Capítulo 1

Ambiente de Desenvolvimento – *Hardware*

Autores: Wu Shin-Ting e Antônio Augusto Fasolo Quevedo

Neste capítulo faremos uma breve apresentação dos recursos de *hardware* do ambiente de desenvolvimento que temos à disposição para realizar os experimentos ao longo deste semestre: o *kit* de rápida prototipagem da Freescale FRDM-KL25Z e a placa auxiliar EA871 (*shield*), desenvolvida pela equipe SATE/FEEC/Unicamp sob a supervisão do Prof. Antonio Augusto Quevedo da FEEC/Unicamp. Esta placa auxiliar é acoplada ao *kit* FRDM-KL25, estendendo a quantidade de periféricos.

1.1 FRDM-KL25Z

FRDM-KL25Z é uma plataforma de suporte à prototipação de projetos baseados em microcontroladores. Além de incluir um microcontrolador MKL25Z128VLK4 da série de Kinetis L, um oscilador de cristal de 8MHz e um circuito de alimentação, ele dispõe de um conjunto básico de periféricos conectados ao microcontrolador e uma interface de depuração OpenSDA que nos permite depurar os programas carregados no microcontrolador através do console de um computador pessoal/*laptop*. Há ainda no *kit* 4 *I/0 headers*, através dos quais pode-se ter acesso a um subconjunto de pinos físicos do microcontrolador. Isso facilita a integração de novos periféricos, a avaliação e os testes de novas ideias, sem nos preocupar com os detalhes da implementação do *hardware*.

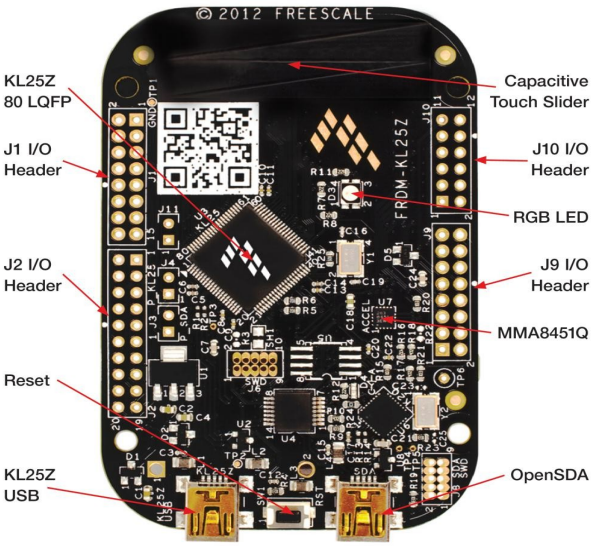


Figura 1: *Kit* de desenvolvimento FRDM-KL25Z

1.1.1 Microcontrolador Kinetis KL2x

O “comandante” da placa é o microcontrolador MKL25Z128VLK4 de 32 *bits*, da série Kinetis L da *Freescale* [1]. Este componente é um *System-on-a-Chip* (SoC), isto é, num *chip* são agregados um núcleo de processamento Cortex-M0+ de arquitetura *ARM*® e vários módulos, como temporizadores, interfaces de comunicação, conversores DA/AD, controlador de interrupções, e unidades de memória. Aliados baixo custo e baixo consumo de energia ao tamanho reduzido e à flexibilidade no desenho de um novo projeto, ele constitui uma alternativa para desenvolver aplicativos portáteis e de alto desempenho.

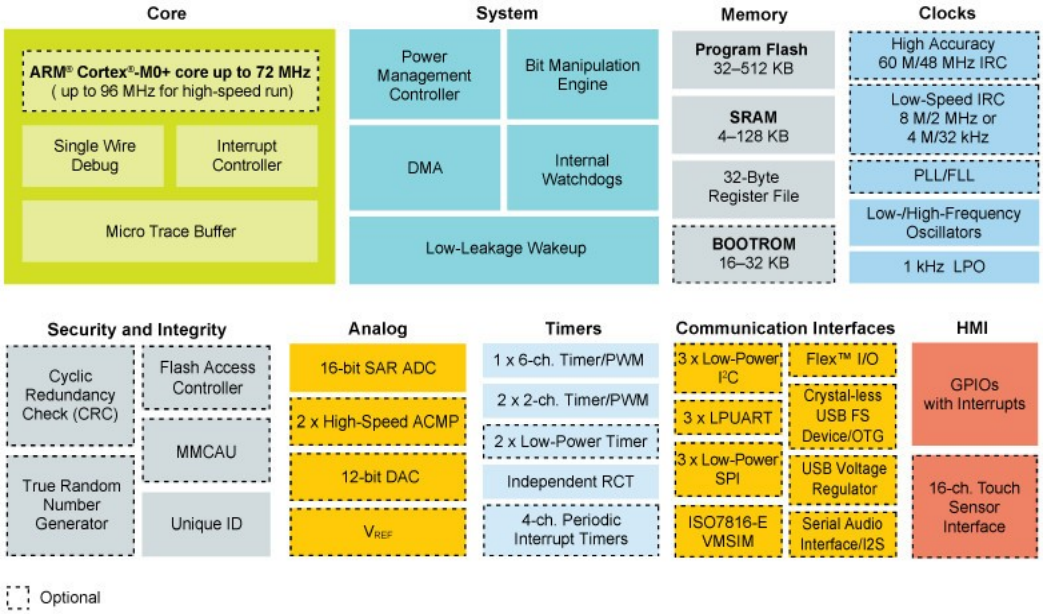


Figura 2: Diagrama de blocos do microcontrolador Kinetis KL2x

1.1.1.1 Processador ARM

O “cérebro” do microcontrolador MKL25Z é o núcleo ARM Cortex-M0+ [2]. O processador ARM (***A**dvanced **R**ISC **M**achine*) é um processador de arquitetura RISC (***R**educed **I**nstruction **S**et **C**omputer*) usado principalmente em microcontroladores, Ele suporta operações aritméticas inteiras (operações lógicas e de deslocamento *bit*-a-*bit*, soma, subtração e multiplicação) sobre operandos de tamanho de 1 (*byte*), 2 (*halfword*), 4 (*word*) e 8 (*doubleword*) *bytes*. E duas representações binárias de operandos são adotadas: quando se trata de um número inteiro sem sinal o formato normal na base binária é utilizada, e quando se trata de um número inteiro com sinal o complemento de 2 é usado.

Mesmo com um número maior de instruções, os processadores RISC têm a vantagem de apresentar um tempo de execução menor, pois suas instruções são usualmente executadas num único ciclo de relógio. Para reduzir a demanda do espaço de memória, o processador ARM suporta, além do repertório de instruções ARM de 32 *bits*, um subconjunto de instruções equivalentes, *Thumb*, de 16 *bits*. Quando o processador está operando no modo *Thumb*, as instruções de 16 *bits* são automaticamente expandidas em suas instruções equivalentes de 32 *bits* antes das suas execuções [5]. Vale ressaltar que o processador

só consegue processar as instruções do modo em que ele se encontra. Instruções de desvio em *assembly*, como BL ou BLX, podem ser utilizados para chavear entre os modos ARM e *Thumb*. Se o *bit* menos significativo do endereço de desvio é 0, o modo é ARM; do contrário, chaveia-se ou mantém-se no modo *Thumb*. O nosso processador é configurado para operar no modo *Thumb*.

