five Serial Ports designated as Serial Ports A, B, C, D, F. All five Serial Ports can operate in an asynchronous mode up to baud rate of the system clock divided by 8. Baud rate means **'Bit per Second of Data Transfer Rate'.** An asynchronous port can handle 7 or 8 data bits. A 9th bit bit address scheme, where an additional bit is sent to mark the first byte of a message, is also supported.

Serial Port A is normally used as a programming port, but may be used either as an asynchronous or as a clocked serial port once application development has been completed and the RCM4000 is operating in the Run Mode.

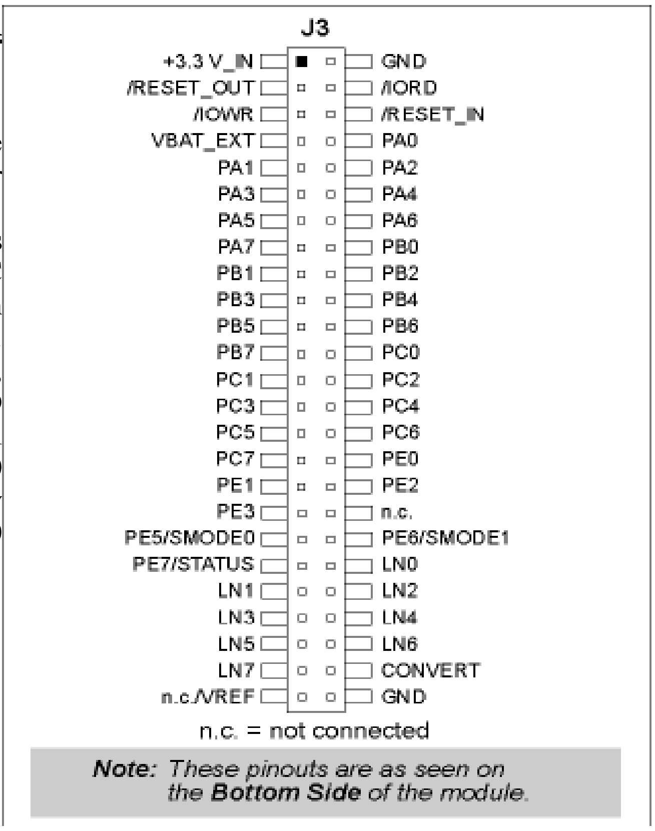
Serial Port B is shared with the A/D Converter of the RCM4000 Module and is set up as a clocked Serial Port. Since this Serial Port is set up for asynchronous Serial Communication of RCM4000 model, user will lose the functionality of the A/D Converter if user tries to use the Serial Port in the Asynchronous mode.

Serial Port C and Serial Port D can also be operated in the clocked Serial Mode. In this mode, a clock line synchronously clocks the data in or out. Either of the two communicating devices supply the clock. In our main Project we have used these two Serial Ports for Serial Communication. We have just sent a character or a bit from Serial Port C to Serial Port D and after getting that bit or character the Serial Port D also sending it intact or by changing in some ways to Serial Port C. I will discuss it in detail when I will discuss about the Software Part of the Project.

Serial Port F, which is available as a matter of course on the RCM4010 and can be used instead of Serial Port C or parallel Port C pins 2 and 3, can also be configured as an SDLC/HDLC Serial Port.
In the Software details of the Project I will discuss how to open the Serial Ports and how to communicate between two Serial Ports via Dynamic C Language.
Now I will discuss in deep about the Serial Port C and D since we have used these two Serial Ports only for our Project purpose. Though by after discussion of Serial Port C and Serial Port D, one can get the full knowledge of Serial Port A and Serial Port B, because these 4 Serial Ports are identical.
As it is discussed earlier that Serial port can be used in Asynchronous or the clocked Serial Mode with an internal or external clock. In the Asynchronous Mode, either 7 or 8 data can be transferred, and a Parity bit and/or an additional address (0) or long stop (1) bit can be appended as well. Parity and the address/long stop bits are also detected when they are received. The Asynchronous mode is Full-Duplex, while the clocked mode can be Half or Full Duplex.
Both transmit and receive have one byte buffering --- a byte may be read while another byte is being received, or the next byte to be transmitted can be loaded while the current byte is still being transferred out. The byte is available in the buffer after the final bit is sampled.
There are various Registers to operate those Serial Ports for example *Serial Port Status Register(SxSR)* to know the Status of the Serial Ports at a particular time. The status of each Serial Port is available in this register and contains information on whether a receive buffer was overrun, a parity error was received, and the transmit buffer is empty or busy sending a byte. The status is updated when the final bit is sent out. I am not going into the details of the various registers. One can download those informations from the Rabbit Website since all those things are available there.

