# ALS-SDA-ARM7-08

# USER MANUAL

**ADVANCED ELECTRONIC SYSTEMS**
**#143, 9TH MAIN ROAD, LAGGERE CROSS,**
**NEAR RAJAGOPAL NAGAR OLD POLICE STATION,**
**PEENYA INDUSTRIAL AREA, BANGALORE-560 058,**
**PHONE: 91-80-41625285/41539484.**
**E-mail: sales@alsindia.net**
**URL: www.alsindia.net**

# INDEX

# CHAPTER1: INTRODUCTION

The **ALS/EVBRD/ARM7T8** board is a study Board which includes LPC2148 ARM7TDMI-S micro-controller with USB 2.0 Full speed device, multiple UARTs, SPI, I2C & on-chip 512K Flash and SRAM up to 40kB, produced by NXP semiconductors.

The **ARM7TDMI-S** is a general-purpose **32-bit** microprocessor, which offers high performance and very low power consumption. The ARM processor is based on Reduced Instruction Set (**RISC**) architecture, And the instruction set and related decode mechanism are much simpler than those of micro programmed Complex Instruction Set Computers. This simplicity results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective processor Core.

Pipeline techniques are employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

The ARM7TDMI-S processor also employs a unique architectural strategy known as THUMB, which makes it ideally suited to high-volume applications with memory restrictions, or applications where code density is an issue. The key idea behind THUMB is that of a super-reduced instruction set. Essentially, the ARM7TDMI-S processor has two Instruction sets:

- The standard **32-bit** ARM instruction set.
- A **16-bit** THUMB instruction set.

The THUMB set's 16-bit instruction length allows it to approach twice the density of standard ARM code while retaining most of the Arm's performance advantage over a traditional 16-bit processor using 16-bit registers. This is possible because THUMB code operates on the same 32-bit register set as ARM code. THUMB code is able to provide up to 65% of the code size of ARM, and 160% of the performance of an equivalent ARM Processor connected to a 16-bit memory system.

## 1.1 MICROCONTROLLER FEATURES

Following are the features of 16 bit /32 bit LPC2148 Arm Microcontroller

- **PHILIPS LPC2148** is a 16-bit or 32-bit Microcontroller in a LQFP64-pin Package.
- 40 kB of on-chip static RAM and 512 kB of on-chip flash memory. 128-bit wide interface/accelerator enables high-speed 60 MHz operation.
- The LPC2148 provides 100000 erase/write cycles and 20 years of Data-retention.
- In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software. Single flash sector or full chip erase takes 400ms and Flash programming takes 1ms per 256-byte line.
- USB 2.0 Full speed compliant device controller with 2 kB of endpoint RAM. In addition, the LPC2148 provides 8 kB of on-chip RAM accessible to USB by DMA.
- Embedded ICE-RT and Embedded Trace Macro cell (**ETM**) interfaces offer real time debugging with on-chip Real Monitor software and high-speed real-time tracing of instruction execution.
- Two 10-bit ADCs provide a total of 14 analog inputs, with conversion times as low as 2.44µs per channel.
- Single 10-bit DAC provides variable analog output.
- Two 32-bit Timers/External event Counters (with four Capture and four Compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.

- Multiple serial interfaces including two UARTs (16C550 equivalent), two Fast I2C-bus(400 kbit/s), SPI and SSP with buffering and variable data length capabilities.
- Vectored interrupt controller (VIC) with configurable priorities and vector addresses. Up to 45 numbers of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 µs.
- On-chip integrated oscillator operates with an external crystal in range from 1 MHz to 30 MHz and with an external oscillator up to 50 MHz.
- Power saving modes include Idle and Power-down.
- Individual power enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt, USB, Brown-Out Detect (BOD) or Real-Time Clock (RTC).
- Single power supply chip with Power-On Reset (POR) and BOD circuits: CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O pads.

## 1.2 BOARD SPECIFICATIONS:

**1.2.1.** Study of ARM evaluation system

- **LPC2148** 16/32 BIT ARM7TDMI-S with 512K bytes Program Flash, 42K bytes RAM
- 12MHz Crystal allows easy communication setup
- One on board voltage regulator for generating 3.3V. Input to this will be from External +5V DC Power supply through a 9-pin DSUB connector
- One RS232 serial interface circuit(UART0) with 9 pin DSUB connector: this is used by the Boot loader program, to program **LPC2148** Flash memory without external Programmer and also for application
- Piggy Back module containing **LPC2148** controller. The board can be converted into a **CORTEX-M3(LPC1768)** evaluation board by replacing **LPC2148** piggy back with **LPC1768** piggy back
- Standard JTAG connector with ARM 2×10 pin layout for programming/debugging with ARM-JTAG
- Standard 26-pin FRC connectors to connect to **on-board interface** or some of **ALS standard External Interfaces**
- Reset push-button for resetting the controller

**1.2.2.** Interfacing ADC and DAC

- 8 Bit 8 channel ADC
- 1 channel Internal 10 bit ADC
- 8-bit DAC

**1.2.3.** Interfacing LED and PWM **-** One PWM channel is interfaced to LED with test point to observe the PWM output. Equivalent DC voltage is also available at test point

**1.2.4.** Interfacing real time clock and serial port

- I2C RTC
- Serial RS232 interface circuit using UART0 and UART1

**1.2.5.** Interfacing keyboard and LCD

- 4×4 Key-Matrix
- 16×2 alphanumeric LCD

**1.2.6.** Interfacing EPROM and interrupt
- SPI EEPROM
- Two keys connected to External Interrupt pins of controller, provision for LED
- indication

**1.2.7.** Mailbox: Mailbox implemented using **RTOS (RTX Kernel)** provided by evaluation version of KEIL4 'C' compiler for ARM and onboard hardware. A comprehensive set of RTOS software examples such as: Scheduling algorithms, Events, Semaphores and Mutex are also included

**1.2.8.** Interrupt performance characteristics of ARM and FPGA: An external FPGA module with XILINX SPARTAN6 is interfaced using on board 26pin FRC extension cable. Interrupt from FPGA to ARM controller can be studied along with other experiments

**1.2.9.** Flashing of LED's: Four general purpose flashing LED's

**1.2.10.** Interfacing stepper motor and temperature sensor
- Stepper motor interface with direction control
- Temperature sensor LM335 is interfaced through 8 bit 8 channel ADC

**1.2.11.** Implementing ZIGBEE protocol with ARM**-**An external ZIGBEE interface connects to the ARM evaluation board through a 9 pin D-SUB connector. Allows study of following features,
- Full duplex ZIGBEE wireless communication in AT mode and API mode
- Operating ZIGBEE digital inputs and outputs using both REMOTE ZIGBEE and LOCAL ZIGBEE commands
- ZIGBEE ADC implementation by using both REMOTE and LOCAL AT commands
- The ZIGBEE interface can also be connected to a PC for configurations

**1.2.12.** ADDED Features
- DC motor interface with direction control using relay
- Interface circuit for on board Buzzer
- Two-digit multiplexed 7-segment display interface

**1.2.13**. A number of software examples in 'C-language' to illustrate the functioning of the interfaces. The software examples are compiled using an evaluation version of KEIL4 'C' compiler for ARM

**1.2.14**. Compact elegant plastic enclosure
Serial RS232 cable is included
Optional USB debugger is also available
Operates off 5V DC.

# CHAPTER 2 : HARDWARE DETAILS

## SECTION 2.1:MAIN CONTROLLER BOARD(ALS-SDA-ARM7T8)

### 2.1.1 CONNECTORS DETAILS:

**CN1 CONNECTOR**: 28 pin 14 X 2 HEADER is connected to the controller.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | TDO-JTAG | 15 | GND |
| 2 | TDI-JTAG | 16 | RTCX1 |
| 3 | TMS-JTAG | 17 | RESET |
| 4 | TRST-JTAG | 18 | RTCX2 |
| 5 | TCK-JTAG | 19 | VBAT |
| 6 | P0.11 | 20 | P0.25 |
| 7 | P0.10 | 21 | NC |
| 8 | P0.9 | 22 | NC |
| 9 | P0.8 | 23 | NC |
| 10 | 3.3V | 24 | NC |
| 11 | GND | 25 | NC |
| 12 | 3.3V | 26 | P1.24(2148) |
| 13 | NC | 27 | NC |
| 14 | NC | 28 | NC |

**CN2 CONNECTOR**: 28 pin 14 X 2 HEADER is connected to the controller.

| PIN # | DESCRIPTION | PIN# | DESCRIPTION |
|---|---|---|---|
| 1 | NC | 15 | NC |
| 2 | NC | 16 | NC |
| 3 | NC | 17 | NC |
| 4 | P0.8 | 18 | GND |
| 5 | 3.3V | 19 | 3.3V |
| 6 | NC | 20 | NC |
| 7 | NC | 21 | NC |
| 8 | GND | 22 | NC |
| 9 | NC | 23 | NC |
| 10 | NC | 24 | NC |
| 11 | P1.20 | 25 | P0.22 |
| 12 | P1.21 | 26 | P0.23 |
| 13 | P1.22 | 27 | P0.16 |
| 14 | P1.23 | 28 | NC |

**CN3 CONNECTOR**: 28 pin 14 X 2 HEADER is connected to the controller.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | NC | 15 | P1.17 |
| 2 | NC | 16 | P0.7 |
| 3 | P1.25 | 17 | P0.6 |
| 4 | P0.3 | 18 | P0.5 |
| 5 | P0.14 | 19 | P0.4 |
| 6 | 3.3V | 20 | P0.3 |
| 7 | GND | 21 | P0.2 |
| 8 | P0.31 | 22 | P0.15 |
| 9 | P0.30 | 23 | 3.3V |
| 10 | P0.22 | 24 | GND |
| 11 | P0.28 | 25 | P0.14 |
| 12 | P1.19 | 26 | P0.13 |
| 13 | P1.18 | 27 | P0.12 |
| 14 | P1.16 | 28 | NC |

**CN4 CONNECTOR**: 28 pin 14 X 2 HEADER is connected to the controller.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | P0.21 | 15 | NC |
| 2 | P0.20 | 16 | NC |
| 3 | P0.19 | 17 | NC |
| 4 | P0.18 | 18 | NC |
| 5 | P0.17 | 19 | NC |
| 6 | P0.16 | 20 | NC |
| 7 | NC | 21 | 3.3V |
| 8 | GND | 22 | GND |
| 9 | 3.3V | 23 | TXD0 |
| 10 | NC | 24 | RXD0 |
| 11 | NC | 25 | RTCK-JTAG |
| 12 | NC | 26 | NC |
| 13 | NC | 27 | NC |
| 14 | NC | 28 | NC |

**CN6 CONNECTOR**: 20 pin FRC connected to the controller, Standard JTAG connector for programming/debugging with ARM-JTAG debugger. SHORT jumper JP1 for JTAG to work.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | +3.3V | 11 | RTCK |
| 2 | +3.3V | 12 | GND |
| 3 | TRST | 13 | TDO |
| 4 | GND | 14 | GND |
| 5 | TDI | 15 | RST |
| 6 | GND | 16 | GND |
| 7 | TMS | 17 | GND |
| 8 | GND | 18 | GND |
| 9 | TCK | 19 | GND |
| 10 | GND | 20 | GND |

**CN7 CONNECTOR**: 26 pin FRC connected to the controller, which is compatible with **ALS Standard External Interfaces**. SHORT jumper JP4 while using External Interfaces.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | P0.12 | 14 | P0.17 |
| 2 | P0.13 | 15 | P0.30 |
| 3 | P0.10 | 16 | P0.31 THROUGH JP2(1,2) OR P1.24 THROUGH JP2(2,3) |
| 4 | P0.11 | 17 | P0.28 |
| 5 | P0.8 | 18 | P0.29 |
| 6 | P0.9 | 19 | P1.18 |
| 7 | P0.22 | 20 | P1.19 |
| 8 | P0.23 | 21 | P1.16 |
| 9 | P0.20 | 22 | P1.17 |
| 10 | P0.21 | 23 | P0.14(Short JP13 to use this pin) |
| 11 | P0.18 | 24 | P0.15 |
| 12 | P0.19 | 25 | +5V THROUGH JP4 |
| 13 | P0.16 | 26 | GND |

**CN8 CONNECTOR**: 26 pin FRC connected to connected to on board interfaces. Connect to CN7 using the FRC cable provided.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | P0.12 | 14 | P0.17 |
| 2 | P0.13 | 15 | P0.30 |
| 3 | P0.10 | 16 | P0.31 THROUGH JP2(1,2) OR P1.24 THROUGH JP2(2,3) |
| 4 | P0.11 | 17 | P0.28 |
| 5 | P0.8 | 18 | P0.29 |
| 6 | P0.9 | 19 | P1.18 |
| 7 | P0.22 | 20 | P1.19 |
| 8 | P0.23 | 21 | P1.16 |
| 9 | P0.20 | 22 | P1.17 |
| 10 | P0.21 | 23 | P0.14 |
| 11 | P0.18 | 24 | P0.15 |
| 12 | P0.19 | 25 | +5V |
| 13 | P0.16 | 26 | GND |

### 2.1.2 DB 9 CONNECTOR DETAILS:

**DB1 CONNECTOR**: 9-Pin D-type Male Power connector.

| Pin Number | Description |
|---|---|
| 1,2,3,6,7,8 | No Connection |
| 4 | GND |
| 5 | GND |
| 9 | +5V |

**DB2 CONNECTOR**:**(UART0)** 9-Pin D-type Female connector connects to the COM port of host PC for In System Programming (ISP) application and transferring the data between controller device and host computer. Use a cross cable to connect to PC.

NOTE: DTR and RTS lines are required.

| Pin Number | Description |
|---|---|
| 1,4,7,9 | NC |
| 2 | CONTROLLER Rx0 |
| 3 | CONTROLLER Tx0 |
| 5 | GND |
| 6 | DTR |
| 8 | RTS |

**DB3 CONNECTOR**: **(UART1)** 9-Pin D-type Female connector is used for ZIGBEE interfacing.

| Pin Number | Description |
|---|---|
| 9 | +5V(Through jumper JP8) |
| 1,4,6,7,8 | NC |
| 2 | CONTROLLER Rx1 |
| 3 | CONTROLLER Tx1 |
| 5 | GND |

### 2.1.3 JUMPER DETAILS:

| JUMPERS | CONNECTION | DESCRIPTION |
|---|---|---|
| JP1 (1,2) | Closed | Enable JTAG Programming. |
| JP2 (1,2) | Closed | P0.31 to CN7-16. |
| JP2 (2,3) | Closed | P1.24 to CN7-16. |
| JP3 (1,2) | Closed | Ref Volt applied to DAC. |
| JP4(1,2) | Closed | 5V supply to CN7 connector. |
| JP5 (1,2) | Closed | Connects 3.3v to board. |
| JP6 (1,2) | Closed | External Interrupt (INT0) is given through SW2. |
| JP7 (1,2) | Closed | ISP signal to the controller. |
| JP8 (1,2) | Closed | Connects 5v to DB3 9 pin connector |
| JP9 (1,2) | Closed | Enable internal ADC circuit. |
| JP10 (1,2) | Closed | Enable PWM circuit. |
| JP11(1,2) | Open | DAC output wave form is Bi-polar |
| JP11(1,2) | Closed | DAC output wave form is Uni-polar |
| JP12(1,2) | Closed | External Interrupt (INT1) is given through SW3. |
| JP13(1,2) | Open | Open while resetting controller. |
| JP13(1,2) | Closed | While doing Buzzer, I2C1 RTC and ADC0809 experiment. |
| JP14(1,2) | Closed | Connects SPI ADC P0.17-SCK line |
| JP15(1,2) | Closed | Connects SPI ADC P0.18-MISO line |
| JP16(1,2) | Closed | Connects SPI ADC P0.19-MOSI line |
| JP17(1,2) | Closed | Connects SPI ADC P0.20-SSEL line |
| JP18(1,2) | Closed | Connects I2C P0.11-SCL1 |

| | | |
|---|---|---|
| JP19(1,2) | Closed | Connects I2C P0.14-SDA1 |
| JP20(1,2) | Closed | Short during Buzzer experiment |
| JP21(1,2) | Closed | Short during Flashing LED's experiment |
| JP22(1,2) | Closed | Short(1,2) to use UART1 with LPC2148 Piggy card. |
| JP23(1,2) | Closed | Short(1,2) to use UART1 with LPC2148 Piggy card. |

**Note:** JP2 (1, 2) connects P0.31 to CN7-16. Note that we can use P0.31 as General Purpose Output only. To solve this problem use JP2(2,3) that connects P1.24 to CN7-16 which can be used as input when required.

### 2.1.4 TEST POINT DETAILS:

| | |
|---|---|
| TP1 , TP7 | +5V |
| TP2 | +3.3V |
| TP3 | PWM DC output |
| TP4 | RTC output |
| TP5 , TP6 | Ground |
| TP8 | PWM output |
| TP9 | ADC0809 channel 0 input |

### 2.1.5 POT DETAILS:

| | |
|---|---|
| POT1 | 10K ANVI POT for testing **INTERNAL ADC** of CONTROLLER. |
| POT2 | 50K ANVI POT for **LCD** Contrast. |
| POT3 | 5K ANVI POT for **DAC0800** interface circuit (To vary amplitude). |
| POT4 | To vary ADC0809 channel 0 input |

### 2.1.6ICDETAILS

| | |
|---|---|
| U1 | **LM317** VOLTAGE REGULATOR (3 PIN) |
| U2 | **25LC256** SPI EEPROM(8 PIN) |
| U3,U10,U14 | **74HCT244** OCTAL BUFFER (20 PIN) |
| U4 | **DAC0800** Digital to Analog converter (16 PIN) |
| U5 | **LM358** DUAL OP AMP (8 PIN) |
| U6 | **ICL7660S** SUPER VOLTAGE Converter (8 PIN) |
| U7 | **ULN2803** DRIVER 8 Darlington array(18 PIN) |
| U8 , U9 | **LT543** SEVEN SEGMENT DISPLAY (10 PIN) |

| | |
|---|---|
| U11 | **ADC0809** Analog to Digital converter(28 PIN) |
| U12 | **DS1307** I2C RTC (8 PIN) |
| U13 | **MAX232** RS232 LINE DRIVER(16 PIN) |
| U15 | **4093BK** NAND gate(14 PIN) |

### 2.1.7 SWITCH DETAILS:

**SW1** - Reset Switch.
**SW2** - External Interrupt connected to P0.16 of Controller.
**SW3** - External interrupt to P0.3 pin of Controller.
**SW4 to SW19** - 4×4 key matrix.
**SW20** – 4 way DIP switch for general purpose.
**SW21** – 2 way DIP switch for 7segment display.
**SW22** – 2-WAY dip switch to control RTS & DTR lines for ISP. During programming the switches are kept ON and JP7 is also shorted. In RUN mode the switches are OFF and JP7 is open.

