μNode

User Manual

and

Technical Reference

The **µNode** is a product of:

Q Solutions
107 Gatwick
Constitution St
Cape Town
8001

Table of Contents

1. OVERVIEW OF THE DEVELOPMENT CARD

The price, lack of availability and limited functionality of in-circuit emulators, eprom simulators and development cards inhibit their extensive use. As a result, small institutions, scholars and hobbyist find it a fairly difficult and complex task to develop efficient projects quickly. The µNode Controller Module was designed to provide a solution to this problem by combining these development aids in one unit.

The development card was implemented using the Philips 80C552-16 8-bit microcontroller. This microcontroller contains the built-in peripherals, essential for most computer/controller applications.

The development card contains the following features :

1. High speed serial eprom simulator. (XT/AT compatible)
2. 32K of External RAM and Eprom.
3. PHILIPS PCF8583 Real time clock with alarms. (I^2C bus interface)
4. PHILIPS LTN211 16 character x 2 line LCD display.
5. Battery back-up of user code. (approx. 10 yrs continuous)
6. All important signal lines are accessible for maximum expansion.
7. Power supply unit.

The development card is particularly well suited for process control development, since it accepts compiled code, and can execute it at the maximum clock speed of 16 Mhz. Complex calculations are hence only limited by the strength and efficiency of the compiler. Because of the controller card's low system overhead, it is extremely fast and efficient.

The development card does not contain an Eprom programmer, but instead uses battery non-volatised RAM for program CODE development. Using this technique, a Romulator (EMROM) is implemented, whereby compiled code is downloaded via an RS232 cable into RAM. The development card can then be instructed to execute the code from RAM.

This solution also simplifies the overall board design and prevents the need for high programming voltages, which can affect reliability.

2. THE CONTROLLER/DEVELOPMENT BOARD

The controller card is a single board controller/development system. It contains the microcontroller, RAM/EMROM, watchdog timer, real time clock, battery, 2 serial ports and 3 parallel ports. A DIN 41612 connector allows the computer card to be used in a back-plane configuration, adding memory and I/O boards as required.

There are five main sections to this coverage of the computer board : processor , address decoding and memory, parallel I/O, serial I/O (RS232 and I^2C bus), and power requirements. The block diagram of the computer board is shown in APPENDIX 1.

2.1 PROCESSOR

The computer board was designed around the Phillips 80C552-16 microcontroller, which is based on the 8051 family . The 80C552 contains 256 bytes of RAM, three 16 bit counter/timers, 14 interrupts, 56 I/O lines, 8 channel 10 bit A/D, 1 I^2C serial interface, 2 pulse width modulators and a watchdog timer.

See Phillips 80C51 based 8-bit Microcontroller User Manual (REF: IC20) for a detailed description.

The 80C552 has a 16 bit address and an 8 bit data bus (with the lower address and data bits multiplexed together). When power is applied, the 80C552 will execute code from EPROM or RAM depending on which mode is selected : TRANSFER or EMULATE.

The 80C552 is clocked with an 11.0592 Mhz crystal. This is to facilitate commonly used serial baud rates. If processing speed is critical, a higher crystal frequency e.g., 16 Mhz could be used, or if power conservation is essential, it can be replaced with a lower frequency crystal after development. A 24 Mhz board is also available on request.

2.2 ADDRESS DECODING

The decoding is achieved by means of pre-programmed GAL logic, to ensure minimum address decoding delays, essential for high clock rates. The blown fuses within the GAL decoder, which establish the memory and I/O addresses, ***cannot*** be changed, since the security bit of the device is activated.

In TRANSFER MODE the emulator code, embedded in the 32K eprom, is executed. (Mode selection is achieved by setting the DIP switches accordingly. See *Section 6.)* This mode can also be used in the final design of a product, where the existing eprom is replaced by an eprom which contains the user's final code. Industry production cards are also available on request, where RAM, EPROM and RS232 chips are optional.

Note: The eprom is a ONE TIME programmable type, hence the user is cautioned to develop the final code in EMULATOR mode.

In EMULATOR MODE the development card makes available the entire 64K of external indirect addressable memory map, of which 32K is the on board RAM. All I/O peripherals become available to the developer.

