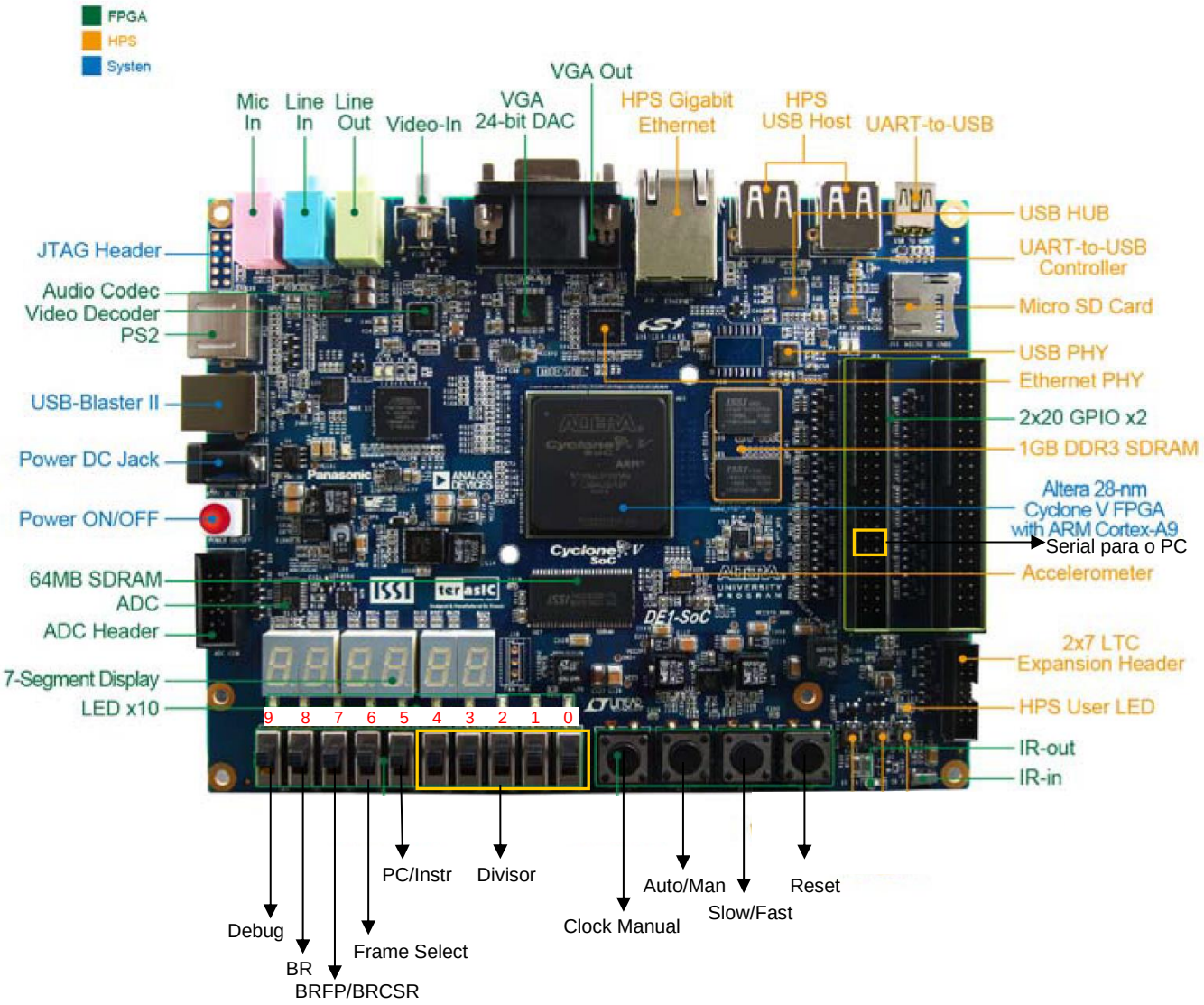
Universidade de Brasília
Departamento de Ciência da Computação

MANUAL DE UTILIZAÇÃO DA DE1-SoC
Processador RISC-V versão 2.4



Frequência do processador: CLK = 50 MHz / (SW[4:0]) Obs.: SW[4:0]=5'b00000 divide por 32
Reset: Assíncrono

Frame Select: SW[6] troca a frame que é apresentada no monitor

Modo Debug: SW[9]=1 mostra bancos de registradores na tela
SW[9]=0 não mostra

SW[8]=0 : Banco de Registradores Inteiros BR
SW[8:7]=2'b10: Banco de Registradores de Ponto Flutuante BRFP
SW[8:7]=2'b11: Banco de Registradores CSR BRCSR

