Exemplo: Leitura de Teclado Matricial com o NIOS II

Por: Professor Édson Mélo
Instituto Federal de Santa Catarina
Departamento de Metalmecânica /IFSC
Coordenador do Projeto FPGA para Todos.
(http://www.fpgaparatodos.com.br)

Colaboração:
Bolsistas Pedro Martins Vieira e
Mateus Antônio L.
Projeto FPGA para Todos

Introdução

Os teclados matriciais são uma solução bastante prática como interface de entrada para sistemas simples que precisam de entrada numérica ou de caracteres, ou de comandos. A sua característica construtiva faz com que uma razoável quantidade de teclas possa ser lida a partir de um número menor de terminais de um sistema digital. Entretanto, esta vantagem tem um custo: um algoritmo ou um sistema sequencial adequado precisa ser desenvolvido para a sua leitura. Este artigo apresenta uma solução para o teste e a leitura de um teclado numérico, de um tipo utilizado em certos aparelhos telefônicos.

Apresentam-se aqui o circuito do teclado, o princípio adotado para a sua leitura, a solução de hardware utilizada, implementada em um sistema embarcado com processador NIOS II e FPGA ALTERA, e o programa que executa a sua leitura e apresenta no console do PC as informações de teclas pressionadas. O conceito básico que se tenta desenvolver não prevê solução para o problema do ruído de trepidação.

Fig. 1: Teste de teclado com o kit de FPGA /NIOS

O sistema foi desenvolvido no **kit de FPGA do Projeto FPGA para Todos**, como mais um exemplo didático de aplicação do conjunto de ferramentas construído pela iniciativa.

Esquema Simplificado do Teclado e Método de Leitura

A figura abaixo mostra o diagrama esquemático do teclado utilizado no experimento, com a típica construção matricial deste tipo de dispositivo. Observe que cada tecla se conecta a uma linha e a uma coluna (todas identificadas na figura para referência futura), estabelecendo contato entre os seus terminais correspondentes quando pressionada. Os resistores apresentados no esquema não estão integrados no teclado, mas são montados em nosso circuito de leitura, e serão explicados posteriormente.

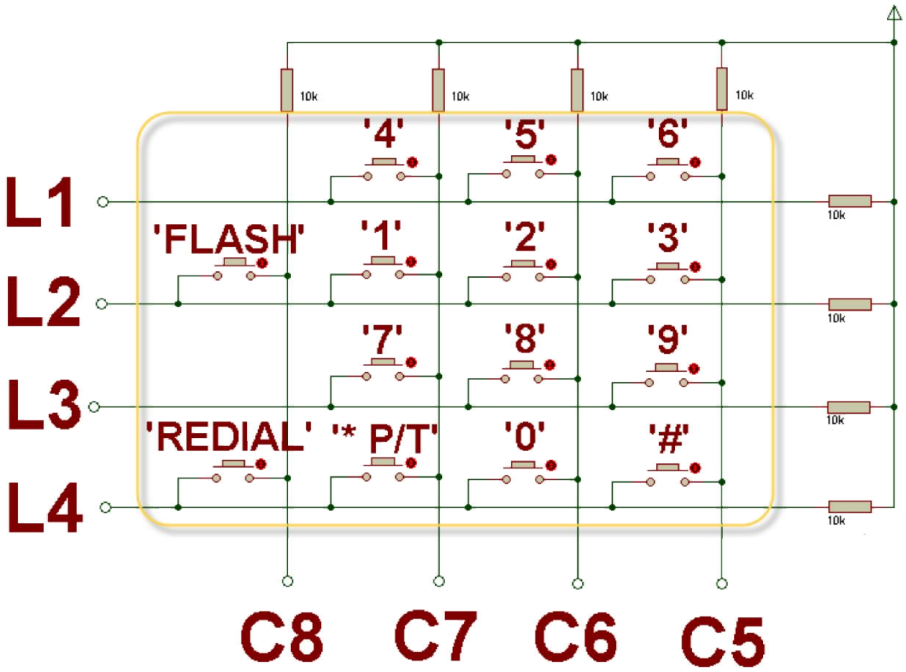

Fig. 2: Diagrama esquemático do teclado, com resistores de "pull-up" externos.

Um aspecto interessante é que o diagrama elétrico do teclado é construído como uma matriz de 4 linhas x 4 colunas, mas fisicamente a disposição das teclas no teclado apresenta-se como cinco linhas de três teclas.

Uma solução para a leitura do teclado deveria executar algum processo de varredura e identificar alguma conexão entre uma linha e uma coluna, e então determinar a tecla pressionada a partir da identificação daquela conexão.

A solução que propomos aqui para teste e verificação do teclado executa uma sequência de três etapas básicas:

1. Testa se há tecla pressionada;
2. Se há uma tecla pressionada, gera um código de identificação de linha e coluna,
3. Identifica a tecla a partir do código gerado.

Para determinar se há uma tecla pressionada, o método consiste em acionar nível lógico baixo ('0') nas linhas do teclado, enquanto as colunas serão verificadas por terminais de entrada do sistema. As colunas serão acionadas por resistores de "pull up", de forma que ficarão normalmente em nível alto, exceto no caso de haver alguma tecla pressionada, como mostrado na figura abaixo. Exemplificando o processo, o acionamento da tecla "7" provocou a formação do código **"1011"** nas colunas (lidas na sequência C8→C7→C6→C5).

