```c
#include <pic.h>
#include "sci.h"

/* Routines for initialisation and use of the SCI
 * for the PIC processor.
 */

/* other options:
 * frame errors
 */

unsigned char sci_Init(unsigned long int baud, unsigned char ninebits)
{
    int X;
    unsigned long tmp;

    /* calculate and set baud rate register */
    /* for asynchronous mode */
    tmp = 16UL * baud;
    X = (int)(FOSC/tmp) - 1;
    if((X>255) || (X<0))
    {
        tmp = 64UL * baud;
        X = (int)(FOSC/tmp) - 1;
        if((X>255) || (X<0))
        {
            return 1;   /* panic - baud rate unobtainable */
        }
        else
            BRGH = 0;   /* low baud rate */
    }
    else
    BRGH = 1;   /* high baud rate */
    SPBRG = X;  /* set the baud rate */

    SYNC = 0;   /* asynchronous */
    SPEN = 1;   /* enable serial port pins */
    CREN = 1;   /* enable reception */
    SREN = 0;   /* no effect */
    TXIE = 0;   /* disable tx interrupts */
    RCIE = 0;   /* disable rx interrupts */
    TX9  = ninebits?1:0;    /* 8- or 9-bit transmission */
    RX9  = ninebits?1:0;    /* 8- or 9-bit reception */
    TXEN = 1;   /* enable the transmitter */

    return 0;
}

void putch(unsigned char byte)
//sci_PutByte(unsigned char byte)
{
    while(!TXIF)    /* set when register is empty */
        continue;
    TXREG = byte;

    return;
}

unsigned char getch(void)
//sci_GetByte(void)
```

```
{
    while(!RCIF)    /* set when register is not empty */
        continue;

    return RCREG;   /* RXD9 and FERR are gone now */
}

unsigned char sci_CheckOERR(void)
{
    if(OERR)    /* re-enable after overrun error */
    {
        CREN = 0;
        CREN = 1;
        return 1;
    }
    return 0;
}

#define sci_PutNinth(bitnine)   (TX9D = bitnine?1:0;)

unsigned char sci_GetNinth(void)
{
    while(!RCIF)
        continue;

    return RX9D;    /* RCIF is not cleared until RCREG is read */
}

unsigned char sci_GetFERR(void)
{
    while(!RCIF)
        continue;

    return FERR;    /* RCIF is not cleared until RCREG is read */
}
```