Display 7 segmentos: SW[5] =0 mostra PC
SW[5] =1 mostra Instrução

Monitoramento dos Sinais de Controle:

UNICICLO	
LED	**Sinal**
LEDR[0]	Clock
LEDR[1]	Slow/Fast
LEDR[2]	Manual/Automático
LEDR[3]	Não usado
LEDR[4:9]	Não usado

MULTICICLO	
LED	**Sinal**
LEDR[0]	Clock
LEDR[1]	Slow/Fast
LEDR[2]	Manual/Automático
LEDR[3]	Não usado
LEDR[4:9]	Estado

PIPELINE	
LED	**Sinal**
LEDR[0]	Clock
LEDR[1]	Slow/Fast
LEDR[2]	Manual/Automático
LEDR[3]	Não usado
LEDR[4:9]	Não usado

Simulação por forma de onda: Waveform1.vwf ou **Waveform2.vwf**

Break Point para Debug Manual:

Use a instrução `ebreak` diretamente no seu programa.
Pressione KEY[2] para continuar a execução do programa ou KEY[3] para ir passo a passo.

No Arquivo `Parametros.v` defina a organização do processador a ser sintetizado

`define UNICICLO
`define MULTICICLO
`define PIPELINE

e a ISA a ser implementada

`define RV32I sintetiza a ISA básica de inteiros
`define RV32IM acrescenta as operações de multiplicação, divisão e resto
`define RV32IMF acrescenta as operações de ponto flutuante

Mapeamento da Memória:

Memória de Instruções (CodeMemory)

Endereço	Tamanho	Uso
0x0040 0000	64 kibytes	.text UserCodeBlock
...		
0x0040 FFFF		
...		Não Alocado

Memória de Dados (DataMemory)

Endereço	Tamanho	Uso
0x1001 0000	128 kibytes	.data UserDataBlock
...		
0x1002 FFFF		
0x1003 0000	32 kibytes	UserDataBlock1
...		
0x1003 7FFF		
0x1003 0000	64 Mibytes	SRAM
...		
0x1400 FFFF		
...		Não Alocado

Valores definidos na inicialização do Processador:
Endereço da Base da Pilha: `sp=0x1003 7FFC`
Endereço do Contador de Programa: `PC=0x0040 0000`

Arquivos Default:
Memória de Programa : de1_text.mif
Memória de Dados : de1_data.mif

Os arquivos .mif devem ser gerados pelo MIF Dump Memory do RARS.

O arquivo SYSTEMv24.s contém a rotina de tratamento de exceções/interrupções e serviços do sistema (ecalls), e o arquivo MACROSv24.s as macros e endereços necessários.

Deve-se configurar o Rars para:
-Habilitar Settings/Self-Modifying Code
-Desabilitar Settings/Exception Handler

E/S Mapeada em Memória (MMIO)

Endereço	Tamanho	Uso
0xFF00 0000	76800 bytes	Frame0 Memória de vídeo da VGA
...		
0xFF01 2C00		
0xFF10 0000	76800 bytes	Frame1 Memória de vídeo da VGA
...		
0xFF11 2C00		
...		Não Alocado
0xFF20 0000	4 bytes	KDMMIO Key Ctrl
0xFF20 0004	4 bytes	KDMMIO Key Buffer
0xFF20 0008	4 bytes	KDMMIO Disp Ctrl
0xFF20 000C	4 bytes	KDMMIO Disp Buffer
...		Não Alocado
0xFF20 0100	4 bytes	Buffer0 Teclado
0xFF20 0104	4 bytes	Buffer1Teclado
...		Não Alocado
0xFF20 0120	1 byte	Rx RS232
0xFF20 0121	1 byte	Tx RS232
0xFF20 0122	1 byte	Rx/Tx Crtl
...		Não Alocado
0xFF20 0160	4 bytes	Áudio inL
0xFF20 0164	4 bytes	Áudio inR
0xFF20 0168	4 bytes	Áudio outL
0xFF20 016C	4 bytes	Áudio outR
0xFF20 0170	4 bytes	Áudio Ctrl1
0xFF20 0174	4 bytes	Áudio Ctrl2
0xFF20 0178	4 bytes	NOTE_SYSCALL_ADDRESS
0xFF20 017C	4 bytes	NOTE_CLOCK
0xFF20 0180	4 bytes	NOTE_MELODY
0xFF20 0184	4 bytes	MUSIC_TEMPO_ ADDRESS
0xFF20 0188	4 bytes	MUSIC_ADDRESS
0xFF20 018C	4 bytes	PAUSE_ADDRESS
...	...	Não Alocado
0xFF20 0200	4 bytes	ADC_CH0
0xFF20 0204	4 bytes	ADC_CH1
0xFF20 0208	4 bytes	ADC_CH2
0xFF20 020C	4 bytes	ADC_CH3
0xFF20 0210	4 bytes	ADC_CH4
0xFF20 0214	4 bytes	ADC_CH5
0xFF20 0218	4 bytes	ADC_CH6
0xFF20 021C	4 bytes	ADC_CH7
...	...	Não Alocado
0xFF20 0500	4 bytes	IrDA_CTRL
0xFF20 0504	4 bytes	IrDA_RX
0xFF20 0508	4 bytes	IrDA_TX
0xFF20 0514	4 bytes	LFSR_ADDRESS
0xFF20 0520	4 bytes	KEYMAP0 00 a 1F
0xFF20 0524	4 bytes	KEYMAP1 20 a 3F
0xFF20 0528	4 bytes	KEYMAP2 40 a 5F
0xFF20 052C	4 bytes	KEYMAP3 60 a 7F
0xFF20 0604	4 bytes	VIDEO FRAME SELECT