### 2.1.8 POWERMATE DETAILS:

**PM1**:5 Pin Power mate (High Current Output Lines for Stepper Motor).

| Pin Number | Description |
|---|---|
| 1 | +5V |
| 2 | OUT1(A) |
| 3 | OUT2(B) |
| 4 | OUT3(C) |
| 5 | OUT4(D) |

### 2.1.9 RELIAMATE DETAILS:

**RM1:** 16 pin Single female Berg for LCD Interface.

| PIN # | DESCRIPTION | PIN # | DESCRIPTION |
|---|---|---|---|
| 1 | GND | 9,10 | NO CONNECTION |
| 2 | +5V | 11 | DATA LINE D4(P0.4) |
| 3 | 50K ANVI POT (POT2) | 12 | DATA LINE D5(P0.5) |
| 4 | RS(P0.2) | 13 | DATA LINE D6(P0.6) |
| 5 | GND | 14 | DATA LINE D7(P0.7) |
| 6 | CSE(P0.3) | 15 | +5V(Back light) |
| 7,8 | NO CONNECTION | 16 | GND |

**RM2:** Analog input for ADC0809 channel 1

| Pin Number | Description |
|---|---|
| 1 | Input signal |
| 2 | GND |

**RM3:** DAC 0800 Interface

| Pin Number | Description |
|---|---|
| 1 | Connected to DAC O/P |
| 2 | GND |

**RM4:** DC MOTOR

| Pin Number | Description |
|---|---|
| 1 | Connected to POLE1 of Rly1 |
| 2 | Connected to POLE2 of Rly1 |

## SECTION 2.2 : ZIGBEE INTERFACE BOARD(NIFC-67)

**CONNECTORS DETAILS:**
**POWERMATE :PM1**

| | |
|---|---|
| PM1/1 | +5V |
| PM1/2 | GND |
| PM1/3 | NC |
| PM1/4 | NC |

**RELIAMATE DETAILS:**
**RM1:** Relay output

**JUMPERS DETAILS:**

| | |
|---|---|
| JP1(2 PIN) | Short – provide power to NIFC-67 board from ALS ARM 2148 board.<br>Open – provide power to NIFC-67 board using external power supply(PM1) |

**DB1 CONNECTOR**: **(UART1)** 9-Pin D-type Female connector is used for ZIGBEE interfacing.

| Pin Number | Description |
|---|---|
| 2 | XB_Rx |
| 3 | XB_Tx |
| 5 | SIGNAL GND |
| 1,4,6,7,8 | NC |
| 9 | 9->+5V to NIFC-67 from ARM7T8 board through jumper JP1 |

## SECTION 2.3 : FPGA SPARTAN6 INTERFACE BOARD(NIFC-68)

### Connector details:

CN1: 30-pin male berg stick connector

| Pin name | CN1 Pin no | XC6SLX4 pin number | Description |
|---|---|---|---|
| GND | 1,2,29 & 30 | - | Ground pins |
| P1 | 3 | 1 | I/o pin |
| P2 | 4 | 2 | I/o pin |
| P5 | 5 | 5 | I/o pin |
| P6 | 6 | 6 | I/o pin |
| P7 | 7 | 7 | I/o pin |
| P8 | 8 | 8 | I/o pin |
| P9 | 9 | 9 | I/o pin |
| P10 | 10 | 10 | I/o pin |
| P11 | 11 | 11 | I/o pin |
| P12 | 12 | 12 | I/o pin |
| P14 | 13 | 14 | I/o pin |
| P15 | 14 | 15 | I/o pin |
| P16 | 15 | 16 | I/o pin |
| P17 | 16 | 17 | I/o pin |
| P21 | 17 | 21 | I/o pin |
| P22 | 18 | 22 | I/o pin |
| P23 | 19 | 23 | I/o pin |
| P24 | 20 | 24 | I/o pin |
| P26 | 21 | 26 | I/o pin |
| P27 | 22 | 27 | I/o pin |
| P29 | 23 | 29 | I/o pin |
| P30 | 24 | 30 | I/o pin |
| P32 | 25 | 32 | I/o pin |
| P33 | 26 | 33 | I/o pin |
| P34 | 27 | 34 | I/o pin |
| P35 | 28 | 35 | I/o pin |

CN2: 30-pin male bergstick connector

| Pin name | CN2 Pin no | XC6SLX4 pin number | Description |
|---|---|---|---|
| GND | 1,2,29 & 30 | - | Ground pins |
| P38 | 3 | 38 | I/o pin |
| P40 | 4 | 40 | I/o pin |
| P41 | 5 | 41 | I/o pin |
| P43 | 6 | 43 | I/o pin |
| P44 | 7 | 44 | I/o pin |
| P45 | 8 | 45 | I/o pin |
| P46 | 9 | 46 | I/o pin |
| P47 | 10 | 47 | I/o pin |
| P48 | 11 | 48 | I/o pin |
| P50 | 12 | 50 | I/o pin |
| 10MHZCLK | 13 | 51 | 10Mhz clock |
| CLK2 | 14 | 55 | 500khz and 100khz selection by JP2 |
| CLK1 | 15 | 56 | 5Mhz and 1Mhz selection by JP1 |
| P57 | 16 | 57 | I/o pin |
| P59 | 17 | 59 | I/o pin |
| P58 | 18 | 58 | I/o pin |
| P62 | 19 | 62 | I/o pin |
| P61 | 20 | 61 | I/o pin |
| P66 | 21 | 66 | I/o pin |
| P64 | 22 | 64 | I/o pin |
| P69 | 23 | 69 | I/o pin |
| P67 | 24 | 67 | I/o pin |
| NC | 25,26,27&28 | - | NC |

CN3: 30-pin male bergstick connector

| Pin name | CN3 Pin no | XC6SLX4 pin number | Description |
|---|---|---|---|
| GND | 1,2,28,29 & 30 | - | Ground pins |
| P74 | 3 | 74 | I/o pin |
| P75 | 4 | 75 | I/o pin |
| P78 | 5 | 78 | I/o pin |
| P79 | 6 | 79 | I/o pin |
| P80 | 7 | 80 | I/o pin |
| P81 | 8 | 81 | I/o pin |
| P82 | 9 | 82 | I/o pin |
| P83 | 10 | 83 | I/o pin |
| P84 | 11 | 84 | I/o pin |
| P85 | 12 | 85 | I/o pin |
| P87 | 13 | 87 | I/o pin |
| P88 | 14 | 88 | I/o pin |
| P92 | 15 | 92 | I/o pin |
| P93 | 16 | 93 | I/o pin |
| P94 | 17 | 94 | I/o pin |
| P95 | 18 | 95 | I/o pin |
| P97 | 19 | 97 | I/o pin |
| P98 | 20 | 98 | I/o pin |
| P99 | 21 | 99 | I/o pin |
| P100 | 22 | 100 | I/o pin |
| P101 | 23 | 101 | I/o pin |
| P102 | 24 | 102 | I/o pin |
| P104 | 25 | 104 | I/o pin |
| P105 | 26 | 105 | I/o pin |
| NC | 27 | - | NC |

CN4: 30-pin male bergstick connector

| Pin name | CN4 Pin no | XC6SLX4 pin number | Description |
|---|---|---|---|
| GND | 1,2,29 & 30 | - | Ground pins |
| NC | 3 | - | NC |
| P111 | 4 | 111 | I/o pin |
| P112 | 5 | 112 | I/o pin |
| P114 | 6 | 114 | I/o pin |
| P115 | 7 | 115 | I/o pin |
| P116 | 8 | 116 | I/o pin |
| P117 | 9 | 117 | I/o pin |
| P118 | 10 | 118 | I/o pin |
| P119 | 11 | 119 | I/o pin |
| P120 | 12 | 120 | I/o pin |
| P121 | 13 | 121 | I/o pin |
| P123 | 14 | 123 | I/o pin |
| P124 | 15 | 124 | I/o pin |
| P126 | 16 | 126 | I/o pin |
| P127 | 17 | 127 | I/o pin |
| P131 | 18 | 131 | I/o pin |
| P132 | 19 | 132 | I/o pin |
| P133 | 20 | 133 | I/o pin |
| P134 | 21 | 134 | I/o pin |
| P137 | 22 | 137 | I/o pin |
| P138 | 23 | 138 | I/o pin |
| P139 | 24 | 139 | I/o pin |
| P140 | 25 | 140 | I/o pin |
| P141 | 26 | 141 | I/o pin |
| P142 | 27 | 142 | I/o pin |
| P143 | 28 | 143 | I/o pin |

CN5: 10-pin FRC connector

| Pin name | Pin no | Description |
|---|---|---|
| +3.3V | 1 | Supply |
| GND | 2,3 & 9 | Ground |
| TCK | 4 | Test clock |
| TMS | 5 | Test mode select |
| TDO | 6 | Test data out |
| TDI | 7 | Test data IN |
| NC | 8 & 10 | - |

CN6: 26-pin FRC connector

| Pin name | Pin no | Description |
|---|---|---|
| IN1 to IN24 | 1 to 24 | I/O pins |
| VCC | 25 | Through jumper VCC is given to 26-pin connector |
| GND | 26 | Ground pin to 26-pin connector |

PM1 : 4 way power mate connector. Power supply to the board

| Pin name | Pin no |
|---|---|
| +5V dc (VCC) | 1 |
| GND | 2 |
| NC | 3,4 |

## Jumper details:

JP1 : Selection of frequencies - short(1,2) for 5MHz and short(2,3) for 1MHz.
JP2 : Selection of frequencies - short(1,2) for 500kHz and short(2,3) for 100kHz.
JP3 : VCC is routed to 26-pin connector.

## Switch details:

SW1 : 8-way DIP switch to provide digital inputs.
SW2 : Press to configure the IC(xc6slx4). Led LD1(by default on) is off during configuration.

## LED details:

LD1 : Indicates completion of transferring bit stream stored in flash PROM into the XC6SLX4 FPGA.
LD2 to LD9 : 8 led's to indicate the status of DIP switch inputs.
LD10 to LD17 : 8 output led's.
LD18 : Indicates +5V.
LD19 : Indicates +3.3V.
LD20 : Indicates +2.5V.

## Test point details:

TP1 : +5V.
TP2 : +3.3V.
TP3 : +2.5V.
TP4 : +1.2V.
TP5 : GND.
INT : Interrupt test point – can be routed through flying lead to external controller board.

# CHAPTER 3: CABLE DETAILS

## 3.1 APPLYING POWER:

Use the following procedure to apply power. Connect a 9-pin DSUB Female Connector to a 9-pin DSUB Male connector **DB1** provided on the Board. The color code for the supply is shown in table below:

| **PIN NUMBERS** | **POWER** | **COLOUR CODE** |
|---|---|---|
| 9 | +5V | BLUE |
| 4,5 | GND | YELLOW / BLACK |

## 3.2 SERIAL COMMUNICATION:

The RS232 Cross cable connections required for establishing communication between Evaluation Board and a display terminal/host computer system is given below (ON BOARD it is DB2).

Open the Hyper Terminal & set the host computer system baud rate to **9600**, data length to **8 bit**, parity bit to **none** and stop bits to **1**.

| **ARM7 LPC2148 PROJECT BOARD** (**DB2**) **PIN NO.** (9 PIN MALE) | **COMPUTER** (COM PORT) **PIN NO.** (9-PIN FEMALE) |
|---|---|
| 3-TXD | 2-RXD |
| 2-RXD | 3-TXD |
| 6-DTR | 4-DTR |
| 8-RTS | 7-RTS |
| 5-SIGNAL GND | 5-SIGNAL GND |
| 1,4,7,9 | NC |

The ALS made RS232 cross both ends male cable connections required for establishing communication between ARM controller Board and NIFC-67(ZIGBEE board) - (ON BOARD it is DB3) and to provide supply to NIFC-67 board through ARM controller board.

| **ALS-SDA-ARM7T8 Board** (**DB3**) PIN NO. (9 PIN FEMALE) | **NIFC-67** COM PORT - **DB1** (9-PIN FEMALE) |
|---|---|
| 2-CNTR_RXD1 | 2-XB_Rx |
| 3-CNTR_TXD1 | 3-XB_Tx |
| 5-SIGNAL GND | 5-SIGNAL GND |
| 1,4,6,7,8 - NC | 1,4,6,7,8 - NC |
| 9-> +5V through jumper JP8 | 9->+5V to NIFC-67 from ARM7T8 board through jumper JP1 |

# CHAPTER4 INSTALLATION

## 4.1 KEIL UVISION4 ID INSTALLATION:

Installation of keiluVision4 as follows.

- Go to **EXE** folder and then **uvision4.2** in the CD and run **Keil4 Arm.exe** file.
- **Next**
- Click on the option "I agree to all the terms of..." and then give **Next**
- **Next**
- Give name and the mail id (it might be any mail id) and then **Next**
- Click **Finish** to complete the installation.

## 4.2 PROJECT CREATION IN KEILUV4 IDE:

Create a project folder before creating NEW project.

- Open **Keil uVision4 IDE** software by double clicking on "Keil Uvision4" icon.
- Go to **"Project"** then to **"New uVision Project"** and save it with a name in the respective project folder, already you created.
- Select the device as **"NXP"** In that **"LPC2148"** then press **OK** and then press "**YES"**
- button to add "**startup.s"** file.
- In **startup** file go to **Configuration Wizard**. In **Configuration Wizard** window uncheck **PLL Setup** and check **VPBDIV Setup.**
- Go to "**File**" In that **"New"** to open an editor window. Create your source file and use the header file **"lpc21xx.h"** in the source file and save the file. **Colour syntax highlighting** will be enabled once the file is saved with a extension such as "**.C** ".
- Right click on **"Source Group 1"** and select the option **"Add Existing Files to Group**
- **Source Group 1"**add the ***.C source file**(s) to the group.

After adding the source file you can see the file in Project Window.

Then go to **"Project"** in that "**Translate**" to compile the File (s). Check out the **Build output** window.
Right click on **Target1** and select **options for Target Target1**.
Then go to option "Target" in that

- Xtal 12.0MHz
- Select "Use MicroLIB".
- Select IROM1 (starting 0x0 size 0x80000).
- Select IRAM1 (starting 0x40000000 size 0x8000).

Then go to option "Output"

- Select "Create Hex file"

Then go to option "Linker"

- Select "Use Memory Layout for Target Dialog".

To come out of this window press **OK.**
Go to "**Project**" in that "**Build Target**" for building all source files such as ".C","ASM", ".h", files, etc…This will create the *****.HEX file if no warnings & no Errors. Check out the **Build output** window.

### 4.3 FLASH MAGIC VERSION 6.1: Installation of Flash Magic as follows.

Go to **EXE** folder and then **Flash Magic 6.1** folder in the CD & run **FlashMagic.exe**

- **Next**
- Click on the option "I Accept the Agreement" and then give **Next**
- Then it asks the **Destination** location**,** Click **Next.**
- Further Select start menu folder**,** Click **Next.**
- Select **"Create a desktop icon"** then **Next**
- It prompts **"Ready to Install"** Click **INSTALL.**
- Click **Finish** to complete the installation.

### 4.4 ISP PROGRAMMING:

**FLASH MAGIC** software can be used to download the HEX files to the Flash memory of controller.

**HowtoDownload?**

Connect the serial cross cable from 9-pin DSUB Female connector (DB2) to the PC COM port. Push both SW22/1, 2 to ON position, JP7 should be shorted while using ISP programming. Connect DC +5V Power, through the 9-pin DSUB Male connector (DB1) applied from an external source. Switch ON the power. Open JP13 while downloading the software.

**SettingsinFLASHMAGIC:**
**Options -> Advanced options -> Hardware Config**

Enable these options only

- Use DTR and RTS to control RST and ISP pin
- Keep RTS asserted while COM port open Press OK then do the below settings

**Step1**. Communications:

- Device : LPC2148
- Com Port : COM1(Check and connect)
- Baud Rate : 9600
- Interface : None(ISP)
- Oscillator : 12MHz

**Step2**. ERASE:

- Select "Erase Blocks Used By Hex File".

**Step3**. Hex file:

- Browse and select the Hex file which you want to download.

**Step4**. Options

- Select "Verify after programming".

**Step5**. Start:

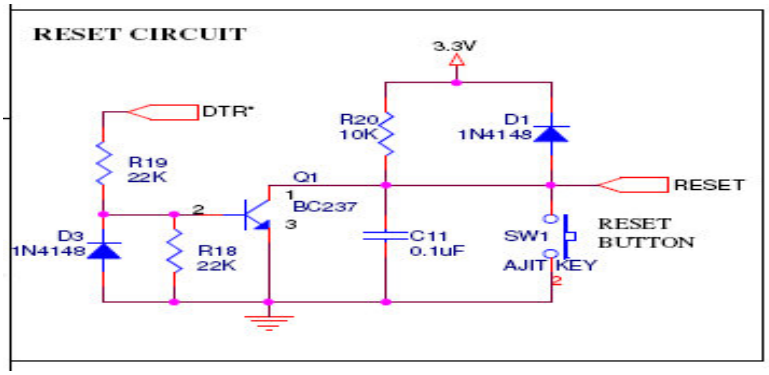
- Click Start to download the hex file to the controller.

After downloading the code the program starts executing in the hardware, then remove the ISP jumper JP7.

# CHAPTER5: ONBOARD INTERFACES

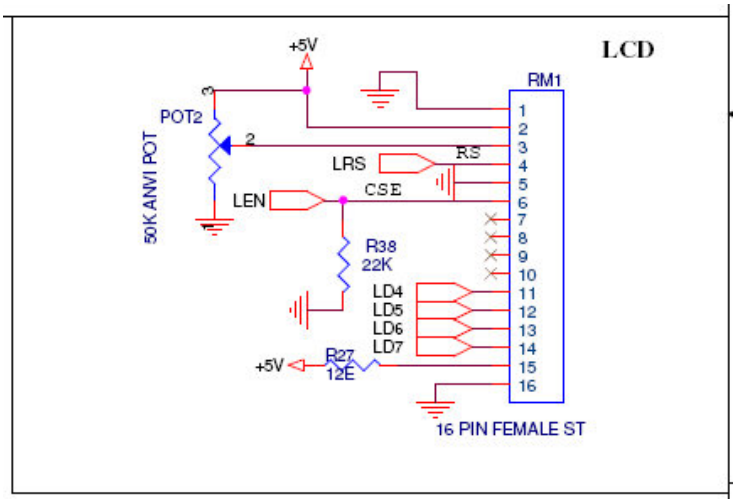
## 5.1 Push Button Switch (RESET)

SW1: This push button is connected to the RST (pin-57) of the Micro controller. It is used to reset the controller.




