

micromagic systems

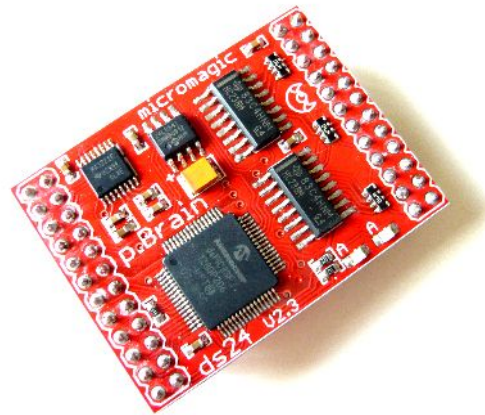
Animatronic & Puppet control systems for Film & Television

p.Brain-ds24 V2.3 Robot Controller - User Guide V1.0

Page 1 of 28

Contents

- (1) [Product Image](#)
- (1) [Description](#)
- (2) [Key Features](#)
- (2) [Board Layout](#)
- (3) [Connector CN1](#)
- (3) [Connector CN2](#)
- (4) [Micro Controller Schematic](#)
- (4) [Analogue Capture / Digital IO X 8](#)
- (5) [UART 1 / Serial Port 1](#)
- (5) [SPI Port / Digital IO X 4](#)
- (5) [LED Indicators](#)
- (6) [EEPROM Schematic](#)
- (6) [EEPROM Operation](#)
- (6) [RS232 Schematic](#)
- (6) [UART2 / Serial Port 2](#)
- (7) [PWM Multiplexor Schematic](#)
- (7) [PWM Multiplexor Operation](#)
- (9) [Code Examples](#)
- (9) [Software License Agreement](#)
- (9) [Processor Initialisation](#)
- (14) [Port & Hardware Initialisation](#)
- (18) [RS232 Communications](#)
- (20) [EEPROM Read & Write](#)
- (23) [Delay Routines](#)
- (24) [PWM Multiplexer](#)
- (28) [Legal \(Please Read First\)](#)



This controller is designed for people who have an understanding of PIC microcontrollers, associated peripherals and general PWM techniques, It is not suitable for the beginner or novice.

Description

The p.Brain-ds24 has been designed for use in advanced hobby robotics, where multiple R/C type servos need to be controlled with a standard PWM signal, such as hexapod robots. At the core of the p.Brain-ds24 is a microchip dsPIC33F 16bit micro controller with 128Kb programme flash, 8Kb RAM and 32Kb external EEPROM. This micro controller can be programmed using microchips ICD2 programmer and MPLAB IDE. Although the micro controller can be

P.Brain-ds24 (v2.3) User Guide

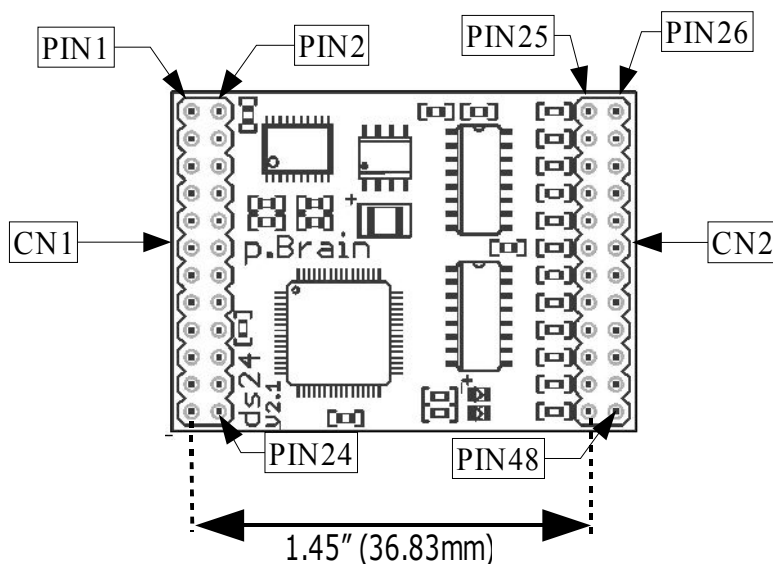
programmed in assembler, I suggest using a C compiler such as Hi-techs (www.htsoft.com) All programming examples are written for the Hi-tech compiler.

The p.Brain-ds24 has 24 PWM channels for R/C servo control, along with a variety of other peripherals such as UART's, I2C, SPI, ADC, Digital I/O, which are all brought out to two 24 pin, 0.1" pitch headers. The p.Brain-ds24 can then be plugged into a motherboard to distribute the PWM signals and peripherals accordingly such as micromagic's "p.Brain-SMB" or a user defined motherboard. The p.Brain-ds24 has an on board 3.3V regulator, and can be run from 4 to 9V on the VIN pin. There is a 3.3V output pin available which can supply 100mA for user peripherals.

Key Features

- Compact size (approx 44 x 32 mm)
- On Board 3.3V regulator
- dsPIC33FJ128GP206 16bit, 40Mips Processor
- 128Kb Programme Flash, 8Kb RAM, 64Kb External EEPROM
- External 8Mhz Ceramic Resonator or Internal 7.37Mhz R/C with PLL to 40MIPS
- UART1, Inverted TTL
- UART2, RS232, Inverted TTL or 2 x Digital IO
- I2C (Internally connected to 8Kbyte EEPROM)
- 4 x Digital IO with pull-up, or configured as SPI port
- 8 x Digital IO with pull-up, or 8 x 12 bit Analogue capture (ADC)
- 2 x On Board LED's
- Convenient 0.1" pitch pin headers for easy integration.

p.Brain LAYOUT



P.Brain-ds24 (v2.3) User Guide

Connector CN1 (24pin 0.1" (2.54mm) pitch)