<u>**RCM4000 Digital Inputs and Outputs:**</u>

The figure is showing the RCM4000 pin-outs for header J3.
Here there are four Parallel Ports designated by PA (0 to 7), PB (0 to 7), PC (0 to 7), PE (0- & 5-7). In the Pin Diagram there are also the Pins for A/D Converter. Those are LN, LN, LN2, LN3, LN4, LN5, LN6, LN7. those are the 8 pins for A/D Converter. In the next topic about parallel ports we will see that how Rabbit 4000 Microprocessor in being communicated by various Parallel Ports in the RCM4000 Module.



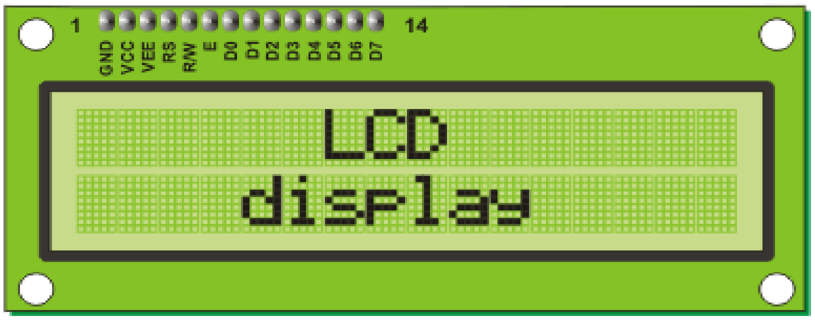
LCD Display:

For the LCD Display Screen we have used **HD44780U** manufactured by Hitachi. The **HD7840U dot-matrix liquid crystal display controller** and driver LSI displays alphanumerics, and symbols. It can be configured to drive a dot-mtrix liquid Crystal display under the control of a 4-bit or 8-bit microprocessor. Since all the functions such as display RAM, character generator, and liquid crystal driver, required for driving a dot-matrix liquid Crystal display are internally provided on one chip, a minimal system can be interfaced with this controller. These components are specialized for being used with the micro controllers, which means that they can not be activated by standard IC circuits. They are used for writing different messages on a miniature LCD.
In our project we have used a LCD which can display 4 lines and in each line 16 characters. In addition, it is possible to display symbols, that user makes up on it own. Automatic shifting message on display (Shift Right, Shift Left), appearance of the pointer, backlight, etc. are considered as useful characteristics.

Pin Diagram and Pin Configuration of HD44780U:

There are pins along one side of the small printed board used for connection to the microcontroller. There are total of 14 pins marked with numbers (16 in case the background light is built in). The pin diagram and their function is given below.

Here the LCD we have used for our project purpose it has 16 pins. The two pins are named as Cathode(K) and Anode(A). Already discussed the functions of those two pins. Now I will discuss the functions of various pins in LCD.

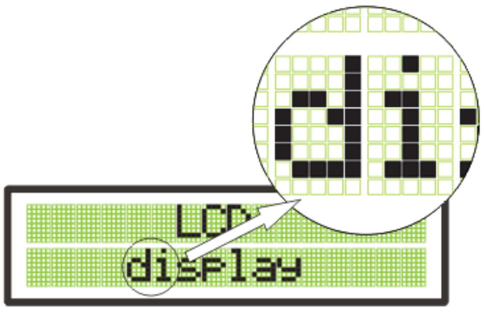


Function	Pin Number	Name	Logic State	Description
Ground	1	Vss	-	0V
Power Supply	2	Vdd	-	+5 V
Contrast	3	Vee	-	0- Vdd
Control of operating	4	RS	0	D0-D7 are interpreted as comments
			1	D0-D7 are interpreted ad Data
	5	R/W	0	Write data (From controller to LCD)
			1	Read Data (From LCD to controller)

Function	**Pin Number**	**Name**	**Logic State**	**Description**
	6	E	0 1 From 1 to 0	Access to LCD disabled Normal Operating Data/commands are transferred to LCD
Data/ commands	7	D0	0/1	Bit 0 LSB
	8	D1	0/1	Bit 1
	9	D2	0/1	Bit 2
	10	D3	0/1	Bit 3
	11	D4	0/1	Bit 4
	12	D5	0/1	Bit 5
	13	D6	0/1	Bit 6
	14	D7	0/1	Bit 7

LCD Screen:

LCD Screen consists of four lines with 16 characters each. Each character consists of 5X8 or 5X11 dot matrix. In our project we have configured the LCD 5X8 Matrix display because it is most commonly used. Contrast on display depends on the power supply voltage and whether messages are displayed in one or two lines. For that reason, variable voltage 0-Vdd is applied on pin marked as Vee. Trimmer potentiometer is usually used for that purpose. Some versions of displays have built in backlight (blue or green diodes). When used during operating, a resistor for current limitation should be used (like with any LE diode).