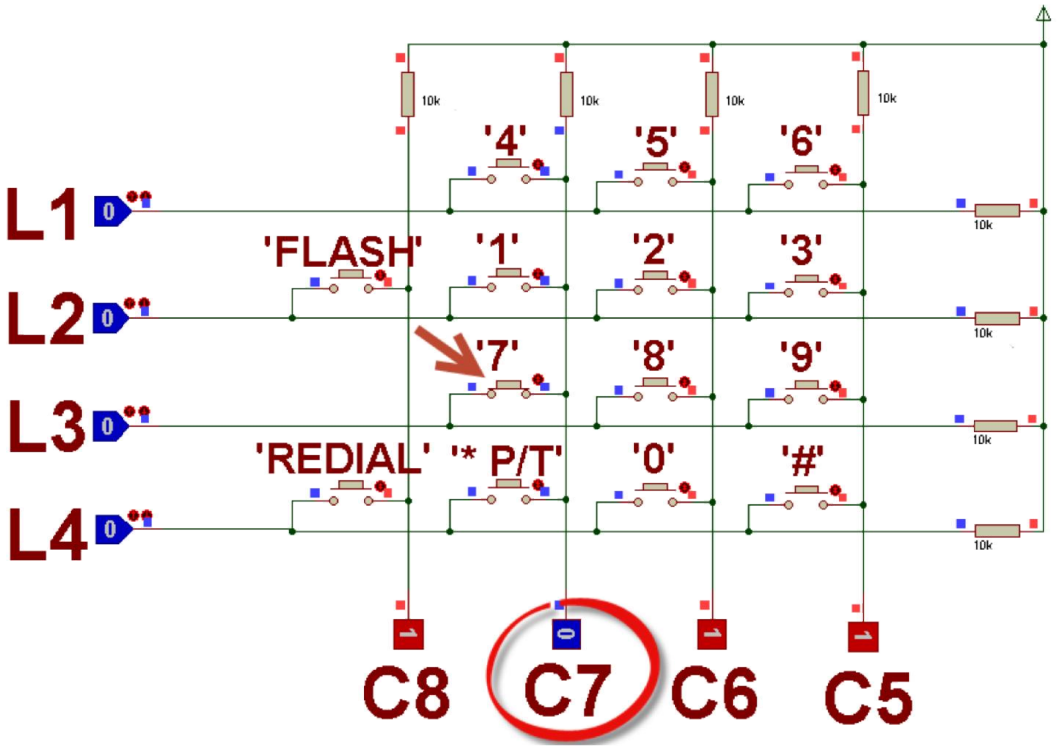

Fig. 3: Simulação do teste de colunas, com acionamento da tecla '7'.

Se a leitura das colunas resulta em algum valor diferente de **"1111"**, há alguma tecla pressionada, e se passa à etapa seguinte. Antes, porém, a informação de estado das colunas (o código formado por sua leitura), deve ser armazenado.

A condição de acionamento dos terminais do teclado é alternada agora. As linhas passam a ser lidas pelo sistema, enquanto as colunas são acionadas em nível lógico baixo, por pinos do sistema que agora devem ser de saída. Devido ao "pull up" das linhas, elas estarão em nível alto,

exceto aquela conectada (por um botão acionado) a alguma coluna. Na **Fig 4**, por exemplo, mostramos o resultado do acionamento do botão "7", que faz o estado das linhas ser lido como **"1011"** (lidas na sequência L4→L3→L2→L1).

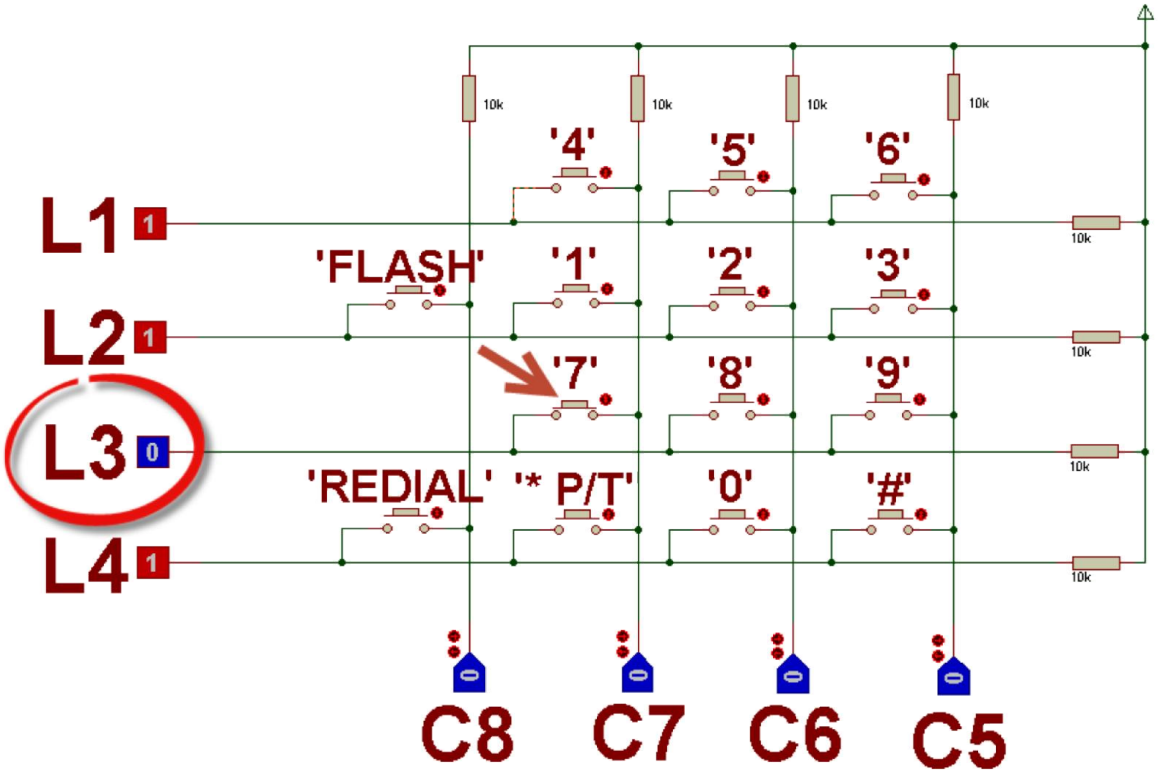

Fig. 4: Simulação do teste das linhas, com acionamento da tecla "7".

Na última etapa do processo, se agrupam os códigos formados na leitura das linhas e das colunas em um valor único de oito bits (por exemplo **"10111011"**) e se busca em uma tabela a tecla correspondente a este código (para o nosso exemplo, encontraríamos a tecla "7"). Para o nosso teclado em particular, a tabela seguinte mostra os códigos de formação para todas as teclas. Observe que "*" e "P/T" utilizam um mesmo código, e que não há tecla associada a dois dos códigos possíveis.