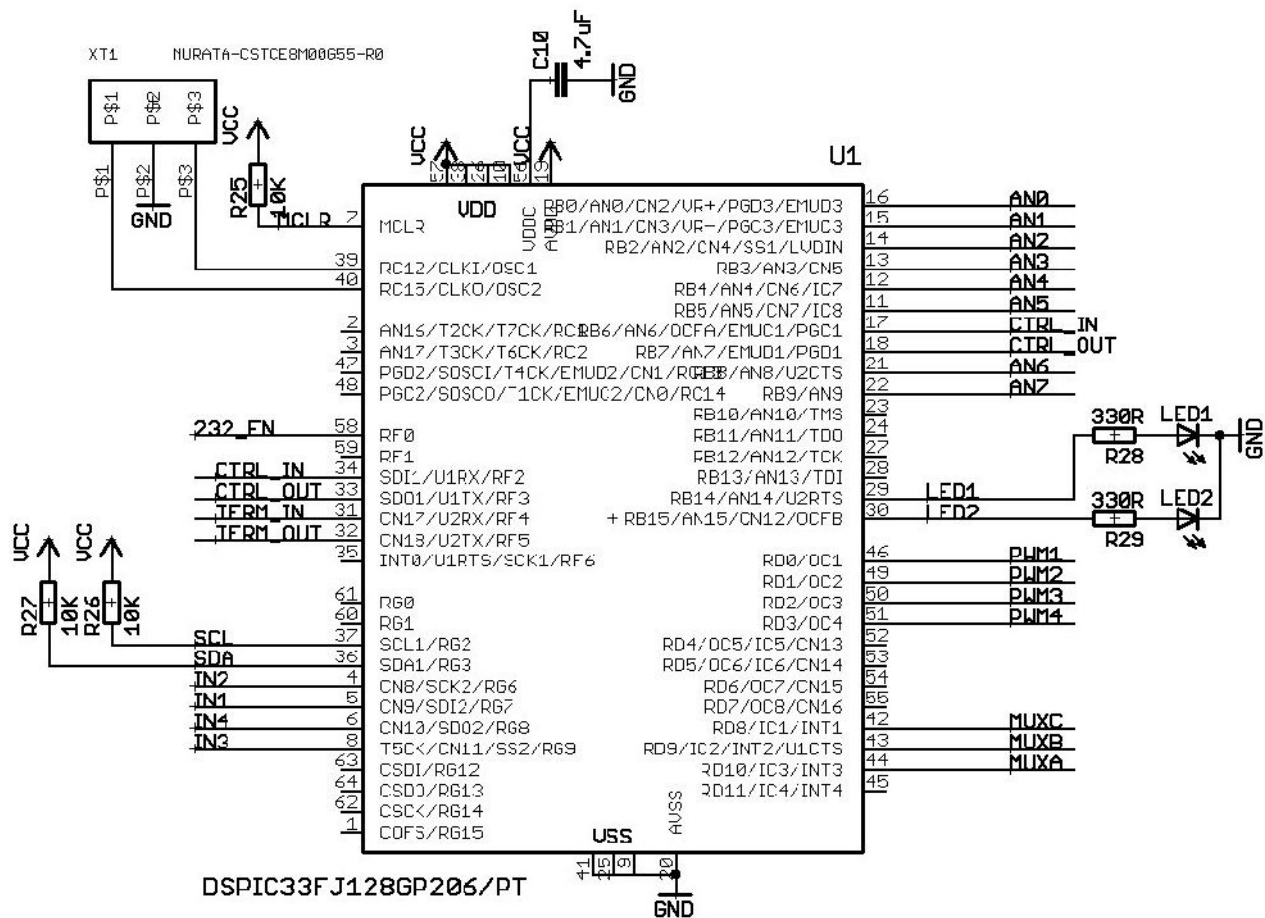
PIN	NAME	DESCRIPTION
1	VIN	Positive Power 4 to 9V DC
2	GND	Power Ground
3	U2_232_RX	UART2 RS232 Receive
4	U2_232_TX	UART2 RS232 Transmit
5	U2RX/DIG14	UART2 TTL Receive
6	U2TX/DIG15	UART2 TTL Transmit
7	U1RX/PGC	UART1 TTL Receive & ISSP Clock
8	U1TX/PGD	UART1 TTL Transmit & ISSP Data
9	SCL	I2C SCL
10	SDA	I2C SDA
11	VREG3.3V	Regulated 3.3V output (100ma MAX)
12	MCLR	ISSP MCLR
13	DIG0/SDI	Digital I/O 0 or SPI Data IN
14	DIG1/SCK	Digital I/O 1 or SPI Clock
15	DIG2/SS	Digital I/O 2 or SPI Select
16	DIG3/SDO	Digital I/O 3 or SPI Data Out
17	DIG4/ANA0	Digital I/O 4 or Analogue Capture 0 or Input Capture 0
18	DIG5/ANA1	Digital I/O 5 or Analogue Capture 1 or Input Capture 1
19	DIG6/ANA2	Digital I/O 6 or Analogue Capture 2
20	DIG7/ANA3	Digital I/O 7 or Analogue Capture 3
21	DIG8/ANA4	Digital I/O 8 or Analogue Capture 4
22	DIG9/ANA5	Digital I/O 9 or Analogue Capture 5
23	DIG10/ANA6	Digital I/O 10 or Analogue Capture 6
24	DIG11/ANA7	Digital I/O 11 or Analogue Capture 7

Connector CN2 (24pin 0.1" (2.54mm) pitch)

PIN	NAME	DESCRIPTION
1	PWM1	Pulse Width Modulated Output 1
2	PWM13	Pulse Width Modulated Output 13
3	PWM2	Pulse Width Modulated Output 2
4	PWM14	Pulse Width Modulated Output 14
5	PWM3	Pulse Width Modulated Output 3
6	PWM15	Pulse Width Modulated Output 15
7	PWM4	Pulse Width Modulated Output 4
8	PWM16	Pulse Width Modulated Output 16
9	PWM5	Pulse Width Modulated Output 5
10	PWM17	Pulse Width Modulated Output 17
11	PWM6	Pulse Width Modulated Output 6
12	PWM18	Pulse Width Modulated Output 18
13	PWM7	Pulse Width Modulated Output 7
14	PWM19	Pulse Width Modulated Output 19
15	PWM8	Pulse Width Modulated Output 8
16	PWM20	Pulse Width Modulated Output 20
17	PWM9	Pulse Width Modulated Output 9
18	PWM21	Pulse Width Modulated Output 21
19	PWM10	Pulse Width Modulated Output 10
20	PWM22	Pulse Width Modulated Output 22
21	PWM11	Pulse Width Modulated Output 11
22	PWM23	Pulse Width Modulated Output 23
23	PWM12	Pulse Width Modulated Output 12
24	PWM24	Pulse Width Modulated Output 24

P.Brain-ds24 (v2.3) User Guide

Micro controller Schematic



This schematic displays the micro controller connections to the PWM block, EEPROM, LED's and various ports on CN1 which are described in more detail in the following pages.

Analogue Capture / Digital IO x 8

There are eight p.Brain pins that can be configured as either analogue capture, digital IO or up to two input capture channels:

PORT PIN NAME	p.Brain NAME	DESCRIPTION
RB4/AN4/CN6/IC7	DIG4/AN0	Digital IO 4, Ana Chan 0, Input Capture 0
RB5/AN5/CN7/IC8	DIG5/AN1	Digital IO 5, Ana Chan 1, Input Capture 1
RB2/AN2/CN4	DIG6/AN2	Digital IO 6, Ana Chan 2
RB3/AN3/CN5	DIG7/AN3	Digital IO 7, Ana Chan 3
RB0/AN0/CN2	DIG8/AN4	Digital IO 8, Ana Chan 4
RB1/AN1/CN3	DIG9/AN5	Digital IO 9, Ana Chan 5
RB9/AN9	DIG10/AN6	Digital IO 10, Ana Chan 6
RB8/AN8	DIG11/AN7	Digital IO 11, Ana Chan 7

As can be seen in the table above, the assigned analogue pin names on the p.Brain header do not necessarily match the actual analogue channel on the dsPIC. This can be easily resolved in software with a simple lookup table. Also, DIG4/AN0 and DIG5/AN1 can be configured as input capture ports if needed. DIG4 thru DIG9 have a programmable weak pull-up and change

P.Brain-ds24 (v2.3) User Guide

notification interrupt capabilities. DIG10 and DIG11 have no internal pull up resistors and may require external pull up resistors.

For further information on analogue capture, input capture and change notification pins, please see microchips data sheet at www.microchip.com or the supplied sample code.

UART1 / Serial Port 1

Serial port 1 can only operate at Inverted TTL levels. In order to use the port the following micro controller pins are required:

PORT PIN NAME	p.Brain NAME	DESCRIPTION
SDI1/U1RX/ RF2	CTRL_IN	TTL Data In
SDO1/U1TX/ RF3	CTRL_OUT	TTL Data Out

To use serial port 1, RF3 needs to be configured as an output, and RF2 as an input. Also the associated UART registers require configuration. CTRL_IN and CTRL_OUT are also connected to the micro controllers PGC1 and PGD1 respectively, which are used during ISSP. If you are designing your own motherboard for the p.Brain, bare in mind if you intend to use UART1 to connect to another device, you will need to disconnect the U1RX & U1TX pins on CN1 for ISSP.

For further information on accessing ports or using the UART please refer to the data sheet at www.microchip.com or see the supplied [code samples](#).

SPI Port / Digital IO x 4

PORT PIN NAME	p.Brain NAME	DESCRIPTION
CN9/SDI2/ RG6	DIG0/SDI	Digital I/O 0 or SPI Data IN
CN8/SCK2/ RG7	DIG1/SCK	Digital I/O 1 or SPI Clock
CN11/SS2/ RG9	DIG2/SS	Digital I/O 2 or SPI Select
CN10/SD02/ RG8	DIG3/SDO	Digital I/O 3 or SPI Data Out

These four pins can be configured as Digital I/O or as an SPI port for interfacing to external peripherals. Each of these pins has a programmable weak pull-up and change notification interrupt capabilities. For further information on the SPI port or change notification pins, please see microchips data sheet at www.microchip.com or the supplied [code samples](#).

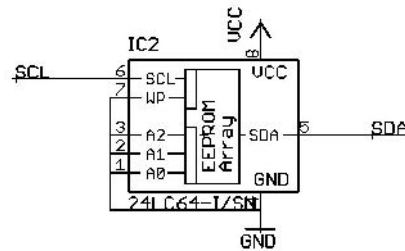
LED Indicators

PORT PIN NAME	SCHEMATIC NAME	DESCRIPTION
RB14	LED1	Red Led
RB15	LED2	Green Led

There are two LED's on the p.Brain module accessible by the user. These LED's have there anode's connected to the port pins defined above. To turn on an LED, enable the relative port bit as an output, and set the port pin high. For further information on accessing ports please see microchips data sheet at www.microchip.com or the supplied [code samples](#).