LCD Basic Commands:

All data transferred to LCD through outputs D0-D7 will be interpreted as commands or as data, which depends on logic state on pin RS. If RS=1,then bits D0-D7 are address of characters that should be displayed. and if RS=0, then Bits D0-D7 are commands which determine display mode. Now lists of commands which LCD recognizes are given in the table.

Commands	**RS**	**RW**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Clear Display	0	0	0	0	0	0	0	0	0	1
Cursor Home	0	0	0	0	0	0	0	0	1	x

Commands	**RS**	**RW**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S
Display On/Off Control	0	0	0	0	0	0	1	D	U	B
Cursor/Display Shift	0	0	0	0	0	1	D/C	R/L	x	x
Function Set	0	0	0	0	1	DL	N	F	x	x
Set CGRAM address	0	0	0	1	CGRAM ADDRESS					
Set DDRAM address	0	0	1	DDRAM ADDRESS						
Read "BUSY" Flag	0	1	BF	DDRAM ADDRESS						
Write to CGRAM or DDRAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0
Read from CGRAM or DDRAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0

I/D	1 = Increment by 1 0 = Decrement by 1	R/L	1 = Shift Right 0 = Shift Left
S	1 = Display Shift ON 0 = Display Shift OFF	DL	1 = 8-bit Interface 0 = 4-bit Interface
D	1 = Display ON 0 = Display OFF	N	1 = Display in two lines 0 = Display in one line
B	1 = Cursor Blink ON 0 = Cursor Blink OFF	F	1 = Character format 5X10 dots 0= character format 5X7 dots
U	1= Cursor ON 0 = Cursor OFF	D/C	1 = Display Shift 0 = Cursor Shift

Depending upon the various bit pattern of the pins there are several commands to make the LCD ready for displaying the output. There are corresponding HEX values for those commands and we have to send those HEX values by the Programming in Dynamic C.

1. **Clear Display:** Hex value is 01H
2. **Cursor Home:** Hex Value is 02H
3. **Entry Mode Set:**
   i. *Decrement Cursor {Shift Cursor Left}:* Hex Value is 04H
   ii. *Increment Cursor [Shift Cursor Right]:* Hex Value is 06H
   iii. *Shift Display Right [One character]:* Hex Value is 05H
   iv. *Shift Display Left [One character]:* Hex value is 07H
4. **Display/Cursor On/Off:**
   1. *Display Off cursor Off:* Hex Value is 08H
   2. *Display Off, cursor On:* Hex Value is 0AH
   3. *Display On, Cursor Off:* Hex Value is 0CH
   4. *Display On, Cursor On:* Hex Value is 0EH
   5. *Display On, Cursor Blinking:* Hex Value is 0FH
   6. *Display Off, Cursor Blinking:* Hex Value is 0BH

5. **Cursor/Display Shift:**

1. *Shift Cursor position to Left:*	Hex Value is 10H
2. *Shift Cursor Position to Right:*	Hex Value is 14H
3. *Shift Entire Display to Left:*	Hex Value is 18H
4. *Shift Entire Display to Right:*	Hex Value is 1CH

