# Usando o LCD Shield Arduino no kit FRDM-Kl25Z da Freescale



Por Eng. Roberto Bairros dos Santos

www.bairrospd.com

Data: 15/10/2016

# Sumário

Introdução	3
Compatibilidade com a placa UNO do Arduino	6
O componente LCDHTA	8
Como configurar o componente LCDHTA.	10
A programação	13
Referências:	16

## Introdução.

Este tutorial irá amostrar como usar o shield com LCD e teclado mais comum para a linha Arduino montado no KIT FRDM-KL25Z da Freescale.

O KIT FRDM-KL25Z da Freescale é um kit didático similar ao Arduino, possui um hardware compatível com a placa Arduino UNO de forma a permitir usar os kits, sensores e tudo mais que existe da linha Arduino.



A vantagem do KIT FRDM-KL25Z da Freescale é possuir um microcontrolador MKL25Z128VLK4 de 32 bits ARM CORTEX-MO com 128K de memória Flash e 16KSRAM, com ADC de 16 bits e 53 pinos de IO digital.

A placa do kit já vem com acelerômetro, três leds coloridos, um sensor de toque que simula um potenciômetro.

Você pode conferir os detalhes da placa no link <u>www.freescale.com/FRDM-KL25Z</u>.

O kit já vem com o cabo de programação, como o Arduino.



Figure 2. FRDM-KL25Z Feature Call-outs



A programação é feita via software chamado CodeWarrior baixado grátis no link dado anteriormente.



#### Compatibilidade com a placa UNO do Arduino.

O KIT apresenta um hardware compatível com a placa Arduino UNO de forma que você pode encaixar todos os shields do Arduino diretamente na placa do KIT.

A pinagem é mostrada na figura a seguir.



A interligação do shield é mostrada a seguir.



A Relação entre os endereços dos pinos do LCD e os pinos do kit é mostrada na figura a seguir.

Arduino	KL25z
D9	PTD5
D8	PTA13
D4	PTA4
D5	PTA5
D6	PTC8
D7	PTC9
	Arduino D9 D8 D4 D5 D6 D7

#### O componente LCDHTA

Este trabalho foi baseado na publicação de Erik Stiger no link a seguir.

https://mcuoneclipse.com/2012/12/22/hd44780-2x16-character-display-for-kinetis-and-freedom-board/

Para usar o LCD com o CodeWarrior é preciso usar dois componentes criados pelo Erik Stiger o LCDHTA e o WAIT disponível no link descrito a seguir.

http://steinerberg.com/EmbeddedComponents/LCDHTA/home.htm

http://steinerberg.com/EmbeddedComponents/Wait/home.htm

Faça o download dos dois arquivos \*.PEupd Importe para o CodeWarrior usando o menu descrito na figura a seguir.

Process	or Expert	Window	/ Helj	ρ
Sh	ow Views			
Hide Views				
Import Component(s)				
The state		N	+1 0	

Você pode ver o post do link a seguir que mostra detalhes de como baixar toda a biblioteca criada pelo senhor Erik Stiger.

http://mcuoneclipse.com/2013/05/09/processor-expert-component-peupd-files-on-github/

Você pode pegar este componente na aba da livraria como indica a figura a seguir.

Note que ao carregar o LCDHTA o componente WAIT será carregado também, este é um componente que gera atraso de tempo em ms equivale ao delay(ms) do Arduino!

<u>W</u> indow <u>H</u> elp			
• 🔌 🏇 • 🔗 • ⊿ 🖢 •	- {□ - \$\$ - \$ +   ≥	Quick Access	部   記 C/C++ 参 Debug
S Component Inspector - EN1	Scomponents Library 🛛		B 👗 🗸 🗆 🗆
Categories Alphabetical Assista	ant Processors		
Component	Component Level		^
> Board Support			
🔽 🚌 CPU External Devices			
> 🗁 Communication			
🗸 🗁 Display			
ChLCD	Low		
ConsolelO	High		
FontDisplay	High		
FreeMaster	High		
GDisplay	High		
GFont	High		
CentChLCD	Low		
C CDHTA	High		
led 🛞	Low		
EDbit	Low		

#### Como configurar o componente LCDHTA.

O senhor Erik Stiger diz que 'HTA' é a abreviatura da Universidade de Lucerne onde ele leciona!