A fim de aumentar a vazão da execução das instruções, o fluxo de processamento de instruções no processador ARM é de dois estágios: enquanto se busca (*fetch*) uma nova instrução, é executada (*execute*) a instrução anterior [6]. Com isso, o tempo total de execução de uma instrução é, em média, um ciclo de relógio. Observe que o tempo é um tempo médio porque nem todas as instruções, como as de desvio, permitem esta paralelização.

A arquitetura ARM é dita mapeada em memória, uma vez que tanto os periféricos quanto as unidades de memória conectados ao processador compartilham o mesmo espaço de endereços de 32 *bits*. Como a menor unidade endereçável é um *byte*, cada endereço deste espaço é associado a um *byte*. Portanto, o nosso processador consegue endereçar até um espaço de 2^{32} *bytes*, ou 4GB. O espaço de endereços é dividido em várias partições onde são mapeadas diferentes periféricos integrados no microcontrolador MKL25Z, como ilustra a Figura 3. O sufixo 128 no nome do nosso microcontrolador, MKL25Z128, indica que o microcontrolador é provido de uma memória *flash* de 128KB lotada no subespaço de endereços 0x00000000-0x07FFFFFF e uma memória RAM de 16KB mapeada no subespaço 0x1FFFF000-0x20002FFF. Os endereços das instruções armazenadas na memória *flash* processáveis pelo nosso processador são, portanto, menores que 0x08000000 e os endereços dos dados dos programas armazenados na memória RAM, são os que estão no intervalo 0x1FFFF000-0x20002FFF.

System 32-bit Address Range	Destination Slave	Access
0x0000_0000–0x07FF_FFFF[1]	Program flash and read-only data (Includes exception vectors in first 196 bytes)	All masters
0x0800_0000–0x1FFF_EFFF	Reserved	—
0x1FFF_F000-0x1FFF_FFFF[2]	SRAM_L: Lower SRAM	All masters
0x2000_0000-0x2000_2FFF[2]	SRAM_U: Upper SRAM	All masters
0x2000_3000–0x3FFF_FFFF	Reserved	–
0x4000_0000–0x4007_FFFF	AIPS Peripherals	Cortex-M0+ core & DMA
0x4008_0000–0x400F_EFFF	Reserved	–
0x400F_F000–0x400F_FFFF	General purpose input/output (GPIO)	Cortex-M0+ core & DMA
0x4010_0000–0x43FF_FFFF	Reserved	–
0x4400_0000–0x5FFF_FFFF	Bit Manipulation Engine (BME) access to AIPS Peripherals for slots 0-127[3]	Cortex-M0+ core
0x6000_0000–0xDFFF_FFFF	Reserved	–
0xE000_0000–0xE00F_FFFF	Private Peripherals	Cortex-M0+ core
0xE010_0000–0xEFFF_FFFF	Reserved	–
0xF000_0000–0xF000_0FFF	Micro Trace Buffer (MTB) registers	Cortex-M0+ core

System 32-bit Address Range	Destination Slave	Access
0xF000_1000–0xF000_1FFF	MTB Data Watchpoint and Trace (MTBDWT) registers	Cortex-M0+ core
0xF000_2000–0xF000_2FFF	ROM table	Cortex-M0+ core
0xF000_3000–0xF000_3FFF	Miscellaneous Control Module (MCM)	Cortex-M0+ core
0xF000_4000–0xF7FF_FFFF	Reserved	–
0xF800_0000–0xFFFF_FFFF	IOPORT: GPIO (single cycle)	Cortex-M0+ core

Figura 3: Mapeamento dos módulos no espaço de endereços disponíveis (Tabela 3-27 em [1])

Entendendo a forma de endereçamento no processador ARM, vamos ver como estes endereços devem ser vistos por um programador que declara os seus dados como um dos 4 tipos suportados pelo processador. Cada dado do tipo *byte* é endereçável por um endereço único. Um dado do tipo *halfword* ocupa 2 *bytes*, 1 *byte* de endereço A e outro *byte* de endereço A+1. Um dado do tipo *word* ocupa 4 *bytes*, nos endereços A, A+1, A+2 e A+3. E, finalmente, um dado do tipo *doubleword* "gasta" 8 endereços, A, A+1 … A+7. Daí a afirmação de que os endereços dos dados do tipo *halfword*, *word* e *doubleword* são divisíveis por (alinhados com múltiplos de) 2, 4 e 8, respectivamente.

Quando os dados têm tamanho maior que um *byte*, há duas alternativas de armazenamento destes *bytes*.: *little-endian* e *big-endian*. Em *little-endian* o *byte* menos significativo é armazenado no *byte* de endereço menor (Figura 4.a), enquanto no sistema de memória *big-endian*, o *byte* menos significativo tem endereço mais alto (Figura 4.b). No nosso microcontrolador o sistema de memória é configurado em *little-endian*.

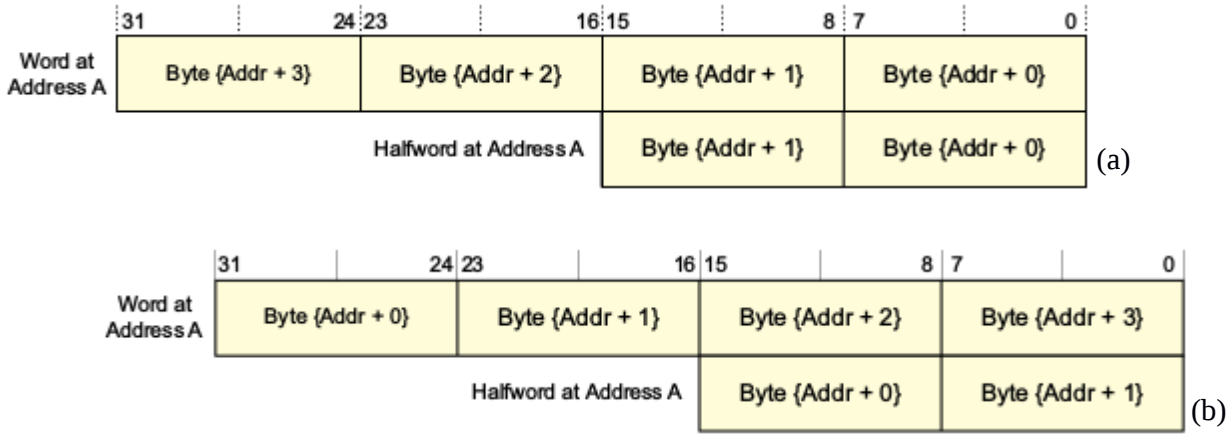


Figura 4: Sistema de memória *little-endian* (a) e *big-endian* (b)

Além dos repertórios de instruções e de um espaço de endereços para referenciar os elementos do microcontrolador, o processador dispõe de 13 registradores de 32 *bits* de propósito geral (R0-R12) e registradores de 32 *bits* de propósito específico no modo ARM: SP (*stack pointer*), LR (*link register*), PC (*program counter*) e CPSR (*current program status register*). E no modo *Thumb*, 8 registradores de propósito geral (R0-R7) e os mesmos registradores de propósito específico. Todos são de 32 *bits*. Figura 5 mostra comparativamente os dois conjuntos de registradores.