6. Function Set:

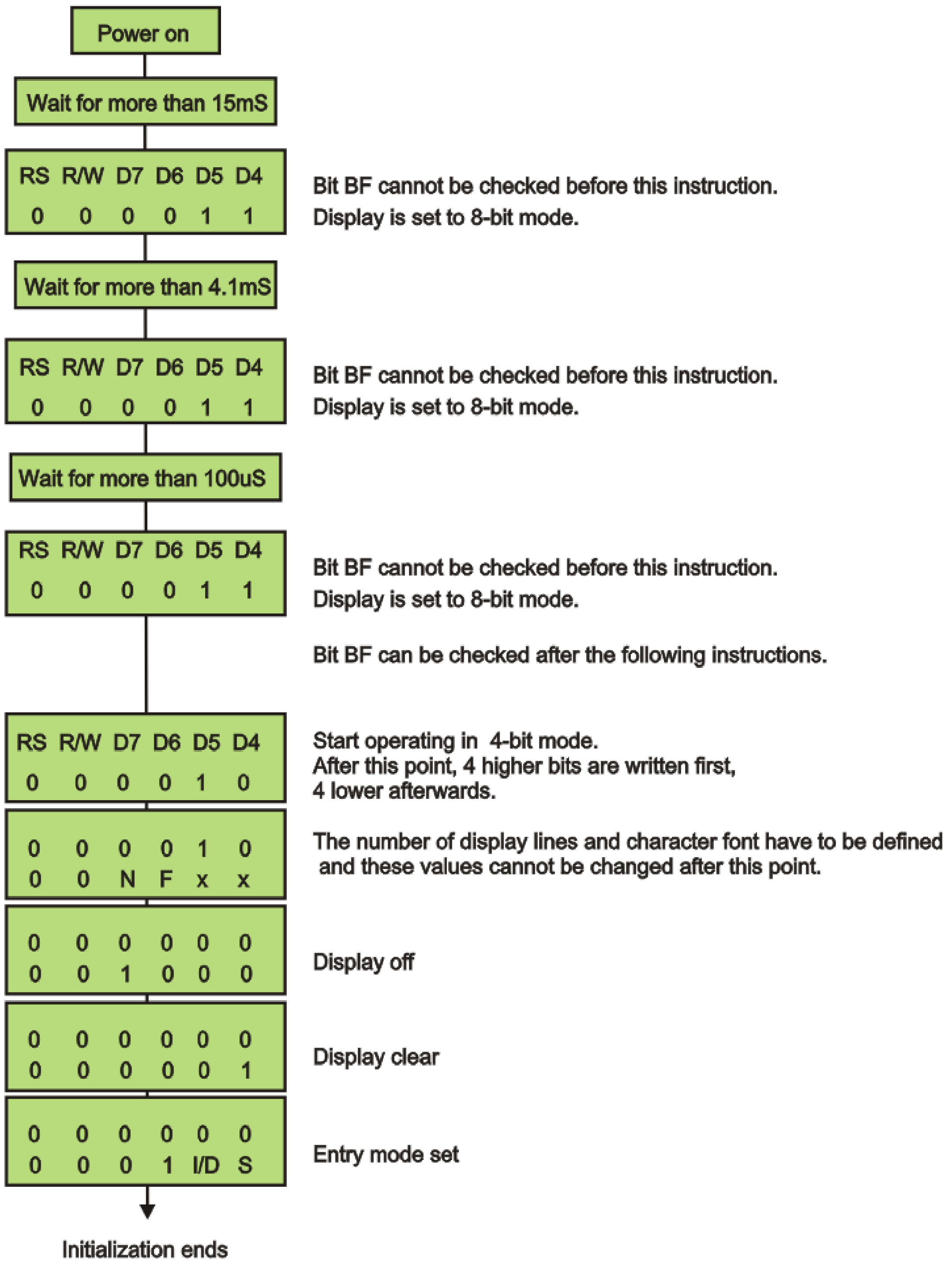
In our Project we have used 5X7 Dot Matrix. So in the Function Set I am writing the proper HEX Code for only 5X7 Matrix.

Mode of Interfacing of LCD:

Depending on how many lines are used for connection among the D0 to D7, to the microcontroller, there are 8-bit and 4-bit Mode of Interfacing of LCD. The appropriate mode is determined at the beginning of the process in a phase called "I**NITIALIZATION**". In our Project we have used the LCD in **4-bit Mode**. In case of 8-bit Mode the data are transferred through outputs D0-D7. But in case of 4-bit Mode, for the sake of saving valuable I/O pins of the microcontroller, there are only 4 higher bits (D4-D7) used for communication, while other may be left unconnected. Consequently, each data is sent to LCD in two steps: four higher bits are sent first (that normally would be sent through lines D4-D7), four lower bits are sent afterwards. With the help of initialization, LCD will correctly connect and interprete each data received. Besides, with regards to the fact that data are rarely read from LCD (data mainly are transferred from microcontroller to LCD) one more I/O pin may be saved by simple connecting R/W pin to the Ground. Such saving has its price. Even though message displaying will be normally performed, it will not be possible to read from busy flag since it is not possible to read from display.

Initialization of LCD:

Before displaying any comments or output in the LCD Display, at first we have to give some commands to the LCD ti initialize it. Only then we can use the LCD for our necessary purpose. If we want to configure the LCD for 4-bit Mode Communication between LCD and Rabbit 4000 then we have to write DL= 0. If we want to configure the LCD for 5X7 Dots then we have to write F=0, If we want to off the Display then we have to write D=0 and etc. In the below block diagram or Flow chart we can get how to initialize a LCD for 4-bit Mode of Communication.