- **Chamadas do Sistema (ECALLS):**

No endereço apontado por UTVEC encontra-se a rotina de tratamento de exceções/interrupções e chamadas do sistema:

Serviço	$v0	Argumentos	Resultados
Print Integer	1 101	a0=inteiro a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4 o número inteiro complemento de 2 a0 na posição (a1,a2) com as cores a3={0...0BBGGGRRRbbgggrrr} BGR fundo; bgr frente
Print Float	2 102	fa0=float a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4 o número float em fa0 na posição (a1,a2) com as cores a3
Print String	4 104	a0=endereço string a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4 a string terminada em NULL (.string) presente no endereço a0 na posição (a1,a2) com as cores a3
Read Int	5 105		Retorna em a0 o valor inteiro com sinal lido do teclado.
Read Float	6 106		Retorna em fa0 o valor float lido do teclado.
Read String	8 108	a0 endereço do buffer de entrada a1 número de caracteres máximo	Retorna no endereço a0 o conjunto de caracteres lidos, terminando com /0.
Print Char	11 111	a0=char (ASCII) a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4 o caractere a0 (ASCII) na posição (a1,a2) com as cores a3
Exit	10 110		Retorna ao sistema operacional. Na DE1-SoC trava o processador.
Read Char	12 112		Retorna em a0 código ASCII do caractere da tecla pressionada
Time	30 130		Retorna 64 bits correspondente ao tempo do sistema em ms a0 = parte menos significativa a1 = parte mais significativa
MIDI Out assíncrono	31 131	a0 = pitch a1 = duração ms a2 = instrumento (1) a3 = volume	Gera o som definido e retorna imediatamente
Sleep	32 132	a0=tempo(ms)	Aguarda a0 milissegundos
MIDI Out síncrono	33 133	a0 = pitch a1 = duração ms a2 = instrumento (1) a3 = volume	Gera o som definido e retorna apenas após o término
Print Integer Hexadecimal	34 134	a0=inteiro a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4, em hexadecimal, o número inteiro de 32 bits em a0 na posição (a1,a2) com as cores a3
Print Integer Unsigned	36 136	a0=inteiro a1=coluna a2=linha a3=cores a4=frame	Imprime na frame a4, em hexadecimal, o número inteiro de 32 bits sem sinal em a0 na posição (a1,a2) com as cores a3
Rand	41 141		a0 = número randômico de 32 bits

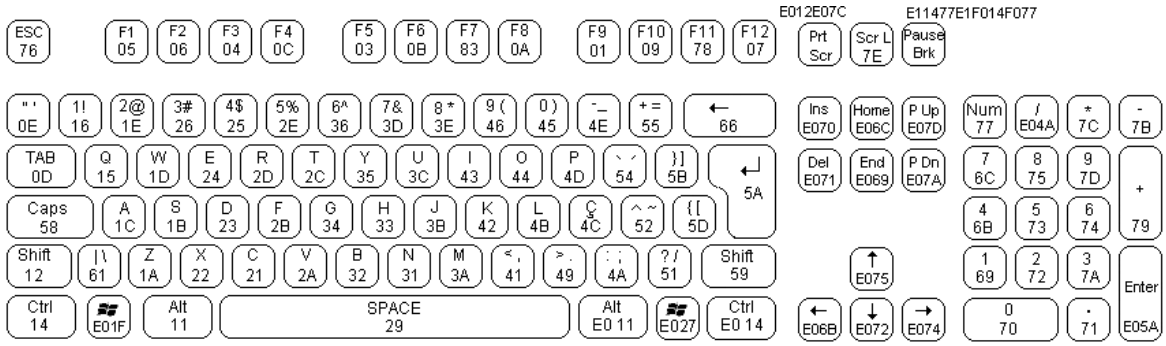
Ecalls próprios do SYSTEMv24.v

Draw Line	47 147	a0=x0 a1=y0 a2=x1 a3=y1 a4=cor a5=frame	Desenha uma linha reta do ponto (a0,a1) até o ponto (a2,a3) com a cor a4 na frame a5
Clear Screen	48 148	a0 = cor a1=frame	Limpa a frame a1(0 ou 1) com a cor a0

Os ecalls 1xx são para utilização com o Bitmap Display Tool e o Keyboard Display MMIO Tool do Rars, e funcionam de maneira idêntica com a DE1-SoC.