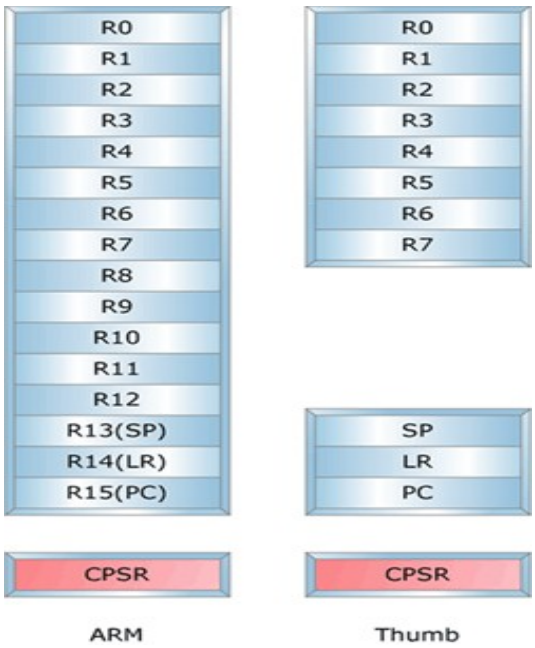


Figura 5: Registradores no modo ARM e no modo Thumb (Fonte: [5])

A Figura 6 mostra os *bits* do CPSR em que os *bits* de estado N (condição de sinal negativo), Z (condição do valor zero), C (condição de transporte) e V (condição de transbordamento) são armazenados.

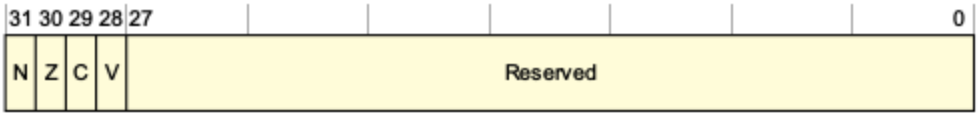


Figura 6: *Bits* de estado no registrador CPSR.

Uma estratégia aplicada no processador ARM para melhorar o tempo de execução das operações aritméticas inteiras é que as instruções para operações aritméticas inteiras só aceitam como operandos aqueles que estiverem armazenados nos registradores. Instruções específicas de *load* e *store* são usadas para transferência de dados entre os registradores e as unidades de memória.

Tratamento de exceções e interrupções é suportado pelo processador MKL25Z. Exceções ocorrem quando o fluxo normal da execução de instruções é interrompido por eventos internos do processador, como uma falha na busca ou na execução de uma instrução. Interrupções, por sua vez, são eventos externos do processador como uma transição no sinal de entrada. O tratamento dos dois é, no entanto, bem similar: o estado corrente é salvo, carrega-se no contador de programa PC o endereço da rotina de serviço disponível na tabela de vetores, executa-se a rotina de serviço, restaura-se o estado anterior ao do desvio, e retorna-se ao endereço em que aconteceu o desvio. O processador suporta 16 interrupções internas e 32 interrupções externas ao núcleo, diferenciáveis em 7 níveis de prioridade de atendimento, de -3 a 3. Quanto maior o nível de um evento, menor a prioridade do seu atendimento. De acordo com a seção B.1.5.1 em [3], o nível de prioridade máximo, -3, é atribuído ao evento *Reset*, -2 ao evento *NMI* (*non-maskarable interrupt*) e -1 ao evento *HardFault*. Os outros eventos têm o seu nível de prioridade, de 0 a 3, programável. Tabela 3-7 em [1] sintetiza todos os eventos processáveis em MKL25Z, com respectivos números de exceção e os endereços onde estão armazenados os endereços das respectivas rotinas de serviço. Por exemplo, o endereço do ponteiro do topo de pilha (SP) e do contador de programa (PC) ficam residentes, respectivamente, nos endereços 0x00000000 e 0x00000004 do espaço de endereços. Pela Figura 3, a memória *flash* está mapeada neste subespaço de endereços; portanto,

podemos concluir que estes endereços e os endereços das rotinas de serviço ficam armazenados na memória *flash*.

1.1.1.2 Módulos de Periféricos

Pela Figura 2 há uma série de módulos agregados ao processador (núcleo Cortex M0+) do microcontrolador MKL25Z da família Kinetis L. Esses módulos são também conhecidos como módulos de periférico da família Kinetis L. Entre os diversos módulos disponíveis, destacamos os módulos de conversão analógico-digital (ADC) e digital-analógico (DCA) (Seção 3.7 em [1]), os temporizadores PIT, TPM, LPTMR, RTC (Seção 3.8 em [1]), módulos de interface de comunicação serial USB, UART, SPI e I2C (Seção 3.9 em [1]) e módulos de interface com usuário através dos sensores de toque TSI e pinos digitais GPIO (Seção 3.10 em [1]). Todos estes módulos podem ser vistos como circuitos dedicados, providos de registradores de controle, de estado e de dados independentes.

Com exceção do módulo de interface dos pinos digitais GPIO, os registradores são mapeados no subespaço de endereços 0x40000000-0x4007FFFF. Se quisermos acessar os dados de algum deles, basta utilizarmos o endereço em que cada um é mapeado. Por exemplo, o conversor ADC do MKL25Z consegue multiplexar até 24 sinais analógicos na entrada. Configurando o campo ADCH (*bits* 4:0) do registrador ADC0_SC1A mapeado no endereço 0x4003B000, como mostra a Seção 28.3.1 em [1], o projetista pode selecionar, via *software*, um sinal dentre os 24.

O microcontrolador MKL25Z suporta 5 portas de 32 pinos digitais, GPIOA, GPIOB, GPIOC, GPIOD e GPIOE. Os registradores que configuram estes pinos digitais, como setar a direção do sinal digital, atribuir um valor lógico a um pino configurado como saída, são mapeados no subespaço de endereços 0x400FF000-0x400FFFFF. Por exemplo, se quisermos setar o pino 5 da porta GPIOA como entrada, poderemos fazer isso via *software* setando o *bit* 5 do registrador GPIOA_PDDR que está mapeado no endereço 0x400FF014, como mostra a Seção 41.2.6 em [1]. O microcontrolador apresenta ainda a alternativa de acessar estes pinos digitais com zero estado de espera. Pinos digitais com esta característica específica de velocidade, denominados *fast* GPIO (FGPIO), são mapeados nos subespaços de endereços 0x80000000-0xFFFFFFFF, como mostra a Figura 3.