**5.2 16X2 LCD Interface :** A 16X2 Alphanumeric LCD Display with back light is provided along with the Evaluation Board. The LCD is interfaced using 4 – bit mode.

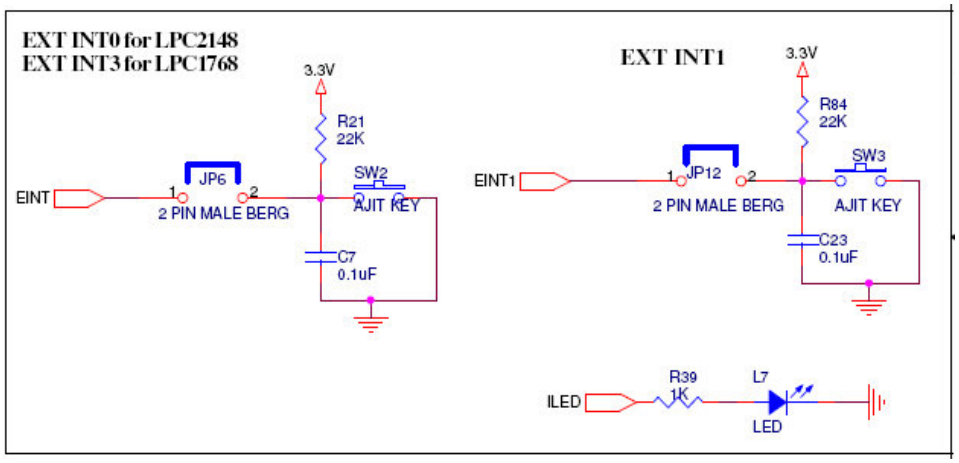
RS = 0 for sending Command to the LCD, controlled by port P0.2
RS = 1 for sending Data to the LCD, controlled by port P0.2
R/W = 1 for reading from the LCD
R/W = 0 for writing to the LCD, normally it is grounded
EN = 0 for disabling the LCD
EN = 1 for enabling the LCD, controlled by port P0.3
D4 to D7 (P0.4 to P0.7) – data lines



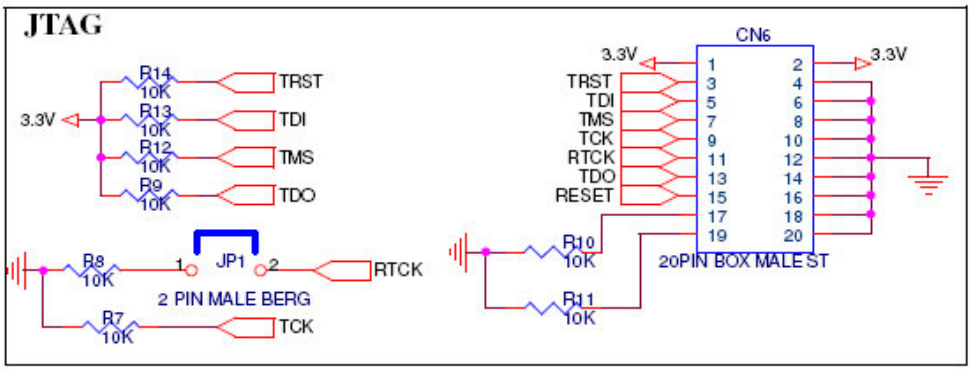
## 5.3 External Interrupt :

**External Interrupt 0:** By using P0.16 port line we are generating external interrupt (EINT0). Short JP6, when we press the switch SW2 the port line goes low & the external interrupt occurs at port line P0.16.

**External Interrupt 1:** By using P0.3 port line we are generating external interrupt (EINT1). Short JP12, when we press the switch SW3 the port line goes low & the external interrupt occurs at port line P0.3.




**5.4 JTAG:** The JTAG connector allows the software debugger to talk via a JTAG **(Joint Test Action Group)** port directly to the core. Instructions may be inserted and executed by the core thus allowing LPC2148 memory to be programmed with code and executed step by step by the host software. The debug communication channel allows the JTAG port to be used for sending and receiving data without affecting the normal program flow. Short the jumper JP1 (1, 2) in the target board to enable JTAG programming.



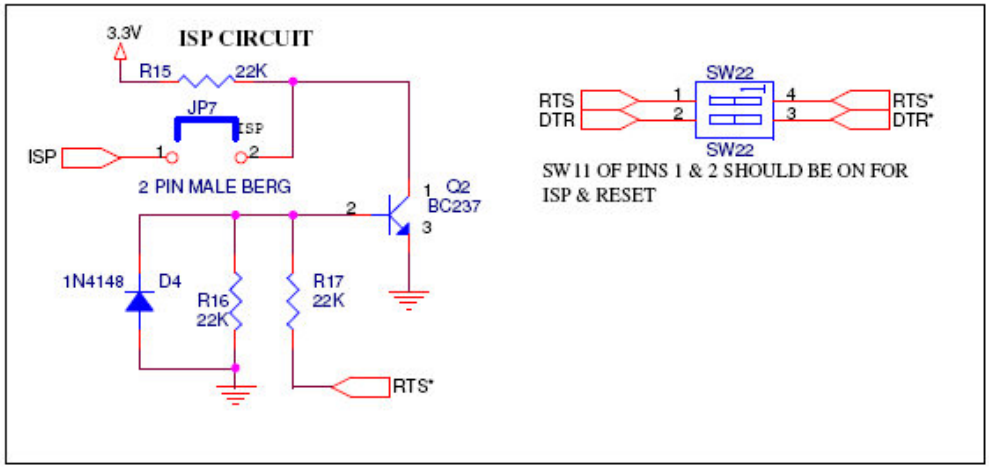
## 5.5 ISP Circuit:

A LOW level after reset at pin P0.14 is considered an external hardware request to start the ISP command handler. Assuming that power supply pins are on their nominal levels when the rising edge on RESET pin is generated, it may take up to 3 ms before P0.14 is sampled and the decision on whether to continue with user code or ISP handler is made.

During programming the controller using Flash magic software, jumper – JP7 needs to be shorted. This jumper connects the ISP line P0.14 to ground level during the Flash magic attempt to program the flash.

**SW22** – is to isolate the hand shaking signals from board signals (RTS, DTR) and connector. Keep this switch ON before programming the controller flash memory with the application code. Keep it open to Run the loaded program and reset. Especially if UART0 is using for any communication purpose, user must keep these switch open to execute UART0 related code.

RTS – Controls the ISP line of the controller P0.14
DTR – Used to interface controller reset.

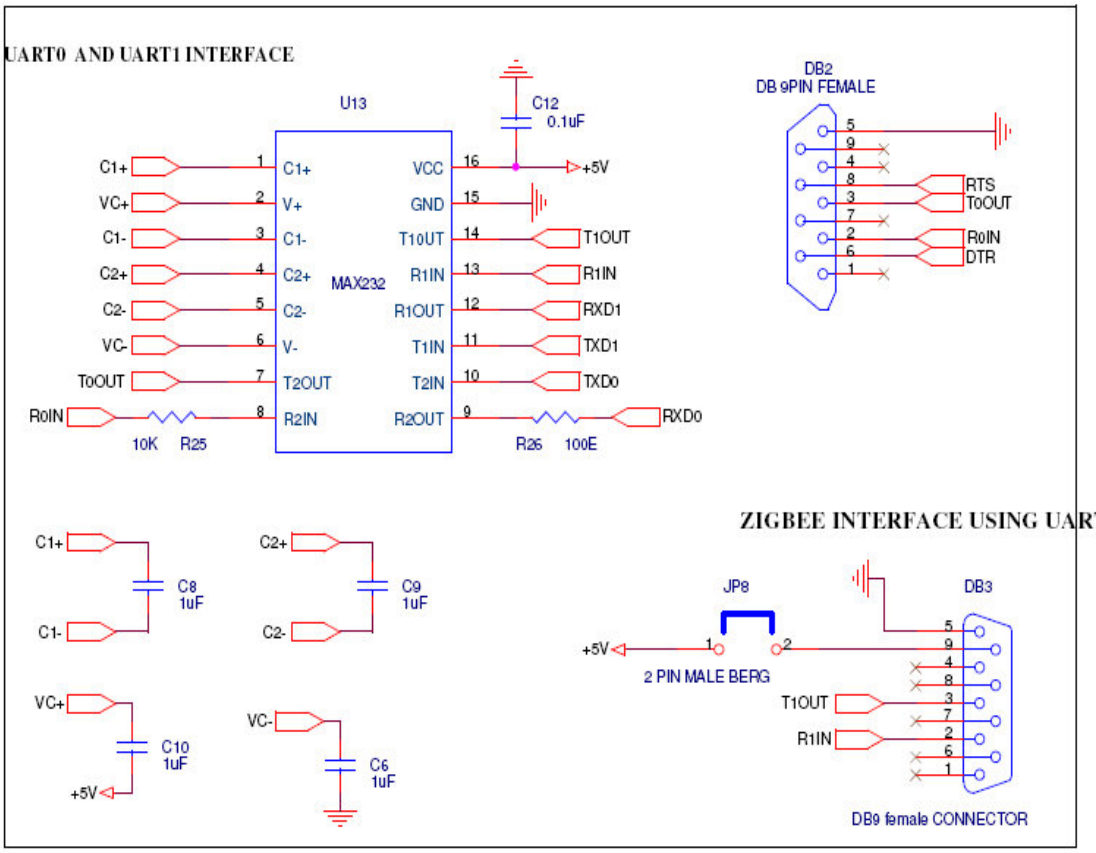


## 5.6 UART0 :

The board has an RS-232 serial communication port. The RS-232 transmits and receives signals that appear on the female 9-pin DB connectors (DB2). Use a standard RS-232 cross cable to connect the board to the computer's serial port. The controller U1 provides serial I/O data at TTL levels to the MAX232 (U13) device, which in turn converts the logic value to the appropriate RS-232 voltage level.

## 5.7 UART1 :

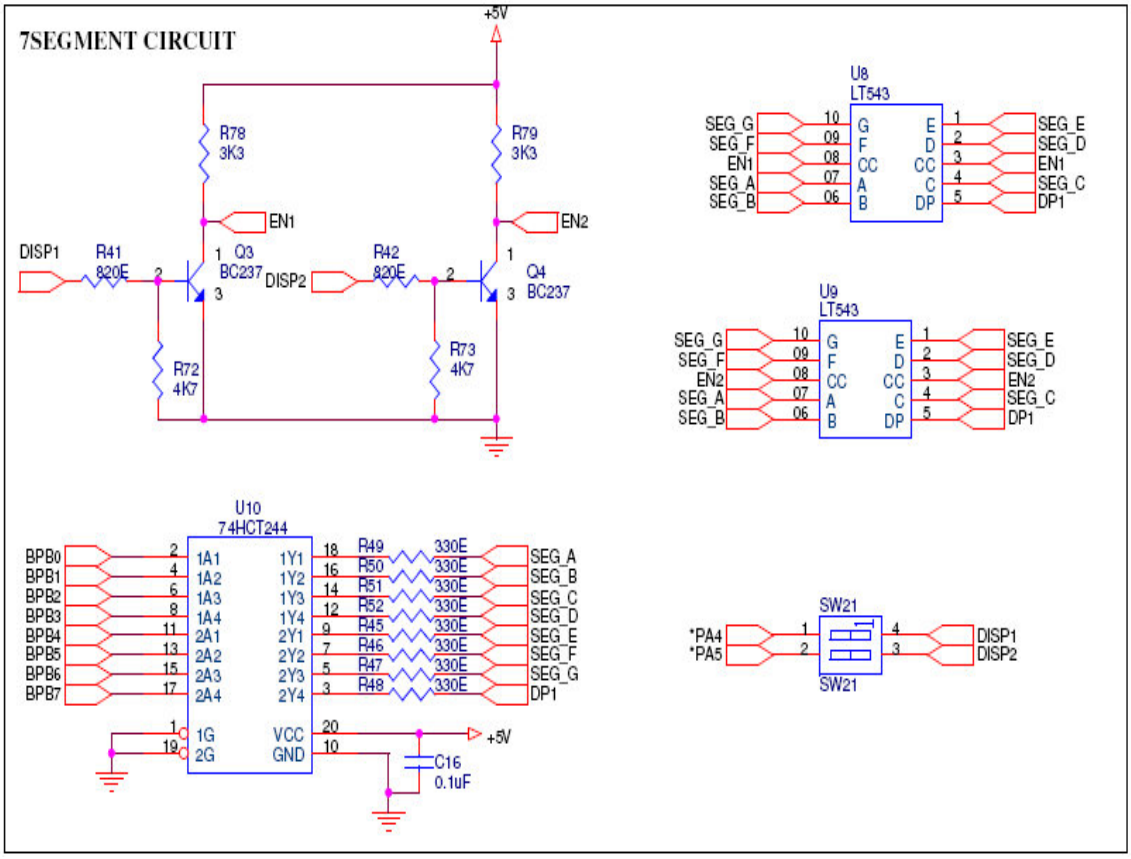
The board has an RS-232 serial communication port. The RS-232 transmits and receives signals that appear on the female 9-pin DB connectors (DB3). Use a standard RS-232 cross cable to connect the board to the computer's serial port. The controller U1 provides serial I/O data at TTL levels to the MAX232 (U13) device, which in turn converts the logic value to the appropriate RS-232 voltage level.



**5.8 Two Digit Multiplexed 7- Segment Displays:** There are two multiplexed 7-segment displays LT543 (**U8 and U9**) on the board. Each display has 8-inputs SEG_A (Pin-7), SEG_B (Pin-6), SEG_C (Pin-4), SEG_D (Pin-2), SEG_E (Pin-1), SEG_F (Pin-9), SEG_G (Pin-10) and DP (Pin-5). The remaining pins pin-3 & 8 is Common Cathode. The port lines P0.28 and P0.29 are used to select one of the 2 digits as shown in the table below. The port lines P0.16 to P0.23 are used as segment lines for the EIGHT digits through the 74HCT244 buffer (**U10**).

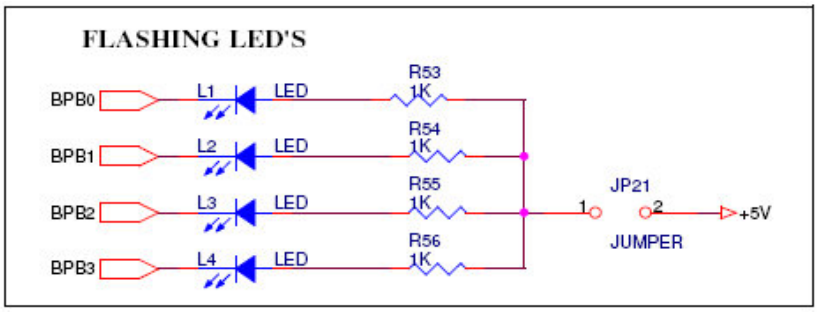
**Selection Of seven segment displays:**

| P0.28 | P0.29 | Display unit selected |
|---|---|---|
| 1 | 0 | U8 |
| 0 | 1 | U9 |




**5.9 Flashing LED's:**

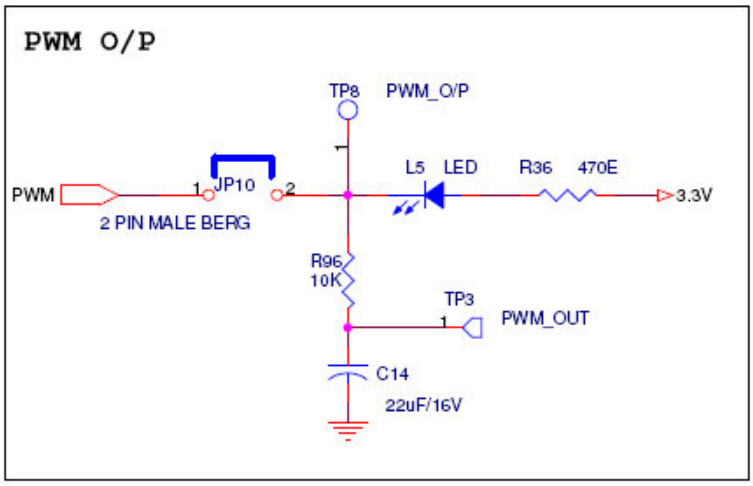
Light Emitting Diodes (LED's) are components most commonly used for displaying the port line status. There are 4 LEDs on the board; these lines are connected to the Port lines P0.16 (PB0) to P0.19 (PB3) through buffer.




## 5.10 PWM :

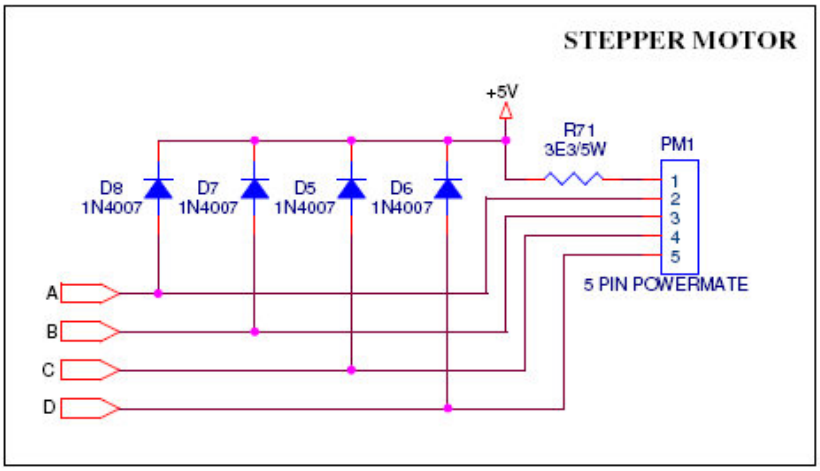
Using P0.8 (**PWM4**) port line we are testing PWM. Here match register (PWMMR4) is used to adjust the duty cycle (T-on to T-off ratio). Initially the duty cycle goes on increasing up to some specified value, when it reaches maximum value the duty cycle goes on decreasing and this process continues. Observe the output on CRO at TP8 and also LED (L11) will change status slowly.