O software oferece os métodos descritos abaixo onde estão assinalados os métodos que serão usados neste tutorial.



Antes de configurar os pinos você deverá alterar uma configuração da CPU pois o pino PTA4 também usada como interrupção NMI!

Vá na aba do PE Cpu altere a configuração NMI desabilitando-a, como mostra a figura a seguir.

C/C++ - LCD_03/Sources/main.c - CodeWarrior Development Studio					- 6	s ×
Eile Edit Source Refactor Search Project MQX Tools Processor Expert	<u>R</u> un	Window Help				
📸 🔻 🔚 🕼 (Active) 🚽 🐔	<i>§</i> -	🔍 🔅 🕈 🕶 🛷 🔹 🌛 🖄	• 🖗 • 🍫 • 🗢 • 🔤 -	Quick Access	📑 🖬 🗟 C/C++	🅸 Debug
The CodeWarrior Projects 🖾 📼		S Component Inspector - Cpu	🛿 🗞 Components Library		Basic Advanced	~
	~	Properties Methods Events	Build options Resources			
😪 Components - LCD_03 🕱 📃 💼 🔮 🍸 📟		Name	Value	Details		
> 🗁 OSs	^	CPU type	M 28VLK4			
Processors		> Clock settings				
> @ Cpu:MKL25Z128VLK4		Initialization priority	Tabled	1		
		Watchdog disable				
> > Referenced Components		CPULinterrunts/resets				
V 🖨 LCD1:LCDHTA	-	V NMI interrupt	Disabled			
> I EN1:BitlO[LCDHTA\CtrlPin]		Interrupt	INT_NMI	Property is disabled		
RS1:BitIOILCDHTA\CtrlPin]		7 Hard Fault	Disabled		,	
> DB41:BitlOILCDHTA\DBpin1		> Bus Fault	Disabled			
> DB51:BitlOILCDHTA\DBpin]		> Usage Fault	Disabled			
> DB61:BitlO[LCDHTA\DBpin]		> Supervisor Call	Disabled			
		> Pendable Service	Disabled			

O próximo passo é configurar as propriedades do componente LCD1:LCDHTA para operar com 4 bits sem usar o R/W para isto desabilite o campo Ready from Display e o campo Dat/Control Bus opção DB0...DB3 já que somente os bits DB4 a DB7 serão usados, o restante não precisa alterar.

Component Inspector - LCD1 🔅	S Components Library		Basic Advanced	2
roperties Methods Events				
Name	Value	Details		
Component name	LCD1			
HW Interface				
LCD Type	generic			
LCD Lines	2	D		
Characters per Line	16	D		
✓ Line Addresses				
Line 1	00	H		
Line 2	40	H		
Line 3	10	H		
Line 4	50	H		
> LCD Enable Signal	Disabled			
Kead from Display	Disabled			
Wart (us)	0	D		
E signal	CtrlPin			
> E2	Disabled			
RS signal	CtrlPin			
v Data/Control Bus				
Data/Control Bus Width	4bit			
> DB0DB3	Disabled			
<ul> <li>DB4DB7</li> </ul>	Enabled			
DB4	DBpin			
DB5	DBpin			
DB6	DBpin			
DB7	DBpin			
> Bits/Byte Bus	Disabled			
<ul> <li>System Interface</li> </ul>				
Wait	WAIT1			

Agora você já pode alterar os pinos das I/O do componente LCD1:LCDHTA como descrito abaixo, estes são os pinos da placa compatíveis com o shield do LCD Arduino!



Com o componente configurado não esqueça de gerar o código do PE, depois parta para o programa exemplo.