1.1.1.3 Multiplexação dos Pinos Físicos

Pelo exposto, a quantidade de possibilidades de comunicação entre o microcontrolador e o mundo físico externo é muito grande. Porém, a quantidade dos pinos físicos PTXy, onde X é a letra designadora da porta e y o número de pino da porta, presentes no encapsulamento do MKL25Z é limitada em 80 pinos (Figura 7). Além dos pinos digitais das portas GPIOx, há pinos de comunicação dos módulos seriais, pinos dos circuitos conversores e pinos dos temporizadores que precisam ser associados aos pinos físicos para viabilizar a transferência de sinais físicos. A solução adotada para estabelecer uma correspondência m:n, onde m>>n, foi a multiplexação. No microcontrolador MKL25Z, é associado a cada pino físico até 8 pinos dos circuitos integrados nele. A tabela da seção 10.3.1 em [1] sintetiza todas as associações implementadas. E o uso dos pinos físicos é multiplexável via *software* através do módulo PORT. Há um registrador de controle PORTx_PCRn para cada pino físico e há um campo MUX (*bits* [10:8]) neste registrador através do qual consegue-se multiplexar os sinais do pino via *software*. Por exemplo, se quisermos que o pino físico 5 da porta A seja associado ao pino digital 5 do GPIOA, basta escrevermos o valor 0b001 (ALT 1) nos *bits* [10:8] (campo MUX) do endereço 0x40049014 em que o registrador PORTA_PCR5 está mapeado.

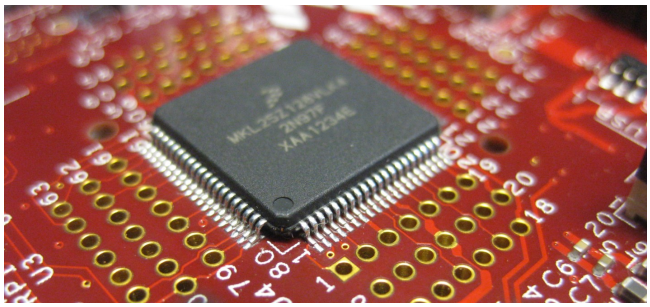


Figura 7: Encapsulamento de MKL25Z128

1.1.1.4 Módulo de Geração de Sinais de Relógio

O “coração” do microcontrolador é constituído de dois módulos geradores dos sinais de relógio necessários para estabelecer bases de tempo para todo o restante dos módulos: o módulo MCG (*Multipurpose Clock Generator*) e o módulo OSC (*system Oscillator*). Figura 8 ilustra as diferentes possibilidades de gerar as bases de tempo para diferentes módulos do MKL25Z.

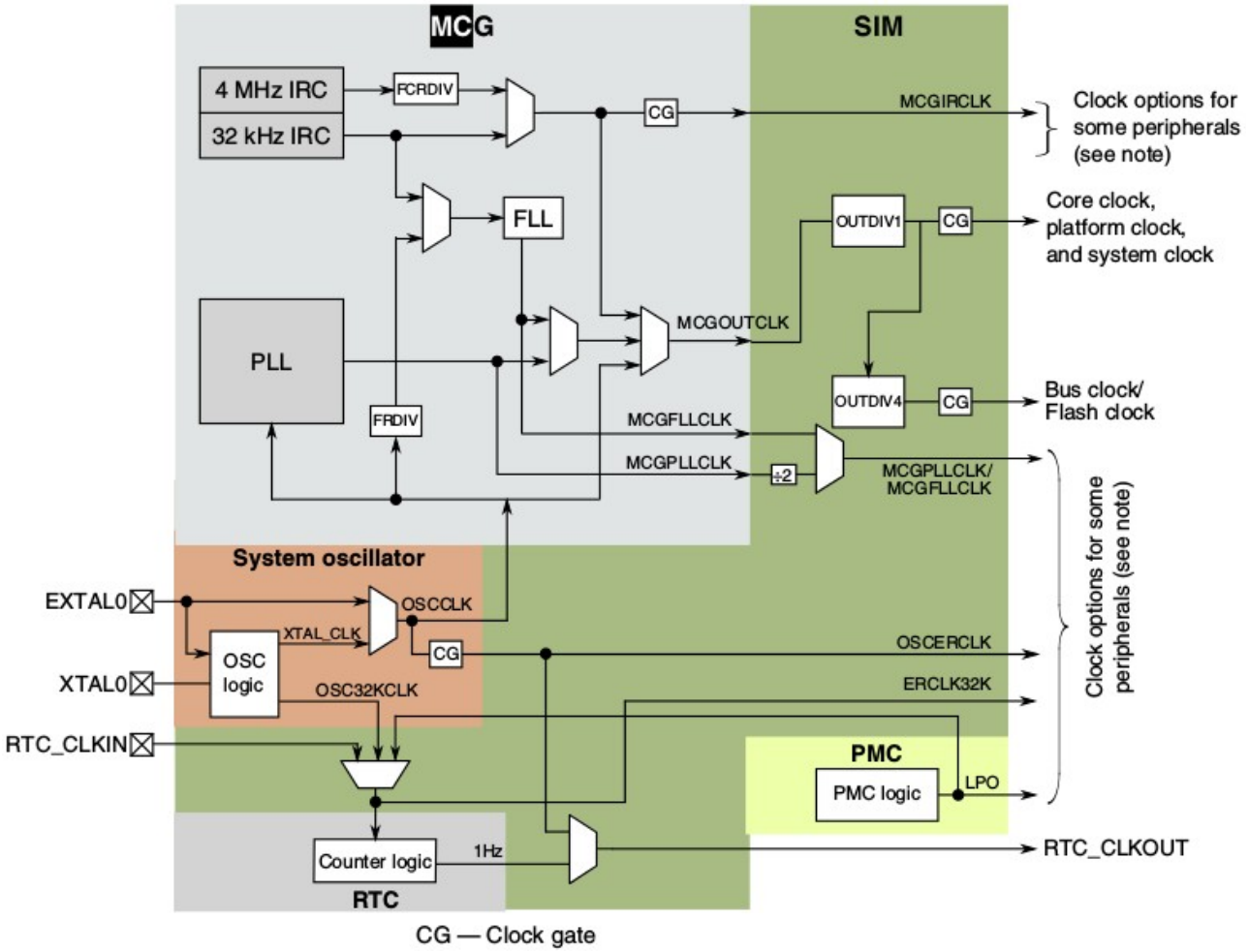


Figura 8: Fontes de sinais de relógio (Fonte: Figura 5-1 em [1]).