Corresponding DC voltage can also be monitored at TP3



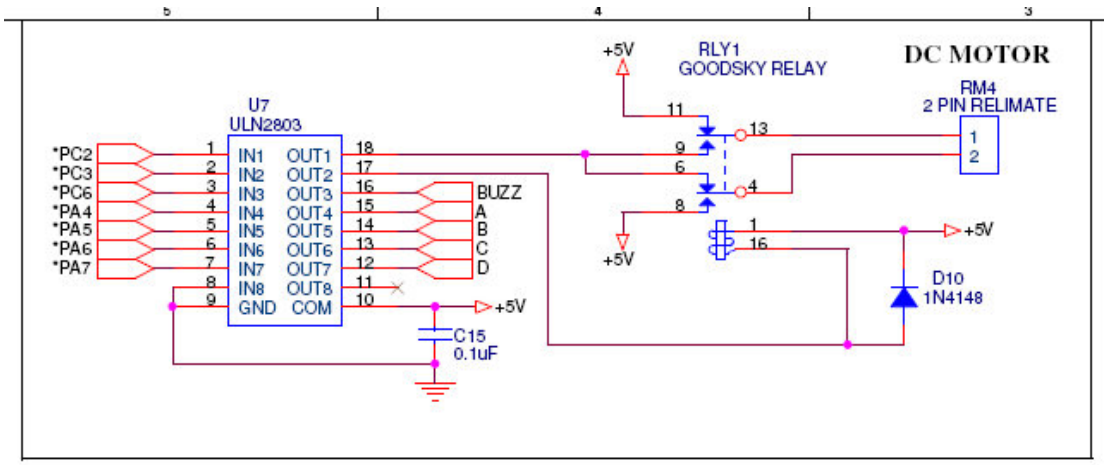

## 5.11 Stepper motor Interface:

The Stepper motor can be interfaced to the board by connecting it into the Power Mate PM1. It is interfaced through the high current driver ULN2803. These lines will have high current (max 300 mA) with low voltage level of 0.7V. The rotating direction of the stepper motor can be changed through software. Port lines used for Stepper motor are P0.28 – P0.31.

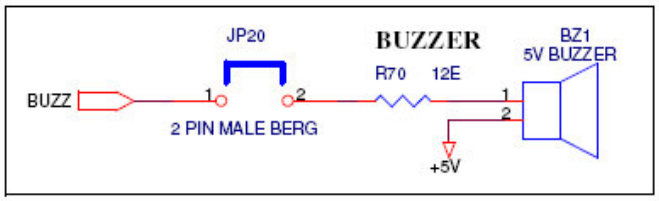



## 5.12 DC motor Interface:

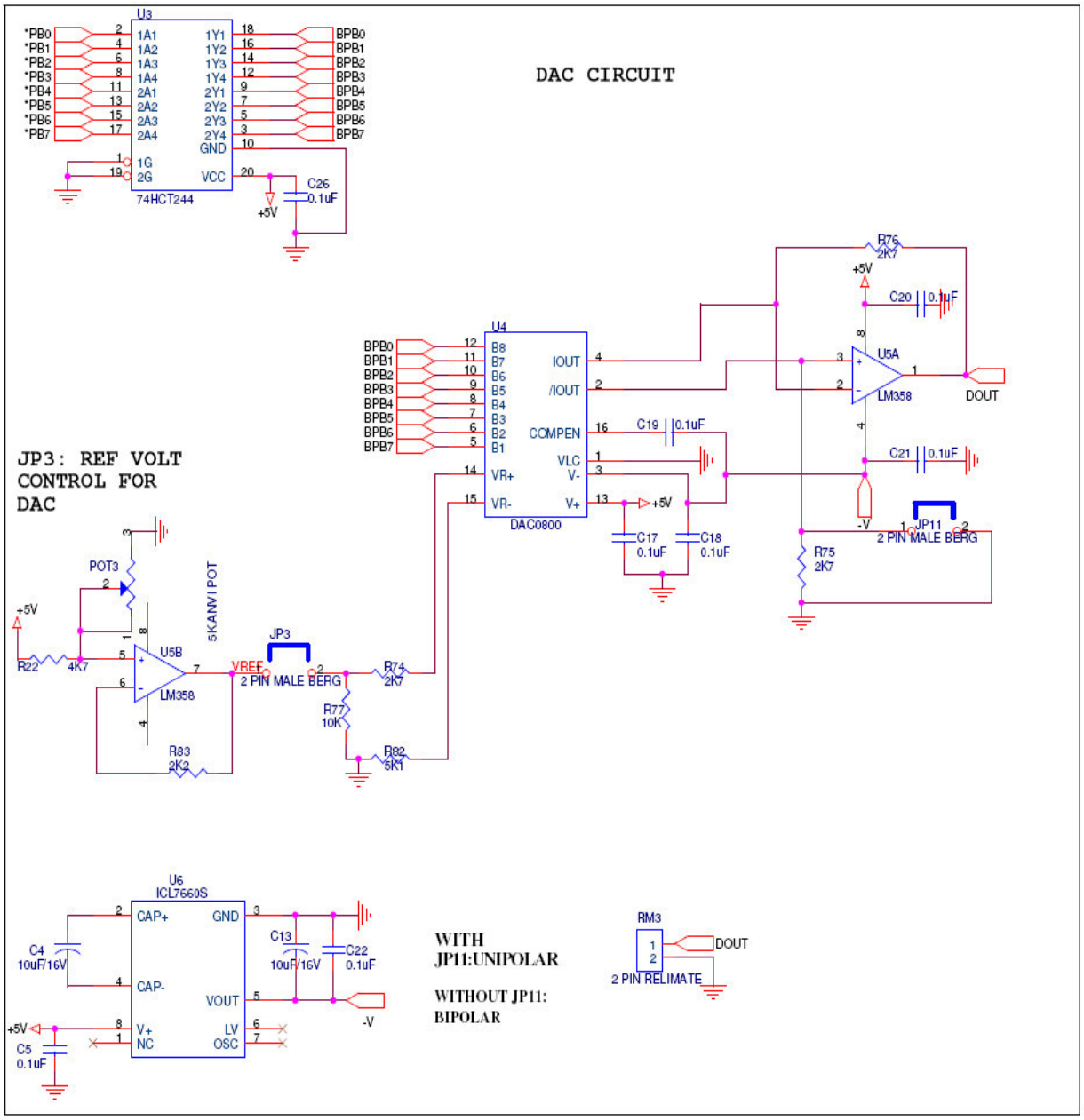
The **DC Motor** can be interfaced to the board by connecting it to the Reliamate RM4. It is interfaced through the high current driver ULN2803. These lines will have high current (max 300 mA) with low voltage level of 0.7V. The direction of the rotation can be changed through software using Relay **RLY1**. Port lines used for DC motor are P0.10 and P0.11.



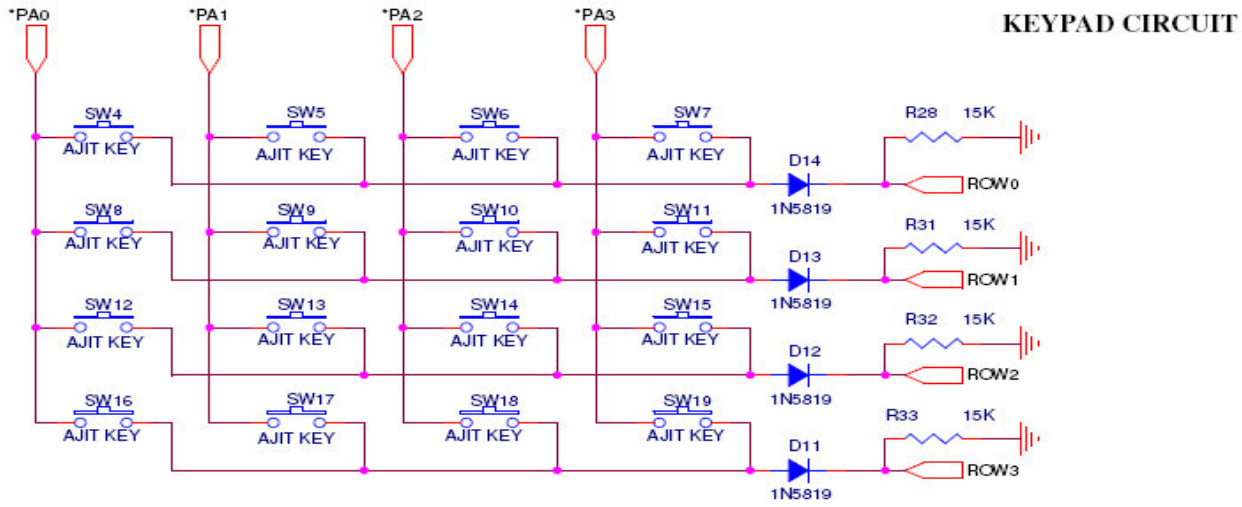

**5.13 Buzzer Interface:** The BUZZER is connected through port line P0.14.



**5.14 DAC0800 INTERFACE:** **DAC0800** is used to convert the digital data into analog signals. Digital data from specified port lines is given to DAC input. Amplitude of output waveform can be varied by varying **POT3** (5K Pot) that is by varying the reference voltage of DAC0800 when JP3 is closed. For **Bipolar** mode open jumper JP11. For **Unipolar** mode short jumper JP11. Port lines used for DAC are P0.16 – P0.23.

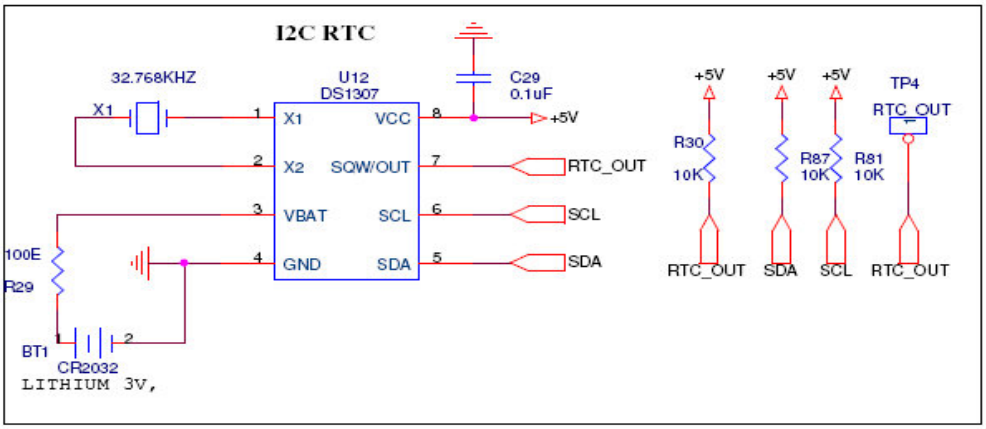


**5.15 Keypad (4 Rows X 4 Columns):** The switches **SW4** to **SW19** are organized as 4 rows X 4 columns matrix. One end of all the switches are connected to port lines P1.20 – P1.23, which is configured as rows. The other end of the matrix is connected to the port lines P1.16 – P1.19 which is configured as columns. The interface diagram for keypad is shown below.

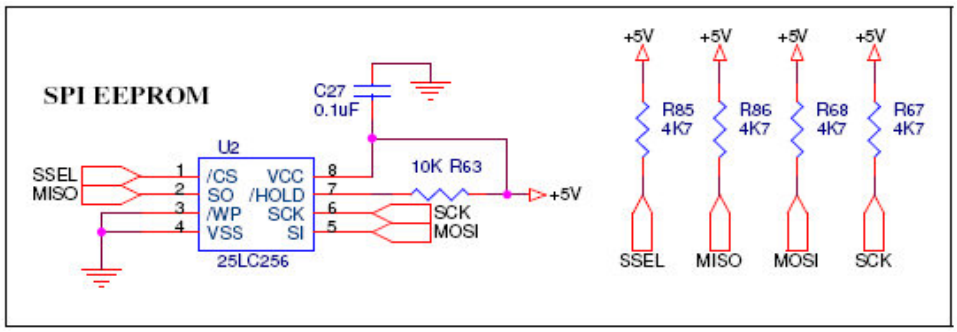



## 5.16 Real time clock(RTC) interfacing using I2C compatible RTC DS307

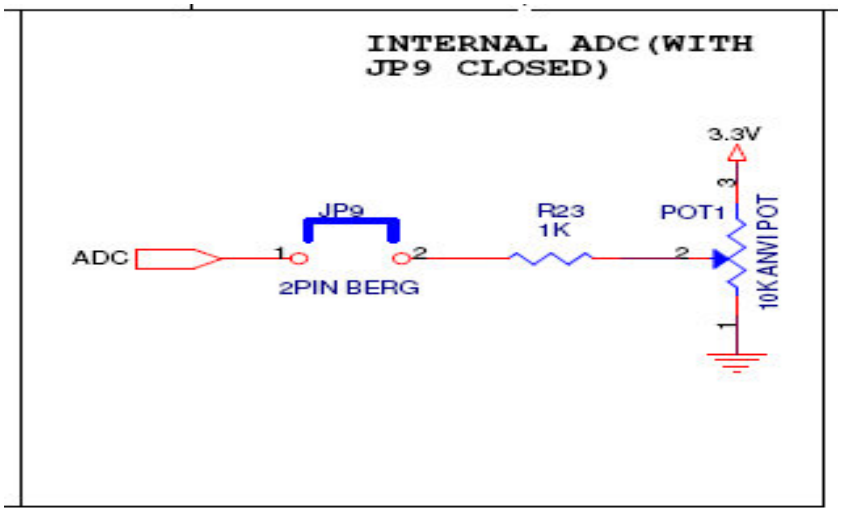
The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the Month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the battery supply.
I2C1 is used for this example. SDA1 – P0.14 and SCL1 – P0.11



**5.17 EEPROM Interfacing using SPI compatible 25LC256:** 25LC256 is 256 Kbit Serial Electrically Erasable PROMs. The memory is accessed via a simple Serial Peripheral Interface (SPI) compatible serial bus. The bus signals required are a clock input (SCK) plus separate data in (SI) and data out (SO) lines. Access to the device is controlled through a Chip Select (CS) input. Communication to the device can be paused via the hold pin (HOLD). While the device is paused, transitions on its inputs will be ignored, with the exception of Chip Select, allowing the host to service higher priority interrupts.
SPI1 is used for the communication. SCK1 – P0.17, MISO1 – P0.18, MOSI1 – P0.19,
SSEL – P0.20.

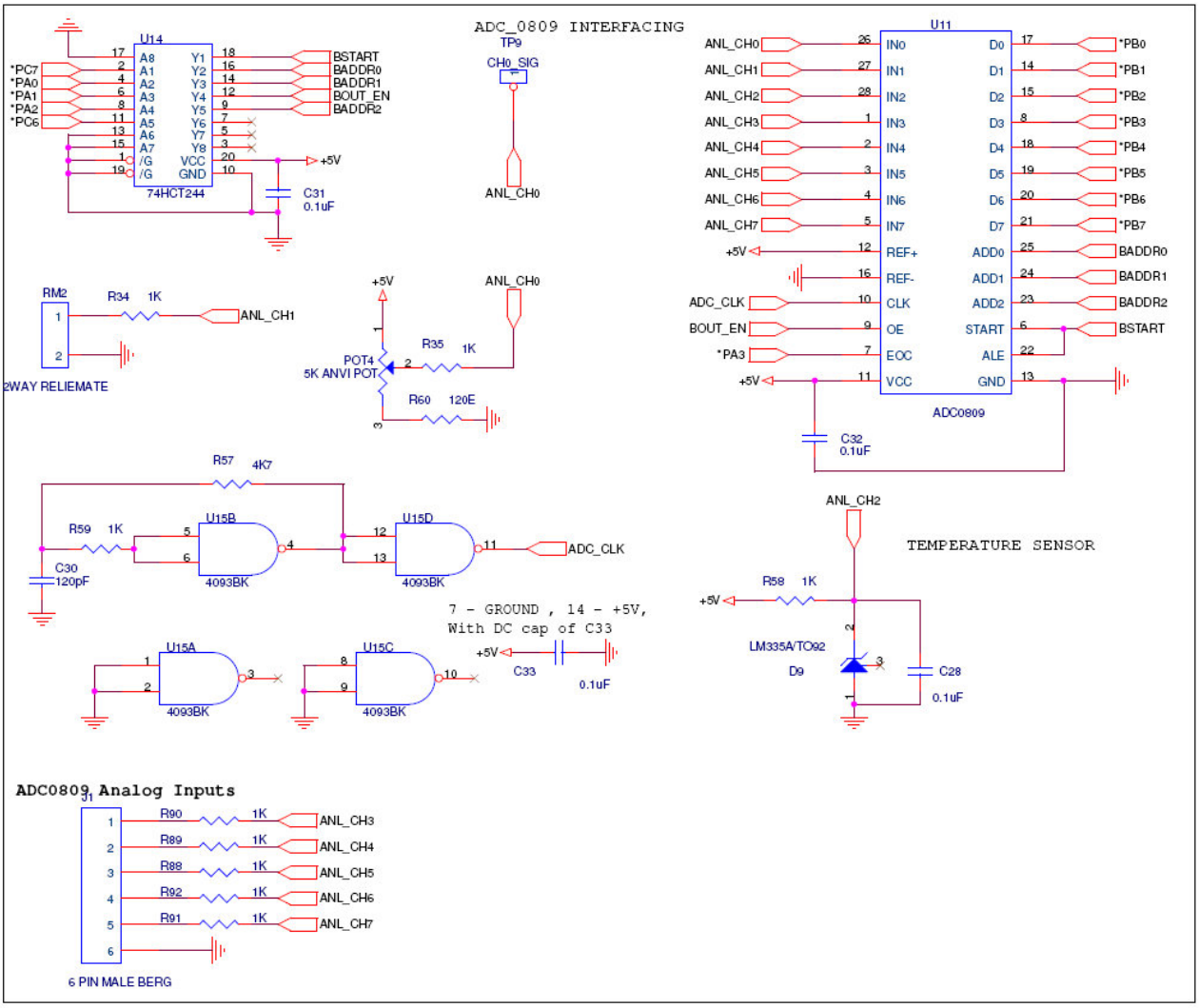



**5.18 Internal ADC interface:** On board there is one interface for internal ADC. AD0.4 (pin P0.25) of controller is used to convert the analog input voltage varied using POT1 to digital value. A 0.00 to 3.3V is the input voltage range. 000 to 3FF is the converted digital voltage range here. Short JP9(1, 2) to use this interface.



**5.19 ADC using ADC0809:** The ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

**Implementation is done for all 8 channels,**
**channel 0** – onboard potentiometer is provided.
**Channel1** – Reliamate is provided on board to apply input from external variable power supply.
**Channel 2** – Temperature sensor is connected to channel 2.
**channel 3 to channel 7** = input terminals brought to J1 male bergs to apply input from external variable power supply.