Tabela 1: Códigos de formação para o teclado telefônico.

Num.	Tecla	Código (Colunas.Linhas)
0	0	1101.0111
1	1	1011.1101
2	2	1101.1101
3	3	1110.1101

4	4	1011.1110
5	5	1101.1110
6	6	1110.1110
7	7	1011.1011
8	8	1101.1011
9	9	1110.1011
10	* e P/T	1011.0111
11	FLASH	0111.1101
12	#	1110.0111
13	REDIAL	0111.0111
14		0111.1011
15		0111.1110

Desenvolvendo uma Aplicação em FPGA /NIOS II.

Interface entre o SOPC e o Teclado Matricial

De forma a permitir a leitura do teclado com um sistema embarcado desenvolvido no kit de FPGA, utilizando o método proposto acima, precisamos conectá-lo a uma porta bidirecional, em que os pinos possam alternar entre o funcionamento como entrada e como saída.

Um dos SOPCs didáticos construídos para o kit do Projeto FPGA para Todos, denominado **SOPC_BASE2**, integra duas portas bidirecionais, denominadas PORT_A e PORT_B. No projeto do sistema, na ferramenta QSYS da ALTERA, estas portas foram configuradas para este modo de operação, como mostrado na figura abaixo, capturada na etapa de elaboração da PORT_A.

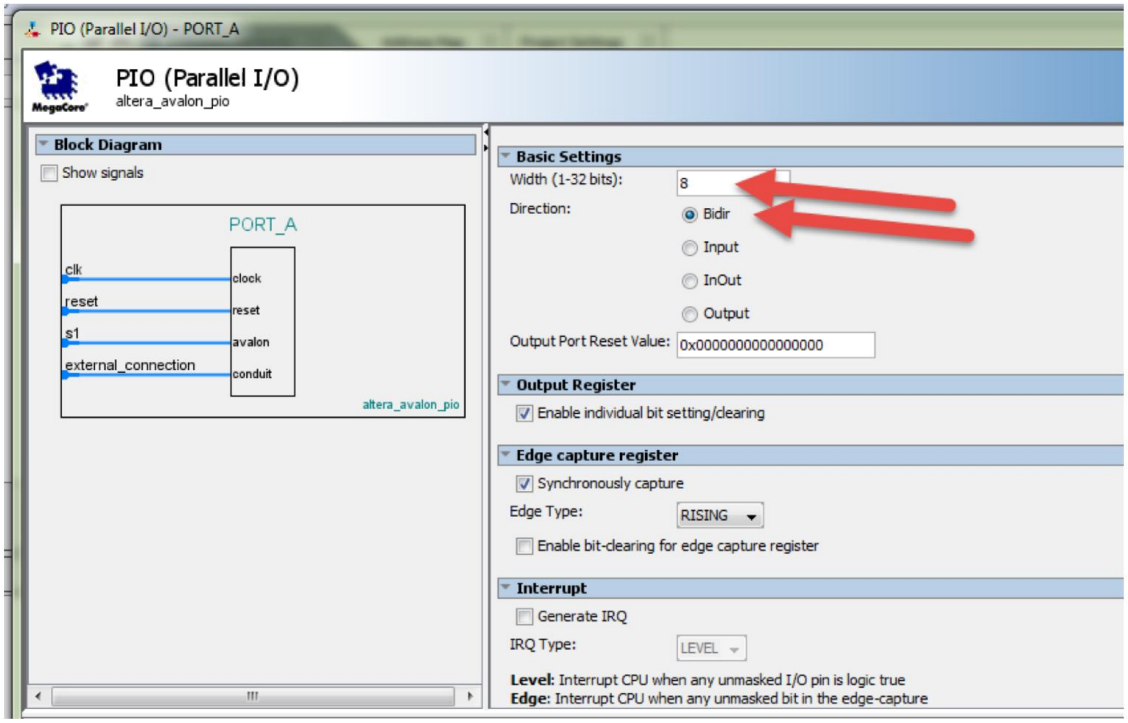

Fig. 5: Configuração de porta bidirecional no NIOS II.

Em nosso exemplo, conectamos o teclado à porta PORT_A, com os resistores de “pull-up” externos, como mostrado abaixo, de modo a poder gerar e identificar os códigos de linha e coluna apresentados no desenvolvimento do método de leitura do teclado.

Fig. 6: Conexão do teclado ao kit, com os resistores de "pull-up".

Programação da Leitura do Teclado – Exemplo

Apresentamos aqui um programa simples que identifica teclas pressionadas no teclado e exibe na tela do PC, no console da interface JTAG do NIOS II, os valores correspondentes. É uma demonstração básica, e o principal componente do projeto é a função *identifica_tecla()*, que pode ser estudada mais detidamente, para compreensão do processo de leitura do teclado, ou pode simplesmente ser chamada em algum projeto de aplicação prática.

Definições e Declarações Iniciais

O trecho inicial do código define as bibliotecas do sistema necessárias, alguns símbolos que simplificarão a programação, e o protótipo da função identifica_tecla(). Ressaltamos também as macros que dão nomes mais adequados (*config_DDR, entrada, saida e delay)* para as funções que acessam os drives do NIOS para as portas e para a temporização do sistema.

```
#include "sys/alt_stdio.h"
#include <altera_avalon_pio_regs.h>
#include <system.h>
#include <unistd.h>

#define PORTA                           PORT_A_BASE
#define PORTB                           PORT_B_BASE

#define PORT_TECLADO                    PORTA
#define PORT_LCD                        PORTB

#define config_DDR(porta,dado)    IOWR_ALTERA_AVALON_PIO_DIRECTION(porta, dado)
#define entrada(porta)            IORD_ALTERA_AVALON_PIO_DATA(porta)
#define saida(porta,dado)         IOWR_ALTERA_AVALON_PIO_DATA(porta, dado)

#define cbi(porta,n)              IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(porta, 1<<n)
#define sbi(porta,n)              IOWR_ALTERA_AVALON_PIO_SET_BITS(porta, 1<<n)

#define delay(intervalo)          usleep (1000L*intervalo)

char tecla, codigo;


// Protótipo para função de leitura de teclado:
int identifica_tecla ();
```

A função *identifica_tecla()*

Apresenta-se aqui uma função simples e útil para teste e leitura do teclado matricial do tipo

telefônico, que implementa o método proposto acima. Com esta função, torna-se prático desenvolver muitos diferentes projetos de aplicação com este periférico.