Power on
Wait for more than 15mS
RS R/W D7 D6 D5 D4
0 0 0 0 1 1
Bit BF cannot be checked before this instruction.
Display is set to 8-bit mode.
Wait for more than 4.1mS
RS R/W D7 D6 D5 D4
0 0 0 0 1 1
Bit BF cannot be checked before this instruction.
Display is set to 8-bit mode.
Wait for more than 100uS
RS R/W D7 D6 D5 D4
0 0 0 0 1 1
Bit BF cannot be checked before this instruction.
Display is set to 8-bit mode.
Bit BF can be checked after the following instructions.
RS R/W D7 D6 D5 D4
0 0 0 0 1 0
Start operating in 4-bit mode.
After this point, 4 higher bits are written first,
4 lower afterwards.
0 0 0 0 1 0
0 0 N F x x
The number of display lines and character font have to be defined and these values cannot be changed after this point.
0 0 0 0 0 0
0 0 1 0 0 0
Display off
0 0 0 0 0 0
0 0 0 0 0 1
Display clear
0 0 0 0 0 0
0 0 0 1 I/D S
Entry mode set
Initialization ends

The Software details of the Initialization and various command and Display Data sending to LCD from Rabbit 4000 Micro processor will be discussed in the Software details of the Project.
In the next part I will discuss how LCD and LED is being configured with RCM4000 Prototype board through Parallel ports.

Interfacing of LCD and LED through Parallel Ports or Details about parallel Port A:

As discussed earlier that in Rabbit 4000 Microprocessor there are five Parallel Ports named as **Parallel Port A**, **Parallel Port B**, **Parallel Port C**, **Parallel Port D** and **Parallel Port E**. The below block diagram will describe how various parallel ports are I/O Ports are connected with Rabbit 4000 Microprocessor in the RCM4000 Prototype Board.

Each parallel Port has 8 pins. From Pin 0 to Pin 7. Parallel ports have some dual characteristics. But these ports are used as an I/O ports i.e. we can attach various external Hardwares in RCM4000 Prototype board for our Project purpose. To configure those Hardwares and to make the proper use of those Hardwares we have to program those External equipments through various parallel Ports by Dynamic C. There are various registers in those Parallel Ports which are helping us to configure those Parallel Ports and send or get some data to or from those External Hardwares. Now I will discuss in brief how to make an Handshaking environment between those External Hardwares and Rabbit 4000 Microprocessor.

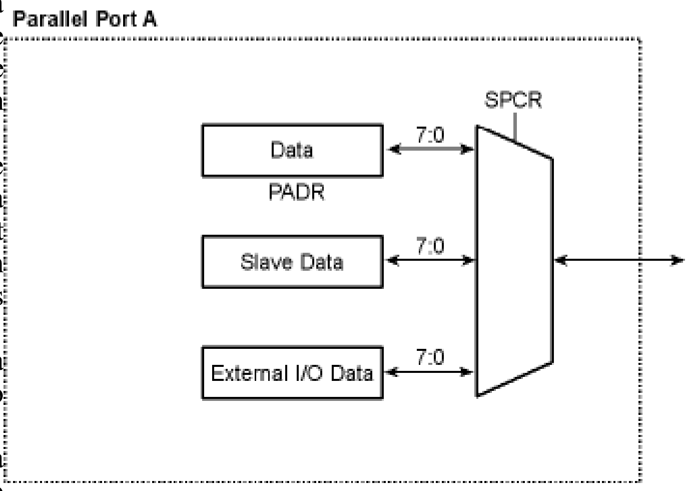
LED Interfacing:

In the RCM4000 Prototype Board there are two three LED-s. One is for Power Supply Purpose. And the other two are for our Program Tasting Purpose. Those two LED-s are treated as External Hardwares. And those two LED-s (DS2 and DS3) are connected with pin number 2 and Pin Number 3 of Parallel Port B. One can highly On or OFF or blink those LED-s by configuring the Parallel Port B as an I/O port by the register PBDR(Port B Data Register). Though in our main Project we have not used those LED-s, but in my Project period i.e. 2 and half months I have written many Programs and there I have used those LED-s for many times. If one can see those Programs carefully he or she will be highly realized that what is happening in time of LED Interfacing. We have just make the 2nd and 3rd Pin or bit of Parallel Port B as an Output Port. Then we have to send 1 for LED On and send 0 for LED OFF.