See Technical Reference: Memory Map.

2.3 PORT I/O AND CHIP SELECTS

The 80C552 contains 56 port I/O pins. A few of these pins are dedicated to a particular function, hence the user is cautioned when using interrupt pins as I/O. These port pins are bi-directional in structure and are CMOS compatible. *See 80C552 Manual* for port loading capabilities.

The development card also provides three free chip selects which are available to the developer. These are pre-decoded by the GAL. The addresses are so chosen as to facilitate I/O peripherals and possibly a large memory block. *(Refer to the MEMORY MAP section for the appropriate addresses of these chip selects.)*

The upper (16K) of memory is decoded into user select S2. This can be used together with external decoding circuitry, for adding large paged memory structures, especially for industrial or DATA logging applications.

2.4 SERIAL I/O(RS232) AND I^2C SERIAL I/O

There are two serial ports on the 80C552 microcontroller, the RS232 serial port, and the I^2C bus serial port.

2.4.1 RS232 SERIAL I/O

This serial port is functionally identical to a USART, i.e. the baud rate generator, data bits , parity bit and stop bit are fully programmable. No external line driver is needed, as this is provided.

In the TRANSFER mode this serial I/O is configured for 19200 or 9600 baud, depending on the keyboard selection. (using the default 11.0592 Mhz crystal.) The USART is accessible in the EMULATOR mode. Care must be taken when changing crystal values, as this affects both modes.

Possible applications are data capture, line printer(printer connected in null modem configuration) or driving a large LCD/LED display.

2.4.2 I^2C SERIAL I/O

This is a very high speed (100 kHz, with 400 kHz "fast mode") intelligent bi-directional serial I/O. Many devices with I^2C compatible I/O can be connected together in a ring configuration using the master/slave principle . Within this ring, any master configurable device can communicate with any other device on the bus. This is particularly useful in multiprocessing, as it reduces software and hardware overheads considerably.

See library section for software support. This is an important I/O, and could be used extensively if properly understood. Refer to Philips i^2c data book (REF: IC 14a , 14b) for a thorough investigation.

The built in real time clock (RTC PCF8583) is an I^2C bus device, and is accessed during TRANSFER MODE. I^2C bus test routines are built into the unit and can be accessed in the TRANSFER MODE.

NOTE:
In order to make as many interrupts as possible available to the developer, the RTC alarm shares EXTERNAL INTERRUPT 1 with the keypad decoder. Below are hints to use to overcome the situation.

HINTS:
When servicing external interrupt 1, in the case where the interrupt could have been generated by the RTC, the Control/Status register of the RTC can be examined to establish the source of the interrupt. Bit 1 in this byte, (Control/Status register) is the alarm flag. In cases where the RTC alarm is not being used, it can be assumed that the interrupt was generated by the keypad. *(See TECH REF " SOFTWARE" for details on the routine test_x1 which tests this bit)*

2.5 POWER REGULATOR UNIT

The power supply unit essentially consists of a linear voltage regulator. This unit is capable of providing a maximum of 1 Amp. It would be advisable if the development board could be supplied with a maximum of 12V and a minimum of 8V, as to reduce power dissipation across the regulator, and to ensure board reliability.

HINT:
Use only CMOS or ALS(TTL) components for all add-on boards. Insert a heat sink under the regulator if large voltage drop across regulator.

Supply board with 8V maximum.

3. OPERATING INSTRUCTIONS

Connect the power and RS232 cables to the µNODE card. See TECHNICAL REFERENCE: µNode Layout for details.

When power is supplied, the µNODE card automatically resets the 80C552 microcontroller. Pressing the RESET key is also a way of properly initiating the reset condition. The operation of the 80C552 then depends on the D.I.P switch settings:

a) Previous mode: STARTUP
Present mode : TRANSFER

D.I.P switch 1 : position [0], the following will happen.

The 80C552 will display the signature logo of the product, and then will program it's USART with the following parameters [19200 bit/s, 8 bits, no parity, 1 stop bit] for reception. The real time clock will then be read, and the time in international format is displayed on the LCD.

The keyboard can now be used for changing the transfer rate or correcting the displayed time.