A função identifica_tecla() retorna um número inteiro, entre 0 e 16, que deve ser interpretado da forma seguinte:

- o valor 16 significa que não há uma tecla pressionada;
- o valor 15 significa que há uma combinação de teclas pressionadas;
- os valores entre 0 e 14 correspondem a alguma das teclas pressionadas, conforme a Tabela 1, previamente apresentada.

A listagem completa da função identifica_tecla(), incluida em um programa simples de teste do teclado, é apresentada no Anexo 1, e discutida ligeiramente aqui.

Inicialmente define-se, no trecho mostrado abaixo, uma matriz ("array"), de nome tab_cod_tecla[], com os códigos de conexão das teclas, conforme sua ligação à porta PORT_A.

```
//  Matriz de formação dos códigos de tecla:
//  PA7.PA6.PA5.PA4: linhas do teclado
//        PA3.PA2.PA1.PA0: colunas do teclado.
char tab_cod_tecla [15] = {
                      0b11010111  /* 0 */,
                      0b10111101  /* 1 */,
                      0b11011101  /* 2 */,
                      0b11101101  /* 3 */,
                      0b10111110  /* 4 */,
                      0b11011110  /* 5 */,
                      0b11101110  /* 6 */,
                      0b10111011  /* 7 */,
                      0b11011011  /* 8 */,
                      0b11101011  /* 9 */,
                      0b10110111  /* *  e também P/T  */,
                      0b01111101  /* FLASH */,
                      0b11100111  /* # */,
                      0b01111101  /* REDIAL */,
                      0b01110111  /*   */
              };
```

Na função identifica_tecla(), podem-se facilmente verificar os três estágios da verificação e leitura do teclado, previamente discutidos. No trecho inicial da função, mostrado abaixo, acionam-se as linhas em nível 0, enquanto as colunas são lidas para a variável *codigo_tecla*. Se *codigo_tecla* for 1111B, nenhuma tecla é pressionada, e a função é terminada, com valor de retorno 16.

```
// Configuração da porta para verificação de linhas:
config_DDR (PORTA, 0b11110000);  // Bits 7 a 4 como saídas,....
saida (PORTA, 0b00001111); // 0 nas saídas, e "pull up" nas entradas...
usleep (3);
```

```
// Testa se há uma tecla pressionada, retorna em
// caso contrário. Se não há uma tecla pressionada,
// retorna com o valor 16
codigo_tecla = entrada (PORTA) & 0b00001111;
if (  codigo_tecla == 0b00001111 )
        return 16;
```

Caso *codigo_tecla* resulte diferente de 1111B, há uma tecla pressionada, e a função prossegue com a reconfiguração dos pinos das portas (saídas e entradas). Agora, as colunas são acionadas em '0', e o estado das linhas é lido e composto com o anterior das colunas na variável *codigo_tecla* (os quatro primeiros bits correspondem ao estado das linhas, e os quatro restantes, ao das colunas).

```
//  Configuração da porta para verificação de colunas:
config_DDR (PORTA, 0b00001111); // Pinos 7 a 4 entrada......
saida (PORTA, 0b11110000); // Saídas em 0, entradas com "pull up".
usleep (3);

codigo_tecla = codigo_tecla | (entrada (PORTA) & 0b11110000);
```

Na última etapa da leitura, o valor formado em *codigo_tecla* é buscado na tabela de códigos, para a identificação da tecla correspondente. Caso haja combinação de teclas pressionadas, a tabela não conterá o código formado, e a função retornará com o valor 15, que indica esta condição.

```
    for (indice = 0; indice < 15; indice++)
          if (tab_cod_tecla[indice] == codigo_tecla)
                  break;

    return indice;
    // Obs.:  tecla não válida se indice = 15.
}
```

Exemplo de Aplicação: Teclado e Interface JTAG.

Como uma primeira aplicação, bastante elementar, da função identifica_tecla(), o programa *leitura_teclado*, cuja função principal é listada abaixo, envia para o PC, através da interface JTAG do sistema, a informação do estado do teclado, a cada segundo. O programa utiliza uma estrutura *switch...case* para determinar que tecla é pressionada, ou que não há tecla válida, se for o caso.

Entenda que o programa utiliza a versão reduzida da biblioteca padrão do C, daí a chamada às funções de saída em console *alt_putstr() e alt_printf()*.

O funcionamento do programa é mostrado em uma captura instantânea na **Fig.** 7, e em vídeo disponível no site YOUTUBE, no endereço http://youtu.be/XjmVCaq7pgw.

```
int main() {

  alt_putstr("Teste de Teclado com o NIOS II!\n");

  while (1)  {
        tecla = identifica_tecla();
        switch ( tecla ) {
             // Apresenta a informação de tecla pressionada (ou não):
           case 16:  alt_printf ("Sem tecla pressionada."); break;
           case 15:  alt_printf ("Comb. de teclas."); break;
           case 14:  alt_printf ("Tecla: REDIAL"); break;
           case 12:  alt_printf ("Tecla: #");  break;
           case 11:  alt_printf ("Tecla: FLASH"); break;
           case 10:  alt_printf ("Tecla: *"); break;
           default:  alt_printf ("Tecla: %x", tecla); break;
       }

        alt_printf ("\n");

        // 1000 ms até a próxima leitura..
       delay (1000);

  }
```