P.Brain-ds24 (v2.3) User Guide

EEPROM Schematic

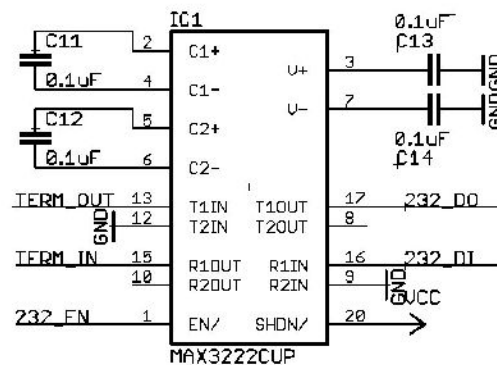


EEPROM Operation

PORT PIN NAME	EEPROM PIN	DESCRIPTION
RG2/SCL1	SCL	Serial Clock
RG3/SDA1	SDA	Serial Data

The on board EEPROM is a microchip 24LC64 device which can be accessed using the pins in the above table. The write protect pin and address pins on the EEPROM have been tied to GND, which gives a base address of 0xA0. Communication with the device is achieved using the Synchronous Serial Port 1 in I2C master mode. RG2 & 3 must be configured as inputs, and the SSP port configured as I2C master mode. For further information on accessing ports or the 24LCxx series EEPROM's please see microchips data sheet at www.microchip.com or the supplied *code samples*.

RS232 Schematic



The RS232 transceiver is connected to UART2 on the PIC micro controller. See below for operation information.

UART2 / Serial Port 2

Serial port 2 can be configured as either RS232 or Inverted TTL. In order to use the port the following micro controller pins are required:

PORT PIN NAME	p.Brain NAME	DESCRIPTION
CN17/U2RX/ RF4	TERM_IN	TTL Data In
CN18/U2TX/ RF5	TERM_OUT	TTL Data Out

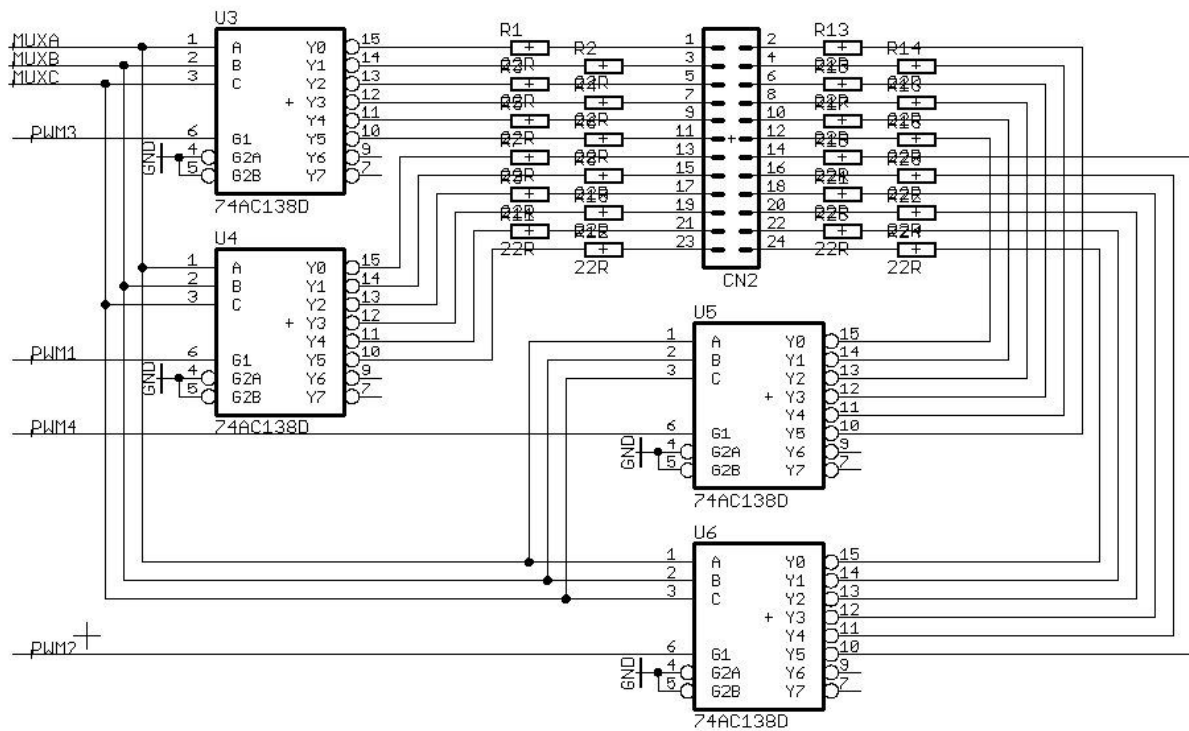
P.Brain-ds24 (v2.3) User Guide

RF0	232_EN	RS232 Enable
-----	--------	--------------

To use serial port 2, RF0,4 need to be configured as outputs, and RF5 as an input. Also the associated UART registers require configuration. By clearing the 232_EN pin, the RS232 transceiver is enabled which means the TERM_IN pin is driven by the transceiver. When the 232 transceiver is enabled, the U2RX pin must not be connected to any other source, however, the U2TX pin will still transmit inverted TTL data. If this UART is not required, these pins can be configured as Digital I/O DIG14 = RF4 & DIG15 = RF5.

For further information on accessing ports or using the UART please refer to the data sheet at www.microchip.com or see the supplied [code samples](#).

PWM Multiplexor Schematic



PWM Multiplexor Operation

In order to get 24 channels of PWM from the dsPIC, four 1 to 8 channel multiplexors are used, however, only the first 6 channels are used on each multiplexor. Each multiplexor has 3 address lines A, B and C and a gait line which is tied to one of the PWM outputs from the micro controller, PWM1, PWM2, PWM3 and PWM4. There are two more gait lines on the multiplexors which are tied to ground and so are not used. With this arrangement, four PWM channels are driven at a time, one on each multiplexor. Once the PWM cycle is complete for the current four channels, the address select lines are incremented and the next four channels can be driven. This is repeated six times to give the full 24 channels.

P.Brain-ds24 (v2.3) User Guide

The average R/C PWM signal is 1 to 2ms long, and repeats 50 times per second. With the multiplexor arrangement we need to output 6 PWM signals sequentially, so if we were to say the longest PWM time is 2ms then the maximum time for 6 channels would be 12ms which would give a maximum refresh rate of 83Hz. Given that there would be some time required to service interrupts and setup registers, a more realistic refresh rate would be 80Hz.

- FPS = Servo Refresh Rate in Hz
- MAXPWM = Maximum PWM length in mili seconds
- CHANS = Number of sequential channels (remember, each sequential channel drives 4 PWM outputs)

$$\text{FPS} = 1 / (\text{MAXPWM} * \text{CHANS}) = 1 / (2\text{m} * 6) = 83.33\text{Hz}$$

This of course gives the maximum refresh rate for all 24 channels at 2ms PWM pulse width. With less channels, higher refresh rates can be achieved, however, 50 or 60hz is sufficient.

You will notice in the multiplexor schematic that PWM1 does not correspond to the first multiplexor, and the multiplexor outputs are not necessarily in ascending order in reference to CN2 connector. This is due to PCB routing constraints and may seem confusing at first, however, all PWM outputs can be re-mapped in software to the correct CN2 output pin. (see example code)

PORT PIN NAME	SCHEMATIC NAME	DESCRIPTION
RD0 /OC1	PWM1	Output Compare Register PWM output 1
RD1 /OC2	PWM2	Output Compare Register PWM output 2
RD2 /OC3	PWM3	Output Compare Register PWM output 3
RD3 /OC4	PWM4	Output Compare Register PWM output 4
RD10 /IC3/INT3	MUXA	Multiplexor Block Address A (LSB)
RD9 /IC2/INT2/U1CTS	MUXB	Multiplexor Block Address B
RD8 /IC1/INT1	MUXC	Multiplexor Block Address C (MSB)

In order to access the multiplexor block, RD0,1,2,3,8,9,10 must be configured as outputs in the port configuration registers. Also output compare registers 1 thru 4 must be configured for PWM output. For further information on accessing ports, and output compare registers please see microchips data sheet at www.microchip.com or the supplied [code samples](#).