LCD Display Interfacing:

In the previous discussion I have discussed about LCD in details. Now I will discuss the connection between LCD Display screen and Rabbit 4000 Microprocessor. How the LCD have configured with Parallel Port A. Here to make the interfacing environment between LCD and Rabbit 4000 Microprocessor we have used Parallel Port A.
Parallel Port A is a byte-wide port that can be used as an input or an output port. Parallel Port A is also used as the data bus for the slave port and auxiliary I/O bus. The ***Slave Port Control Register (SPCR)*** is used to configure how Parallel Port A is used. Parallel Port A is an input at startup or reset. If the SMODE pins have selected the slave port bootstrap mode, Parallel Port A will be the slave port data bus until disabled by the processor. Parallel Port A can also be used as an external I/O data bus to isolate external I/O from the main data bus.
In the block diagram of Parallel Port A we can find that it can be used for the three purposes. Here the Register PADR (Port A Data Register) helps to read or write the data. But at first we have to choose that which operation mode we want in Parallel Port A.
Here for our Project purpose we have selected parallel Port A as a General_purpose 8-bit data input Port. This port uses Pins PA0 through PA7. These pins can be used as follows:

1. General-Purpose 8-bit data input (Write 0x080 to SPCR)
2. General-Purpose 8-bitr data output (Write 0x084 to SPCR)



3. Slave Port Data Bus (Write to SPCR)
4. Data Bus of the Auxiliary I/O bus (Write 0x08C to SPCR)

Once Parallel Port A is set up, data can be read or written by accessing PADR. Note that Parallel Port A is not available for general-purpose I/O while the slave port or the auxiliary I/O bus is selected. Selecting these options for Parallel Port A affects Parallel Port B because Parallel Port B is then used for address and control signals.

Analog to Digital Converter:

The RCM4000 has an on-board ASS7870 A/D Converter whose scaling and filtering are done via the motherboard on which the RCM4000 module is mounted. The A/D converter multiplexes converted signals from eight single-ended or four differential inputs to Serial Port B on the Rabbit 4000. The eight analog input pins, LN0–LN7, each have an input impedance of 6–7 MΩ, depending on whether they are used as single-ended or differential inputs. The input signal can range from -2 V to +2 V (differential mode) or from 0 V to +2 V (single-ended mode).

ADS7870 Brief Description:

The ADS7870 (US patents 6140872, 6060874) is a complete low-power data acquisition system on a single chip. It consists of a 4-channel differential/8-channel single-ended multiplexer, precision programmable gain amplifier, 12-bit successive approximation analog-to-digital (A/D) converter, and a precision voltage reference. The programmable-gain amplifier provides high input impedance, excellent gain accuracy, good common-mode rejection, and low noise. For many low-level signals, no external amplification or impedance buffering is needed between the signal source and the A/D input. The offset voltage of the PGA is auto zeroed. Gains of 1, 2, 4, 5, 8, 10, 16, and 20 V/V allow signals as low as 125 mV to produce full-scale digital outputs. The ADS7870 contains an internal reference, which is trimmed for high initial accuracy and stability vs temperature. Drift is typically 10 ppm/°C. An external reference can be used in situations where multiple ADS7870s share a common reference. The serial interface allows the use of SPI™, QSPI™, Microwire™, and 8051-family protocols, without glue logic.

The **key features** of ADS7870 are listed below:

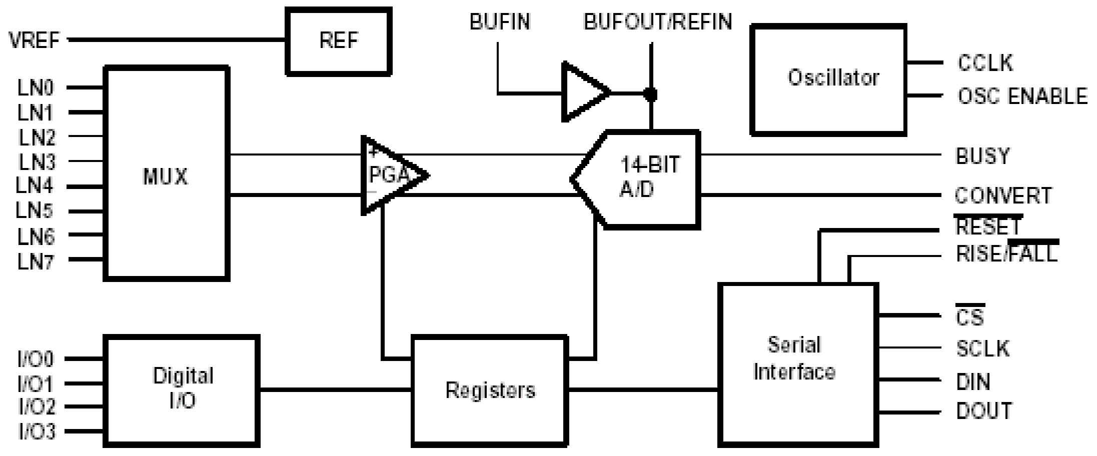
1. PGA Gains: 1, 2, 4, 5, 8, 10, 16, 20V/V