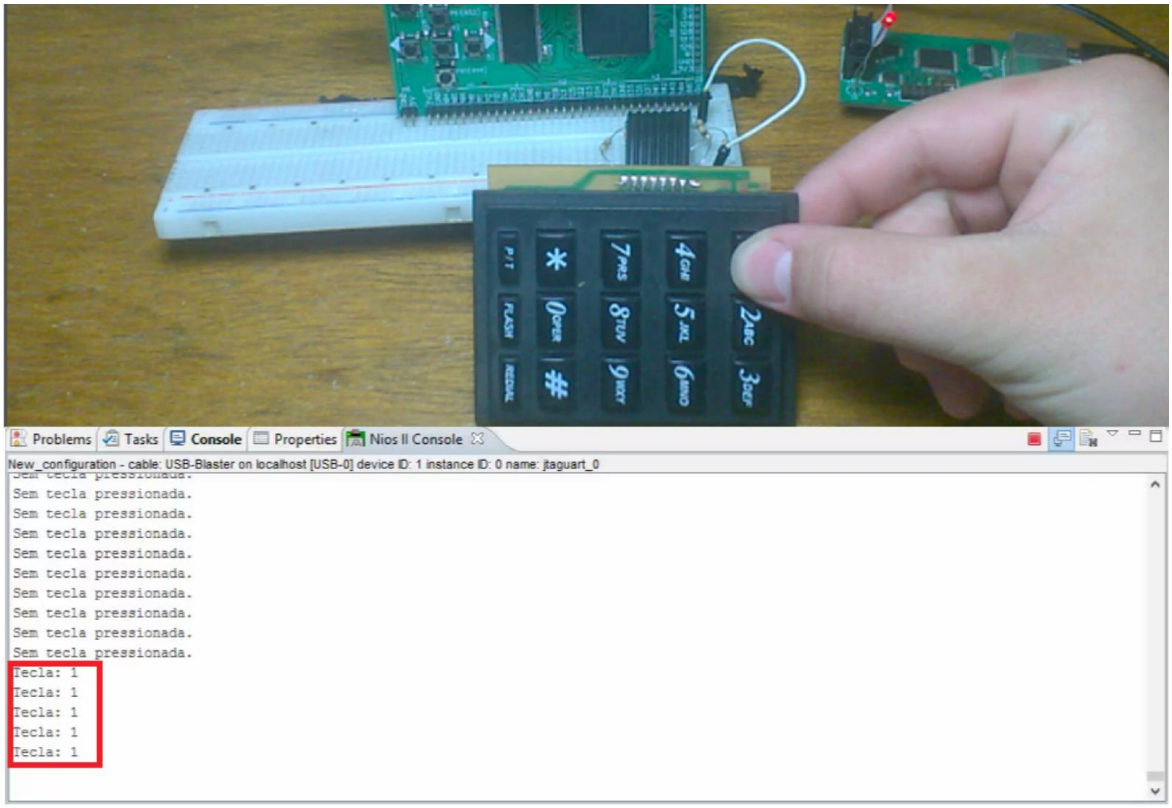

Fig. 7: Sistema de leitura de teclado com NIOS II e captura de tela de funcionamento.

Outro Exemplo de Aplicação: Teclado e LCD

Em outra aplicação interessante, e demonstrando o uso do teclado em um projeto ligeiramente mais complexo, propomos aqui um sistema para a entrada de códigos numéricos a serem exibidos no LCD. Neste projeto, usamos as funções para LCD discutidas em outro artigo, brevemente disponível no portal FPGA para Todos.

A figura abaixo mostra o conjunto, que integra o kit de FPGA, o teclado conectado à porta PORT_A, como previamente discutido, e um mostrador LCD do tipo semigráfico, 16x2, à PORT_B. Para conexão do LCD ao kit, utiliza-se a placa adaptadora para LCD do projeto.

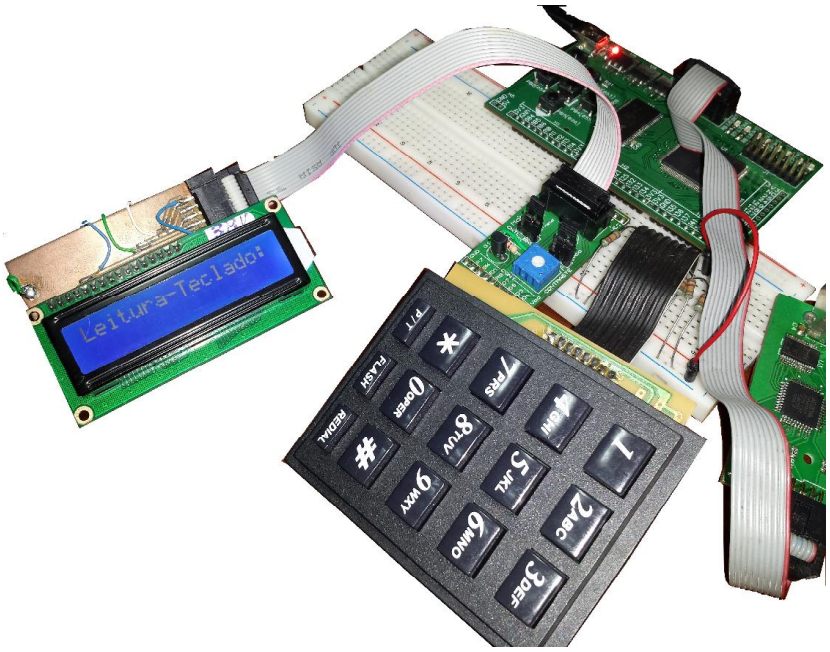

Fig. 8: Montagem de kit com teclado e LCD.