P.Brain-ds24 (v2.3) User Guide

Code Examples

All code examples are written for the Hitec dsPICC compiler. All code examples are subject to the following license agreement.

Software License Agreement

The software supplied herewith by micromagic systems limited (the "Company") for its p.Brain controller is intended and supplied to you, the Company's customer, for use solely and exclusively on micromagic systems p.Brain controller products. The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

Processor Initialisation

In the following code example you will see how to initialize the processor oscillator and configuration bits. The processor is configured to run at 32Mhz using the 8Mhz external resonator. The main programme then uses functions for port initialisation, serial communications, time delays and EEPROM access. These functions are defined in the code examples sections below.

```
// *****
// DEFINITIONS
// *****

#define PIC_CLK      32000000          // CLOCK SPEED
#define FCY          (PIC_CLK/2)     // INSTRUCTION SPEED

// SOME USEFULL TYPE DEFINITIONS
typedef BYTE        unsigned char
typedef WORD        unsigned int

// *****
// HEADER FILES
// *****

#include <htc.h>
#include <stdio.h>
#include <stdlib.h>

// *****
// CONST STRING DATA
// *****

const char build_title[] = {"Example"};
const char build_author[] = {"micromagic systems ltd"};
const char build_date[] = {__DATE__};
const char build_time[] = {__TIME__};
const char build_vers[] = {"1.0"};
```

P.Brain-ds24 (v2.3) User Guide

```

// *****
// CONFIGURATION BITS
// *****

__CONFIG( FOSCSEL, OSCPLL & IESODIS & TEMPEN ); // PLL ON
__CONFIG( FOSC, POSCXT & FCKSMDIS ); // EXTERNAL XT OSCILATOR
__CONFIG( FWDT, WDTDIS ); // WATCHDOG TIMER DISABLED
__CONFIG( FPOR, PWRT128 ); // POWERUP TIMER = 128ms
__CONFIG( FGS, GCPU ); // NO CODE PROTECT

// *****
// PSECT. NOT NECESSARY BUT CAN BE USEFULL FOR DEBUGGING
// *****

#pragma psect text=main

// *****
// PORT DEFINITIONS
// *****

#define PORTB_SETUP 0b0000001111111111
#define PORTC_SETUP 0b0000000000000000
#define PORTD_SETUP 0b0000000000000000
#define PORTF_SETUP 0b0000000000010100
#define PORTG_SETUP 0b0000001111001100

#define PORT_CNPU1 0b0000111111111100
#define PORT_CNPU2 0b0000000000000000

#define RED_LED RB14 // LED1
#define GRN_LED RB15 // LED2

#define PWM1 RD0 // USED BY PWM BLOCK
#define PWM2 RD1
#define PWM3 RD2
#define PWM4 RD3

#define MUX_A0 RD10 // PWM BLOCK MULTIPLEXOR PINS
#define MUX_A1 RD9
#define MUX_A2 RD8

#define EE_WP RD7 // EEPROM WRITE PROTECT
#define SCL RG2 // I2C SERIAL CLOCK
#define SDA RG3 // I2C SERIAL DATA

#define RS232_EN RF0 // RS232 ENABLE

#define JP0 !RG8
#define JP1 !RG9
#define JP2 !RG6
#define JP3 !RG7

// *****
// SERIAL PORT DEFINITIONS AND GLOBAL VARIABLES
// *****

// THE FOLLOWING BAUD RATE VALUES ARE FOR CLOCK SPEEDS OF 32 MHZ
// PLEASE CONSULT THE dsPIC33F DATA SHEET FOR BAUD RATE CALCULATIONS
// OR USE THE Pic Baud Calculator TOOL FROM MICROMAGIC SYSTEMS.
// NOTE: DO NOT USE BRGH=1 SETTING ON THIS PROCESSOR, IT DOES NOT WORK
// ON EARLY SILICON REVISIONS

#define B9600 103 // 0.16% ERROR
#define B19200 51 // 0.16% ERROR
#define B38400 25 // 0.16% ERROR
#define B57600 16 // 2.12% ERROR

```

P.Brain-ds24 (v2.3) User Guide

```

#define B115200 8 // -3.55% ERROR

// MOST PC BASED UARTS WILL ACCEPT BAUD RATE ERRORS OF +/- 3.55%, HOWEVER
// IF YOUR HOST SYSTEM ALSO HAS A BAUD RATE ERROR, YOU MAY EXPERIENCE PROBLEMS.

#define TX2_BUFFER_SIZE 32 // MUST BE 2^n EG. 2,4,8,16,32,64...
#define RX2_BUFFER_SIZE 32 // MUST BE 2^n EG. 2,4,8,16,32,64...

volatile BYTE gTx2Buffer[ TX2_BUFFER_SIZE ];
volatile WORD gTx2Cntr, gTx2Ptr;
volatile bit gTx2BufferFull;

volatile BYTE gRx2Buffer2[ RX2_BUFFER_SIZE ];
volatile WORD gRx2Cntr2, gRx2Ptr;
volatile bit gRx2BufferFull;

// *****
// PWM DEFINITIONS & GLOBAL VARIABLES
// *****

#define OC_TIMER_PRE_SCALER 8 // OUTPUT COMPARE TIMER PRE SCALER, USED BY TIMER 3.
#define PWM_MUL (double) (FCY/OC_TIMER_PRE_SCALER/1000000) // CLOCK CYCLES PER USECOND
#define PWM_RANGE (WORD) (PWM_MUL * 1000) // PWM RANGE IN uS
#define PWM_MID (WORD) (PWM_MUL * 1500) // PWM CNETRE IN uS
#define PWM_MIN (WORD) (PWM_MID - (PWM_RANGE/2) ) // PWM MIN
#define PWM_MAX (WORD) (PWM_MID + (PWM_RANGE/2) ) // PWM MAX

#define SERVO_MAX (int) (PWM_RANGE) // USEFULL DEFINITIONS FOR
#define SERVO_MID (int) (PWM_RANGE/2) // SERVO TRAVEL RANGE
#define SERVO_MIN 0 //

BYTE gMuxAddress; // PWM MULTIPLEXOR ADDRESS, 0 to 5
BYTE gServoEnable[24]; // SERVO ENABLE ARRAY, NON ZERO + ENABLE PWM OUTPUT
WORD gServoPos[24]; // SERVO POSITION ARRAY.
// CONSTANT SERVO REMAP ARAY
// THIS ARRAY IS THE CONFIGURATION FOR THE p.Brain-SMB
// MOTHERBOARD.
const BYTE cServoRemap[24] = {16,3,15,2,14,1,13,0,12,23,11,22,10,21,9,20,8,19,5,17,4,7,18,6};

// *****
// USEFULL DELAY MACROS
// *****

#define dly1u NOP();NOP();NOP();NOP();NOP();NOP();NOP();NOP()
#define dly500n NOP();NOP();NOP();NOP()

// *****
// FUNCTION PROTOTYPES
// *****

void Initialise( void );

void putch( char pC ); // SERIAL ROUTINES
char getch( void );
char getche( void );
int kbhit( void );
void interrupt tx2isr( void );
void interrupt rx2isr( void );

void i2cWaitForIdle( void ) // EEPROM ROUTINES
void i2cStart( void )
void i2cRepStart( void )
void i2cStop( void )
BYTE i2cRead( BYTE pAck )
WORD i2cWrite( BYTE pData )
void my_eeprom_write_byte( WORD pAddr, BYTE pData )
BYTE my_eeprom_read_byte( WORD pAddr )