- **Interface Teclado PS2**

O teclado PS2 utiliza um protocolo que envia serialmente um código de 8 bits (scancode) para cada tecla pressionada. O "despressionamento" de uma tecla é identificada pelo envio do marcador 0xF0 antes do scancode da tecla que foi solta.
A figura abaixo mostra o mapeamento dos scancodes (em hexa) das teclas.



Teclas especiais (Pause, Prt Scr, setas direcionais, etc,) utilizam códigos de 2 ou mais bytes.

Há 3 formas implementadas para acessar os dados fornecidos por um teclado PS2.

1) Buffer de scancodes:
Nos endereços `Buffer0Teclado` e `Buffer1Teclado` está um buffer de 8 bytes que armazena os scancodes enviados pelo teclado.

Buffer1			
8	7	6	5

Buffer0			
4	3	2	1

O código mais recente está localizado no byte 1 do Buffer0 e o mais antigo no byte 4 do Buffer1, permitindo assim a análise da sequência de pressionamento das teclas ao longo do tempo.

2) Mapeamento das teclas (KeyMap):
Nas words dos endereços `KeyMap0`, `KeyMap1`, `KeyMap2` e `KeyMap3` está o mapeamento em 128 bits de todos os scancodes das teclas, onde bit 1 significa que a tecla está acionada no momento, 0 a tecla não está pressionada.
Ex.: Memoria[KeyMap0] & (1<<(29-1)) indica se a tecla W (scancode 0x1D = 29) está acionada ou não
A tecla ENTER (0x5A) é mapeada no bit 90, isto é, Memoria[KeyMap2] & (1<<(90-64)-1)
Deste modo pode-se verificar o pressionamento simultâneo de teclas.

Obs.: Cuidar, pois vários teclados PS2 comerciais não permitem o acionamento simultâneo de determinadas combinações de 3 ou mais teclas.

3) Compatibilidade com o Keyboard Display MMIO Tool do Rars
O endereço `KDMMIO_Ctrl` possui no seu bit menos significativo a informação se há ou não uma tecla pressionada.
O endereço `KDMMIO_Data` possui o código ASCII da tecla pressionada. Permitindo assim a detecção do pressionamento de uma tecla de cada vez.

Executar o `testePS2.s` e analisar o valor dos registradores mostrados na tela que fica mais claro as 3 formas de adquirir os dados do teclado

- ## Interface VGA:

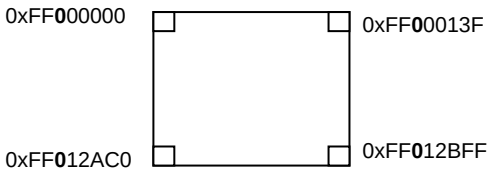
A interface VGA é idêntica ao Bitmap Display Tool do Rars.

Estão implementados 2 frames buffers localizados nos endereços `VGAADDRESSINI0` = 0xFF000_0000 e `VGAADDRESSINI1` = 0xFF10_0000 selecionáveis pelo endereço `VGAFRAMESELECT`, que deve conter a word 0 ou 1, ou pela chave SW[6].

O arquivo frame0.mif é carregado na memória VGA frame0 e o arquivo frame1.mif na frame1 ao inicializar o processador.

Mapeamento da tela VGA para resolução 240 linhas(Y) × 320 colunas(X) pixels:

Mapeamento: (X,Y) => Endereço Base (0xFF**0**0 0000) + Y × 320 + X
Com X de 0 a 319 e Y de 0 a 239.



Cor do Pixel: 8 bits

7-6	5-3	2-0
BB	GGG	RRR

Obs.: A cor 0xC7 (Magenta) corresponde ao transparente, tanto na DE1-SoC quanto no Bitmap Display Tool do Rars.
Ver exemplo: `testeVGA.s`

Obs:
O programa bmp2oac2.exe converte um arquivo .bmp (24 bits/pixel gerado pelo **paint.net**) para:

a) .bin, para uso no Rars (leitura de arquivo)
b) .mif para carregar diretamente na memória de vídeo VGA (via definição do arquivo default)
c) .s em uma linha Assembly para incluir no .data

O programa bmp2isc.exe converte um arquivo .bmp (24 bits/pixel gerado pelo **paint.net**) para:

a) .data para incluir diretamente no segmento .data

- ## Interface de áudio CODEC:

As amostras de 16 bits de áudio estéreo são lidas e escritas nos endereços indicados na tabela de mapa de memória a uma frequência de 44.1kHz.
As amostras são números de 16 bits similares às amostras de um arquivo de áudio no formato RAW.