O programa completo deverá ser disponibilizado no portal, em um artigo específico sobre o uso de LCD com o NIOS no kit do projeto. Discutimos aqui, em linhas gerais, trechos da função principal [*main()*] do programa. Inicialmente, configura-se a porta a ser utilizada (PORTB, que é associada ao nome PORT_LCD em uma diretiva de definição no início do código-fonte) como porta de saída, e se chama a função de configuração do LCD. A tela do LCD é limpa, com um código de comando 0x01, e uma mensagem inicial é exibida, na sua primeira linha.

```
// Configuração do LCD e Mensagem Inicial:
config_DDR (PORT_LCD, 0b11111111); // PORT_LCD: porta de saída.
configuraLCD();
envia_byte (1, 0x01); // Limpa o LCD...
```

```
envia_string ("Leitura-Teclado:");
// Prepara para enviar uma nova linha:
envia_byte (1, 0xC0);      // Cursor na segunda linha...
```

A seguir, se inicia um trecho de repetição contínua, em que se aguarda que uma tecla seja pressionada e depois solta, e os códigos numéricos digitados são acrescentados, sequencialmente, em linha no LCD. As teclas '*', REDIAL, FLASH e P/T são ignoradas, mas a tecla '#' comanda uma limpeza da segunda linha do LCD, e o cursor é reposicionado para se começar a exibição de uma nova sequência numérica.

```
// Loop: Escreve linhas com os códigos digitados no teclado numérico:
   while(1) //Teste...

   {
         // Espera o acionamento de uma tecla:" );
         while ( (tecla = identifica_tecla()) == 16)
                       ;  // O código 16 informa que não há tecla...

         // Imprime o valor da tecla, ou executa outra função:
         switch (tecla)  {
               case 15: break; // Tecla inválida, nada a executar...
               case 14: break; //
               case 13: break; //
               case 11: break; //
               case 10: break; //
               case 12: // Caractere "#" é comando para limpar linha...
                     envia_byte(1, 0xC0);
                     envia_string ("                 ");
                     envia_byte(1, 0xC0);
                     break;

               default:
                     // Envia o código ASCII correspondente à tecla numérica:
                     envia_byte (0, tecla + '0'); //
                     break;
         }

         // Quando houver tecla pressionada, espera que a
         // mesma seja solta, antes de prosseguir..
         while ( identifica_tecla() == tecla)
                                    ;
   }
}
```

O funcionamento do programa é exibido na figura ao lado, e em vídeo disponível no Youtube. Destacamos que as rotinas para o LCD não são descritas aqui, mas podem ser estudadas no próprio projeto, que pode ser obtido do portal FPGA Para Todos. Um artigo dedicado ao uso do LCD com o

Fig. 9: Tela de saída para LCD.

NIOS II, no kit do projeto, estará disponível brevemente.

Conclusão

O processo de leitura de um teclado matricial em um projeto baseado em microcontrolador é simples, e discutido e exemplificado aqui através de aplicações de cunho didático, em um sistema com o processador NIOS II. É interessante observar, entretanto, que os conceitos básicos foram apresentados de forma independente de arquitetura de hardware adotada, demandando apenas o uso de uma porta bidirecional de E/S. Este tipo de porta é disponivel na maioria dos microcontroladores comerciais. Assim, é perfeitamente viável a aplicação do artigo em projetos com arquiteturas de processamento, como AVR, PIC, ARM.

Espera-se que o material apresentado possa servir como referência e suporte para o desenvolvimento de projetos que envolvam o uso deste tipo de componente. Pretende-se ainda que o artigo e os exemplos de projeto, disponibilizados no portal **FPGA para Todos** possam também ser aplicados como material didático, na demonstração de aspectos da construção de sistemas embarcados (pull up, portas bidirecionais, e outros conceitos) e de programação básica de acesso a um periférico (configuração de portas, acesso de E/S).

Finalizando, é importante observar que a aplicação imediata deste material, em uma abordagem educacional, é fortemente viabilizada pela disponibilidade do sistema físico, na forma dos kits de FPGA e periféricos do projeto FPGA para Todos. Particularmente, uma pequena placa periférica que implementa um teclado simples de 12 teclas, e também uma para conexão facilitada de um display LCD são disponibilizadas pelo projeto. Todas estas placas têm arquitetura aberta, e as placas para sua montagem podem ser conseguidas/produzidas com relativa facilidade.

Anexos

As listagens dos programas discutidos no texto são apresentadas aqui, para estudo. Para teste dos projetos, deve-se gravar o circuito do sistema embarcado (o "SOPC") no kit de FPGA e a seguir, no ambiente de desenvolvimento do NIOS, montar de gravar o projeto correspondente a cada programa. Este processo é descrito em um roteiro disponível na seção de "download" do artigo no **Portal FPGA para Todos**.