```

P.Brain-ds24 (v2.3) User Guide

```

void    my_eeprom_write_buffer( WORD  pAddr, BYTE *pData )

void    DelayUs( WORD pDelay )          // DELAY ROUTINES
void    DelayMs( WORD pDelay )

// *****
// MAIN CODE
// *****

void    main()
{
BYTE    lT;
int     lI;

//     SETUP THE CLOCK PLL (PHASE LOCKED LOOP).
//     THERE ARE CERTAIN RESTRICTIONS TO THE CLOCK SPEED AT EACH STAGE
//     OF THE PLL:
//     PLLPRE : CLK MUST BE BETWEEN 0.8 > 8.0Mhz HERE
//     PLLFBD : ( MUL FACTOR ) CLK MUST BE 100 > 200 Mhz Here
//     PLLPOST : PLL OUTPUT CLK MUST BE 12.5 > 80 Mhz.

                //     WITH OUR INPUT CLOCK OF 8Mhz
CLKDIV = 0;      //     PLLPRE = /2 = 4Mhz
PLLFBD = 0x001e; //     PLL x 32 = 128Mhz
PLLPOST1 = 1;   //     PLLPOST = /4 = 32Mhz

While( !LOCK ) //     NOW WE SHOULD WAIT FOR THE PLL TO LOCK.
    continue;

Initialise(); //     CONFIGURE OUR PORTS & HARDWARE

GRN_LED = 1; //     MAIN PROGRAMME BEGINS HERE

U2RXIE = 1; //     ENABLE UART2 RECEIVER INTERRUPTS
RS232_EN = 0; //     ENABLE RS232 TRANSCEIVER

                //     DISPLAY MESSAGE
printf("Hello World!\r\n\r\nPress SPACE bar to continue.");

while( 1 ) //     WAIT FOR SPACE BAR
{
    if( kbhit() ) //     WAS KEYBOARD HIT?
    {
        lC = getche(); //     GET & CHECK KEY
        if( lC == ' ' )
            break;
    }
}

lI = 0;
while(1) //     LOOP
{
    if( TlIF )
    {
        TlIF = 0; //     RESET T1 FLAG
        RED_LED ^= 1; //     FLASH LED

        //     FILL THE SERVO BUFFER WITH SOME SERVO DATA
        //     THIS ROUTINE JUST MOVES THE SERVOS BACK AND FORWARD
        //     THROUGH THEIR FULL RANGE.

        if( gServoBuffer[0] == SERVO_MAX )
            lI = -1;
        else if( gServoBuffer[0] == SERVO_MIN )
            lI = 1;
    }
}

```

P.Brain-ds24 (v2.3) User Guide

```
for( lT = 0; lT < 24; lT++ )
    gServoBuffer[ lT ] += lI;

//          COPY & RE-MAP SERVO DATA TO gServoPos BUFFER
DistributeServoOutputs( &gServoBuffer );

StartPWMFrame();          //          START PWM FRAME
}
}
```

P.Brain-ds24 (v2.3) User Guide

Port & Hardware Initialisation

The following example demonstrates the initialization of the processor ports and hardware configuration registers. This function is called "Initialise()"

```

void Initialise( void )
{
// *****
// PORT SETUP
// *****

PORTB = 0;
PORTC = 0;
PORTD = 0;
PORTE = 0;
PORTG = 0;

TRISB = PORTB_SETUP; // CONFIGURE PORT FOR I/O
TRISC = PORTC_SETUP;
TRISD = PORTD_SETUP;
TRISF = PORTE_SETUP;
TRISG = PORTG_SETUP;

CNEN1 = 0; // CHANGE NOTIFICATION ENABLE REGISTER
CNEN2 = 0;
CNPU1 = PORT_CNPU1; // CHANGE NOTIFICATION PULL-UPS ENABLE REGISTER
CNPU2 = PORT_CNPU2;

ODCD = 0; // ALL PORTS ARE NOT OPEN DRAIN
ODCF = 0;
ODCG = 0;

// *****
// INTERRUPT SETUP
// *****

INTCON1 = 0; // CLEAR INTERRUPT CONTROL REGISTERS
INTCON2 = 0;

IFS0 = 0; // CLEAR ALL INTERRUPT REQUEST FLAG REGISTERS
IFS1 = 0;
IFS2 = 0;
IFS3 = 0;
IFS4 = 0;

IEC0 = 0; // CLEAR ALL INTERRUPT ENABLE CONTROL REGISTERS
IEC1 = 0;
IEC2 = 0;
IEC3 = 0;
IEC4 = 0;

IPC0 = 0; // CLEAR ALL INTERRUPT PRIORITY REGISTERS
IPC1 = 0;
IPC2 = 0;
IPC3 = 0;
IPC4 = 0;
IPC5 = 0;
IPC6 = 0;
IPC7 = 0;
IPC8 = 0;
IPC9 = 0;
IPC10 = 0;
IPC11 = 0;
IPC12 = 0;
IPC13 = 0;
IPC14 = 0;

```

P.Brain-ds24 (v2.3) User Guide

```

IPC15 = 0;
IPC16 = 0;
IPC17 = 0;

// *****
//   TIMER MODULES SETUP
// *****

//   TIMER 1 CONFIGURED AS MAIN LOOP TICK TIMER

T1CON = 0;           //
T1CKPS0 = 1;        //   PRESCALER 1:8
PR1 = TIMERVERAL;
T1ON = 1;           //   T1 ON

//   T3 USED BY OC/PWM GENERATION.
//   USED AS INTERRUPT WHEN ALL FOUR OC/PWM OUTPTUS ARE DISABLED
//   SO AS TO CREATE NEXT PWM FRAME.

T3CON = 0;
T3CKPS0 = 1;        //   PRESCALER 1:8
PR3 = 0xffff;
T3IP0 = 1;          //   T3 INTERRUPT PRIORITY 3
T3IP1 = 1;
T3IP2 = 0;
T3ON = 1;           //   T3 ON

T2CON = 0;          //   UNUSED TIMERS
T4CON = 0;
T5CON = 0;
T6CON = 0;
T7CON = 0;
T8CON = 0;
T9CON = 0;

// *****
//   INPUT CAPTURE SETUP
// *****

IC1CON = 0;         //   ALL INPUT CAPTURES OFF
IC2CON = 0;
IC3CON = 0;
IC4CON = 0;
IC5CON = 0;
IC6CON = 0;
IC7CON = 0;
IC8CON = 0;

// *****
//   OUTPUT COMPARE SETUP
// *****

OC1CON = 0;         //   OUTPUT COMPARE 1
OC1_TSEL = 1;       //   USE TIMER3
OC1IP0 = 0;         //   INTERRUPT PRIORITY 2
OC1IP1 = 1;
OC1IP2 = 0;

OC2CON = 0;         //   OUTPUT COMPARE 2
OC2_TSEL = 1;       //   USE TIMER3
OC2IP0 = 0;         //   INTERRUPT PRIORITY 2
OC2IP1 = 1;
OC2IP2 = 0;

OC3CON = 0;         //   OUTPUT COMPARE 3
OC3_TSEL = 1;       //   USE TIMER3
OC3IP0 = 0;         //   INTERRUPT PRIORITY 2

```

P.Brain-ds24 (v2.3) User Guide

```

OC3IP1 = 1;
OC3IP2 = 0;

OC4CON = 0;           //      OUTPUT COMPARE 4
OC4_TSEL = 1;        //      USE TIMER3
OC4IP0 = 0;          //      INTERRUPT PRIORITY 2
OC4IP1 = 1;
OC4IP2 = 0;

OC5CON = 0;          //      UNUSED OUTPUT COMARE
OC6CON = 0;
OC7CON = 0;
OC8CON = 0;

//      *****
//      SPI MODULE SETUP
//      *****

SPI1STAT = 0; //      UNUSED
SPI2STAT = 0;

//      *****
//      I2C MODULE SETUP
//      *****

I2C1CON = 0;         //      I2C1 CONFIGURATION
I2C1STAT = 0;
I2C1_EN = 1;         //      ENABLE
//      CALCULATE BIT RATE
I2C1BRG = ((FCY/400000) - (FCY/1111111)) - 1;
I2C1_BCL = 0;        //      CLEAR BUS COLLISION FLAG
I2C1MSK = 0;         //      CLEAR MASK
I2C1ADD = 0;         //      CLEAR ADDRESS

//      *****
//      DATA CONVERTER INTERFACE MODULE SETUP
//      *****

DCICON1 = 0; //      UNUSED

//      *****
//      ADC
//      *****

AD1CON1 = 0; //      UNUSED
AD1CON2 = 0;
AD1CON3 = 0;
AD1CON4 = 0;
AD1CHS123 = 0;
AD1CHS0 = 0;

AD1PCFGL = 0xffff; //      ADC PORT PIN CONFIG: 0 = ADC, 1 = DIG.
AD1PCFGH = 0xffff; //      ADC PORT PIN CONFIG: 0 = ADC, 1 = DIG.

//      *****
//      UART1 SETUP
//      *****

U1MODE = 0;         //      UART1 CONFIGURATION
U1_SPEN = 1;        //      ENABLE
U1STA = 0;          //      CLEAR STATUS REGISTER
U1_TXEN = 1;        //      ENABLE TX
U1_WAKE = 1;        //      UART WILL WAKE PROCESSOR FROM IDLE MODE
U1BRG = B38400;     //      SET BAUD RATE