Os sinais de relógio utilizados do núcleo, no barramento e nas unidades de memória proveem do módulo MCG. São duas tecnologias implementadas no módulo MCG para gerar sinais de diferentes frequências a partir de um oscilador de cristal externo: FLL (*Frequency-Locked Loop*), até 48MHz, e PLL (*Phase-Locked Loop*), até 100MHz. Via *software*, é possível configurar o módulo MCG para operar num dos 9 modos de operação: FEI (*FLL Engaged Internal*), FEE (*FLL Engaged External*), FBI (*FLL Bypassed Internal*), FBE

(*FLL Bypassed External*), PBE (P*LL Bypassed External*), PEE (P*LL Engaged External*), BLPI (*Bypassed Low Power Internal*), BLPE (*Bypassed Low Power External*) e Stop. Descrições detalhadas de cada um destes modos podem ser encontradas na seção 24.4.1 em [1]. O principal sinal gerado pelo módulo MCG é MCGOUTCLK que é usado como base de tempo da grande maioria dos módulos do microcontrolador.

Veja na Figura 9 que o modo de operação do módulo MCG é FEI após **Reset**, e a Seção 4.1.3 da referência [4] contém a informação de que a frequência de MCGOUTCLK é 20.971520MHz quando a frequência do IRC é ajustada em 32.768 kHz (32768 = 2^{15}) pela fábrica, como é o caso do MKL25Z. Após a inicialização pode-se alterar, via *software*, os modos de operação do MCG respeitando o diagrama de transição de modos apresentado na Figura 8. Os registradores de controle do módulo MCG estão mapeados no subespaço de endereços 0x40064000-0x4006400F.

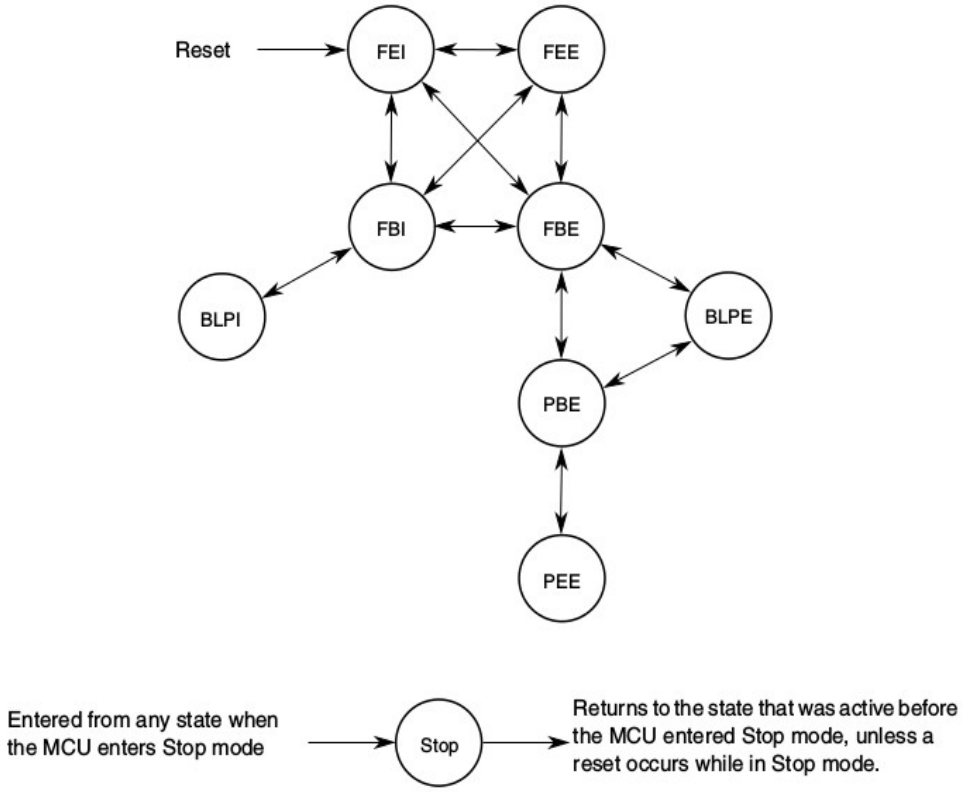


Figura 9: Diagrama de estados dos possíveis modos de operação do MCG (Fonte: Figura 24-16 em [1])

1.1.1.5 Módulo de Controle Integrador

Além do “cérebro” e do “coração”, o nosso microcontrolador tem ainda um módulo integrador dos seus “membros”, permitindo controle individualizado dos sinais de relógio. Este módulo é o SIM (*System Integration Module*). Todos os módulos de periférico têm os seus sinais de relógio desabilitados na inicialização (**Reset**). Cabe ao projetista habilitar as bases de tempo de cada circuito. Por exemplo, se o projetista decidir utilizar o pino digital 5 do módulo GPIOA, ele precisa habilitar o sinal de relógio deste módulo através do *bit* 9 do registrador de controle SIM_SCGC5 do módulo SIM que está mapeado no endereço 0x40048038.

1.1.1.6 Controlador de Interrupções

Com o objetivo de aumentar a eficiência e reduzir o consumo de energia, é integrado ao processador Cortex-M0+ o controlador de interrupções NVIC (*Nested Vectored Interrupt Controller*). Este controlador não só consegue "aliviar" o processador do processamento de até 240 interrupções configuradas com até um dos 256 níveis de prioridade, como também provê um mecanismo de desvio mais eficiente entre o fluxo normal de execução e o fluxo de tratamento específico de um evento e a passagem direta de um fluxo de tratamento de evento para outro fluxo de tratamento com uso da técnica de *tail-chaining*. A técnica *tail-chaining* consiste essencialmente em eliminar o procedimento de retornar da primeira rotina de serviço ao fluxo normal, salvar novamente o mesmo contexto para então entrar na segunda rotina de serviço. Figura 10 mostra o papel intermediador entre os módulos de periféricos e o núcleo de processamento.

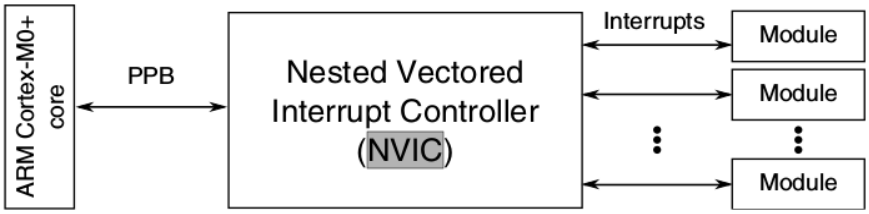


Figura 10: NVIC (Fonte: Figura 3-2 em [1])

Como outros módulos, o módulo NVIC possui um conjunto de registradores de controle através dos quais é possível adequar as funções do módulo para um fim específico. Estes registradores estão mapeados no subespaço de endereços 0xE0000000-0xE00FFFFF, reservado para periféricos particulares, conforme mostra a Tabela B3-18 da referência [3].