# CHAPTER6: DEMO PROGRAMS IN KEIL uVISION4 uVISION4 IDE:

For all the demo programs make sure that the corresponding settings have to be made:

- Both the pins of SW22 should be in ON position for ISP programming.
- Short JP7 for ISP programming.
- Short JP4 to connect +5v to Interface Board.
- Short JP13 while testing the on-board Buzzer interface.
- Inter connect **CN7** to **CN8** by means of 26 core FRC cable to use **On board interfaces**.
- Use only **CN7** connector for **External NIFC's**. Use 26 core 2 feet flat cable

**Note:**

- **Do not short any pins of JP14, JP15, JP16, JP17, JP18 and JP19 when you make connection from CN7 to CN8.**
- **While using SPI EEPROM and I2C RTC do not make connection from CN7 to CN8.**
- **For all experiments in which LCD is involved do not short jumper JP12.**
- **For Key-matrix, ADC0809 and Dip switch interfaces turn on the pin 1 of switch sw20 before doing reset and do not reset the controller when pin 1 of switch sw20 in OFF condition because On the LPC2148, the ETM pin functions are multiplexed with P1.25-16. To have them come up as port pins, do not connect a bias resistor to P1.20/TRACESYNC, and ensure that any external driver connected to P1.20/TRACESYNC is either driving high, or is in high-impedance state, during Reset.**

**Note:**

- **All ZIGBEE related information, configuration using X-CTU 6.1, operating modes, properties are provided in separate manual NIFC-67_REV00_Manual.**
- **All Spartan6 related information, installation, configuration, properties are provided in separate manual NIFC-68_REV00_Manual.**
- **The RTOS Mailbox related information, project creation in RTX and downloading procedure is provided in separate manual RTOS-ALS-SDA-ARM7-08.**

## 6.1 To test onboard ADC using ADC0809:

Implementation is done for all 8 channels.

Do the following settings for all channels.

- short Jumper JP13.
- Do not short Jumper JP3.
- Do not short Jumper JP20.
- Do not short Jumper JP21.
- Turn ON the pin 1 of switch sw20
- Turn OFF all the pins of switch SW21

**Channel 0:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH0
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH0

**Result :** Vary the analog input using ANVI potentiometer POT4 and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 1:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH1
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH1

**Result:** Connect variable power supply to reliamate RM2 and vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 2:**
Load the temp_sensor.hex file using programming tool FLASH MAGIC.
folder name: ADC0809_Temp_sen
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC0809_Temp_sen

**Result:** Observe the temperature data on ALS-SDA-ARM708 board LCD.

**Channel 3:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name: ADC_0809_CH3
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH3

**Result:** Connect variable power supply to male berg J1-pin1 with respect to ground J1-pin6,vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 4:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH4
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH4

Result: Connect variable power supply to male berg J1-pin2 with respect to ground J1-pin6,vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 5:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH5
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH5

Result: Connect variable power supply to male berg J1-pin3 with respect to ground J1-pin6,vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 6:**
Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH6

path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH6

Result: Connect variable power supply to male berg J1-pin4 with respect to ground J1-pin6,vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

**Channel 7:**

Load the ADC_0809.hex file using programming tool FLASH MAGIC.
folder name :ADC_0809_CH7
path: ALS-SDA-ARM7-08\SOFTWARE\Exp01_ADC_0809\ADC_0809_CH7

Result: Connect variable power supply to male berg J1-pin5 with respect to ground J1-pin6,vary the analog input voltage in the range 0 to 5V and observe the Digital data on ALS-SDA-ARM708 board LCD.

## 6.2 To test I2C Real time clock(RTC):

File name : E:\MAHADEVA_R\PROJECTS\ALS-SDA-ARM7-08\SOFTWARE\Exp02_I2C1_RTC
load the I2C_RTC.hex file using downloading tool Flash Magic.

**Settings:**

- **Do not** connect CN7 to CN8.
- Short jumper JP13
- Short jumper JP18(1,2), JP19(1,2).

Result: After loading the program, turn OFF the switch SW22(1,2) . RUN the program, Open hyper terminal to observe the RTC output.
When at power ON or at reset condition the following message will display on hyper terminal.
"Peripheral testing
1.Test RTC WRITE
2.Test RTC READ"

press '1' to update the contents.
Press '2' to read the RTC values as shown below:


## 6.3 ZIGBEE interfacing:

**Trouble shooting in NIFC-67 board.**

- When at power ON or at RESET condition If L2 & L3 are not blinking properly(for COORDINATOR) and L3 is not turned ON(for END DEVICE), check the Zigbee module for its proper insertion and check the 3.3V supply.
- When you setup a network, make sure that there is one coordinator board & a number of End devices.
- All the input, output, ADC lines of the coordinator and End devices must be configured using X-CTU.
- Connect the NIFC-67(DB1) to ALS-SDA-ARM7-08 REV00(DB3) using RS232 cross both ends male DB9 cable. and supply the DC power +5V, through 4 way powermate to PM1 of NIFC-67.
- If you are giving +5V to NIFC-67 through SDA-ARM7-08 REV00 through RS232 cross both ends male DB9 cable short jumper JP1 in NIFC-67 and short JP4 in SDA-ARM7-08 REV00 board.

**Note: All ZIGBEE related information, configuration using X-CTU 6.1, operating modes, properties are provided in separate manual NIFC-67_REV00_Manual.**

**Test Setup Requirements:**

- NIFC-67 REV00 board – 2/3 No's.
- ALS-SDA-ARM7-08 REV00 – 1/2 No's.
- 4 way powermate cable – 1 No.
- RS232 cross both ends male DB9 cable.

**Settings:**

- short jumper JP22(1,2) and JP23(1,2)

### 6.3.1 ZIGBEE data transmission in AT mode:

**Coordinator AT to end device AT communication:** Coordinator AT ZIGBEE module is connected to ALS-SDA-ARM7-08 board through DB9 male to male cable and end device AT ZIGBEE module is connected to PC using X-CTU software. Coordinator sends data using broadcast address and same data received by END DEVICE and display it on X-CTU.

**Setup requirement**: 2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module -configure the Zigbee module as COORDINATOR AT using firmware update procedure.

Now take another NIFC-67 module configure that Zigbee modules as END DEVICE AT using firmware update procedure and Set the sleep time ST = FFFE.
click on write icon to apply changes.

After Zigbee module configurations,
Connect the NIFC-67 in which COORDINATOR AT is connected to ALS-SDA-ARM7-08 REV00 using DB9 male to male cable.

Connect the NIFC-67 in which END DEVICE is connected to PC using RS-232 cross cable. open X-CTU and observe the output in console window.

Load the AT_transmit.hex file using programming tool FLASH MAGIC.
Folder name: AT_MODE_TRANSMIT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\AT_MODE_TRANSMIT

**Result:** Coordinator AT sends data using broadcast address continuously and same data received by END DEVICE AT and display it on X-CTU.

**End device AT to Coordinator AT communication:** End device AT ZIGBEE module is connected to ALS-SDA-ARM7-08 board through DB9 male to male cable and Coordinator AT is connected to PC using X-CTU software. End device AT sends data using Coordinator address and same data received by Coordinator AT and display it on X-CTU.

**Setup requirement**:2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module configure the Zigbee module as COORDINATOR AT using firmware update procedure.

Now take another NIFC-67 module configure that Zigbee modules as END DEVICE AT using firmware update procedure and Set the sleep time ST = FFFE.
click on write icon to apply changes.

After Zigbee module configurations,
Connect the NIFC-67 in which END DEVICE AT is connected to ALS-SDA-ARM7-08 REV00 using DB9 male to male cable.
Connect the NIFC-67 in which COORDINATOR is connected to PC using RS-232 cross cable. open X-CTU and observe the output in console window.

Load the AT_transmit.hex file using programming tool FLASH MAGIC.
Folder name: AT_MODE_TRANSMIT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\AT_MODE_TRANSMIT

**Result:** END DEVICE AT sends data using address of coordinator continuously and same data received by Coordinator AT and display it on X-CTU.

## ZIGBEE module operation in API mode:

**6.3.2 Relay control using LOCAL AT COMMAND(0X08):** Coordinator/ end device is connected to ALS-SDA-ARM7-08 board through DB9 male to male cable. Coordinator/ end device sends D4 command with high or low using 0x08 local AT command, and relay connected in end device/coordinator module will turn ON and turn OFF continuously.
Operation is same for both coordinator and end device.

**Setup requirement**:1 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of NIFC-67 Now configure the Zigbee module as COORDINATOR API/End device API using firmware update procedure.

After Zigbee module configurations.
Connect the NIFC-67 to ALS-SDA-ARM7-08 board through DB9 male to male cable.

Load the Relay_cntrl_Local_AT.HEX file using programming tool FLASH MAGIC.
folder name : Relay_cntrl_Local_AT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Relay_cntrl_Local_AT

Result: Relay will be turned ON and turned OFF continuously and following messages will be displayed on ALS-SDA-ARM7-08 REV00 LCD.
"Relay ON" (during relay ON).
"Relay OFF" (during relay OFF).

**Note:** Local AT (0x08) operation is same for both COORDINATOR and END DEVICE.

**6.3.3 Relay control using REMOTE AT COMMAND(0X17):**coordinator can set or clear the digital lines of end devices which are connected to network using remote AT command(0x17). Coordinator is connected to ALS-SDA-ARM7-08 board and two end devices are powered up, then coordinator send D4 command with high or low using broadcast address with remote AT and relays which are connected to end device modules will turn ON and turn OFF continuously.

**Setup requirement**:2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module, Now configure the Zigbee module as COORDINATOR API using firmware update procedure.

Connect RS-232 cross cable to DB1 of another NIFC-67 module, Now configure the Zigbee module as END DEVICE API using firmware update procedure.

Set the sleep time ST = FFFE.

After Zigbee module configurations.
Connect the NIFC-67 in which COORDINATOR zigbee is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

power up the another NIFC-67 in which END DEVICE Zigbee module is connected.

Load the Relay_cntrl.HEX file using programming tool FLASH MAGIC.
folder name: Relay_cntrl_Remote_AT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Relay_cntrl_Remote_AT

Result: Relay which is connected to END DEVICE Zigbee module NIFC-67 will be turned ON and turned OFF continuously and following messages will be displayed on ALS-SDA-ARM7-08 REV00 LCD.
"Relay ON" (during relay ON).
"Relay OFF" (during relay OFF).

**6.3.4 ADC using LOCAL AT command(0x08):** Coordinator/ end device is connected to ALS-SDA-ARM7-08 REV00 board. coordinator/ end device send IS command using 0x08 local AT command and collect the ADC values of local zigbee device which is connected to controller and display it on LCD.
Operation is same for both coordinator and end device.

**Setup requirement**:1 No NIFC-67 module and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of NIFC-67, Now configure the Zigbee module as Coordinator/ end device API using firmware update procedure.

Write the default setting by clicking on ICON shown below which is available in configuration window.


configure D0 as ADC[2]
configure D1 as ADC[2]
click on write icon to apply changes.

After Zigbee module configurations.
Connect the NIFC-67 to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

Load the adc_local.hex file using programming tool FLASH MAGIC.
folder name : ADC_local_AT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\ADC_local_AT

Result: vary the potentiometers POT1 and POT2 in NIFC-67 board and observe the ADC value on ALS-SDA-ARM7-08 REV00 board LCD.

**6.3.5 ADC using Remote AT command(0x17):** Coordinator is connected to ALS-SDA-ARM7-08 REV00 board and one or two end devices are powered up. Coordinator send NI command using broadcast address with remote AT and collect the addresses and node identifiers of end devices, then coordinator send IS command using particular addresses of end devices and collect the information regarding ADC's and digital I/O's. Here implementations done for ADC's so coordinator send IS command and collect ADC values and display it on LCD. Implementations are made to connect two end devices with one coordinator.

**Setup requirement**: 2/3 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module, Now configure the Zigbee module as COORDINATOR API using firmware update procedure.

Now take 1 or 2 no. NIFC-67 modules configure those Zigbee modules as END DEVICE API using firmware update procedure.
Write the default setting by clicking on "default configuration" ICON.
configure D0 as ADC[2].
configure D1 as ADC[2].
Set the sleep time ST = FFFE.
set node identifiers NI(should be 4 characters).
click on write icon to apply changes.

After Zigbee module configurations,
Connect the NIFC-67 in which COORDINATOR is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

power up the NIFC-67 modules in which END DEVICES are connected.

Load the ADC_Remote_AT.HEX file using programming tool FLASH MAGIC.
folder name: ADC_Remote_AT
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\ADC_Remote_AT

Result: vary the potentiometers POT1 and POT2 provided in NIFC-67 modules in which END DEVICE zigbee modules are connected. And observe the ADC values on ALS-SDA-ARM7-08 REV00 board LCD in following format.

NI = "node identifier of device1" ADC0 = "ADC0 value of device1"
ADC0 = "ADC1 value of device1"

delay() ;

NI = "node identifier of device2" ADC0 = "ADC0 value of device2"
ADC0 = "ADC1 value of device2"

**6.3.6 Digital I/O status using local AT command(0X08):** coordinator/ end device is connected to ALS-SDA-ARM7-08 board. coordinator/ end device send IS command using 0x08 local AT command. Then it will collect the status of digital I/O's and display it on LCD.

Operation is same for both coordinator and end device.

**Setup requirement**:1 No NIFC-67 module and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of NIFC-67, Now configure the Zigbee module as coordinator/end device API using firmware update procedure.

Write the default settings by clicking on "default configuration" ICON which is available in configuration window.
configure P1(DIO11) as Digital input[3].
configure P2(DIO12) as Digital input[3].

set node identifiers NI(should be 4 characters).
click on write icon to apply changes.

After Zigbee module configurations.
Connect the NIFC-67 to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

Load the digi_io_local.HEX file using programming tool FLASH MAGIC.
folder name : DIGI_IO_LOCAL
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\DIGI_IO_LOCAL

Result: press the switch SW2 & SW3 provided in NIFC-67 module and observe the digital line status on ALS-SDA-ARM7-08 REV00 LCD in following format

"DIO11 = HIGH/LOW"

"DIO12 = LOW/HIGH".

**6.3.7 Digital I/O status using Remote AT command(0X17):** Coordinator can collect the status of digital lines of end devices which are connected to the network using remote AT command(0x17). Coordinator is connected to ALS-SDA-ARM7-08 REV00 board and one end device is powered up. Coordinator send NI command using broadcast address with remote AT command and collect the address and node identifier of end device, then coordinator send IS command using particular address of end device and collect the status of digital I/O's of end device. display the digital line status on ALS-SDA-ARM7-08 REV00 board LCD.

**Setup requirement**: 2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67, Now configure the Zigbee module as COORDINATOR API using firmware update procedure.

Now take 1 no. NIFC-67 module configure this Zigbee modules as END DEVICE API using firmware update procedure.
Write the default settings by clicking on "default configuration" ICON.

Set the sleep time ST = FFFE.
configure P1(DIO11) as Digital input[3].
configure P2(DIO12) as Digital input[3].

set node identifiers NI(should be 4 characters).
click on write icon to apply changes.

After Zigbee module configurations,
Connect the NIFC-67 in which COORDINATOR is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

power up the NIFC-67 module in which END DEVICE is connected.

Load the digi_io_remote.hex file using programming tool FLASH MAGIC.
folder name: DIGI_IO_REMOTE
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\DIGI_IO_REMOTE

Result: Press the switch SW2 & SW3 provided in NIFC-67 module in which END DEVICE is connected and observe the digital line status on ALS-SDA-ARM7-08 REV00 LCD in following format.

"DIO11 = HIGH/LOW"

"DIO12 = HIGH /LOW".

**6.3.8 Data transfer using Transmit Request(0x10):** Using transmit request 0X10 command the information is transferred over a ZIGBEE link. Possible ways of communication are explained below.

**COORDINATOR TO END DEVICE COMMUNICATION:** Coordinator ZIGBEE module is connected to ALS-SDA-ARM7-08 REV00 board through male to male DB9 cable and END DEVICE is connected to PC using X-CTU software. Coordinator send data to END DEVICE using broadcast address and same data received by END DEVICE and display it on X-CTU.

**Setup requirement**: 2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module. configure the Zigbee module as COORDINATOR API using firmware update procedure.
Now take another NIFC-67 module configure that Zigbee modules as END DEVICE API using firmware update procedure and Set the sleep time ST = FFFE.
click on write icon to apply changes.

After Zigbee module configurations,
Connect the NIFC-67 in which COORDINATOR is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

Connect the NIFC-67 in which END DEVICE is connected to PC using RS-232 cross cable.
open X-CTU and observe the output in console window.

Load the Transmit_RQ.HEX file using programming tool FLASH MAGIC.
folder name: COORD_ENDDEVICE
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Transmit_Request\COORD_ENDDEVICE

Result: Coordinator sends data to END DEVICE using broadcast address continuously and same data received by END DEVICE and display it on X-CTU.

**END DEVICE TO COORDINATOR COMMUNICATION:** End device ZIGBEE module is connected to ALS-SDA-ARM7-08 REV00 board through male to male DB9 cable and coordinator is connected to PC using X-CTU software. End device sends data to coordinator using address of coordinator and same data received by coordinator and display it on X-CTU.

**Setup requirement**:2 No's NIFC-67 modules and 1 No. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module. configure the Zigbee module as END DEVICE API using firmware update procedure and Set the sleep time ST = FFFE.

Now take another NIFC-67 module configure that Zigbee module as COORDINATOR API using firmware update procedure.

After Zigbee module configurations,
Connect the NIFC-67 in which END DEVICE is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

Connect the NIFC-67 in which COORDINATOR is connected to PC using RS-232 cross cable. open X-CTU and observe the output in console window.

Load the Transmit_RQ.HEX file using programming tool FLASH MAGIC.
folder name : ENDDEVICE_COORD
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Transmit_Request\ENDDEVICE_COORD

Result: END DEVICE sends data to Coordinator using address of Coordinator continuously and same data received by Coordinator and displayed it on X-CTU.

**FULL DUPLEX COMMUNICATION:** Coordinator is connected to one ALS-SDA-ARM7-08 REV00 BOARD, End device is connected to another ALS-SDA-ARM7-08 REV00 BOARD. The data transferred by Coordinator is received by end device and display it through serial, the data transferred by end device is received by Coordinator and display it through serial.

**Setup requirement**:2 No's NIFC-67 modules and 2 Nos. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
Connect RS-232 cross cable to DB1 of one NIFC-67 module, configure the Zigbee module as COORDINATOR API using firmware update procedure.
Now take another NIFC-67 module configure that Zigbee modules as END DEVICE API using firmware update procedure and Set the sleep time ST = FFFE.

After Zigbee module configurations,
Connect the NIFC-67 in which END DEVICE is connected to one ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

Load the Transmit_RQ.HEX file using programming tool FLASH MAGIC.
folder name: ENDDEVICE_COORD
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Transmit_Request\ENDDEVICE_COORD

Connect RS-232 cross cable to DB2 of ALS-SDA-ARM7-08 REV00, establish the serial communication and open hyper terminal to observe the received data.

Connect the NIFC-67 in which COORDINATOR is connected to another ALS-SDA-ARM7-08 REV00 using male to male DB9 cable

Load the Transmit_RQ.HEX file using programming tool FLASH MAGIC.
folder name: COORD_ENDDEVICE
path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Transmit_Request\COORD_ENDDEVICE

Connect RS-232 cross cable to DB2 of ALS-SDA-ARM7-08 REV00, establish the serial communication and open hyper terminal to observe the received data.