Press the [B] key (baud rate), to toggle the transfer rate to 9600 baud. This is useful for user of XT compatible machines, which cannot work at 19200 baud. Pressing the [B] key toggles the rate back to 19200 baud.

Press the [C] key (change time) and the bottom line will indicate that the user must enter the correct time. At the end of the sixth entry the new time is displayed and the real time clock is updated. To set the clock accurately, enter the sixth key when the seconds are correct. To escape from the [C] key sequence, press the [RESET] key.

The µNODE card is now ready for reception. The user should now enter the following command at the computer keyboard if using an IBM compatible. { COPY *.bin [COM1, COM2,..] }. Use any communication package if using a non IBM machine. After the binary file is sent, the LCD displays the numbers of bytes received. Check this against the size of the file by doing a [dir., cat,..] on the file. If the sizes are identical, the µNODE can now enter the EMULATOR mode, if not, make sure the baud rates are identical.

b) Previous mode : TRANSFER
Present mode : EMULATOR, (watchdog enabled)
D.I.P switch 1 : [1] ,
D.I.P switch 3 : [1] , the following will happen,

The 80C552 will execute the user code from RAM.

NOTE: The 80C552 will reset, and execute the user code from address 0000H.

The real time clock can be accessed via the I^2C bus if needed. All peripherals and add-on memory becomes available.

c) Previous mode : TRANSFER
Present mode : EMULATOR, (watchdog disabled)
D.I.P switch 1 : [1] ,
D.I.P switch 3 : [0] , the following will happen,

The 80C552 will execute the user code from RAM, but from an undetermined location. Press RESET to execute code correctly.

NOTE: The 80C552 does not reset, and will NOT execute the user code from address 0000H, as the watchdog timer does not reset the 80C552 microcontroller.

d) Previous mode : EMULATOR
Present mode : TRANSFER
D.I.P switch : [0] , the following will happen,

The 80C552 will not enter the TRANSFER mode, i.e. it will not RESET and wait for reception. To activate the proper operation of this mode, press the [RESET] key. Code can now be downloaded.

4. D.I.P SWITCH SELECTION SETTINGS

There are three DIP switches to be set on the controller module.

S1 (as mentioned above) is used to select the mode of operation between TRANSFER and EMULATE mode.

S2 is used to select the CODE size of the user software between 16K and 32K. NOTE: If 32K mode is used, no on board RAM is available.

S3 enables/disables the watch-dog timer (T3) of the 80C552.

See TECHNICAL REFERENCE: DIP-SWITCH S1, S2, S3

5. SOFTWARE SUPPORT AND LIBRARIES

The following software packages can be used for developing 8051 support software. The list below is categorised from the most efficient and easy to use.

1. Franklin C (version 3.9) with RTX tiny multitasking operating system.
2. Franklin C (version 3.7)
3. Intel PL/M -51 (revision -003)
4. Intel Assembler ASM51
5. Archimedes C-51 Cross- Compiler for 8051 Microcontroller. (8th ed.)
6. Cybernetics ASM51 with software simulator.

All the above software packages allow the user to generate a pure Intel HEX FILE. This can then be converted into a binary format with the HEXBIN.EXE utility.

Using high level language compilers will make complex maths functions, string formatting and better memory management possible.

The libraries that supplement the µNODE card, are in the form of C include files and Assembler files. These assembler files can be linked into the user's code or added to the compiler library files by the developer, using the appropriate library manager.

The list below represents the C include and ASM files *(See TECHNICAL REFERENCE: SOFTWARE)*:

1. All master and slave routines for the I^2C bus configuration.
2. An updated **putchar** and **getkey** dedicated for LCD or RS232 formats.
3. A **watchdog** refresh routine.

4. LCD initialisation and configuration routines.
5. RTC initialisation and **show_time** and **show_date** routines.
6. An RS232 initialisation routine.
7. A **test_x1** routine to determine the origin of external interrupt 1.