1. Listagem do Programa *Leitura_teclado:*

```
/*
*
*   Projeto:  Leitura de Teclado Matricial
*
*      Este programa permite a leitura de um teclado matricial
*   de 5x3 teclas, do tipo telefônico.  Pode ser facilmente
*   adaptado para outra estrutura de teclado matricial.
*
*   Por:  Prof.  Édson Mélo, em março de 2014
*/



#include "sys/alt_stdio.h"
#include <altera_avalon_pio_regs.h>
#include <system.h>
#include <unistd.h>

#define PORTA PORT_A_BASE
#define PORTB PORT_B_BASE

#define config_DDR(porta,dado)   IOWR_ALTERA_AVALON_PIO_DIRECTION(porta, dado)
#define entrada(porta)           IORD_ALTERA_AVALON_PIO_DATA(porta)
#define saida(porta,dado)        IOWR_ALTERA_AVALON_PIO_DATA(porta, dado)

#define delay(intervalo)         usleep (1000L*intervalo)
char tecla;

int main()
{
       alt_putstr("Teste de Teclado com o NIOS II!\n");

       while (1)  {
              tecla = identifica_tecla();
              switch ( tecla ) {
                 // Apresenta a informação de tecla pressionada (ou não):
                case 16:  alt_printf ("Sem tecla pressionada."); break;
                case 15:  alt_printf ("Comb. de teclas."); break;
                case 14:  alt_printf ("Tecla: REDIAL"); break;
                case 12:  alt_printf ("Tecla: #");  break;
                case 11:  alt_printf ("Tecla: FLASH"); break;
                case 10:  alt_printf ("Tecla: *"); break;
                default:  alt_printf ("Tecla: %x", tecla); break;
```

```
        }

            alt_printf ("\n");

            // 100 ms até a próxima leitura..
        delay (1000);

      }
}


int identifica_tecla () {
    int indice;

    //  Matriz de formação dos códigos de tecla:
    //  8765:   linhas do teclado
    //      4321:  colunas do teclado.
    char tab_cod_tecla [15] = {
                              0b11010111  /* 0 */,
                              0b10111101  /* 1 */,
                              0b11011101  /* 2 */,
                              0b11101101  /* 3 */,
                              0b10111110  /* 4 */,
                              0b11011110  /* 5 */,
                              0b11101110  /* 6 */,
                              0b10111011  /* 7 */,
                              0b11011011  /* 8 */,
                              0b11101011  /* 9 */,
                              0b10110111  /* *  e também P/T  */,
                              0b01111101  /* FLASH */,
                              0b11100111  /* # */,
                              0b01111101  /* REDIAL */,
                              0b01110111  /*   */
            };

    char codigo_tecla;  // Código formado pelo acionamento de tecla.

    // Configuração da porta para verificação de linhas:
    config_DDR (PORTA, 0b11110000);  // Bits 7 a 4 como saídas, e 3 a 0, como entradas.
    saida (PORTA, 0b00001111);           // 0 nas saídas, e "pull up" nas entradas...
    usleep (3);

    // Testa se há uma tecla pressionada, retorna em
    // caso contrário. Se não há uma tecla pressionada,
    // retorna com o valor 16
    codigo_tecla = entrada (PORTA) & 0b00001111;
    if (  codigo_tecla == 0b00001111 )
         return 16;

    //  Configuração da porta para verificação de colunas:
    config_DDR (PORTA, 0b00001111);   // Pinos 7 a 4 como entrada, e 3 a 0 como saídas.
    saida (PORTA, 0b11110000);            // Saídas em 0, entradas com "pull up".
    usleep (3);

    codigo_tecla = codigo_tecla | ( entrada (PORTA) & 0b11110000);

    for (indice = 0; indice < 15; indice++)
        if (tab_cod_tecla[indice] == codigo_tecla)
             break;

    return indice;
    // Obs.:  tecla não válida se indice = 15.
}
```