Result: The data transferred by Coordinator is received by end device and display it through serial, the data transferred by end device is received by Coordinator and display it through serial.

**6.3.9 ARM Eval board to ARM Eval board communication using zigbee:** End device Zigbee module is connected to one ALS-SDA-ARM7-08 REV00 board and coordinator is connected to another ALS-SDA-ARM7-08 REV00 board. The internal ADC data is processed in controller board (connected to end device).This ADC data is transferred to Coordinator by end device using transmit request command(0X10).ADC data is received by coordinator and same is displayed on controller board LCD. This demonstrates board to board communication using zigbee protocol.

**Setup requirement**:2 No's NIFC-67 modules and 2 Nos. ALS-SDA-ARM7-08 REV00.

**Zigbee Configuration:**
short jumper JP9 in ALS-SDA-ARM7-08 REV00 board(for internal ADC operation).

Connect RS-232 cross cable to DB1 of one NIFC-67 module, configure the Zigbee module as COORDINATOR API using firmware update procedure.

Now take another NIFC-67 module configure that Zigbee module as END DEVICE API using firmware update procedure and Set the sleep time ST = FFFE.

After Zigbee module configurations,
Connect the NIFC-67 in which END DEVICE is connected to ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

path:ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Internal_ADC\Data_sender
Load the data_sender. HEX file using programming tool FLASH MAGIC.
folder name : Data_sender

Connect the NIFC-67 in which COORDINATOR is connected to another ALS-SDA-ARM7-08 REV00 using male to male DB9 cable.

path: ALS-SDA-ARM7-08\SOFTWARE\Exp03_ZIGBEE\Internal_ADC\data_receive
Load the data_receive. HEX file using programming tool FLASH MAGIC.
folder name : data_receive

Result: Vary the pot POT1 in ALS-SDA-ARM7-08 REV00 board in which END DEVICE Zigbee module is connected and observe the Digital output on another ALS-SDA-ARM7-08 REV00 LCD in which COORDINATOR is connected in following format.

DIGITL O/P = xxxx
ANALOG I/P = x.xx

## 6.4 To test stepper motor

Load the STPM.hex file using programming tool FLASH MAGIC.
folder name: Exp04_Stepper_motor
path: ALS-SDA-ARM7-08\SOFTWARE\Exp04_Stepper_motor

Connect the Female Powermate of the stepper motor to the male Powermate **PM1** present on the board.

Settings:

- Turn OFF both the pins of switch SW21
- Short jumper JP2(1,2)

**Result**: The stepper motor rotates in **Clockwise and Anti Clockwise** direction. This process is continuous in loop.

## 6.5A To test DC motor

Load the dcmotor.hex file using programming tool FLASH MAGIC.
folder name: Exp05A_DC_Motor
path: ALS-SDA-ARM7-08\SOFTWARE\Exp05A_DC_Motor

settings:

- Do not short JP18 either (1,2) or (2,3)

Connect the Female relimate of the DC motor to the male Relimate RM4 present on the board.

**Result:** The DC motor rotates **Clockwise** when the relay is in OFF status & **Anti Clockwise** when the relay is in ON status. This process is continuous in loop.

## 6.5B To test DC motor with Speed control

Load the DC_Motor.hex file using programming tool FLASH MAGIC.
folder name: EXP5B_DC_MTR_Speed_CNTRL
path:\ALS-SDA-ARM7-08\SOFTWARE\EXP5B_DC_MTR_Speed_CNTRL
settings:

- Do not short JP18 either (1,2) or (2,3)

Connect the Female relimate of the DC motor to the male Relimate RM4 present on the board.

**Result:** The DC motor rotates **Clockwise** & **Anti Clockwise** with speed variation. This process is continuous in loop.

## 6.6 TO test 7 Segment Display:

Load the SevenSeg.hex file using programming tool FLASH MAGIC.
folder name: Exp06_7Seg_disp
path: ALS-SDA-ARM7-08\SOFTWARE\Exp06_7Seg_disp

Settings:

- Turn ON all the pins of switch SW21.
- Do not short jumper JP3

**Result:** Press SW16 and observe display changing from **00 to FF.** For every press of SW16 the display will change ***Ex:*** **00 , 11 , 22 , 33** etc up to **FF** and then back to **00.** This process is continuous in loop.

## 6.7 To test 7 segment display as Decimal counter:

Load the deci_cntr.hex file using programming tool FLASH MAGIC.
folder name: Exp07_DECI_CNTR
path: ALS-SDA-ARM7-08\SOFTWARE\Exp07_DECI_CNTR

Settings:

- Turn ON all the pins of switch SW21.
- Do not short jumper JP3

**Result:** observe display changing for every 1 second from **00 to 99** and then back to **00.** This process is continuous in loop.

## 6.8 TO TEST DAC0800:

settings:

- short jumper JP3.
- Turn OFF all the pins of switch SW21.

### 6.8.1 Sine Wave:

Load the sinewave.hex file using programming tool FLASH MAGIC.
folder name: Exp08_DAC\sinewave
path: ALS-SDA-ARM7-08\SOFTWARE\Exp08_DAC\sinewave

**Result:** Press the reset switch to run the program Observe the sinwaveform at the Pin-1 of RM3 using Oscilloscope (CRO) with respect to GND pin-2 of RM3. When jumper JP11 is shorted it is Unipolar mode when jumper JP11 is opened it is Bipolar mode.

### 6.8.2 Square Wave:

Load the square.hex file using programming tool FLASH MAGIC.
folder name: Exp08_DAC\square
path: ALS-SDA-ARM7-08\SOFTWARE\Exp08_DAC\square

**Result:** Press the reset switch to run the program. Observe the square waveform at the Pin-1 of RM3 using Oscilloscope (CRO) with respect to GND pin-2 of RM3. When jumper JP11 is shorted it is Unipolar mode when jumper JP11 is opened it is Bipolar mode.

### 6.8.3 Triangle Wave:

Load the triangular.hex file using programming tool FLASH MAGIC.
folder name: Exp08_DAC\triangular
path: ALS-SDA-ARM7-08\SOFTWARE\Exp08_DAC\triangular

**Result:** Press the reset switch to run the program. Observe the triangle waveform at the Pin-1 of RM3 using Oscilloscope (CRO) with respect to GND pin-2 of RM3. When jumper JP11 is shorted it is Unipolar mode when jumper JP11 is opened it is Bipolar mode.

## 6.9 To test LCD :

Load the LCD.hex file using programming tool FLASH MAGIC.
folder name: Exp09_LCD
path: ALS-SDA-ARM7-08\SOFTWARE\Exp09_LCD

**Result:** The following message will be displayed on LCD

**" ALS BENGALURU"**
**"LCD INTERFACING"**

## 6.10 External interrupt0 Interface

Load the EXT_INT0.hex file using programming tool FLASH MAGIC.
folder name: Exp10_EXT_INT0
path: ALS-SDA-ARM7-08\SOFTWARE\Exp10_EXT_INT0

Settings:

- short jumper JP6
- Turn on the pin 1 of switch SW20\
- Do not connect FRC from CN7 to CN8

**Result:** When switch SW2 is pressed, the port line goes low & the external interrupt occurs at port line P0.16. To show the external interrupt has occurred LED L7 has been used. (LED L7 toggles at each Press of the SW2).

## 6.11 External interrupt1 Interface

Load the EXT_INT1.hex file using programming tool FLASH MAGIC.
folder name: Exp11_EXT_INT1
path: ALS-SDA-ARM7-08\SOFTWARE\Exp10_EXT_INT1

Settings:

- short jumper JP12
- Turn on the pin 1 of switch sw20

**Result:** When switch SW3 is pressed, the port line goes low & the external interrupt occurs at port line P0.3. To show the external interrupt has occurred LED L7 has been used. (LED L7 toggles at each Press of the SW3)

## 6.12 To test internal ADC:

settings:

- short jumper JP9

### 6.12.1 Polled Mode:

Load the INT_ADC.hex file using programming tool FLASH MAGIC.
folder name: Exp12_INT_ADC
path: ALS-SDA-ARM7-08\SOFTWARE\Exp12_INT_ADC

**Result:** Vary the POT1 (10K) and observe the corresponding analog input value & digital output value on LCD. (The input can be varied from 0.00V to 3.30V, and the corresponding output will be 000 to 3FF.)

### 6.12.2 Interrupt Mode

Load the adc_in_intr_mode.hex file using programming tool FLASH MAGIC.
folder name: Exp12_INT_ADC\adc_in_intr_mode
path: ALS-SDA-ARM7-08\SOFTWARE\Exp12_INT_ADC\adc_in_intr_mode

**Result:** Vary the POT1 (10K) and observe the corresponding analog input value & digital output value on LCD. (The input can be varied from 0.00V to 3.30V, and the corresponding output will be 000 to 3FF.)

## 6.13 TO TEST Pulse Width Modulation (PWM):

Load the pwm.hex file using programming tool FLASH MAGIC.
folder name: Exp13_PWM
path: ALS-SDA-ARM7-08\SOFTWARE\Exp13_PWM

settings:

- JP10 jumper has to be shorted.

**Result:** Check the output waveform on CRO at TP8 with respect to GND and check the corresponding DC voltage at test point TP3 with respect to GND also observe the LED L5 intensity which varies according to PWM output.

## 6.14 TO TEST Keypad (4X4 Matrix):

Load the 4x4keypad.hex file using programming tool FLASH MAGIC.
folder name: Exp14_4x4keypad
path: ALS-SDA-ARM7-08\SOFTWARE\Exp14_4x4keypad

settings:

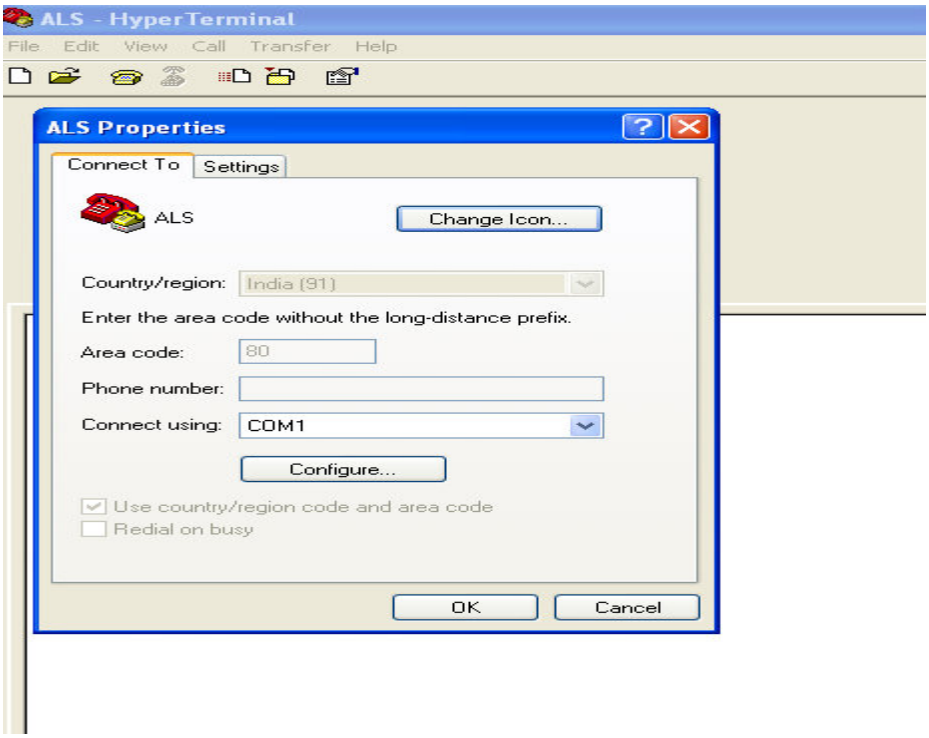
- Turn on the pin 1 of switch sw20

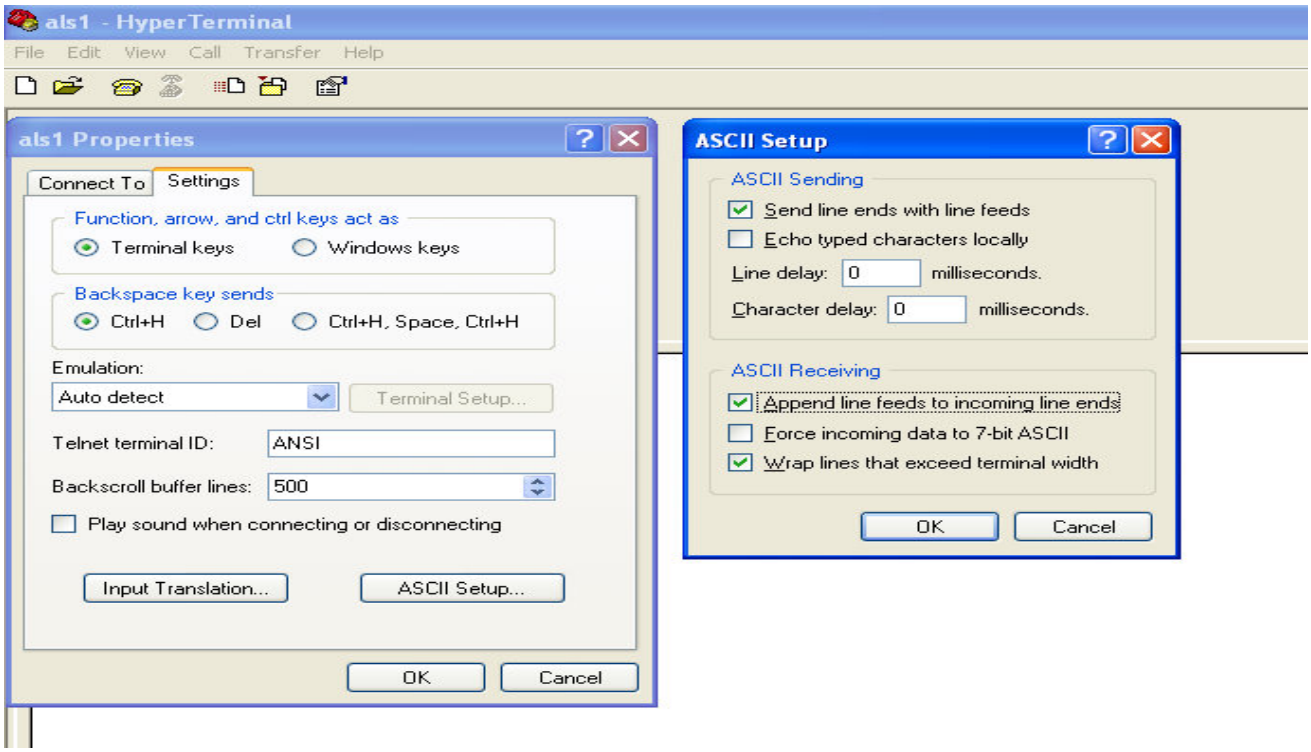
**Result:** When keys SW04 to SW19 are pressed the corresponding outputs '0 to F' will be displayed on the LCD.

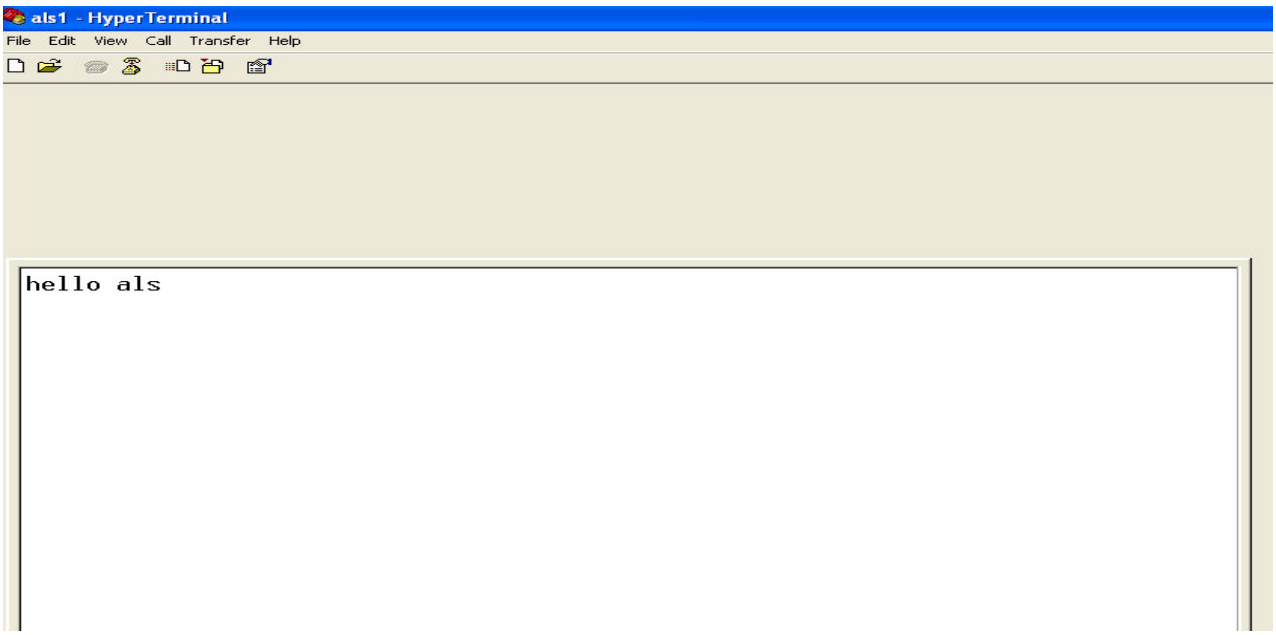
## 6.15 TO TEST UART0:

Load the uart0.hex file using programming tool FLASH MAGIC.
folder name: Exp15_uart0
path: ALS-SDA-ARM7-08\SOFTWARE\Exp15_uart0

**Result:** After downloading the program, Push both pins of dip-switch SW22 to OFF position. do not short the jumper JP7. Press RESET (SW1). Open the hyper terminal, set the Com port, baud rate and other settings as shown below. Pres any key on the PC keyboard the same will be displayed on hyper terminal.




als1 - HyperTerminal
File Edit View Call Transfer Help
als1 Properties
Connect To
Settings
Function, arrow, and ctrl keys act as
Terminal keys
Windows keys
Backspace key sends
Ctrl+H
Del
Ctrl+H, Space, Ctrl+H
Emulation:
Auto detect
Terminal Setup...
Telnet terminal ID:
ANSI
Backscroll buffer lines:
500
Play sound when connecting or disconnecting
Input Translation...
ASCII Setup...
OK
Cancel
ASCII Setup
ASCII Sending
Send line ends with line feeds
Echo typed characters locally
Line delay:
0
milliseconds.
Character delay:
0
milliseconds.
ASCII Receiving
Append line feeds to incoming line ends
Force incoming data to 7-bit ASCII
Wrap lines that exceed terminal width
OK
Cancel