6. TECHNICAL REFERENCE

6.1 POWER SUPPLY:

The **3-pin** power connector **(J1)** is configured as follows:

>J1.1: 9V rail

>J1.2: ANALOGUE GND (AGND)

>J1.3: GND

The **96-pin** connector **(P1)** includes six power rails:

>P1A.01: GND

>P1A.02: ANALOGUE GND (AGND)

>P1A.03: Vcc

>P1A.04: GND

>P1B.17: GND

>P1B.18: 9V rail

NOTES:

- The 9V rail takes a **minimum of 8V**. To reduce power dissipation in the regulator, keep this voltage as low as possible.
- If custom boards are fitted at P1, and rely on Vcc (P1.A3) for power, the regulator will need a heat sink.
- Unloaded, the board draws less than 200mA with a 11.059MHz Crystal.
- The regulator can source 1A, if adequately cooled.
- If Vcc is being supplied to the unit at P1A.03 by an external supply, do not connect to the 9V rail.
- To avoid ground loops, do not connect to GND of the unit at more than one pin.
- In the absence of power to the unit, the on board battery keeps V+, the supply rail for the Real Time Clock and RAM, at **3.6**V.

6.2 THE REAL TIME CLOCK (RTC)

The real time clock (*Philips PCF8583* Refer to PHILIPS "I²C BUS COMPATIBLE IC's IC12a/b") communicates with the microcontroller using the **I²C** bus. It generates an alarm interrupt on **INT1**, multiplexed with the keypad decoder's interrupt. When servicing an interrupt that could have been caused by either device, the **CONTROL/STATUS** register of the RTC can be read and the alarm interrupt bit masked. (*test_alarm ( )* below, does this.)

Routines that make use of the RTC have been written in assembler, and can be linked into a high level language program. The following functions are available. (using *K.&R.* C declaration style.)

```
bit     init_RTC    ( void );          /* 1 if RTC present        */
void    show_time   ( char position );  /* position: LCD cursor    */
void    show_date   ( char position );
bit     test_alarm  ( void );          /*returns alarm status bit */
void    set_time    ( void );          /*enter time at keypad     */
```

6.3 ROM

The **ROM** contains the **HOST** code that runs during **MODE 0 (TRANSFER)**. This code displays the logo " **µNode** ", and the time on the top line of the **LCD** display.

GUEST code, to be run during **MODE 1 (EMULATE)**, can now be downloaded via the **RS232** port (**J2**). Once code has been received, the number of bytes received is reported on line 2 of the LCD display.

The following keys have functions:

A: I²C slave bus test.

B: Toggle Baud rate between 19.2 kBd and 9.6 kBd.

C: Change Time.

6.4 LCD DISPLAY

The LCD display used is the Philips LTN211R-10, 2 X 16 character dot matrix display. *(Refer to PHILIPS': LIQUID CRYSTAL DISPLAYS AND DRIVER IC's FOR LCD. REF: LDC01)*

The COMMAND register of the display can be written to or read from at location 8202H, and the DATA register can be written to at location 8201H. The assembler routine, *putchar*, can be used to write one character to the DATA or CONTROL register, and can be called as is in a C program. The declaration in C is:

extern char putchar (char);

The function of putchar is to print one character to the device selected by the variable *io_stream* (defined as **PUBLIC** in the module **PUTCHAR**):

io_stream= 0 : Output goes to ***LCD DATA*** *(screen)*

io_stream= 1 : Output goes to **LCD COMMAND REGISTER**

io_stream= 2 : Output goes to **RS232 PORT (S0BUF)**

This function is required by the function *printf ( )* found in most C compilers' run time library. To call this function from assembler, first set up *io_stream*, move the character to be printed into **R7**, and then call *putchar*. Control will be returned to the calling routine once the byte has been transmitted or displayed. (If *io_stream= 0* or *1*, the function waits for the "**BUSY**" bit in the LCD module's control register to clear. In the case of *io_stream = 2*, the **TI** bit of the serial **UART** is polled until set, and then cleared.)

R1 is the contrast control of the LCD display.

The following functions are also available:

void set_lcd (char command);

void init_lcd ();

The first function writes "*command*" to the **COMMAND** register of the LCD module. *"Init_lcd ()"* sets up the LCD into the following state:

8 BIT DATA; **DISPLAY ON;**

CURSOR OFF; **AUTO INCREMENT CURSOR;**

FREEZE DISPLAY; **SCREEN CLEAR.**

6.5 ADDRESS BUS

The entire address bus is available at **P1**. The lower byte of the address bus is latched on the board for external use.