1.1.2 Periféricos

O diagrama de blocos da Figura 11 mostra esquematicamente os componentes que já estão integrados n FRDM-KL25. Dentre os recursos disponíveis podemos destacar, além do microprocessador *Freescale* MKL25Z128VLK4 (versão 80 pinos LQFP):

- Interface OpenSDA via USB integrada para simplificar a carga de programas e depuração.
- *Clock* a cristal externo de 8MHz (multiplicado internamente via PLL).
- 64 pinos de acesso para várias funções da placa: alimentação, controle, e interfaces.
- Porta miniUSB para implementação de USB OTG.
- Porta serial RS-232 com a UART física do processador ligada à mesma USB da interface OpenSDA, aparecendo enumerada no computador como uma porta USB COM.
- Botão RESET.
- Alimentação selecionável entre fonte externa ou pela USB.
- Acelerômetro de 3 eixos
- *led* RGB.
- *Slider* capacitivo.

É interessante observar que o circuito do *OpenSDA* é baseado num outro microcontrolador da família Kinetis K20 da *Freescale*, K20DX128VFM5.

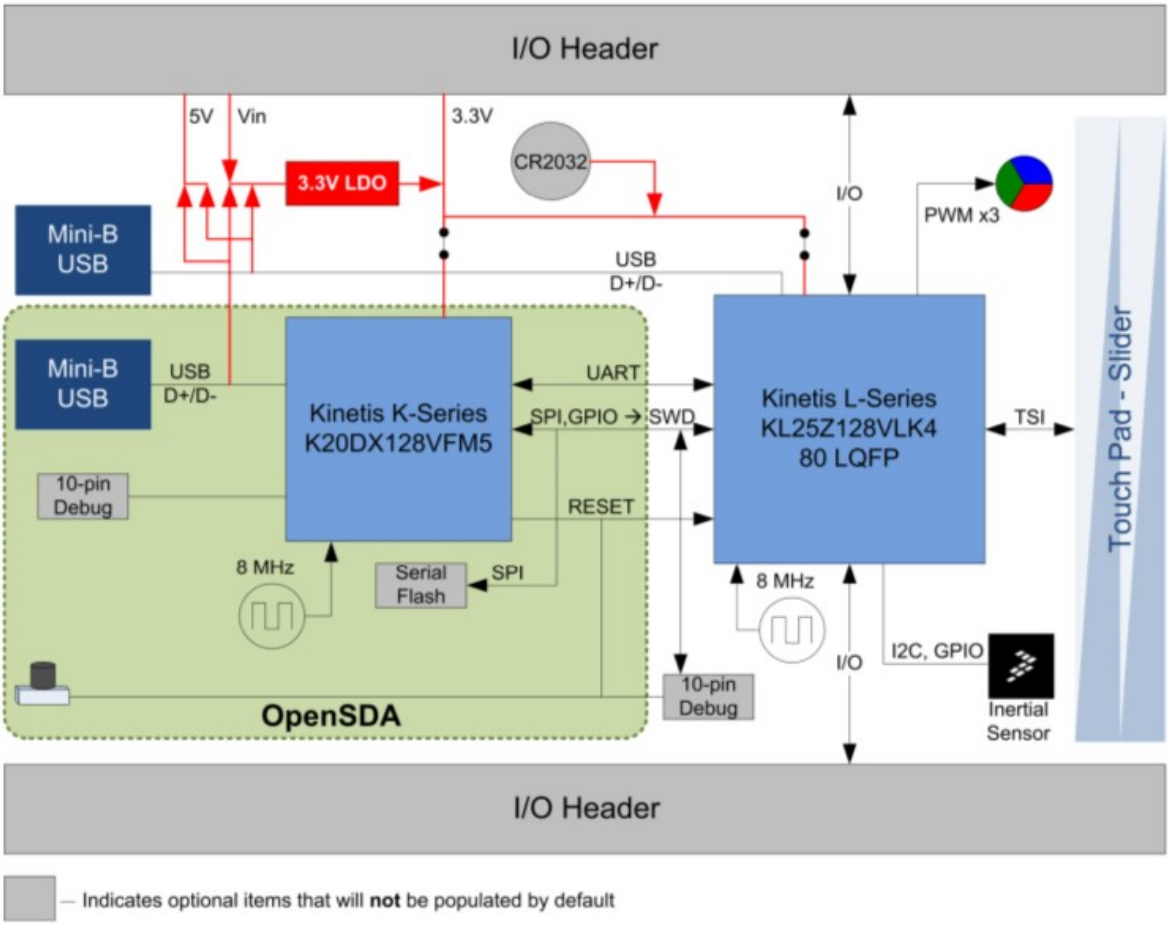


Figura 11: Diagrama de blocos do *kit* FRDM-KL25Z (Fonte: Figure 1 em [7])

Com o *hardware* já montado, só precisamos inserir programas que controlam o fluxo de sinais entre o microcontrolador e os periféricos integrados a FRDM-KL25Z através dos pinos físicos em que eles estão ligados. Portanto, sob o ponto de vista de programação, o primeiro passo para elaborar um programa é identificar em quais pinos físicos os periféricos estão conectados. Figura 12 mostra a conexão física do acelerômetro (MMA8451Q) nos pinos PTE24 (pino 24 da porta E), PTE25, PTA14 e PTA15 para se comunicar com o microcontrolador através do protocolo de comunicação serial I2C.

MMA8451Q	KL25Z128
SCL	PTE24
SDA	PTE25
INT1	PTA14
INT2	PTA15

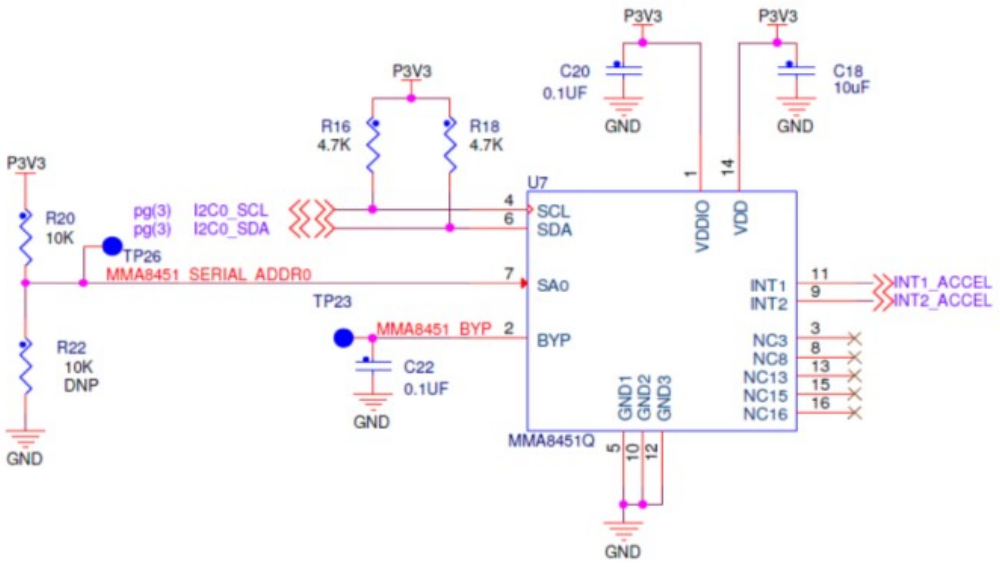


Figura 12: Conexões físicas de acelerômetro (Fonte: [7])