Sincronismo do Processador com o CODEC segue o seguinte protocolo de comunicação:

I) Processador coloca o `Ctrl2` em 0, indicando ao CODEC para enviar uma amostra;
II) Processador aguarda CODEC colocar `Ctrl1` em 1, que indica que uma amostra está pronta;
III) Processador lê os valores nos endereços `inL` e `inR` e escreve os valores nos endereços `outL` e `outR`;
IV) Processador coloca o `Ctrl2` em 1, indicando ao CODEC que terminou de ler e escrever as amostras;
V) Processador aguarda CODEC colocar `Ctrl1` em 0, que indica que o CODEC está pronto novamente
VI) Reinicio do ciclo

Onde o bit [0], nas palavras de controle `Ctrl1` e `Ctrl2`, corresponde ao canal R e o bit [1] ao canal L. Com isto é possível o controle, aquisição e síntese de áudio para os 2 canais de forma independente.

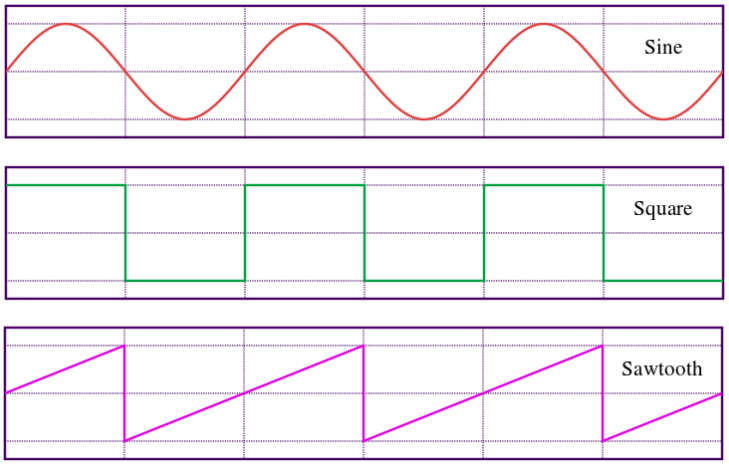
Ver exemplo: `testeAUDIO.s`

- **Interface de áudio: Sintetizador Polifônico (by Lucas Carvalho)**

O sintetizador de áudio utilizado no curso de OAC é um sintetizador polifônico compacto capaz de tocar até 8 notas simultaneamente. Foi baseado no padrão MIDI e é formado por três sistemas:

Oscilador:

Gera formas de onda em várias frequências. É responsável por criar o som base de cada nota em 3 possíveis formatos:

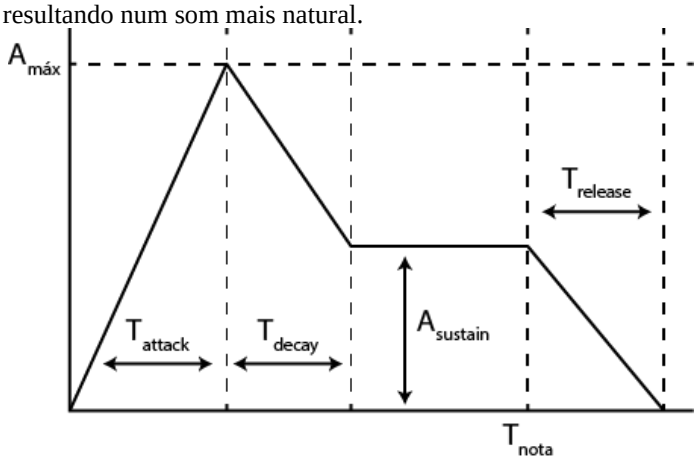


Filtro:

Um filtro passa-baixas pode ser ativado ou não na saída do oscilador. Sua frequência de corte é a própria frequência da nota e seu fator de qualidade é configurável.

Envelope:

O tempo de vida de uma nota é dividido em 4 seções: *Attack*, *Decay*, *Sustain* e *Release*. Elas determinam uma variação na amplitude da onda resultando num som mais natural.