See Memory Map.

6.6 RS232 PORT

The **TX** and **RX** pins of the microcontroller have been converted to **RS232** levels. The RX and TX pins available on the connector **P1** and **J2** are thus at **RS232** line level.

The *'putchar'* function (*See LCD DISPLAY*) can be used to send characters to this port.

6.7 KEYPAD

The 16 key keypad is located at **8000H** in the memory map *(See Memory Map)*. When a key is pressed, an interrupt is generated on **XINT1** (if enabled). This interrupt is shared with the **RTC** *(See RTC)*. The routine *test_alarm ( )* can be used to determine the source of the interrupt.

6.8 DIP-SWITCH S1, S2, S3

- **S1** is used to switch between **MODE 0 (TRANSFER)**, during which **USER CODE** can be downloaded to the unit via **RS232** link, and **MODE 1 (EMULATE)**, in which the **USER CODE** is run.
- **S2** is used to select between two memory map options, **MAP 0** and **MAP 1**. *(See MEMORY MAP)*
- **S3** is used to enable or disable **T3**, the **WATCHDOG TIMER** in the 80C552 microcontroller.

NOTE: The watchdog timer cannot be enabled/disabled from software.

6.9 PHILIPS 80C552

Most of the pins of the **80C552 MICROCONTROLLER** are available at **P1**. *(See **96 PIN CONNECTOR P1**)* This is to enable the designer to develop with the full potential of the device. Refer to the **"PHILIPS 8051 FAMILY MICROCONTROLLER MANUAL"** for details.

6.10 RAM

S2 is used to select the code size required by the programmer. *(See **MEMORY MAP**)*

- **MAP=0** (S2 off) provides the designer with space for **16K** of **CODE** space, and **16K** of onboard **RAM DATA** space.
- **MAP=1** (S2 on) provides **32K** of **CODE** space and **0K** on board **RAM DATA** space.

6.11 MICROCONTROLLER CRYSTAL:

The crystal used is **11.0592 MHz** to accommodate the generation of preferred baud rates for the RS232 port.

6.12 MEMORY MAP

There are **TWO** possible memory maps. The designer selects the memory map implemented by setting **DIP SWITCH S2 (MAP)**. This determines how the **RAM** is used.

ADDRESS	SIZE	MAP= 0	MAP= 1
0000H	16Kb	CODE	CODE
4000H	16Kb	DATA	CODE
8000H	0.5Kb	Keypad	Keypad
8200H	0.5Kb	LCD	LCD
8400H	0.5Kb	SELECT S0	SELECT S0
8600H	14.5Kb	SELECT S1	SELECT S1
B000H	16Kb	SELECT S2	SELECT S2

6.13 SOFTWARE

The following routines are available for use by the developer:

For use with the LCD display, RS232 port, and keypad:

char putchar (char out);
char getkey ();
void init_lcd ();
void set_lcd (char command);
void init_sio0 ();
bit test_x1 ();

A data variable *io_stream* is used to set the destination of the *putchar* routine.

The *init_lcd* function sets up the LCD module for further use by the *putchar* routine, and the function *set_lcd* is used to modify the control register of this module. *(See LCD DISPLAY)*

The *getkey* routine waits for input from the keypad, and *test_x1* returns a bit indicating the origin of external interrupt 1.*(See KEYPAD)*

The function *init_sio0* sets up the RS232 port for communication at 19200 Baud. *(See RS232 PORT)*

The following functions are for use with the I^2C bus and Real Time Clock (RTC):

void init_sio1 (char own_slave_address);
*void start_master (char addr, char length, char *d_ptr);*
*bit start_sio1 (char addr, char length, char *d_ptr);*
bit init_RTC ();
void show_time (char pos);
void show_date (char pos);
bit test_x1 ();