Figura 13 mostra a conexão física dos três *leds* R, G e B com os pinos digitais através dos pinos físicos PTB18, PTB19 e PTD1, respectivamente. O *touchpad* capacitivo está conectado com o módulo TSI (*Touch Sensing Input*) do microcontrolador através dos pinos PTB16 (TSI0_CH9) e PTB17 (TSI0_CH10).

RGB LED	KL25Z128
Red Cathode	PTB18
Green Cathode	PTB19
Blue Cathode	PTD1 [1]

NOTE:

1) PTD1 is also connected to the I/O header on J2 pin 10 (also known as D13).

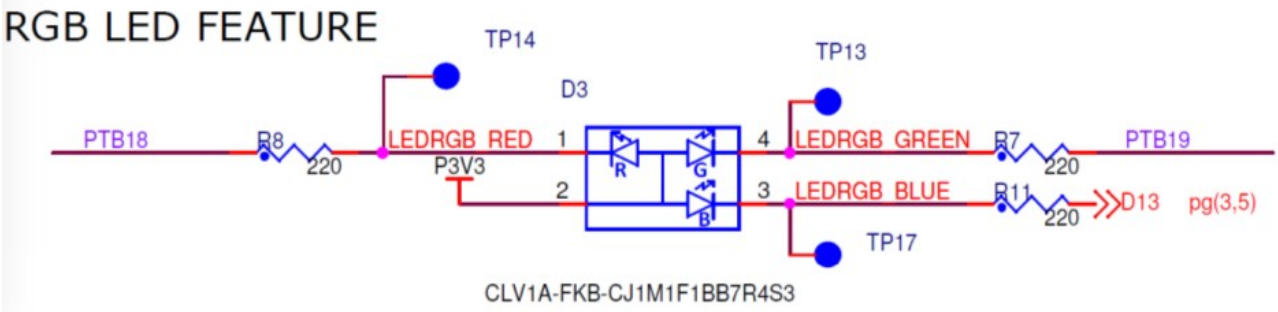


Figura 13: Conexões físicas dos *leds* RGB (Fonte: [7])

Sabendo em quais pinos físicos os periféricos estão conectados e o tipo de função que os pinos desempenham, identificamos os módulos que tais pinos físicos servem e os endereços em que os seus registradores estão mapeados. Tendo os endereços dos registradores de controle, de dados e de estado dos módulos, podemos interagir com os periféricos através destes endereços via *software*. Por exemplo, como o *led* R está conectado no pino físico PTB18 e os sinais digitais, nível lógico 0 (0V) e 1 (3.3V), são suficientes para acendê-lo e apagá-lo, só precisamos habilitar a base de tempo do módulo GPIOB via os registradores do módulo SIM, configurar a função do pino 18 da porta B via módulo PORT e setar o *bit* 18 dos registradores do módulo GPIOB para obter os efeitos desejados.

1.2 Placa Auxiliar EA871

Mesmo com os recursos disponíveis em FRDM KL-25, alguns periféricos que seriam bastante úteis ao curso não estão implementados. Assim, foi decidido que uma placa auxiliar seria criada. Esta placa é conectada à placa principal através dos conectores de I/O e alimentação da mesma. A placa auxiliar é encaixada diretamente sobre a FRDM, como um *shield*. O esquemático no Anexo 1 mostra o circuito da placa auxiliar e o desenho deste circuito na placa do circuito impresso é apresentado no Anexo 2, através do qual identifica-se a posição física de cada componente.

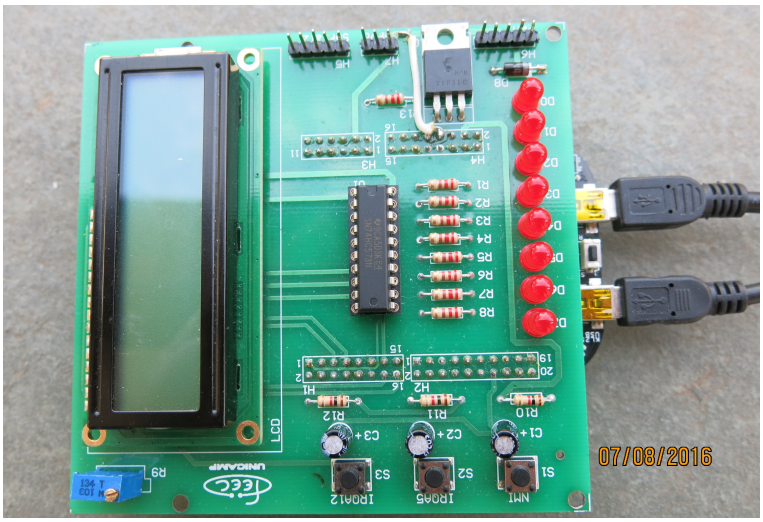


Figura 12: Placa Auxiliar EA871

A placa auxiliar conta com 3 *push-buttons*. Para cada um, há um circuito de *debounce* formado por um resistor de *pull-up* de 10kΩ e um capacitor de 1µF. Os botões estão ligados aos pinos PTA4 (NMI – *non-maskable interrupt*), PTA5 (IRQA5) e PTA12 (IRQA12). A NMI possui um vetor próprio (vetor 2, ou seja, o endereço da sua rotina de serviço é armazenado no endereço 0x00000008 do espaço de endereços conforme a Tabela 3-7 em [1]), enquanto que as interrupções comuns da porta A (no caso IRQA5 e IRQA12) compartilham o mesmo vetor, vetor 30.