2. Listagem do Programa *Teclado_LCD*

```
/*
*
*   Projeto:  Leitura de Teclado Matricial
*
*
*      Este programa permite a leitura de um teclado matricial
*   de 5x3 teclas, do tipo telefônico, gerando saída para um
*   display LCD de 16x2 caracteres.
*       O teclado será conectado a uma porta de saída bidi-
*   recional do sistema, com pinos que podem ser individualmente
*   configurados como de entrada ou de saída. Esta configuração
*   será alterada ciclicamente no programa para executar a
*   varredura das linhas e colunas do teclado.
*        O display de LCD deve ser conectado a uma outra porta
*   bidirecional, configurada desde a partida do programa para
*   função de saída. A aplicação do LCD com o kit de FPGA será
*   melhor discutida em artigo específico dedicado a este tipo
*   de dispositivo.
*
*   Por:  Prof.  Édson Mélo, em março de 2014
*
*/

#include <system.h>
#include <sys/alt_stdio.h>
#include <altera_avalon_pio_regs.h>
#include <unistd.h>

// Diretivas de definição de símbolos úteis, em geral:
#define PORTA                           PORT_A_BASE
#define PORTB                           PORT_B_BASE

// Definição de macros para facilitar acesso aos periféricos do sistema:
#define saida(base,data) IOWR_ALTERA_AVALON_PIO_DATA(base, data)
#define entrada(base) IORD_ALTERA_AVALON_PIO_DATA(base)
#define config_DDR(porta,dado)   IOWR_ALTERA_AVALON_PIO_DIRECTION(porta, dado)
#define cbi(porta,n)                    IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(porta, 1<<n)
#define sbi(porta,n)                    IOWR_ALTERA_AVALON_PIO_SET_BITS(porta, 1<<n)

//  Macro para uma função útil de temporização:
#define delay(retardo) usleep (1000*retardo)

//  Nomes para referência dos periféricos específicos do projeto:
#define PORT_TECLADO                    PORTA
#define PORT_LCD                        PORTB

// Protótipos para funções do LCD:
void lcd_envia_string ( char *msg );
void lcd_envia_byte (int tipo_retardo, unsigned char valor);
void lcd_configura(void);

// Protótipo para função de leitura de teclado:
int identifica_tecla ();

char tecla;
```

```
int main(void)
{

        //  Configuração do LCD e Mensagem Inicial:
        config_DDR (PORT_LCD, 0b11111111);  // Config. de porta.
        lcd_configura();                                    // Função de config. do LCD
        lcd_envia_byte (1, 0x01);                       // Limpa o LCD...

        lcd_envia_string ("Leitura-Teclado:");
        // Prepara para enviar uma nova linha:
        lcd_envia_byte (1, 0xC0);               // Cursor na segunda linha...

        // Loop: Escreve linhas com os códigos digitados no teclado numérico:
        while(1) //Teste...

        {
                // Espera o acionamento de uma tecla:" );
                while ( (tecla = identifica_tecla()) == 16)
                                ;  // O código 16 informa que não há tecla...

                // Imprime o valor da tecla, ou executa outra função:
                switch (tecla)  {
                        case 15:  //  Códigos de teclas ignoradas
                        case 14:  //  (REDIAL, '*', "P/T", "FLASH").
                        case 13:
                        case 11:
                        case 10: break;

                        case 12: // Caractere "#" é comando para limpar
                                        // a segunda linha do LCD, sem afetar a primeira.
                                lcd_envia_byte(1, 0xC0);
                                lcd_envia_string ("                ");
                                lcd_envia_byte(1, 0xC0);
                                break;

                        default:
                                // Envia para o LCD o código ASCII correspondente à
                                // tecla numérica (não se verifica quantos caracteres
                                // já foram impressos no LCD):
                                lcd_envia_byte (0, tecla + '0'); //
                                break;
                }

                // Quando houver tecla pressionada, espera que a
                // mesma seja solta, antes de prosseguir..
                while ( identifica_tecla() == tecla)
                                        ;
        }
}


//=========================================================================================
===
//
//
//   Função para o teclado numérico
//
//              A função identifica_tecla() verifica o teclado matricial de telefone,
//      informando se há uma tecla pressionada, e qual é esta tecla, se existe.
//  Caso uma combinação de teclas esteja pressionada, esta condição será
//  identificada, mas a função não identifica as teclas em questão.
//
//      A função retorna:
//              - 16, se nenhuma tecla é pressionada;
```

```
//            - 15, se alguma combinação de teclas é pressionada;
//            - 13 a 0, dependendo da tecla pressionada:
//                  -- 13: "REDIAL"
//                  -- 12: "FLASH"
//                  -- 11: '#'
//                  -- 10:  '*'  ou "P/T"
//
//            Esta função pode ser facilmente ajustada para outros tipos de teclado
//   matricial, com construção similar ao testado aqui.
//
//=========================================================================================
====

int identifica_tecla () {
    int indice;

    //  Matriz de formação dos códigos de tecla:
    //  8765:   linhas do teclado
    //      4321:  colunas do teclado.
    char tab_cod_tecla [15] = {
                              0b11010111  /* 0 */,
                              0b10111101  /* 1 */,
                              0b11011101  /* 2 */,
                              0b11101101  /* 3 */,
                              0b10111110  /* 4 */,
                              0b11011110  /* 5 */,
                              0b11101110  /* 6 */,
                              0b10111011  /* 7 */,
                              0b11011011  /* 8 */,
                              0b11101011  /* 9 */,
                              0b10110111  /* *  e também P/T  */,
                              0b01111101  /* # */,
                              0b11100111  /* FLASH */,
                              0b01111101  /* REDIAL */,
                              0b01110111  /*   */
            };

    char codigo_tecla;  // Código formado pelo acionamento de tecla.

    // Configuração da porta para verificação de linhas:
    config_DDR (PORT_TECLADO, 0b11110000);  // Bits 7 a 4 (colunas) como saídas,
                                                                    // e 3 a 0 (linhas),
como entradas.
    saida (PORT_TECLADO, 0b00001111);          // 0 nas saídas, e "pull up" nas
entradas...
    usleep (3);

    // Testa se há uma tecla pressionada, retorna em
    // caso contrário. Se não há uma tecla pressionada,
    // retorna com o valor 16
    codigo_tecla = entrada (PORTA) & 0b00001111;
    if (  codigo_tecla == 0b00001111 )
         return 16;

    //  Configuração da porta para verificação de colunas:
    config_DDR (PORT_TECLADO, 0b00001111);   // Pinos 7 a 4 como entrada, e 3 a 0 como
saídas.
    saida (PORT_TECLADO, 0b11110000);            // Saídas em 0, entradas com "pull up".
    usleep (3);

    // Gera código com informação de linha e de coluna:
    codigo_tecla = codigo_tecla | ( entrada (PORT_TECLADO) & 0b11110000);

    // Busca em tabela tecla correspondente ao código:
```

```
    for (indice = 0; indice < 15; indice++)
        if (tab_cod_tecla[indice] == codigo_tecla)
             break;

    return indice;
    // Obs.:  tecla não válida se indice = 15.
}


//=========================================================================================
===
//
//
//   Funções para o LCD:
//
//
//=========================================================================================
====

#define LCD_RS 4
#define LCD_E  5
#define LCD_BK 6



void lcd_envia_string( char *msg)  {
       char letra;
       int conta_letras = 0;

       while ( (letra = msg [conta_letras++])  != 0)
              lcd_envia_byte( 0, letra);

       return;
}

void lcd_envia_byte (int tipo_retardo, unsigned char valor) {

//  Valor: o valor a ser enviado..
//  Tipo_retardo:
//    0:  49 microsegudos, entre dados...
//    1: 2000 microsegundos, após comandos;
//    2: 5000 microsegundos, após comandos de inicialização...
//
//
       char byte_lcd;

       byte_lcd = ~(1 << LCD_RS);   // Zera o bit LCD_RS
       saida(PORT_LCD,byte_lcd);
    if (tipo_retardo == 0) // Mas se for byte de dado,
       {
       byte_lcd |= 1 << LCD_RS; // Ajusta-o para '1'.
              saida(PORT_LCD,byte_lcd);
       }


    byte_lcd &= 0xF0; byte_lcd |= (valor >> 4);  // & 0x0F;
    saida(PORT_LCD,byte_lcd);
    //PULSO_E;
    byte_lcd |= (1<<LCD_E);
    saida(PORT_LCD,byte_lcd);

    byte_lcd  &= ~(1<<LCD_E);
    saida(PORT_LCD,byte_lcd);
```

```
    switch (tipo_retardo)  {

    }

    byte_lcd &= 0xF0; byte_lcd |= valor & 0xF;
    saida(PORT_LCD,byte_lcd);
    //PULSO_E;
    byte_lcd |= (1<<LCD_E);
    saida(PORT_LCD,byte_lcd);
    byte_lcd  &= ~(1<<LCD_E);
    saida(PORT_LCD,byte_lcd);


    switch (tipo_retardo)  {
       case 2:  usleep  (5000); break;// Acumula com os próximos..

       case 1:  usleep (2000); break; // Acumula com o próximo..

       case 0:  usleep (50); break;

    }
}

void lcd_configura(void)  {

       int cont;
       char byte_lcd;

       byte_lcd = ~(1<<LCD_E); // Zera E
       saida(PORT_LCD, byte_lcd);
       byte_lcd &= ~(1<<LCD_RS);//Zera RS
       saida(PORT_LCD, byte_lcd);

  char bytes_config [] = { 0x33, 0x32,     // Nibbles básicos de configuração..
                           0x28,           // 2 linhas, caracteres 8x5
                           0x08,           // Ajusta modo de 4 linhas
                           0x0C,           // Liga display, cursor off, blinking off,
                           0x01            // Reset do dislay e do cursor.
                         };

  delay(15);
  for ( cont =0;cont < 6; cont++) {
         lcd_envia_byte (2, bytes_config [cont]);
  }
}
```