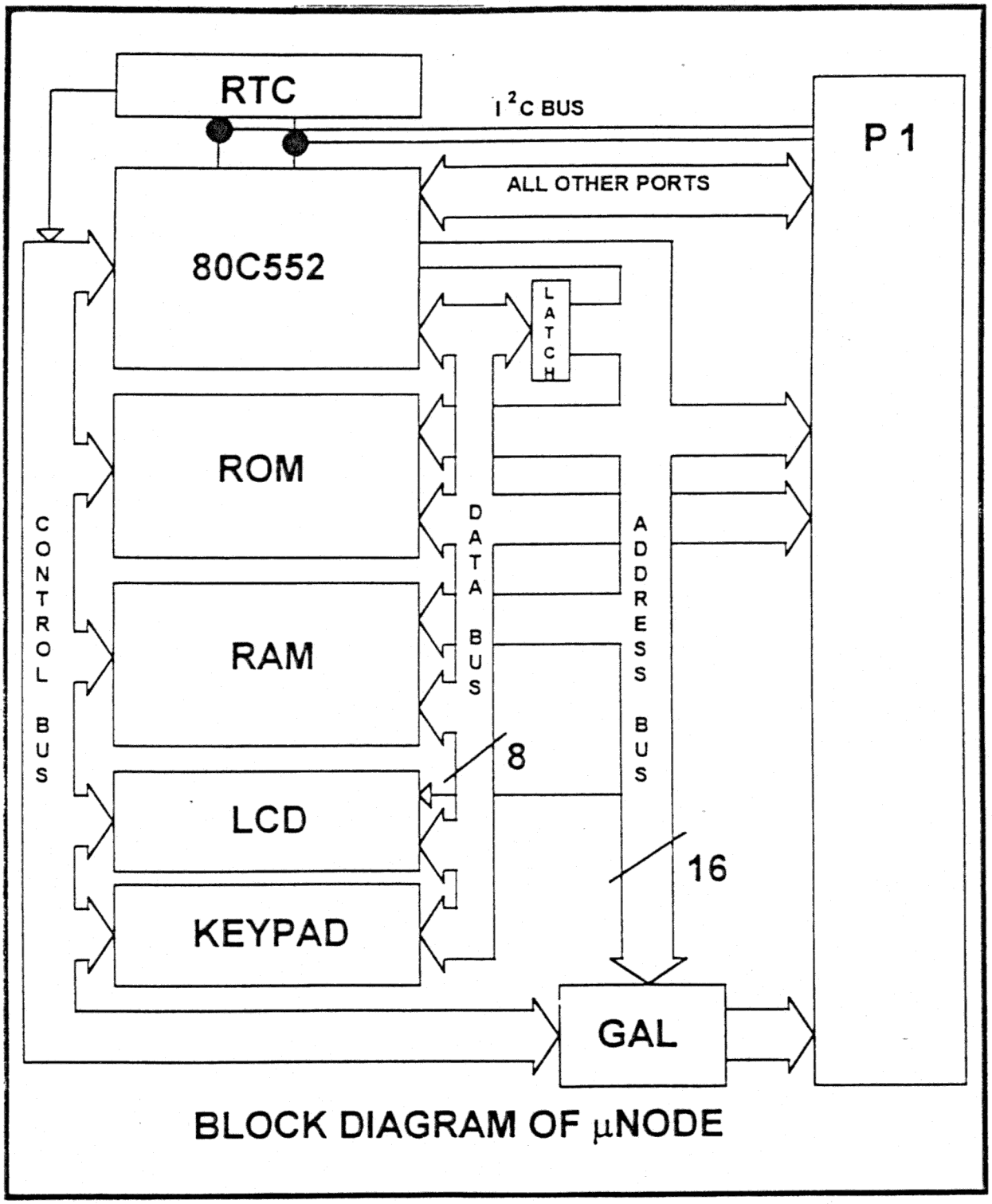
The function *init_sio1* sets up the unit's I^2C bus ready to receive as a slave with the slave address *own_slave_address* and also to respond to a general call address. The function *start_master* is an assembler routine that stores the called unit's address *(addr)* in the relevant special function register, loads the byte counter with the number of bytes to be transmitted *(length)*, and loads the data pointer used during the communication with the parameter *d_ptr*, and generates a start bit on the I^2C bus. The rest of the communication is handled by hardware and an interrupt routine in the module **sio1.obj**.