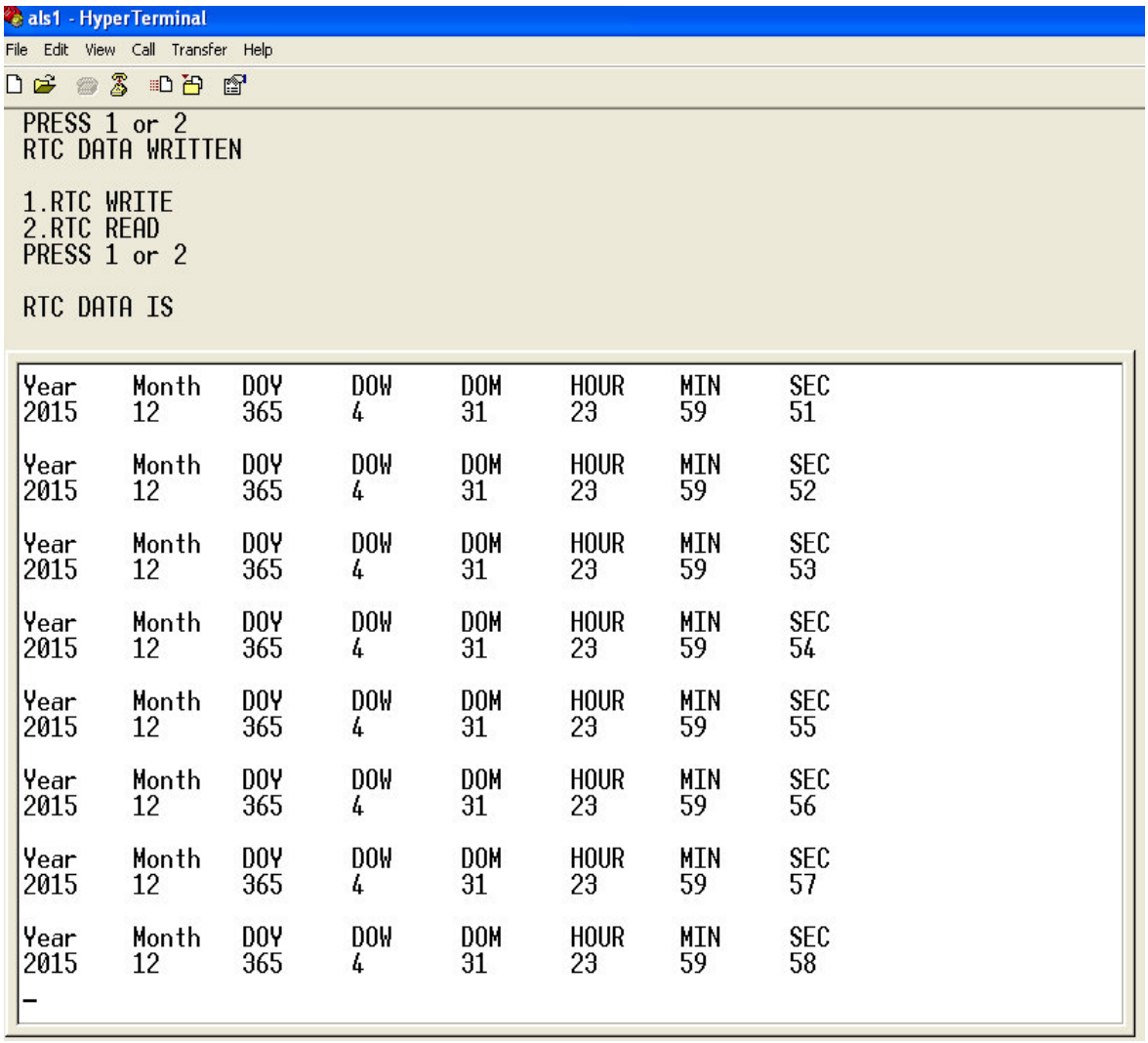

als1 - HyperTerminal
File Edit View Call Transfer Help
hello als

## 6.16 Internal RTC Test:

Load the INT_RTC.hex file using programming tool FLASH MAGIC.
folder name: Exp16_INTERNAL_RTC
path: ALS-SDA-ARM7-08\SOFTWARE\Exp16_INTERNAL_RTC

**Settings:** Make UART0 communication setup. After downloading the hex file, push both pins of dip-switch SW22 to OFF position. Press RESET (SW1).

**Result:** Internal RTC is operated based on the commands sent from PC terminal through UART0. A read and write operation is done and read values are sent to serial port UART0. Until **Esc** key pressed through PC key board a read operation will be continued.



## 6.17 To test SPI EEPROM:

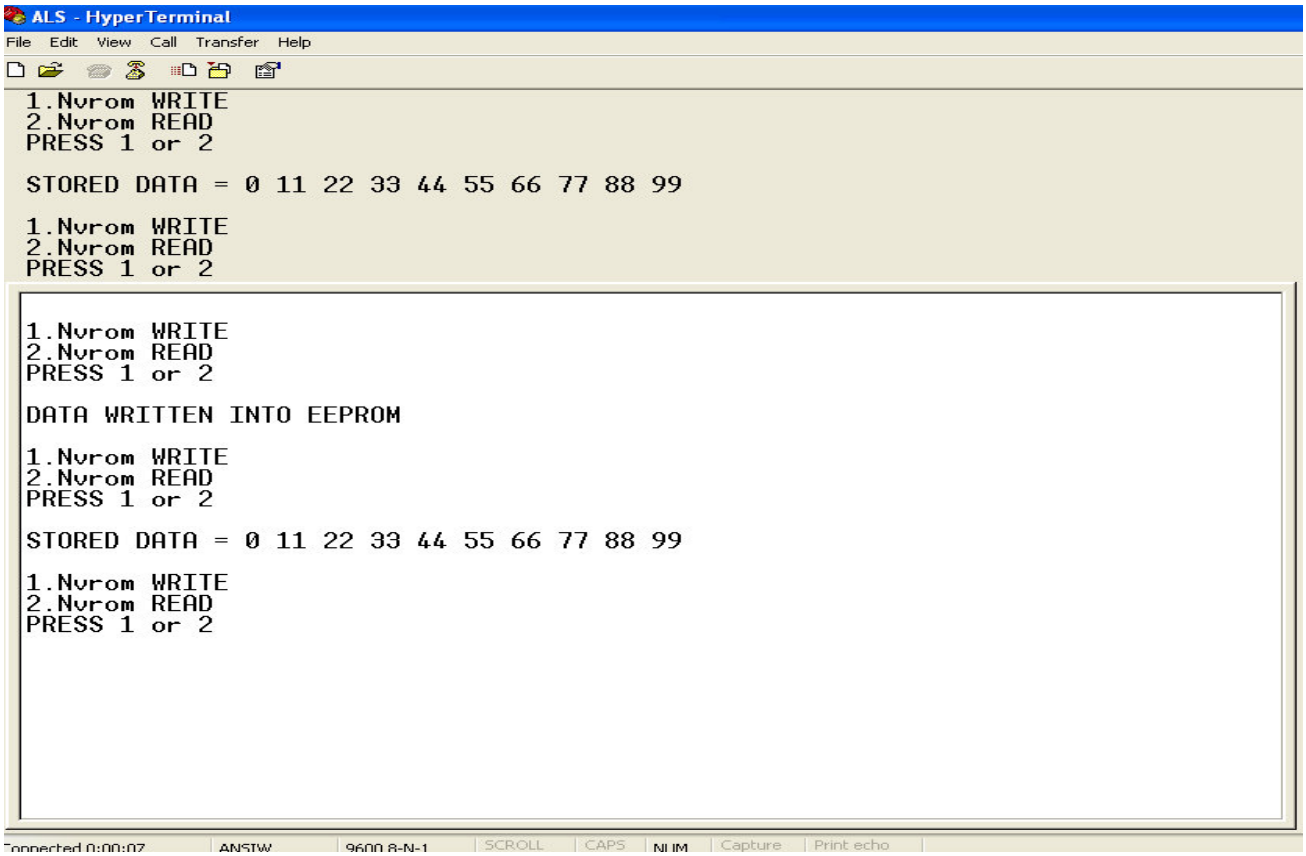
Load the EEPROM.hex file using programming tool FLASH MAGIC.
folder name: Exp17_SPI_EEPROM
path: ALS-SDA-ARM7-08\SOFTWARE\Exp17_SPI_EEPROM

**settings:**

- short jumper JP14(1,2), JP15(1,2), JP16(1,2) and JP17(1,2)
- Remove the connection between CN7 and CN8
- Make UART0 communication setup. After downloading the hex file, push both pins of dip-switch SW22 to OFF position. Press RESET (SW1).

**Result:** The data is written into EEPROM with specific address and data will be read from same memory location and display it in following format:

**STORED DATA = 0 11 22 33 44 55 66 77 88 99**




## 6.18 TO TEST Buzzer (+5V):

Load the Buzzer.hex file using programming tool FLASH MAGIC.
folder name: Exp18_Buzzer
path: ALS-SDA-ARM7-08\SOFTWARE\Exp18_Buzzer

settings:

- short jumper JP13 and JP20

**Result:** Buzzer turn ON and turn OFF for every 1 second, LED L7 toggle correspondingly. This process is continuous in loop.

### 6.19 To test flashing LED's:

Load the LED_Test.hex file using programming tool FLASH MAGIC.
folder name: EXP19_Flashing_LED
path: ALS-SDA-ARM7-08\SOFTWARE\EXP19_Flashing_LED

**settings:**

- short jumper JP21
-

**Result:** observe the toggling of LED's L1 to L4.

### 6.20 To test DIP switch interface:

Load the KEY_TEST.hex file using programming tool FLASH MAGIC.
folder name: EXP20_DIP_SWITCH
path: ALS-SDA-ARM7-08\SOFTWARE\EXP20_DIP_SWITCH

**settings:**
During programing Keep pin1 of DIP switch SW20 in ON position or after programing push pin1 of SW20 to ON position and reset the controller using switch SW1.

**Result:** Turn ON and turn OFF the pins of DIP switch SW20 and observe the key code on LCD.

### 6.21 To test UART1 interface:

Load the UART1.hex file using programming tool FLASH MAGIC.
folder name: EXP21_UART1
path: ALS-SDA-ARM7-08\SOFTWARE\EXP21_UART1

**Result:** After loading the program into the controller connect RS232 cross cable to DB3 and Open the hyper terminal, set the Com port, baud rate and other settings as same as UART0. Pres any printable key on the PC keyboard the same will be displayed on hyperterminal.

## 6.22 RTX APPLICATION EXAMPLES

**TESTING OF SAMPLE PROGRAMS:**

For all the RTOS demo programs make sure that the corresponding settings have to be made:

- Short JP5 to connect +3.3v into the circuit.
- Both Switches of SW22 should be in ON position for ISP programming.
- Short JP7 for ISP programming.
- Once the program loaded into device remove JP7, turn of the dip switch SW22 and press RESET switch SW1.
- Connect the 26 core FRC **CN7** to **CN8** to connect to **On-board interfaces**.

**Note:**

- Do not short any pins of JP14, JP15, JP16, JP17, JP18, JP19, JP22 and JP23 when 26 pin FRC is connected from CN7 to CN8.
- Pin1 of SW20 DIP SWITCH should be turn on pin1 while doing LCD, KEYPAD, ADC0809 examples Because Pin1 of SW20 P1.20 should not be pull down using resister. LOW on P1.20 pin while RESET is LOW enables pins P1.25:16 to operate as Trace port after reset
- Short JP2(1,2) to connect P0.31 from CN7 to CN8.
- Short JP13 to Connect P0.14 from CN7 to CN8 use only while doing I2C, ADC0809 and Buzzer example. Do not reset while this jumper is closed.

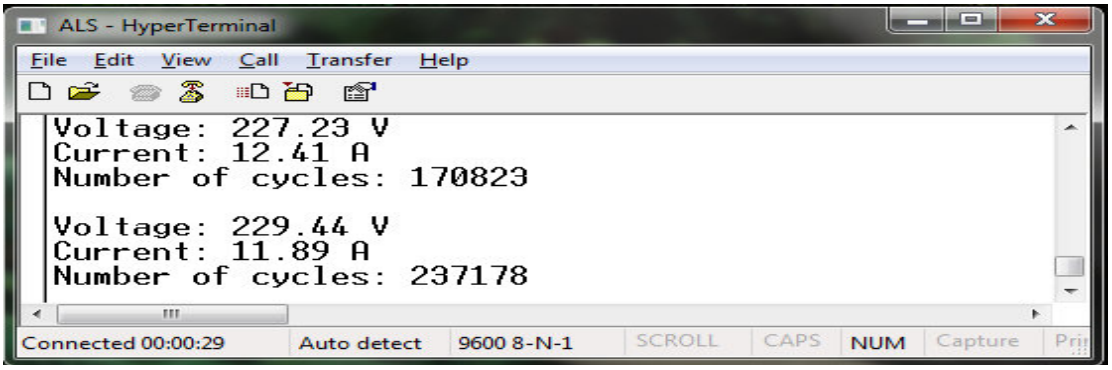
**NOTE: The RTOS Mailbox related information, project creation in RTX and downloading procedure is provided in separate manual RTOS-ALS-SDA-ARM7-08.**

### 6.22.1 RTX_MAILBOX

**Folder Name:** 1.RTX_MAILBOX
**Download Hex File:** RTX_MAILBOX.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Because the message objects in this example are fixed size, the RTX Thread-Safe Fixed-Block Memory Allocation functions are used. They are fully re-entrant and can be used with RTX Kernel with no restrictions.

**Result:** To observe the output on hyper-terminal switch OFF both switches of SW22, open JP7 and press RESET(SW9). Short JP21 and L1, L2 & L3 will turn on for each message displayed on hyper-terminal and you can observe following data.




### 6.22.2 RTX_LCD_MAILBOX

**Folder Name:** 2.RTX_LCD_MAILBOX
**Download Hex File:** RTX_LCD_MAILBOX.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task

inter-communication. Because the message objects in this example are fixed size, the RTX Thread-Safe Fixed-Block Memory Allocation functions are used. They are fully re entrant and can be used with RTX Kernel with no restrictions.

**Result:** A Fixed message is stored in the Mailbox in task1 and Those stored messages are displayed in task2. Press RESET(SW1) and Observe output on LCD.

"ALS,R&D SECTION,"
"BENGALURU-58 ,"

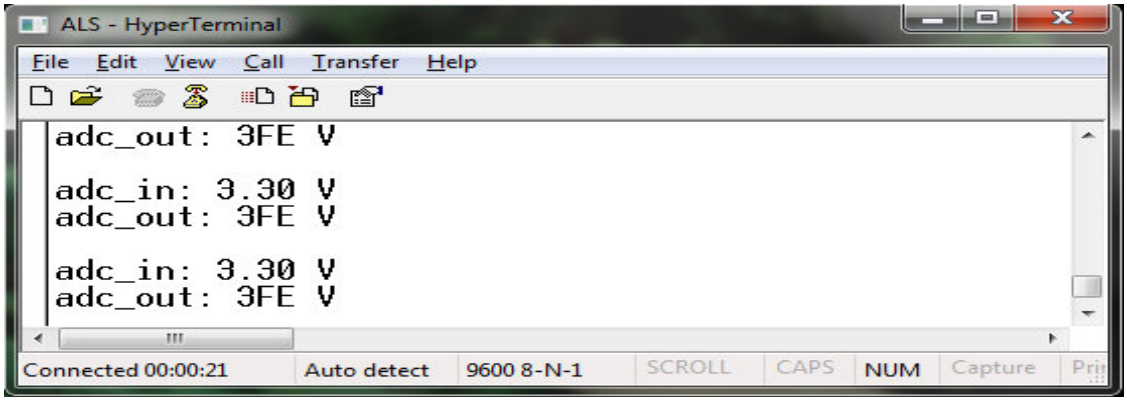
"BENGALURU-58 ,"
"ALS, R&D SECTION," Continuously.

### 6.22.3 RTX_MAILBOX_INT_ADC

**Folder Name:** 17.RTX_MAILBOX_INT_ADC
**Download Hex File:** RTX_MAILBOX_INT_ADC.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Because the message objects in this example are fixed size, the RTX Thread-Safe Fixed-Block Memory Allocation functions are used. They are fully re entrant and can be used with RTX Kernel with no restrictions. Internal ADC converted values are stored in Mailbox in task1 & Those stored Values are displayed in task2.

**Result:** To observe the output on hyper-terminal switch OFF both switches of SW22, open JP7 and press RESET(SW9). Short JP9 and Vary the POT1 and observe output 0 to 3FF for the input voltage 0 – 3.3V.




### 6.22.4 RTX_7SEG_DOWN_COUNTER

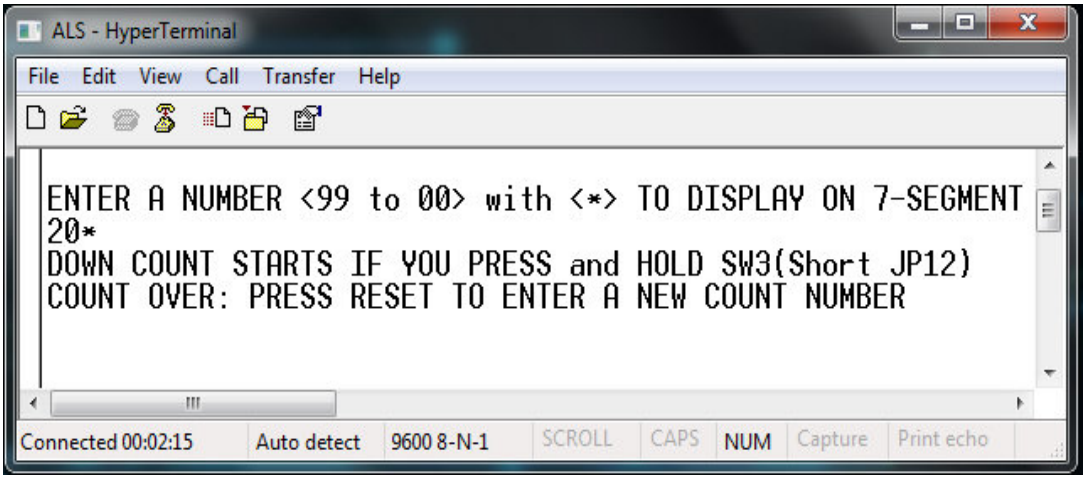
**Folder Name:** 4.RTX_7SEG_DOWN_COUNTER
**Download Hex File:** RTX_7SEG_DOWN_COUNTER.hex

**Description:** In this RTX example Down-Counter is implement using pre-emptive task switching. init_task will create task1(2), task2(1) and task3(1) delete itself from kernel, as the priority of task1 higher than the task2 and task3 it will run first. In task1 input count

number is received from PC keyboard in the format <99 to 00> with <*> as end of string then this data is send to task2 using mailbox and task1 is deleted from kernel. In task2 data is processed and control is passed to task3 as priority is same(cooperative task) and in task3 down counter display is implemented.