```


P.Brain-ds24 (v2.3) User Guide

```

U1TXIP0 = 1;          //      TX INTERRUPT PRIORITY 1
U1TXIP1 = 0;
U1TXIP2 = 0;
U1_TXISEL0 = 1;      //      0 = INT UPON TRANSFER TO SHIFT REG.
                        //      1 = INT UPON FIFO EMPTIED.
U1RXIP0 = 0;          //      RX PRIORITY 6
U1RXIP1 = 1;
U1RXIP2 = 1;

U1_RCISEL0 = 0;      //      INT UPON BYTE RECEIVED
U1_RCISEL1 = 0;      //

//      *****
//      UART 2 SETUP
//      *****

U2MODE = 0;          //      UART2 CONFIGURATION
U2_SPEN = 1;          //      ENABLE
U2_STA = 0;           //      CLEAR STATUS REGISTER
U2_TXEN = 1;          //      TX ENABLE
U2_WAKE = 1;          //      UART WILL WAKE PROCESSOR FROM IDLE MODE
U2BRG = B38400;      //      SET BAUD RATE

U2TXIP0 = 1;          //      TX INTERRUPT PRIORITY 1
U2TXIP1 = 0;
U2TXIP2 = 0;
U2_TXISEL0 = 1;      //      0 = INT UPON TRANSFER TO SHIFT REG.
                        //      1 = INT UPON FIFO EMPTIED.

U2RXIP0 = 0;          //      RX INTERRUPT PRIORITY 6
U2RXIP1 = 1;
U2RXIP2 = 1;

U2_RCISEL0 = 0;      //      INT UPON BYTE RECEIVED
U2_RCISEL1 = 0;

//      SWITCH OFF ALL UNUSED PERIPHERALS. 0 = ON 1 = OFF
//      T5MD T4MD T3MD T2MD T1MD QEIMD PWMMD DCIMD
//      I2C1MD U2MD U1MD SPI2MD SPI1MD C2MD C1MD AD1MD

PMD1 = 0b1000011100011110;

//      IC8MD IC7MD IC6MD IC5MD IC4MD IC3MD IC2MD IC1MD
//      OC8MD OC7MD OC6MD OC5MD OC4MD OC3MD OC2MD OC1MD

PMD2 = 0b1111111111110000;

//      T9MD T8MD T7MD T6MD - - - - - I2C2MD AD2MD

PMD3 = 0b111111111111111;

}

```

P.Brain-ds24 (v2.3) User Guide

RS232 Communications

In this example interrupt driven communications will be configured for the RS232 serial port on UART2. The following are high level serial port functions:

```

//*****
// PUTCHE USED BY PRINTF COMMANDS TO TERMINAL PORT
//*****

void    putch( char pC )
{
while( gTx2Cntr == TX2_BUFFER_SIZE ) //    WAIT FOR SPACE IN BUFFER
    {
        gTX2BufferFull = 1;
        U2TXIE = 1;
    }

U2TXIE = 0; //    DISABLE TX INTERRUPTS
NOP();
gTx2Buffer[ gTx2Ptr ] = pC; //    STORE CHAR IN FIFO BUFFER
gTx2Ptr++; //    INC BUFFER POINTER
gTx2Ptr &= TX2_BUFFER_SIZE-1; //    AND MASK.
gTx2Cntr++; //    INC BUFFER COUNT
U2TXIE = 1; //    ENABLE TX INTERRUPTS
}

//*****
// GETCHE GET BYTE FROM TERMINAL SERIAL PORT
//*****

char    getch( void )
{
char    lC;

while( !gRx2Cntr ) //    WAIT FOR A BYTE IN THE BUFFER
    continue;

U2RXIE = 0; //    DISABLE RX INTERRUPT
NOP();

lC = gRx2Buffer[ ( gRx2Ptr - gRx2Cntr ) & RX2_BUFFER_SIZE-1 ]; //    GET BYTE FROM FIFO BUFFER
gRx2BufferFull = 0; //    CLEAR BUFFER FULL FLAG
gRx2Cntr--; //    DEC BUFFER COUNTER
U2RXIE = 1; //    ENABLE RX INTERRUPTS

return lC; //    RETURN CHAR
}

//*****
// GETCHE GET BYTE FROM TERMINAL SERIAL PORT AND ECHO BACK
//*****

char    getche( void )
{
char    lC;

lC = getch();
putch( lC );

return lC;
}

//*****
// KBHIT FUNCTION, EG. ARE THERE ANY BYTES IN RX FIFO
//*****

int     kbhit( void )
{
return gRx2Cntr; //    RETURN THE FIFO BUFFER COUNTER
}

```

P.Brain-ds24 (v2.3) User Guide

```
}

```

Now for the interrupt service routines:

```

//*****
// UART2 TX ISR
//*****

void interrupt tx2isr( void ) @ U2TX_VCTR
{
if( U2TXIF ) // IS THE TXIF FLAG SET? IT SHOULD BE!
{
while( !U2_TXBF && tx_cntr2 ) // SPACE IN THE UART FIFO? & DATA TO TRANSMIT?
{
// LOAD THE UART BUFFER WITH NEW DATA FROM FIFO BUFFER
U2TXREG = gTx2Buffer[ ( gTx2Ptr - gTx2Cntr ) & TX2_BUFFER_SIZE-1 ];
gTx2BufferFull = 0; // CLEAR BUFFER FULL FLAG
gTx2Cntr--; // DEC BUFFER COUNTER

if( !gTx2Cntr) // IF THERE ARE NO MORE BYTES IN THE FIFO BUFFER
{
U2TXIE = 0; // DISABLE TX INTERRUPTS
}
}

U2TXIF = 0; // CLEAR THE TX INTERRUPT FLAG
}

}

//*****
// UART2 RX ISR
//*****

void interrupt rx2isr( void ) @ U2RX_VCTR
{
BYTE lC;

if( U2RXIF ) // IS THE RX INTERRUPT FLAG SET? IT SHOULD BE!
{
while( U2_URXDA ) // WHILE THERE IS DATA IN THE UART FIFO
{
if( U2_FERR ) // WAS THE DATA CORRUPT?
{
lC = U2RXREG; // YES, SO ABSORB DATA
}
else
{
lC = U2RXREG; // NO, SO STORE DATA IN FIFO BUFFER
if( gRx2Cntr == RX2_BUFFER_SIZE )
{
gRx2BufferFull = 1;
}
else
{
gRx2Buffer[ gRx2Ptr ] = lC;
gRx2Ptr++;
gRx2Ptr &= RX2_BUFFER_SIZE-1;
gRx2Cntr++;
}
}
}

U2RXIF = 0; // CLEAR RX INTERRUPT FLAG
}
}

```

P.Brain-ds24 (v2.3) User Guide

EEPROM Read & Write

These examples demonstrate how to read and write to the on-board EEPROM. First we define the low level I2C access functions:

```
//*****
// WAIT FOR I2C IDLE STATE
//*****

void i2cWaitForIdle( void )
{

while( (I2C1CON & 0x001f) | I2C1_TRSTAT )
    continue;
}

//*****
// START I2C
//*****

void i2cStart()
{

i2cWaitForIdle();

I2C1_SEN = 1;
}

//*****
// I2C REPEAT START
//*****

void i2cRepStart()
{

i2cWaitForIdle();

I2C1_RSEN = 1;
}

//*****
// I2C STOP
//*****

void i2cStop()
{

i2cWaitForIdle();

I2C1_PEN = 1;
}

//*****
// I2C READ BYTE
//*****

BYTE i2cRead( BYTE pAck )
{
BYTE lData;

i2cWaitForIdle();

I2C1_RCEN=1;

i2cWaitForIdle();
```

P.Brain-ds24 (v2.3) User Guide

```

lData = I2C1RCV;

i2cWaitForIdle();

if ( pAck )
    I2C1_ACKDT = 0;
else
    I2C1_ACKDT = 1;

I2C1_ACKEN = 1;          // SEND ACKNOWLEDGE SEQUENCE

return( lData );
}

//*****
// I2C WRITE DATA
//*****

WORD i2cWrite( BYTE pData )
{
i2cWaitForIdle();

I2C1TRN = pData;

return ( !I2C1_ACKSTAT ); // RETURNS 1 IF TX IS ACKNOWLEDGED
}