The function *start_sio1* is written in C and uses *start_master* to initialise communication, and then waits in a busy loop for the end of the communication by testing the global variable **NUMBYTMST**, which indicates the progress of the transmission or reception. This routine will time-out if there is no response (acknowledge) from the called unit.

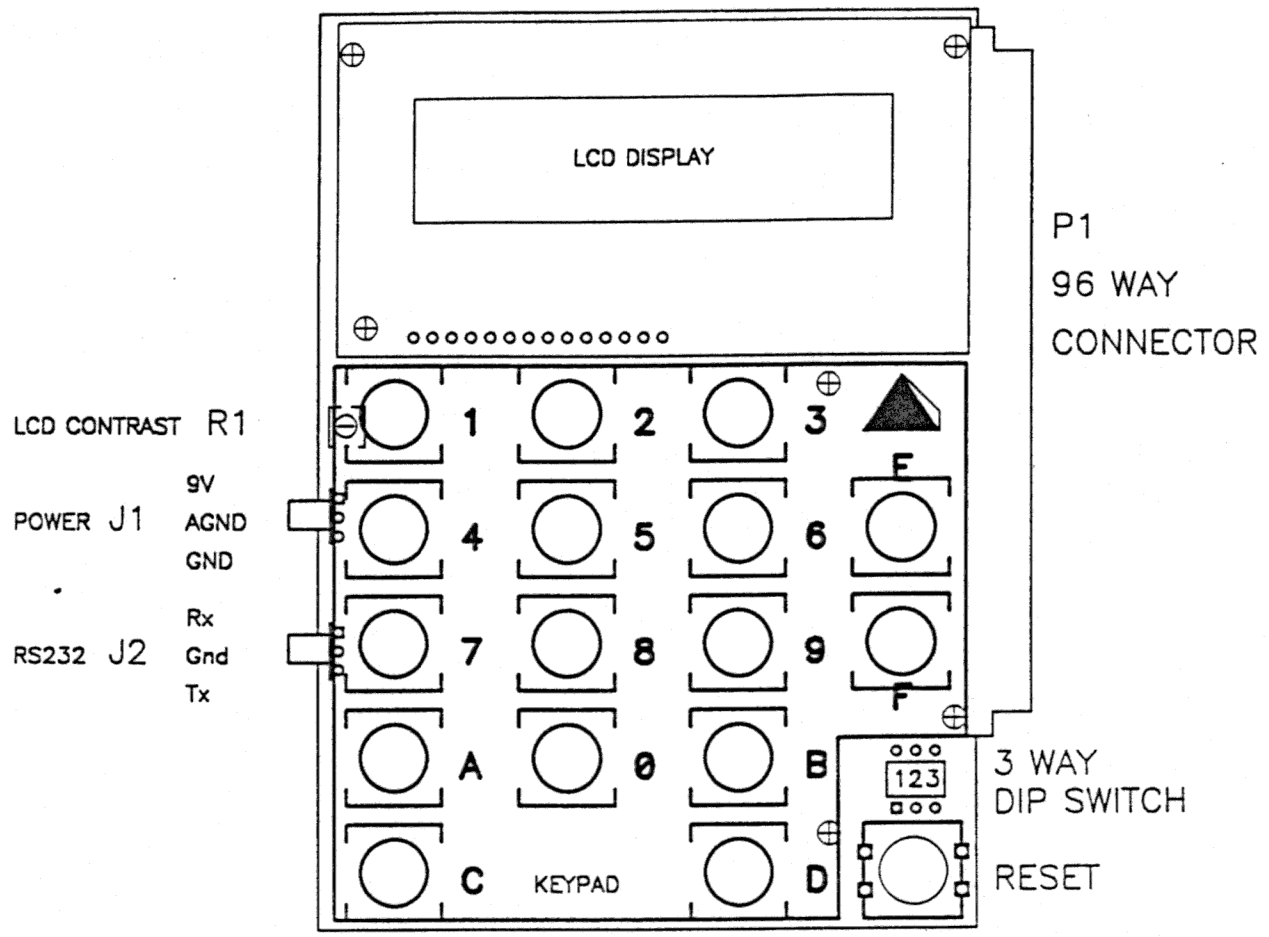
The function *init_RTC* is used to initialise the RTC and returns a bit indicating the presence of the **PCF8583** RTC. The functions *show_time* and *show_date* take a parameter *pos* which is the position on the LCD display that the date or time should be displayed.