Interface

Para a implementação do sintetizador no RISC-V é necessário utilizar o módulo *Sintetizador*, que encapsula todos os sistemas.

```
/*
 * Entradas do CODEC de áudio da própria DE2.
 */
input AUD_DACLRCK;
input AUD_BCLK;
```

Nestas entradas devem ser passados os pinos da DE1-SoC de mesmo nome. São dois pinos de entrada que controlam a taxa de amostragem do CODEC.

```
/*
 * Comando de início/fim de uma nota.
 */
input NOTE_PLAY;
input [6:0] NOTE_PITCH;
```

O sintetizador funciona recebendo comandos para que inicie ou termine uma nota. Por exemplo, caso queira tocar 3 notas simultâneas, você deve enviar 3 comandos de início seguidos.

Para enviar um comando atribua a *NOTE_PLAY* um valor de 1 para iniciar ou 0 para terminar uma nota. *NOTE_PITCH* determina a nota deste comando, seguindo o padrão MIDI (pesquise por *MIDI Note Table*).

Em toda subida de *AUD_DACLRCK*, um comando é lido. Comandos repetidos ou inválidos (terminar uma nota que nunca foi iniciada) são entendidos como se não houvesse comando.

```
/*
 * Configurações do oscilador.
 */
input [1:0] WAVE;
input NOISE_EN;
```

A entrada *WAVE* define qual forma de onda será gerada pelo oscilador, como mostra a seguinte tabela de referência.

0	1	2	3
Mudo	Senoide	Quadrada	Dente-de-serra

O oscilador adicionará um certo ruído à onda gerada se *NOISE_EN* assumir valor 1. Este mecanismo é útil na criação de efeitos sonoros. Sintetizadores de vídeo games antigos utilizavam uma técnica similar para simular explosões, impacto, dano etc.

```
/*
 * Configurações do filtro.
 */
input FILTER_EN;
input [7:0] FILTER_QUALITY;
```

O filtro passa-baixas é ativado se a entrada *FILTER_EN* estiver alta. Sua frequência de corte é a própria frequência da nota sendo filtrada e seu fator qualidade pode ser configurado pela entrada *FILTER_QUALITY*, que é interpretada como ponto fixo Q8.

O fator de qualidade controla a banda de frequências que passa pelo filtro. Caso assuma o valor zero, o filtro se comporta como um passa-baixas comum, sem ressonância. Uma onda quadrada filtrada com fator de qualidade alto se aproxima bastante de uma senoide.

```
/*
 * Configurações do envelope aplicado durante a vida de uma
nota.
 */
input [6:0] ATTACK_DURATION;
input [6:0] DECAY_DURATION;
input [6:0] SUSTAIN_AMPLITUDE;
```

Toda nota tem uma envoltória aplicada a sua onda enquanto é tocada. Uma tecla de piano, ao ser pressionada, gera um som alto inicial que decai e mantém uma altura constante. Quando a tecla é solta a nota some aos poucos até ficar inaudível. Este processo é simulado em sintetizadores por meio do envelope de 4 seções mostrado anteriormente.

A duração de cada fase é configurável, exceto a de *Sustain* cuja amplitude é variável. A escala destes valores é, assim como definido no padrão MIDI, dependente da implementação, ou seja, é necessária uma certa experimentação.

```
/*
 * Amostra de saída do sintetizador.
 */
output [15:0] SAMPLE_OUT;
```

Os sistemas internos do sintetizador processam todas as notas e produzem uma nova amostra a cada subida do clock de amostragem *AUD_DACLRCK*. Esta amostra deve ser repassada ao CODEC da DE1-SoC. Este processo não é parte do sintetizador e já está implementado no processador RISC-V do curso.

- **Interface de áudio: Sintetizador Polifônico (Luiz Henrique Campos)**

1 - Foi criada uma memória de dados de duas portas UserDataBlockDouble. Uma porta de escrita e leitura para o uso da CPU e uma porta somente de leitura para o sintetizador (na pasta item6/core/memoria). FOI RETIRADA ESSA CARACTERÍSTICA

2 - Outra PLL foi usada para gerar o AUD_CTRL_CLK e "afinar" o sintetizador.

3 - Outros Parâmetros adicionados :

```
NOTE_SYSCALL_ADDRESS        = 32'hFF20_0178,
NOTE_CLOCK                  = 32'hFF20_017C,
NOTE_MELODY                 = 32'hFF20_0180,
MUSIC_TEMPO_ADDRESS         = 32'hFF20_0184,
MUSIC_ADDRESS               = 32'hFF20_0188,
PAUSE_ADDRESS               = 32'hFF20_018C
```

4 - Adição do módulo SynthControl que toca a música sequencialmente na memória de dados está na pasta **/core/Sintetizador**.