2. Programmable Input (Up to 4-channel Differential/Up to 8-channel Single-Ended or Some Combination)
3. 1.15-V, 2.048-V or 2.5-V Internal Reference
4. SPI/DSP Compatible Serial Interface (≤20 MHz)
5. Throughput Rate: 52 k samples/sec
6. Error/Overload Indicator
7. Programmable Output 2s Complement/Binary
8. 2.7-V to 5.5-V Single-Supply Operation
9. 4-Bit Digital I/O Via Serial Interface
10. Pin Compatible With ADS7871
11. SSOP-28 Package

There are various Applications of ADS7870 Analog to Digital Converter. Those are :

1. Portable/ Battery Powered Systems.
2. Low Power Instrumentation
3. Low Power Control Systems
4. Smart Sensor Applications.

The Block Diagram of ADS7870 is shown below:



We can see from the Block Diagram that there is 8 channels (Analog Input Channel) in ADS7870. And there is one PGA (Programmable Gain Amplifier) for the gains 1, 2, 4, 5, 8, 10, 16, and 20. So in our Software whatever gain we will fix up that gain will be multiplied with the input voltage getting from different channels. How to use those gains by Software Programming that I will discuss when I will discuss about the Software in details.
Now I will discuss about the various Pins of ADS7870 and their functions. There are 8 Analog Input Channels (LN0 to LN7) in the ADS7870. There are a RESET bar pin for Master resets Zero's all registers. There are also 4 Digital I/O pins. There pins like CONVERT, BUSY, OSC ENABLE, DIN, DOUT, CCLK, SCLK, CS bar etc. for various purpose. I am not going into the details of those purpose. If one are getting interested then he or she can consult the Datasheet of ADS7870 which is available in the official website of Texas Instrumentation.
In the next part I will discuss how we have calibrated the 8 Analog Input Channels and how we have used those gains in our Project purpose.

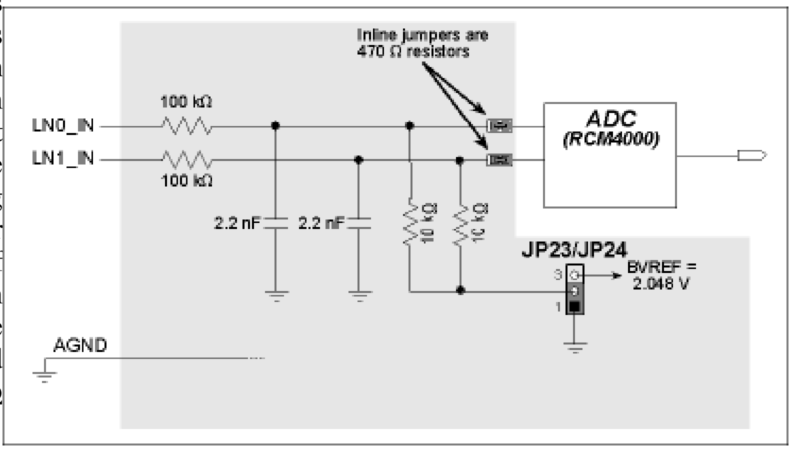
Calibration of all the channels of A/D Converter:

To get the best results from the A/D converter, it is necessary to calibrate each mode (single-ended or differential) for each of its gains. It is imperative that you calibrate each of the A/D converter inputs in the same manner as they are to be used in the application. For example, if you will be performing floating differential measurements or differential measurements using a common analog ground, then calibrate the A/D converter in the corresponding manner. In our Project at first we have calibrated all the 7 channels (Except Channel 7, which is used for Thermistor Purpose) by giving the proper commands through

Dynamic C.
At very beginning of our Project we did not calibrate all the channels. That time we were getting some crazy output. In A/D converter whatever the analog Input voltage we are applying to the channels, if the Gain is 0, then the exact same input voltage one should get. But if one does not calibrate then the output voltage will show some other voltages than Input voltage. In the Software Part of our Project I will discuss how to calibrate different channels for a fixed gain by Dynamic C.

A/D Converter Inputs and Various Gains:

In the figure we can find the resistors which are forming approximate 11:1 attenuator, and the capacitor filters noise pulses from the A/D converter input. The 470 Ω inline jumpers allow other configurations and provide digital isolation when you are not using an A/D converter (Parallel Port D is available). These jumpers optimize using RabbitCore modules with or without A/D converters—if you are designing your own circuit, the best performance for the A/D converter would be realized with 0 Ω resistors.