```

High level EEPROM functions used for storing BYTE data etc:

```

// *****
// WRITE EEPROM BYTE
// *****

void    my_eeprom_write_byte( WORD    pAddr, BYTE pData )
{
EE_WP = 0;          //    DISABLE EEPROM WRITE PROTECT

i2cStart();        //    START I2C
i2cWrite( 0xa0 );  //    WRITE EEPROM ADDRESS
i2cWrite( pAddr >> 8 ); //    WRITE EEPROM DATA ADDRESS
i2cWrite( pAddr & 0xff ); //    WRITE EEPROM DATA
i2cWrite( pData); //    START THE DATA ERASE WRITE CYCLE.
i2cStop();

DelayUs(6000);    //    WAIT FOR WRITE TO END. 6000uS

EE_WP = 1;        //    ENABLE EEPROM WRITE PROTECT
}

// *****
// READ EEPROM DATA
// *****

BYTE    my_eeprom_read_byte( WORD pAddr )
{
BYTE pData;

i2cStart();        //    START I2C
i2cWrite( 0xa0 );  //    WRITE EEPROM ADDRESS
i2cWrite( (addr >> 8) ); //    WRITE DATA ADDRESS
i2cWrite(addr & 0xff);
i2cRepStart();    //    REPEAT START CONDITION
i2cWrite(0xa1);   //    WRITE READ COMMAND
pData = i2cRead(0); //    READ DATA
i2cStop();        //    STOP I2C

return( pData );
}

```

P.Brain-ds24 (v2.3) User Guide

```

}

// *****
// WRITE EEPROM 32 BYTE BUFFER
// *****
// MUST NOT CROSS PAGE BOUNDARY OF 32, EG ONLY 5 LOWER ADDRESS BITS ARE INTERNALY
// INCREMENTED.

void my_eeeprom_write_buffer( WORD pAddr, BYTE *pData )
{
  BYTE pT;

  EE_WP = 0; // DISABLE EEPROM WRITE PROTECT

  i2cStart(); // START I2C
  i2cWrite( 0xa0 ); // WRITE EEPROM COMMAND
  i2cWrite( pAddr >> 8 ); // WRITE DATA ADDRESS START
  i2cWrite( pAddr & 0xff );
  for( pT = 0; pT < 32; pT++ ) // WRITE 32 BYTES OF DATA
    i2cWrite(*pData++);
  i2cStop(); // START THE DATA ERASE WRITE CYCLE.

  DelayUs(6000); // WAIT FOR WRITE TO END. 6000uS

  EE_WP = 1; // ENABLE EEPROM WRITE PROTECT
}

```

P.Brain-ds24 (v2.3) User Guide

Delay Routines

These functions are used for inserting milisecond or microsecond delays.

```
// *****
// DELAY MICRO SECONDS. NEEDS ADJUSTING FOR DIFFERENT PROCESSOR SPEEDS
// *****

void DelayUs( WORD pDelay )
{
while(--pDelay > 0)
    {
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    NOP();
    }
}

// *****
// DELAY MILLI SECONDS.
// *****

void DelayMs( WORD pDelay )
{
WORD lI;

while (pDelay--)
    {
    CLRWDT();
    lI = 4;
    while (lI--)
        {
        DelayUs( 250 ); // ADJUST FOR ERROR
        }
    }
}
```

P.Brain-ds24 (v2.3) User Guide

PWM Multiplexor

This example demonstrates how to configure the interrupt service routines for the PWM multiplexor block. For the purpose of this example, each PWM frame will be started in the main programme loop, once the frame is started, the output compare ISR's will continue until all 24 PWM channels are complete. There are four multiplexors, each with six outputs giving 24 channels of PWM, using multiplexing techniques, PWM signals are created in banks of four, one for each of the four Output Compare registers. See the schematic of the PWM block for further details.

Variables used by the PWM multiplexor:

PWM Definitions

Output compare interrupt service routines:

```
// *****
// OC1 ISR
// *****

void interrupt oc1_isr( void ) @ OC1_VCTR
{
OC1IE = 0;           // DISABLE OCx INTERRUPTS
OC1IF = 0;          // CLEAR INTERRUPT FLAG
NextPWMBank(); // CHECK NEXT PWM BANK
}

// *****
// OC2 ISR
// *****

void interrupt oc2_isr( void ) @ OC2_VCTR
{
OC2IE = 0;
OC2IF = 0;
NextPWMBank();
}

// *****
// OC3 ISR
// *****

void interrupt oc3_isr( void ) @ OC3_VCTR
{
OC3IE = 0;
OC3IF = 0;
NextPWMBank();
}

// *****
// OC4 ISR
// *****

void interrupt oc4_isr( void ) @ OC4_VCTR
{
OC4IE = 0;
OC4IF = 0;
NextPWMBank();
}
```


P.Brain-ds24 (v2.3) User Guide

```

// *****
// IF ALL 4 PWM OUTPUTS IN THE CURRENT BANK ARE DISABLED
// THIS TIMER IS USED TO FORCE AN INTERRUPT TO START
// THE NEXT PWM BANK.
// *****

Void   interrupt      tmr3_isr( void ) @ T3_VCTR
{
T3IF = 0;              // CLEAR INTERRUPT FLAG
T3IE = 0;              // DISABLE INTERRUPT
NextPWMBank(); // CHECK NEXT PWM BANK
}

// *****
// USED TO START THE PWM FRAME, SETS MUX ADDRESS TO 0
// *****

void   StartPWMFrame( void )
{

T3ON = 0;              // SWITCH TIMER OFF
TMR3 = 0;

if( OC1IE || OC2IE || OC3IE || OC4IE ) // CHECK FOR FRAME RATE ERROR
{
// IF ANY OF THE OCIE BITS IS SET WHEN THIS FUNCTION
// IS CALLED, SOMETHING HAS EITHER GONE WRONG, OR THE
// PWM REFRESH RATE IS TOO HIGH. STARTING A NEW FRAME
// BEFORE THE OLD FRAME HAS FINISHED WILL CAUSE SERVO
// GLITCHES AND OTHER ISSUES!

// AT THIS POINT IT IS DOWN TO THE USER TO DECIDE WHAT TO DO!
// SET ERROR FLAG, OR RETURN ETC..
// PWM_error_flag = 1;
// return;
}

OC1IE = 0;            // DISABLE ALL OC INTERRUPTS, INCASE WE IGNORE THE ABOVE ERROR CONDITION
OC2IE = 0;
OC3IE = 0;
OC4IE = 0;

gMmuxAddress = 0;     // SET MUX ADDRESS TO 0

MUX_A2 = 0;NOP();    // IF A2 AND A1 ARE CHANGED WITHOUT AN OP BETWEEN THEM,
MUX_A0 = 0;NOP();    // THE SECOND WILL NOT TAKE EFFECT. OLD PIC BUG ABOUT
MUX_A1 = 0;NOP();    // NOT CHANGING PORT PINS WITHIN CONSECUTIVE INSTRUCTIONS!

SetipOCRegisters(); // CONFIGURE OC REGISTERS FOR NEXT PWM BANK

T3ON = 1;            // START TIMER 3

}

// *****
// THIS FUNCTION CONFIGURESS THE OC REGISTERS WITH THE NEXT BANK OF PWM VALUES
// EACH PWM START TIME IS OFFSET BY A SMALL AMOUNT DEFINED BELOW. THIS IS TO REDUCE
// SIGNAL NOISE ON THE P.BRAIN, BUT IS NOT ENTIRELY NECESSARY.
// *****

#define PWM1_START_COUNT      2
#define PWM2_START_COUNT      12
#define PWM3_START_COUNT      22
#define PWM4_START_COUNT      32