Existem 2 tipos de Configuração para a nota na memória, uma para ecall e outra para a música que será tocada sequencialmente. FOI RETIRADA ESTA CARACTERÌSTICA

Configuração para notas que serão tocadas sequencialmente:

Melody – 1 se for melodia; 0 se for acorde. Se esse bit for 1, a próxima nota só será tocada ao fim desta. Se for zero, a próxima nota será tocada na próxima borda de subida de AUD_DACLRCK.
Instrument – 16 possibilidades de instrumentos (0 - 15).
Volume - 128 possibilidades de amplitude da nota (0 – 127).
Pitch – 128 possibilidades de notas musicais segundo o padrão MIDI (0 – 127).
End – A última nota da música deverá setar esse bit para 1 para o controlador do sintetizador parar de percorrer a memória.
Repeat – Se esse bit estiver setado para 1, a próxima nota será a nota salva no endereço inicial.
Duration (Q4) – Duração em ponto fixo de cada nota. O valor é relativo à duração em milissegundos da Nota Base (Semínima). O valor da duração da Nota Base em milissegundos será passado para o endereço 0xFFFF020C. A partir daí o hardware atribuirá a duração em milissegundos equivalente para cada nota baseado em seus 8 primeiros bits (Duração – Ponto Fixo Q8). Dica: Instale a fonte Maestro Times para visualizar corretamente.

𝅝= Semibreve 4 x Nota Base (0100.0000)

𝅗𝅥= Mínima 2 x Nota Base (0010.0000)

♩= Semínima Nota Base (0001.0000)

♪= Colcheia ½ x Nota base (0000.1000)

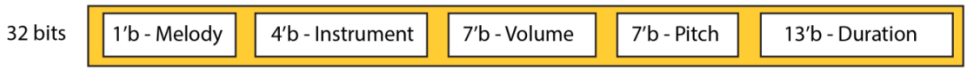
♫= Semicolcheia ¼ x Nota Base (0000.0100)

𝅘𝅥𝅯= Fusa 1/8 x Nota Base (0000.0010)

𝅘𝅥𝅰= Semifusa 1/16 x Nota Base (0000.0001)

Por exemplo, se a duração da Nota base for 400 milissegundos, a duração de (0000.0010) será 400 x 1/8 = 50 milissegundos.

Configuração para notas do ecall (31 e 33):



A word será salva no endereço 0xFF10 0200.
Melody – 1 se for ecall 33; 0 se for ecall 31. Em resumo, se esse bit for 1, a próxima nota só será tocada ao fim desta. Se for zero, a próxima nota será tocada na próxima borda de subida de AUD_DACLRCK.
Instrument – 16 possibilidades de instrumentos (0 - 15). (SÓ 1 IMPLEMENTADO)
Pitch – 128 possibilidades de notas musicais segundo o padrão MIDI (0 – 127).
Duration – Duração da nota em milissegundos.

- **Interface Infravermelho: (Grupo 2 02/2016)**

A sigla IRDA (*Infrared Data Association*) faz referência a um conjunto de protocolos para comunicação sem fio entre dispositivos. A transferência de dados é feita na forma de pacotes que são enviados serialmente. A transmissão começa com o envio de 1 bit de início, seguido por 8 bits de dados e 1 bit de paridade e termina com a transferência de 1 bit de parada. Não é possível enviar e receber pacotes de dados simultaneamente. Para realizar a comunicação entre dispositivos, são necessárias uma porta de recepção (*receiver*) e uma porta de transmissão (*transmitter*).

IRDA_CONTROL	Controle do IRDA	0xFF20 0500
IRDA_RXD	Receptor IRDA	0xFF20 0504
IRDA_TXD	Transmissor IRDA	0xFF20 0508

IRDA_CONTROL (4 bytes)- wReadData[1] indica que tem dado pronto para leitura e wReadData[0] se existe dado sendo transmitido.

IRDA_RXD (4 bytes)- é de onde vai ser lido o dado do Irda

IRDA_TXD (4 bytes)– é o dado que vai ser transmitido pelo Irda (não tem na DE2-70, e não foi testado na DE1-SoC)