C/C++ - LCD_03/Sources/main.c - 0	CodeWarrior Development S	Studio			
<u>File Edit Source Refactor Searce</u>	h <u>P</u> roject MQX Tools P	Processor E <u>x</u> pert <u>R</u> un	<u>W</u> indow	<u>H</u> elp	
📬 🔻 🔚 😨 🛛 🍕 🕶 📷 (Active)	~ 🐔	<i>₿</i> ▼	×	🏇 🕶 🔗 🕶 🌛 🖢	• 🖓 • 🍫 🔶 • 🔿
CodeWarrior Projects			🚫 Com	ponent Inspector - LCD1	🛿 🖠 Components Lib
			Properti	es Methods Events	
ಆ-Components - LCD_03 ⊠	۵ 🗈	🖻 🔁 🔻 🗖	Name		Value
> 🗁 OSs		Generate Process	or Expert C	ode acters per Line	16
✓			•	Addresses	
> 💮 Cpu:MKL25Z128VLK4				Line 1	00

#### A programação.

O programa irá mostrar na primeira linha do LCD a mensagem "Ola KL25Z" e na segunda linha irá mostrar o tempo em segundos!



Antes do programa contido no laço infinito você deve limpar o display com a instrução LCD1\_Clear(), esta função gasta algum tempo executando esta tarefa, por isto, uma função WAIT(1000) de 1000ms deverá ser colocada logo depois do Clear(), por este mesmo motivo o CLEAR deve ser evitado no programa.



Para montar a mensagem com letras e valores numéricos eu prefiro usar a função padrão da linguem "C" sprintf(destivo,string,variáveis) e depois usar a função WriteString(char \*String) do componente LCDHTApara escrever no display, onde a String é a mesma variável destino da função sprintf.

sprintf( destino,string,variaveis);	WriteString(char *String);
LETRAS + NÚMEROS	01a KL25Z 219

Eu também preenchi os espaços em branco para evitar que apareça caracteres indesejáveis na mensagem para isto montei o padrão no formato de comentário para orientar a posição das letras. O exemplo abaixo mostra como escrevo a primeira linha.

```
char strlcd[17];//string a ser usada no programa oara escrever no LCD de 16 colunas.
for(;;) {
   LCD1_GotoXY(1,1);//posiciona primeira linha
   //strcpy(strlcd, "abcdefghijklmnop");
   sprintf(strlcd, "Ola KL252 ");//completar 16 col com espaços vazios
   LCD1_WriteString((char*)strlcd);
```

O programa completo é descrito abaixo, você pode baixar o pdf no site <u>www.bairrospd.com</u> copiar e colar o código.

```
/* Write your code here */
LCD1_Clear();//limpara tudo no inicio, a limpeza demora tempo
WAIT1_Waitms(1000);
int tempo=0;
 char strlcd[17];//string a ser usada no programa oara escrever no LCD de 16 colunas.
       for(;;) {
               LCD1_GotoXY(1,1);//posiciona primeira linha
          //strcpy(strlcd,"abcdefghijklmnop");
                                               ");//<u>completar</u> 16 <u>col</u> <u>com</u> <u>espaços</u> <u>vazios</u>
               sprintf(strlcd,"<u>Ola</u> KL25Z
               LCD1_WriteString((char*)strlcd);
               LCD1_GotoXY(2,1);
          //strcpy(strlcd,"abcdefghijklmnop");
               sprintf(strlcd,"%d",tempo);//completar 16 col com espaços vazios
               LCD1 WriteString((char*)strlcd);
               WAIT1_Waitms(1000);
               tempo++;
```

}//FIM DO LAÇO



#### Referências:

Freescale FRDM-KL25Z Referências: <u>www.freescale.com/FRDM-KL25Z</u>.

Post do site MCU on Eclipse de Erich Styger: <u>https://mcuoneclipse.com/2012/12/22/hd44780-2x16-character-display-for-kinetis-and-freedom-board/</u>

Componente LCDHTA autoria de Erich Styger : http://steinerberg.com/EmbeddedComponents/LCDHTA/home.htm

Componente WAIT autoria de Erich Styger: http://steinerberg.com/EmbeddedComponents/Wait/home.htm