The A/D converter chip can make either single-ended or differential measurements depending on the value of the **opmode** parameter in the software function call. One thing is very Important
that Analog Input Channel 7 does not have the 10 KW resistor installed, and so no resistor attenuator is available, limiting its maximum input voltage to 2 Volt. This input is intended to be used for a Thermistor Purpose, which I am going to discuss in the next topic.
The A/D converter chip can only accept positive voltages, Both differential inputs must be referenced to analog ground, and both inputs must be positive with respect to analog ground.

Now I will discuss what is the main purpose of various Gains. But before that I will tell the Maximum input Voltage range for all gains.

Gain Code	Approx Gain	Actual Gain	Max Input Voltage Range
G0	1	1	0 to +22.528
G1	2	1.8	0 to +11.264
G2	4	3.6	0 to +5.632
G3	5	4.5	0 to +4.506
G4	8	7.2	0 to +2.816
G5	10	9	0 to +2.253
G6	16	14.4	0 to +1.408
G7	20	18	0 to +1.126

From the above chart one can highly get the and take care of choosing gain and Input Voltages to all the channels of the A/D Converter. In our Project we need to show the exact voltage at the output what the voltage we are giving as Input to the various channels of the A/D Converter. So in our main developed Software for my Project we have used Gain 1. But if we are giving voltage which very low to the channels of A/D Converter then we need various other gains. For example from some system we are getting 0.05 Volt and that voltage is being applied and the output voltage and Raw data is being used for some another purpose, then it will be a real problem to get data from this very small voltage. That time to increase the voltage we can use Higher Gain may be Gain 20. Then the voltage will be 0.05X20= 1 volt, which is much more higher than the applied voltage and it will give a perfect output voltage and Raw data for by using which we can do our other necessary jobs.

Thermistor Interfacing with Channel 7 of A/D Converter:

Temperature Controller and Monitor of SENTINEL System:

In our Project we have tasted the Temperature Controller and Monitor Sub-System of the SENTINEL System of GMRT.

Detail Software Description of our Project:

As described earlier We have used Dynamic C to develop the Testing Software for Rabbit Card which will be come in the place of MCM card in near future at GMRT. In this part I will discuss in detail about the Software and what is the real time problem we have faced in time of

programming of A/D Converter through Dynamic C. And also I will describe how we have solved those problems. Also here I will describe those Libraries and function calling which we have used in time of writing Code for our Project.
Dynamic C has been in use worldwide since 1989. It is specially designed for programming embedded systems, and features quick compile and interactive debugging.

Memory Organization and Compiling Program:

In time of compiling the Program which one has written at Dynamic C, one has a choice of doing his or her software development in the Flash Memory or in the Static RAM included on the RCM4000. The Flash Memory and SRAM options are selected with the **OPTIONS > PROGRAM OPTION > COMPILER** menu.
The **Advantage** of working in **RAM** is to save wear the flash memory, which is limited to about 100,000 write cycles. The **Disadvantage** is that the code and data might not both fit in RAM. An application can be compiled in RAM, but cannot run standalone from RAM after the programming cable is disconnected. All standalone applications can only run from flash memory.

Standard Debugging Features:

There are lots of Standard Debugging Features in Dynamic C which help the User to make understood what is happening in time of running of any programming. Those features are:

1. **Breakpoints**: Set breakpoints that can disable interrupts.
2. **Single-Stepping**: Step into or over functions at a source or machine code level µC/OS-II aware. One can enter into the Single-stepping mode after compiling the program successfully by pressing the key <F8>
3. **Code disassembly**: The disassembly window displays addresses, opcodes, mnemonics, and machine cycle times. Switch between debugging at machine-code level and source-code level by simply opening or closing the disassembly window.
4. **Watch Expression**: Watch expressions are compiled when defined, so complex expressions including function calls may be placed into watch expressions. Watch expressions can be updated with or without stopping program execution.
5. **STDIO Window**: printf outputs to this window and keyboard input on the host PC can be detected for debugging purposes. printf output may also be sent to a serial port or file.

Dynamic C Functions call:

The Dynamic C has many Inbuilt Library Functions in various different libraries. One can call and use those library Functions for the betterment of his project. Moreover, to interface between the Rabbit 4000 Microprocessor and varios external Hardwares one must need calling of those library functions. In our Project we have used those functions for various purposes. I am giving some brief ideas about those functions. The RCM4000 was designed to interface with other systems, and so there are no drivers written specifically for the I/O. In our Project we have made interface between the LED and LCD Screen and the Rabbit 4000 Microprocessor.