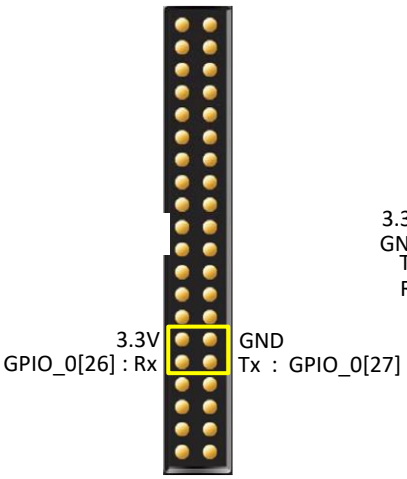
- **Interface LFSR: (Grupo 2 02/2016)**

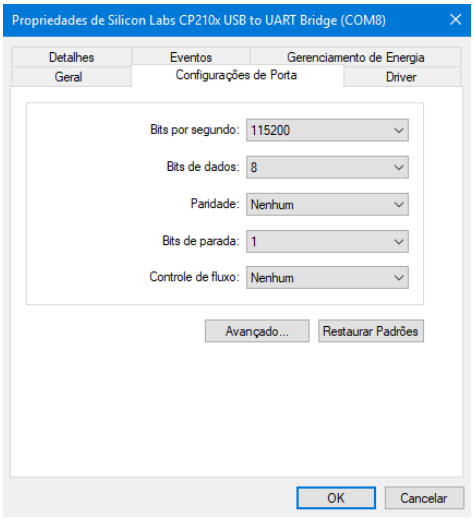
`LFSR_ADDRESS = 0xFF20_0514`

Retorna um número pseudoaleatório que foi gerado pela técnica de *Linear Feedback Shift Register* (LFSR).

- **Interface Serial RS232C / USB:**

Comunicação do processador RISC-V com o PC é feita pela transmissão/recepção serial byte a byte através de um módulo conversor RS232C / USB CP1202 para Arduino, mostrado na figura abaixo:





Este módulo cria uma porta serial COM no PC.
A configuração desta porta, através do Gerenciador de Dispositivos / Portas (COM e LPT), deve seguir ao mostrado na figura acima.
No programa em C, a ser executado no PC, a linha que define o número da porta a ser usada e o baud rate é:

```
  int  cport_nr = 7,        /* usar o número da COM - 1 */
       bdrate   = 115200;
```

Endereços de MMIO do RISC-V:

```
RX_ADDRESS      = 0xFF20_0120
TX_ADDRESS      = 0xFF20_0121
CRTL_ADDRESS    = 0xFF20_0122
```

Byte de Controle:

x	x	x	x	x	Ready	Busy	Start

Procedimentos:

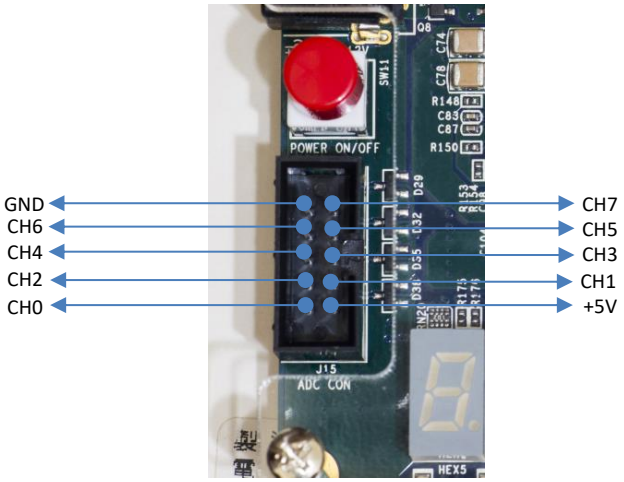
Transmissão: RISC-V → PC
- I- RISC-V escreve o byte a ser enviado no endereço TX
- II- RISC-V ativa o bit Start do controle no endereço CTRL
- III- RISC-V desativa o bit Start do controle no endereço CTRL
- IV- RISC-V aguarda o bit Busy do controle ser desativado para fazer nova transmissão

Recepção: RISC-V ← PC
- I- RISC-V aguarda o bit Ready do controle ser ativado
- II- RISC-V lê o byte recebido do endereço RX
- III- RISC-V aguarda o bit Ready do controle ser desativado

Ver exemplos: `testeRS232.s` e `testeRS232.c`, `demo_tx.c`, `demo_rx.c`
Detalhes em http://www.fpga4fun.com/SerialInterface.html

- **Interface com Conversor Analógico-Digital:**

A DE1-SoC possui um conversor Analógico-Digital (ADC) com 8 canais de 12 bits cada.
O acesso se dá através dos pinos da interface ADC COM abaixo listados.



Os sinais de entrada analógicos devem estar entre 0 e 5V, e os valores convertidos em 12 bits são acessáveis pelos endereços:

```
ADC_CH0 = 0xFF20_0200
ADC_CH1 = 0xFF20_0204
ADC_CH2 = 0xFF20_0208
ADC_CH3 = 0xFF20_020C
ADC_CH4 = 0xFF20_0210
ADC_CH5 = 0xFF20_0214
ADC_CH6 = 0xFF20_0218
ADC_CH7 = 0xFF20_021C
```

A frequência de amostragem deve ser controlada pelo software, pois não há sinal de interrupção.

Ver exemplo: `testeADC.s`