```c
/*********************************************************************
 *   Tittle:            Delfino Evaluation Board Keyboard           *
 *   Filename:          KeyboardMat4x4.c                            *
 *   Date:              20-11-2014                                  *
 *   Last Modify:       20-11-2014                                  *
 *   File Version:      1.0                                         *
 *                                                                  *
 *   Author:            Flaxer Eli                                  *
 *   Company:           Flaxer.net                                  *
 *                                                                  *
 *                                                                  */

#include "DSP28x_Project.h"     // Device Headerfile and Examples Include File

interrupt void Xint3456_isr(void);
#define KB_BUFFER_SIZE  64
struct {
        char Data[KB_BUFFER_SIZE];
        int Haed;
        int Tail;
        } KBCyclicBuffer = {{0}, 0, 0};

/****************************************************************************/
inline static void KeboardWriteCode(char data)
{
  GpioDataRegs.GPBCLEAR.all = (0x0FL << 8);      // Clear 4 data bits GPIO40-GPIO43
  GpioDataRegs.GPBSET.all = ((long)data << 8);  // Set the relevant data bits GPIO40-GPIO43
  DELAY_US(1000);
}
/****************************************************************************/
inline static char KeboardReadCode()
{
  return((GpioDataRegs.GPBDAT.all >> 12) & 0x0FL);      // Read 4 data bits GPIO44-GPIO47
}
/****************************************************************************/
inline void Beep(int MiliSec)
{
  GpioDataRegs.GPASET.bit.GPIO27 = 1;           // Buzzer On
  DELAY_US(1000L*MiliSec);
  GpioDataRegs.GPACLEAR.bit.GPIO27 = 1;         // Buzzer Off
}
/****************************************************************************/
static char scan2ascii(char sc)
{
 switch (sc)
    {
    case  0xE0: return('1');
    case  0xD0: return('2');
    case  0xB0: return('3');
    case  0x70: return('A');
    case  0xE1: return('4');
    case  0xD1: return('5');
    case  0xB1: return('6');
    case  0x71: return('B');
    case  0xE2: return('7');
    case  0xD2: return('8');
    case  0xB2: return('9');
    case  0x72: return('C');
    case  0xE3: return('*');
    case  0xD3: return('0');
    case  0xB3: return('#');
    case  0x73: return('D');
    }
 return(0);
}
/*****************************************************************************/
char ReadKB(char wait)
{
static char code[] = {0xE, 0xD, 0xB, 0x7};
char data;
char i;
  KeboardWriteCode(0x0);
  DELAY_US(1000);
  while(KeboardReadCode() == 0x0F)  // Check 4 data bits GPIO44-GPIO47
    if (!wait) return(0);
  Beep(20);
  for(i=0; i<4; i++)
    {
    KeboardWriteCode(code[i]);
    DELAY_US(1000);
    data = KeboardReadCode();
    if (data != 0x0F)
        break;
    }
  while(KeboardReadCode() != 0x0F); // Wait for button release
  DELAY_US(1000);
  KeboardWriteCode(0x0);
  return(scan2ascii((data << 4) | i));
}
/*****************************************************************************/
char PushButtonRead(int Inx)
{
    switch (Inx)
        {
        case 0:
            return (GpioDataRegs.GPCDAT.bit.GPIO80);
        case 1:
            return (GpioDataRegs.GPCDAT.bit.GPIO81);
        case 2:
            return (GpioDataRegs.GPCDAT.bit.GPIO82);
        case 3:
            return (GpioDataRegs.GPCDAT.bit.GPIO83);
        }
    return(0);
}
/*****************************************************************************/
void ConfigAndInstallKBInt()
{
       EALLOW;  // This is needed to write to EALLOW protected registers
       // Set input qualification period for GPIO44-GPIO47
       GpioCtrlRegs.GPACTRL.bit.QUALPRD3=1;  // Qual period = SYSCLKOUT/2
       GpioCtrlRegs.GPBQSEL1.bit.GPIO44=2;   // 6 samples
       GpioCtrlRegs.GPBQSEL1.bit.GPIO45=2;   // 6 samples
       GpioCtrlRegs.GPBQSEL1.bit.GPIO46=2;   // 6 samples
       GpioCtrlRegs.GPBQSEL1.bit.GPIO47=2;   // 6 samples

       GpioIntRegs.GPIOXINT3SEL.all = 12;    // Xint3 connected to GPIO44 32+12
       GpioIntRegs.GPIOXINT4SEL.all = 13;    // Xint4 connected to GPIO45 32+13
       GpioIntRegs.GPIOXINT5SEL.all = 14;    // Xint5 connected to GPIO46 32+14
       GpioIntRegs.GPIOXINT6SEL.all = 15;    // Xint6 connected to GPIO47 32+15

       PieVectTable.XINT3 = &Xint3456_isr;
       PieVectTable.XINT4 = &Xint3456_isr;
       PieVectTable.XINT5 = &Xint3456_isr;
       PieVectTable.XINT6 = &Xint3456_isr;
       EDIS;   // This is needed to disable write to EALLOW protected registers

       // Enable Xint 3,4,5,6 in the PIE: Group 12 interrupt 1-4
       // Enable int1 which is connected to WAKEINT:
       PieCtrlRegs.PIECTRL.bit.ENPIE = 1;          // Enable the PIE block
       PieCtrlRegs.PIEIER12.bit.INTx1 = 1;         // Enable PIE Group 12 INT1
       PieCtrlRegs.PIEIER12.bit.INTx2 = 1;         // Enable PIE Group 12 INT2
       PieCtrlRegs.PIEIER12.bit.INTx3 = 1;         // Enable PIE Group 12 INT3
       PieCtrlRegs.PIEIER12.bit.INTx4 = 1;         // Enable PIE Group 12 INT4

       // Enable CPU int12
       IER |= M_INT12;

       // Configure XINT3-6
       XIntruptRegs.XINT3CR.bit.POLARITY = 0;      // Falling edge interrupt
       XIntruptRegs.XINT4CR.bit.POLARITY = 0;      // Falling edge interrupt
       XIntruptRegs.XINT5CR.bit.POLARITY = 0;      // Falling edge interrupt
       XIntruptRegs.XINT6CR.bit.POLARITY = 0;      // Falling edge interrupt

       // Enable XINT3-6
       XIntruptRegs.XINT3CR.bit.ENABLE = 1;        // Enable Xint3
       XIntruptRegs.XINT4CR.bit.ENABLE = 1;        // Enable XINT4
       XIntruptRegs.XINT5CR.bit.ENABLE = 1;        // Enable Xint5
       XIntruptRegs.XINT6CR.bit.ENABLE = 1;        // Enable XINT6
}
/******************************************************************************/
interrupt void Xint3456_isr(void)
{
    //Beep(50);
    KBCyclicBuffer.Data[KBCyclicBuffer.Haed++ % KB_BUFFER_SIZE] = ReadKB(0);
    // Acknowledge this interrupt to receive more interrupts from group 12
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}
/******************************************************************************/
```