```

P.Brain-ds24 (v2.3) User Guide

```

void SetipOCRegisters( void )
{

OC1_M2 = 0;           //      DISABLE OC MODULES
OC2_M2 = 0;
OC3_M2 = 0;
OC4_M2 = 0;

//      SETUP T3 TO INTERRUPT IN 50uS IN THE CASE THAT NONE OF THE CURRENT PWM BANK SERVOS
//      ARE ENABLED. IF ANY OF THE FOLLOWING 4 ARE ENABLED, THE T3 INTS ARE DISABLED
//      AND PR3 REG SET TO MAX.

PR3 = (WORD)((double)((FCY/OC_TIMER_PRE_SCALER)/1000000.0) * (50.0));      // IN uSeconds

T3IF = 0;           // CLEAR T3 INTERRUPT FLAG
T3IE = 1;           // T3 INTERRUPTS ENABLED, BUT T3 IS NOT YET RUNNING.

//      IN ORDER TO UNDERSTAND WHY OC1 DOES SERVO CHANNELS 6 TO 11, PLEASE LOOK AT THE PWM
//      MULTIPLEXOR BLOCK IN THE SCHEMATICS SECTION OF THE p.Brain-ds24 USER GUIDE.

//      SETUP COMPARE REGISTERS START AND STOP TIMES.
//      OCR1 = CHANNELS 6 -> 11

OC1R = PWM1_START_COUNT;           // CONFIGURE START TIME, EG RISING EDGE
                                     // CONFIGURE STOP TIME, EG FALLING EDGE
OC1RS = PWM1_START_COUNT + PWM_MIN + gServoPos[ 6 + gMuxAddress ];

if( gServoEnable [ 6 + gMuxAddress ] )      // IF THE SERVO IS ENABLED
{
    OC1_M2 = 1;           // CONFIGURE THE OC MODE
    OC1IE = 1;           // ENABLE OC INTERRUPTS
    T3IE = 0;           // DISABLE T3 INTERRUPTS
    PR3 = 0xffff;       // RESET PR3 TO MAXIMUM
}

//      OCR2 = CHANNELS 23 -> 18
OC2R = PWM2_START_COUNT;
OC2RS = PWM2_START_COUNT + PWM_MIN + gServoPos[ 23 - gMuxAddress ];
if( gServoEnable [ 23 - gMuxAddress ] )
{
    OC2_M2 = 1;
    OC2IE = 1;
    T3IE = 0;
    PR3 = 0xffff;
}

//      OCR3 = CHANNELS 0 -> 5
OC3R = PWM3_START_COUNT;
OC3RS = PWM3_START_COUNT + PWM_MIN + gServoPos[ gMuxAddress ];
if( gServoEnable [ gMuxAddress ] )
{
    OC3_M2 = 1;
    OC3IE = 1;
    T3IE = 0;
    PR3 = 0xffff;
}

//      OCR4 = CHANNELS 17 -> 12
OC4R = PWM4_START_COUNT;
OC4RS = PWM4_START_COUNT + PWM_MIN + gServoPos[ 17 - gMuxAddress ];
if( gServoEnable [ 17 - gMuxAddress ] )
{
    OC4_M2 = 1;
    OC4IE = 1;
    T3IE = 0;
    PR3 = 0xffff;
}
}

```

P.Brain-ds24 (v2.3) User Guide

```

gMuxAddress++; // INC THE MUC ADDRESS.

}

// *****
// ONCE THE PWM FRAME HAS BEEN STARTED, THIS FUNCTION IS CALLED WITHIN EACH OC ISR
// AND WITHIN THE T3 ISR (IF ENABLED) TO DETERMINE IF THE CURRENT PWM BANK IS COMPLET
// EG, ALL FOUR PWM OUTPUTS OF THE BANK ARE FINISHED. IF ALL FOUR CHANNELS OF THE BANK
// ARE COMPLETE, IT THEN CHECKS IF THE FRAME IS COMPLETE.
// *****

void NextPWMBank( void )
{

if( OC1IE || OC2IE || OC3IE || OC4IE ) // IF WE HAVE NOT DONE ALL 4 OUTPUTS, RETURN.
    return;

T3ON = 0; // SWITCH TIMER OFF
TMR3 = 0; // CLEAR TIMER 3

// HAVE WE COMPLETED ALL CHANNELS? IF SO, THE FRAME IS COMPLETE
// SO DO NOT RESET OUTPUT COMPARE MOCULES.
// ONLY FIRST 6 OUTPUTS ON MUX ARE USED.

if( gMuxAddress > 5 )
    return;

if( gMuxAddress & 0x0001 ) // SETUP MUX ADDRESS BITS
    MUX_A0 = 1;
else
    MUX_A0 = 0;

if( gMuxAddress & 0x0002 ) // SETUP MUX ADDRESS BITS
    MUX_A1 = 1;
else
    MUX_A1 = 0;

if( gMuxAddress & 0x0004 ) // SETUP MUX ADDRESS BITS
    MUX_A2 = 1;
else
    MUX_A2 = 0;

SetipOCRegisters(); // CONFIGURE OC REGISTERS AND START NEXT PWM BANK

T3ON = 1; // START TIMER 3
}

// *****
// WHEN DESIGNIN A MOTHERBOARD FOR THE p.Brain-ds24 MODULE, IT IS NOT ALWAYS
// POSSIBLE TO ROUTE THE PWM OUTPUTS TO THE MOTHERBOARD AREA THAT IS DESIRED, THEREFORE
// IT IS NECESSARY TO RE-MAP THE SERVO OUTPUTS IN SOFTWARE.
// *****

void DistributeServoOutputs( WORD *pServoData )
{
WORD lT;

for( lT = 0; lT < 24; lT++ ) // LOOP
    // PLACE SERVO DATA INTO RE-MAPPED LOCATION
    gServoPos[ cServoRemap[ lT ] ] = *pServoData++;
}

```

P.Brain-ds24 (v2.3) User Guide

Legal

Please read fully before purchasing any merchandise from micromagic systems ltd.

PRODUCTS

In no event shall micromagic systems be liable for any claim for incidental, consequential damages, or any injuries sustained due to the use of or improper use of products and / or kits purchased out of or in connection thereof with the manufacture, sale, delivery or use of any product in this catalogue or web site. All micromagic systems products purchased should NOT be used for medical, life-saving, life-support, or any applications that could cause injury, dangerous / hazardous situations or consequential damages resulting from the use of the mechanical, hardware or software products sold or represented by micromagic systems.

All products sold by micromagic systems are for Self Learning Experiences, and for Safe Entertainment.

Prices and Specifications

Note

Product specifications, prices listed and availability of items in our web site and in our printed catalogue are subject to change without notice. All prices shown in our web site and in our printed catalogue are believed accurate at time of publication, but subject to change without any notice. We will always advise you of the new price increase and seek your approval before processing any orders you place.

General Product Terms and Conditions

Call micromagic systems before returning any items.

PRODUCT RETURNS POLICY

All sales are final.

New products are warranted for 30 days. Any return for repair or replacement must be pre-authorized by micromagic systems and under no circumstance will returns be accepted unless so authorized. Return Products under warranty must be pre-approved by MMS and sent via certified mail, prepaid and insured, for your protection. (Please note we cannot refund shipping fees). All electronic kit sales are final. Due to the fact that components of the kit may be damaged during assembly we do not accept returns or refunds on any electronic kits

If you receive damaged merchandise, you must contact micromagic systems within 2 days of receipt of your original order. Specify clearly the reason for your refusal. We will exchange returned merchandise for same new merchandise, or for the item sterling amount within 7 days once we receive the returned damaged items from you. Proof of mailing is advised, as we cannot be held responsible for loss of the returned merchandise in mail transit. All return postage is non-refundable. The merchandise, including packing and wrapping material, being returned should be in the same condition as when you received them. Please contact us via e-mail at matt@micromagicsystems.com. Defective merchandise will be replaced (No cash will be refunded). We reserve the right to refuse to replace any merchandise, which our micromagic systems technicians determine to be damaged by the user, or through inappropriate use of that merchandise.

WARRANTY POLICY

We guarantee all products except electronic kits to be free of defects in workmanship and material for 30 days from the purchase, delivery date. We will repair or replace non-electronic kits (No cash will be refunded), at our option providing there is no evidence of customer misuse or alteration to that product item.

micromagic systems carries a limited 30 day warranty on most all items, some items carry an additional number of warranty days or special restrictions. If you want specific warranty information about a product contact micromagic systems to obtain that information.

We are not able to offer any refunds or accept returns for the following items and products: Electronic Kits.

CANCELLATION POLICY

Please be aware that if you cancel an order you may be responsible for restocking fees and / or shipping charges, including charges for return shipping. Cancelled orders are subject to a 25% or £10.00 minimum restocking fee. Orders cancelled within 24 hours of order placement will not be subject to restocking fees however this does not apply to orders with Express Shipping and Handling.