**Result:** To observe the output on hyper-terminal switch OFF both switches of SW22, open JP7 and press RESET(SW9). Switch ON both switches of SW21, short JP12 open hyper terminal and press RESET(SW1) The output can be seen on 7 segment as on seven segment display:

**Down-Counter:** if we press and hold SW3 continuously 99 - 00




### 6.22.5 RTX_DIP_SWITCH

**Folder Name:** 5.RTX_DIP_SWITCH
**Download Hex File:** RTX_DIP_SWITCH.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Task1 send data to mailbox and task2 is used to display data in mailbox, turn ON any one pin of DIP switch SW20 corresponding keycode is will be processed as data and stored in mailbox, these stored messages are displayed in task2.

**Result:** Press RESET(SW1). Initially following message will be displayed on LCD.

"DIP SWITCH TEST"
"DIP SW = 00"  // if all pins of SW20 are OFF

when you turn ON any one pin of SW20 below message will be displayed on LCD.

"DIP SWITCH TEST"
"DIP SW = 02" // ex: if you turn ON pin2 of SW20

### 6.22.6 RTX_4x4KEY_MAILBOX

**Folder Name:** 6.RTX_4x4KEY_MAILBOX
**Download Hex File:** RTX_4x4KEY_MAILBOX.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Fixed message stored in the Mailbox in task1 by pressing Any KEY on 4x4 Matrix and posted on the mailbox & Those stored messages are displayed in task2. output can be seen on LCD. Press SW4-SW19 Corresponding Key code 00-0F will be displayed on LCD.

**Result:** Initially following message will be displayed on LCD

"4x4 KEYPAD TEST"
"WAITING"

when we press key i.e SW4-SW19 below message will be displayed on LCD

"4x4 KEYPAD TEST"
"KEY = 01" // if we press SW5

### 6.22.7A RTX_ADC0809_MAILBOX

**Folder Name:** 7a.RTX_ADC0809_MAILBOX
**Download Hex File:** RTX_ADC0809_MAILBOX

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Because the message objects in this example are fixed size, the RTX Thread-Safe Fixed-Block Memory Allocation functions are used. They are fully re entrant and can be used with RTX Kernel with no restrictions. In task1 adc conversion is done and data is sent to mailbox and in task2 data read from mail box and displayed on LCD.

**Result**:To observe output on LCD open JP7, short JP13 and Press RESET(SW1). Vary POT4 and observe O/P vary from 00 to FF for 0.0 V to 5.0 V.

### 7A) RTX_ADC0809_TEMP

**Folder Name:** 7b.RTX_ADC0809_TEMP
**Download Hex File**: RTX_ADC0809_TEMP.hex

**Description:** This example program shows how to use the RTX Kernel mailboxes for task inter-communication. Because the message objects in this example are fixed size, the RTX Thread-Safe Fixed-Block Memory Allocation functions are used. They are fully re entrant and can be used with RTX Kernel with no restrictions. In task1 adc conversion is done and data is sent to mailbox and in task2 data read from mail box and displayed on LCD.

Result: Open JP7, short JP13 and Press RESET(SW1). Temperature will be displayed on LCD.

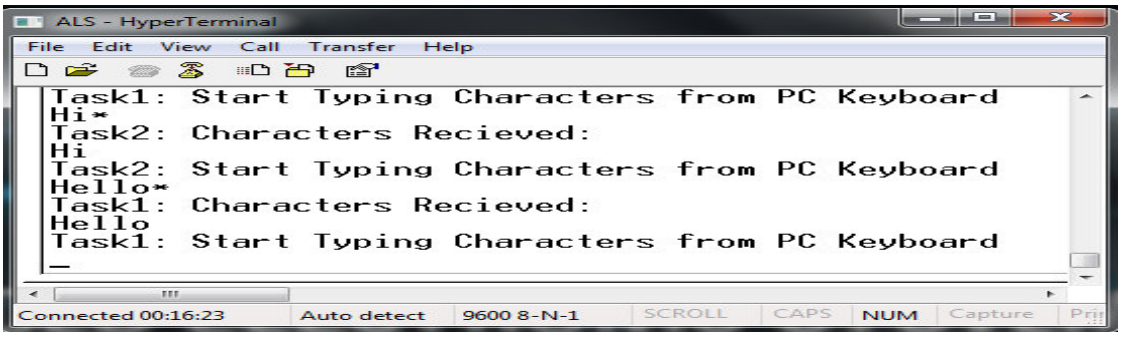
### 6.22.8 RTX_CHATTING

**Folder Name**: 8.RTX_CHATTING
**Download Hex File**: RTX-CHATTING.hex

**Description:** In this RTX example init_task will create task1, task2 and delete itself from the kernel. we have two task send_task and rec_task:
send_task: In this task we will send message to rec_task using mailbox. Message pattern is <Message1><*>.
rec_task: In this task we will read the message sent by send task
using mailbox and reply to that message with same pattern <Message2><*>.

**Result:** To observe the output switch Off both switches of SW11 and Press RESET(SW9). Observe output on hyper terminal.

**Note:** Message format should be <message> followed by <*>




## 6.23 FPGA EXAMPLES:

**Note: All FPGA Spartan6 related information, installation, configuration, properties are provided in separate manual NIFC-68_REV00_Manual.**

**6.23.1 RTC (Interrupt characteristics):** Real time clock is generated in nifc-68(spartan6-xc6slx4). By dividing 100Khx clock a 1 Hz signal is generated internally . An hours , Minutes and seconds counter is implemented .The 1 Hz signal ( period is 1 sec) is used to interrupt the Controller in ALS-SDA-ARM7-08 evaluation board through EINT pin of ARM controller. At every interrupt the Controller reads the Hours , Minutes and Seconds from the FPGA and displays it on Hyperterminal in a PC ,through serial interface .

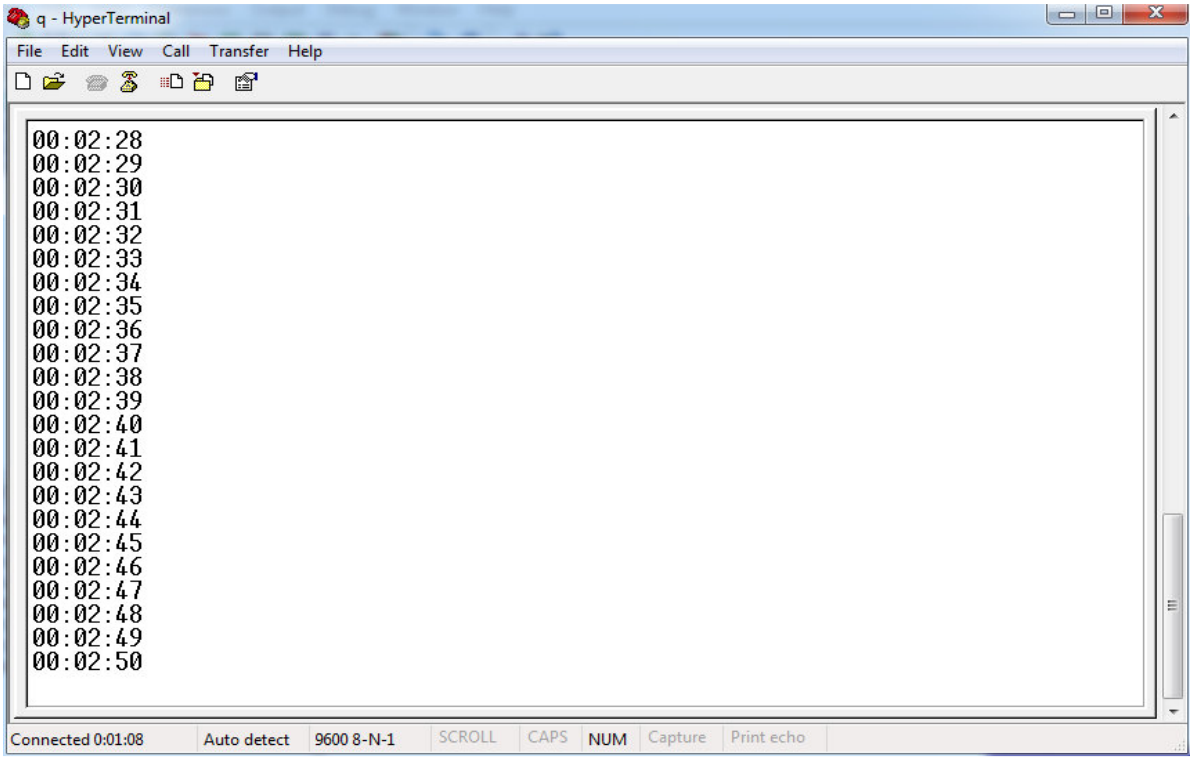
**PIN ASSIGNMENTS:**

| I/O pins | 26-pin connector | Xc6slx4 IC pins | LPC2148 | Description |
|---|---|---|---|---|
| hund_khz | - | P55 | | Short JP2 (2,3) 100khz clock |
| sel[0] | CN5/5 | P138 | P0.8 | Select lines for sec/min/hr |
| sel[1] | CN5/6 | P139 | P0.9 | Select lines for sec/min/hr |
| int_out | - | P2 | P0.3 | One sec Interrupt to 2148. connect INT |
| dout[0] | CN5/13 | P126 | P0.16 | |
| dout[1] | CN5/14 | P124 | P0.17 | |

| | | | | |
|---|---|---|---|---|
| dout[2] | CN5/11 | P127 | P0.18 | 7-bit Data ouput lines |
| dout[3] | CN5/12 | P131 | P0.19 | |
| dout[4] | CN5/9 | P132 | P0.20 | |
| dout[5] | CN5/10 | P133 | P0.21 | |
| dout[6] | CN5/7 | P134 | P0.22 | |

**WORKING procedure.**

- Connect 26-pin FRC cable from CN5 to ALS-SDA-ARM7-08 evaluation board.
- Download rtc_interrupt.mcs(xcf40s)/interrupt.bit(xc6slx4) file to NIFC68 and FPGA_EINT1_RTC.hex file ALS-SDA-ARM7-08.
- Connect flyinglead from **INT** test point of NIFC68 to the pin1 of JP12 of ALS-SDA-ARM7-08.
- Connect one end of DB9 serial cable to DB2 of ALS-SDA-ARM7-08 and another end to PC
- Open the hyperterminal and observe the time.




**6.23.2 RAM INTERFACE 16X8:** RAM is a volatile device to store and retrieve data. Here in this experiment 16x8 RAM is implemented in FPGA . It stores 16 elements of 8-bit data, and uses 4–bit address (2^4 = 16 locations) 0 to 15. The control signals chip select(CS), write(WR) and read(RD) are also used in this design.

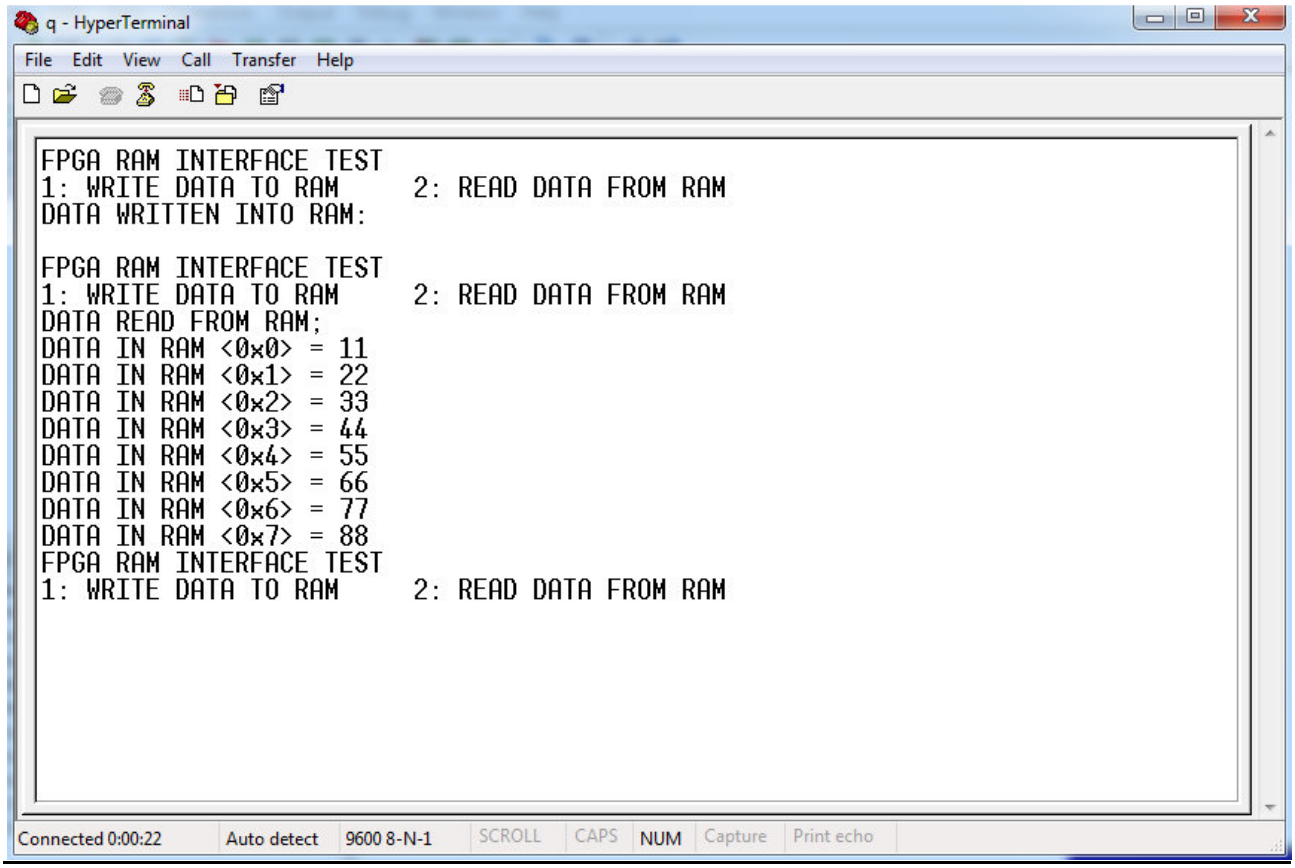
**PIN ASSIGNMENTS:**

| I/O pins | 26-pin connector | Xc6slx4 IC pins | LPC2148 | Description |
|---|---|---|---|---|
| Addr[0] | CN5/5 | P138 | P0.8 | 0th bit address |
| Addr[1] | CN5/6 | P139 | P0.9 | 1st bit address |
| Addr[2] | CN5/3 | P140 | P0.10 | 2nd bit address |
| Addr[3] | CN5/4 | P141 | P0.11 | 3rd bit address |
| Data[0] | CN5/13 | P126 | P0.16 | |

| | | | | |
|---|---|---|---|---|
| Data [1] | CN5/14 | P124 | P0.17 | 8-bit bidirectional data lines |
| Data [2] | CN5/11 | P127 | P0.18 | |
| Data [3] | CN5/12 | P131 | P0.19 | |
| Data [4] | CN5/9 | P132 | P0.20 | |
| Data [5] | CN5/10 | P133 | P0.21 | |
| Data [6] | CN5/7 | P134 | P0.22 | |
| Data[7] | CN5/8 | P137 | P0.23 | |
| CS | CN5/17 | P120 | P0.28 | Active low chip select |
| WR | CN5/18 | P119 | P0.29 | Active high write enable |
| RD | CN5/15 | P123 | P0.30 | Active high read enable |
| LED[0]~LED[7] | - | P21,P23,P26,P29, P32,P34,P38,P43 | - | Output LED's |

**WORKING procedure:**

- Connect 26-pin FRC cable from CN5 to ALS-SDA-ARM7-08 evaluation board.
- Download SRAM.mcs(xcf40s)/ ram_ifc.bit(xc6slx4) file to NIFC68 and fpga_ram.hex file ALS-SDA-ARM7-08.
- Connect one end of DB9 serial cable to DB2 of ALS-SDA-ARM7-08 and another end to PC.
- Switch OFF SW22 and remove JP7 of ALS-SDA-ARM7-08.
- Open the hyperterminal and observe menu will be displayed.
- Press '1' to write to RAM and press '2' to read from RAM.
- During read, for every 2 seconds data byte will be read we can observe the output LED's of NIFC-68.



# CHAPTER 7: TROUBLE SHOOTING

## 7.1 Power Supply:

- JP5 jumper has to be shorted to connect **+3.3V** to controller and other circuits**.**

## 7.2 In System Programming / Download (ISP):

In System Programming or download could not be established properly then check out whether the following conditions are met.

- The cable used for communication should be **cross cable**.
- Switch ON 1 & 2 switches of the SW22 **DTR** and **RTS** for serial communication.
- JP7 should be shorted.
- IC MAX232 is in good condition.

## 7.3 JTAG Programming / Download:

- Short jumper **JP1 (RTCK)** for communication.
- Switch ON 1 &2 pins of the SW22 **DTR** and **RTS** for serial communication.

## 7.4 General Problems:

- Make Proper Jumper Connections as mentioned in Hardware Details.
- Make Proper Connections as mentioned in Demo Programs Set up.

# CHAPTER 8: QUICK REFERENCE

**PORT LINE DETAILS – Used for on board interfaces:**

| SL no. | PROGRAM NAME | PORT LINE |
|---|---|---|
| 1 | LCD | P0.2 - P0.7 |
| 2 | 7SEG DISPLAY | P0.16 - P0.23(data) & P0.28 - P0.29(sel) |
| 3 | GP LED'S | P0.16 – P0.19 |
| 4 | UART0 | P0.0 & P0.1 |
| 5 | UART1 | P0.8 & P0.9 |
| 5 | STEPPER MOTOR | P0.28 – P0.31 |
| 6 | DC MOTOR | P0.10 & P0.11 |
| 8 | BUZZER | P0.14 |
| 9 | EXT-INTERRUPT0 | P0.16 |
| 10 | DAC0800 | P0.16 – P0.23 |
| 11 | PWM | P0.8 |
| 12 | Internal ADC | P0.25 |
| 13 | 4X4 KEY MATRIX | P1.16 – P1.23 |
| 14 | EXT-INTERRUPT1 | P0.3 |
| 15 | I2C | P0.11(SCL) and P0.14(SDA) |
| 16 | SPI | P0.17,P0.18,P0.19,P0.20 |
| 17 | ADC0809 | P0.16-P0.23(data), P1.16,P1.17,P0.14(addrs), P0.15(STRT),P1.18(out_EN),P1.19(EOC) |