The function *test_x1*, as mentioned above, reads, via the I^2C bus, the value in the control register of the RTC and returns a bit indicating the state of the alarm status flag.

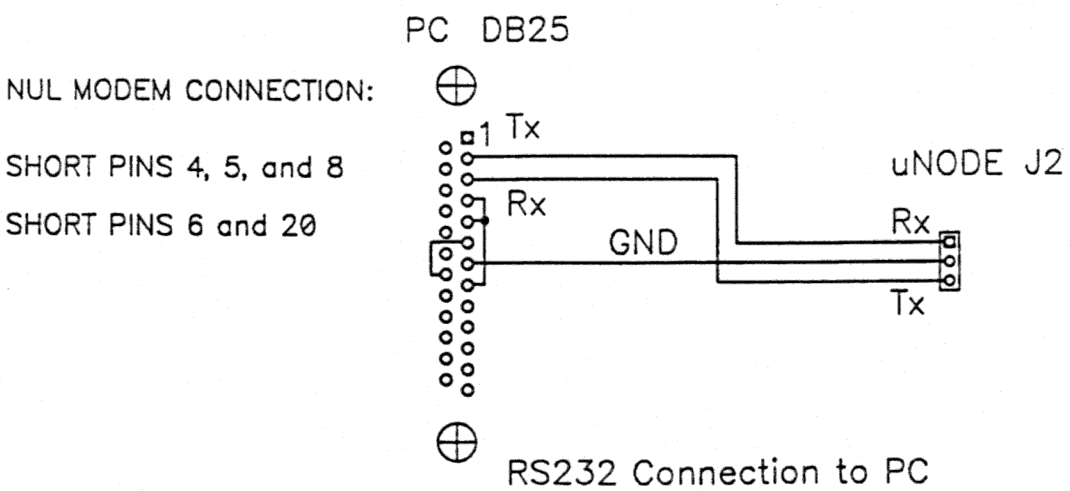
7. APPENDIX 1: BLOCK DIAGRAM



8. APPENDIX 2: LAYOUT



LAYOUT OF uNODE3



RS232 Connection to PC

9. APPENDIX 3: P1 PINOUT

Pin No	ROW A (Bottom)	ROW B (Mid)	ROW C (Top)
1	GND	ALE	P2.7 A15
2	AGND	/PSEN	P2.6 A14
3	VCC	/S0	P2.5 A13
4	GND	/S1	P2.4 A12
5	MODE	/S2	P2.3 A11
6	NC	/RD	P2.2 A10
7	NC	/WR	P2.1 A9
8	NC	RST	P2.0 A8
9	NC	P0.7 AD7	A7 (Decoded)
10	NC	P0.6 AD6	A6 (Decoded)
11	NC	P0.5 AD5	A5 (Decoded)
12	NC	P0.4 AD4	A4 (Decoded)
13	NC	P0.3 AD3	A3 (Decoded)
14	NC	P0.2 AD2	A2 (Decoded)
15	NC	P0.1 AD1	A1 (Decoded)
16	NC	P0.0 AD0	A0 (Decoded)
17	NC	GND	P1.7 SDA
18	NC	9V	P1.6 SCL
19	NC	STADC	P1.5 RT2
20	NC	P3.5 T1	P1.4 T2
21	NC	P3.4 T0	P1.3 CT3I
22	NC	P3.2 /INT0	P1.2 CT2I
23	NC	PW1	P1.1 CT1I
24	NC	PW0	P1.0 CT0I
25	NC	ADC0	P4.7 CMT1
26	NC	ADC1	P4.6 CMT0
27	NC	ADC2	P4.5 Cmsr5
28	NC	ADC3	P4.4 Cmsr4
29	NC	ADC4	P4.3 Cmsr3
30	NC	ADC5	P4.2 Cmsr2
31	Tx (RS232 level)	ADC6	P4.1 Cmsr1
32	Rx (RS232 level)	ADC7	P4.0 Cmsr0

NOTES: **Pins marked NC are reserved for future use.**

10. INDEX