Esta placa possui ainda um *latch* 74HC573 de 8 *bits* com LEDs vermelhos na saída, um *display* LCD (2 linhas de 16 caracteres cada). Usamos a linha de 5V da placa principal para alimentar o *display* LCD e o CI 74HC573. A linha de 3.3V é usada para os circuitos dos *push-buttons*. Os terras das placas são unificados. Apesar de alimentado com 5V, o *latch* interpreta corretamente os níveis lógicos em 3.3V, como os gerados pela placa FRDM-KL25Z. Este *latch* tem seus 8 *bits* de entrada ligados aos pinos PTC0-PTC7 do microcontrolador, e o pino *Enable* conectado ao pino PTC10 do microcontrolador. Quando o sinal *Enable* está em nível alto, as saídas refletem os valores das entradas. Na transição de

nível alto para baixo, os valores instantâneos das entradas são “travados” nas saídas. As linhas de dados do *display* são ligadas no mesmo grupo PTC0-PTC7 usado na entrada do *latch*. Entretanto, o sinal E (*enable*) do LCD é feito pelo pino PTC9. Assim, apesar de os dois periféricos usarem o mesmo conjunto de pinos de dados (PTC0-PTC7), os dados são direcionados para um ou outro de acordo com o sinal dos pinos de *enable* separados. O pino RS do LCD, utilizado para controlar se o mesmo está recebendo um comando ou um caractere, é ligado ao pino PTC8. Há ainda ligações no LCD para o controle de contraste (*trim-pot* de 10kΩ) e para a luz de fundo (resistor).

A placa auxiliar ainda possui um *header*, H5, para acesso a uma das portas seriais da CPU que não estão ligadas ao circuito do OpenSDA na placa FRDM-KL25Z. Este *header* possui ligações para o terra, PTE22 e PTE23 (respectivamente Tx e Rx da UART2 da CPU), além dos pinos PTE20 e PTE21 (usados respectivamente como RTS e CTS para controle de fluxo por *hardware*).

Outro *header*, H6, da placa auxiliar possui conexão com o pino PTB0, para acesso a um canal de *timer*. O mesmo pino PTB0 é ligado ao circuito de base de um transistor de média potência, que permite realizar controle PWM de cargas. Os demais pinos do *header* são para ligar a carga e a sua fonte externa.

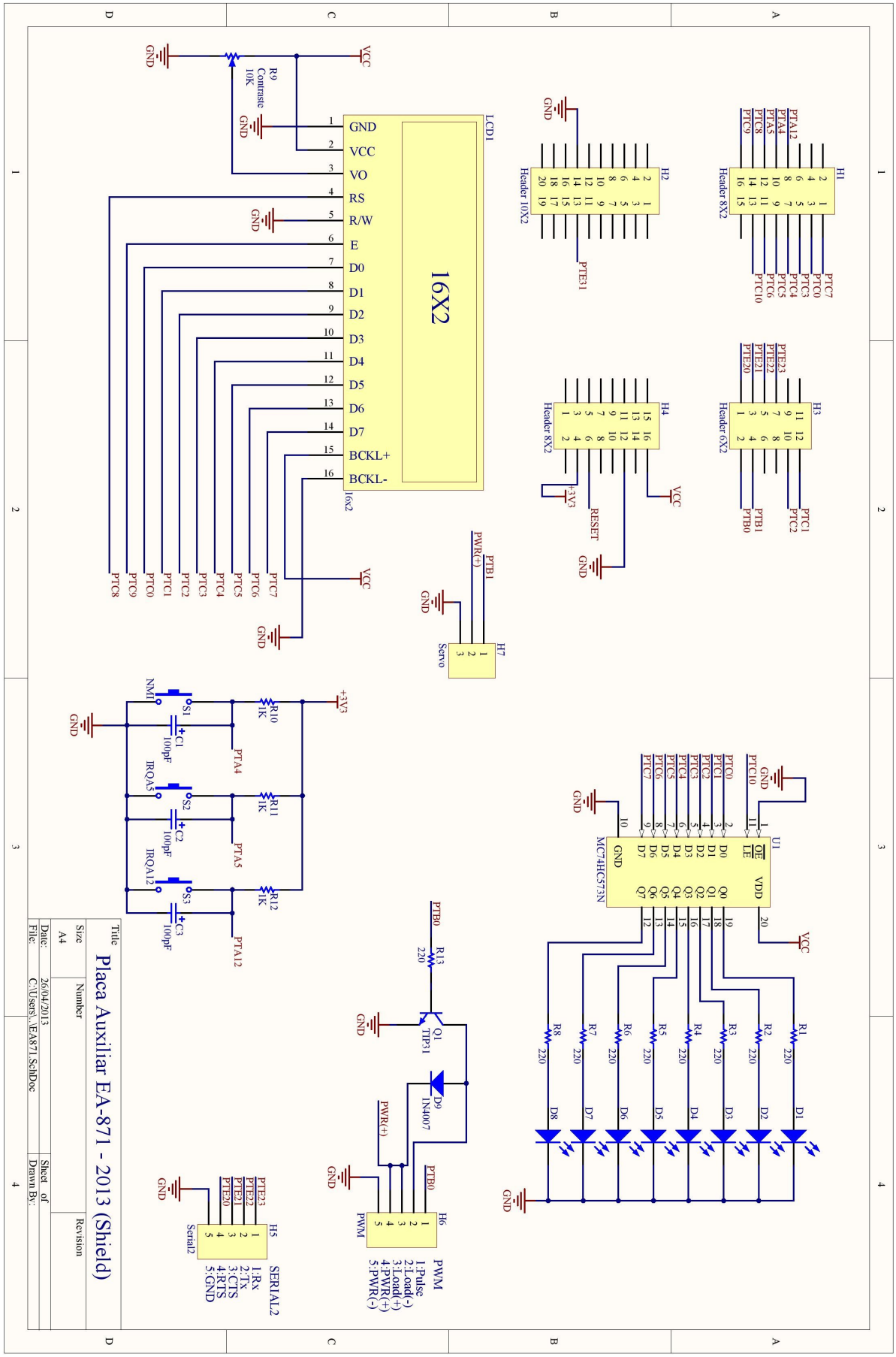
O último *header*, H7, possui conexões para o terra, para a fonte 3V3, e para o pino PTB1, que pode ser configurado para funções de *timer*.

Referências

[1] KL25 Sub-Family Reference Manual – Freescale Semiconductors (doc. Number KL25P80M48SF0RM), Setembro 2012.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KL25P80M48SF0RM.pdf
[2] Cortex-M0+ Technical Reference Manual – ARM Limited.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/Cortex-M0+.pdf
[3] ARMv6-M Architecture Reference Manual – ARM Limited.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/ARMv6-M.pdf
[4] Kinetis L Peripheral Module Quick Reference.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/KLQRUG.pdf
[5] Joe Lemieux. Introduction to ARM Thumb.
http://www.embedded.com/electronics-blogs/beginner-s-corner/4024632/Introduction-to-ARM-thumb
[6] Thiago Lima. Embedded Systems using Microcontrollers.
http://embeddedreference.blogspot.com.br/2013/08/ARMCortexM0PlusLowEnergy.html
[7] Freescale. FRDM-KL25Z User's Manual.
ftp://ftp.dca.fee.unicamp.br/pub/docs/ea871/ARM/FRDMKL25Z.pdf

Agosto de 2016

Anexo 1: Esquemático da Placa Auxiliar



Anexo 2: PCI da Placa Auxiliar