Índice

1. Introdução
   1. O que é um microcontrolador
      1. Mostrar a placa e o chip
      2. Dados relevantes do datasheet
   2. Estrutura de Clock
      1. Geração interna/externa
      2. Divisão dos barramentos dos periféricos
      3. O que é o PLL
   3. Periféricos
      1. GPIO
      2. TIM1
      3. UART (em desenvolvimento)
      4. ADC (em desenvolvimento)
      5. DAC (em desenvolvimento)
   4. Processador
      1. Diferença entre ponto fixo e ponto flutuante
      2. Interrupções
2. Programação
   1. STM32CubeMx (ferramenta comum a todos)
   2. Eclipse GNU ARM GCC-ARM-None-EABI Open OCD
      1. Instalação
      2. Compilação
      3. Depuração
3. Geração de PWM (duty cycle fixo)
4. Modulação Senoidal e aplicação a conversores estáticos
   1. Aplicação a Conversores Estáticos
5. Comunicação (em desenvolvimento)
6. Aquisição de Dados (em desenvolvimento)
7. Bibliografia e Referências

Lista de Figuras

[Figura 1: Kit STM32F4 Discovery 6](#_Toc473628454)

[Figura 2: Estrutura de blocos do microcontrolador STM32F407VG 7](#_Toc473628455)

[Figura 3: Esquema de alimentação 8](#_Toc473628456)

[Figura 4: Estrutura de entrada e saída de uma porta I/O 10](#_Toc473628457)

[Figura 5: Estrutura de Clock 12](#_Toc473628458)

[Figura 6: Alimentação dos clocks HSE/LSE 13](#_Toc473628459)

[Figura 7: Diagrama de Blocos da divisão do barramento dos periféricos 13](#_Toc473628460)

[Figura 8: Matriz de Barramentos 14](#_Toc473628461)

[Figura 9: Seleção de Alternate Functions (pinos 0 a 7) 16](#_Toc473628462)

[Figura 10: Seleção de Alternate Functions (pinos 8 a 15) 16](#_Toc473628463)

[Figura 11: Operação do contador e frequência de chaveamento 18](#_Toc473628464)

[Figura 12: Sinais referentes ao contador (PSC = 0) 19](#_Toc473628465)

[Figura 13: Sinais referentes ao contador (PSC = 1) 19](#_Toc473628466)

[Figura 14: Sinais referentes ao contador (PSC = 0) 20](#_Toc473628467)

[Figura 15: Sinais referentes ao funcionamento do contador (PSC = 0) 20](#_Toc473628468)

[Figura 16: Funcionamento do TIM1 no modo PWM edge-aligned com ARR = 8 21](#_Toc473628469)

[Figura 17: Funcionamento do TIM1 no Modo PWM center-aligned com ARR = 8 22](#_Toc473628470)

[Figura 18: Representação de um número em ponto flutuante 23](#_Toc473628471)

[Figura 19: Estrutura da mantissa 23](#_Toc473628472)

[Figura 20: Página da web para baixar a IDE 24](#_Toc473628473)

[Figura 21: Pasta compactada com os arquivos baixados 25](#_Toc473628474)

[Figura 22:Página da Web para baixar o compilador 25](#_Toc473628475)

[Figura 23: Configurações recomendadas de instalação 26](#_Toc473628476)

[Figura 24: Website para baixar interface de depuração OPENOCD 26](#_Toc473628477)

[Figura 25: Pasta compactada baixada 27](#_Toc473628478)

[Figura 26: Website para baixar o Windows Building Tools 27](#_Toc473628479)

[Figura 27: Website para baixar o programa STM32CubeMX 28](#_Toc473628480)

[Figura 28: Seleção do pacote de firmware 28](#_Toc473628481)

[Figura 29: Seleção do diretório para workspace 29](#_Toc473628482)

[Figura 30:Software para a instalação 29](#_Toc473628483)

[Figura 31: Confirmação de instalação de software não assinado 30](#_Toc473628484)

[Figura 32: Reiniciar o Eclipse 30](#_Toc473628485)

[Figura 33: Abrir uma nova Perspectiva 30](#_Toc473628486)

[Figura 34: Selecionar opção Packs 31](#_Toc473628487)

[Figura 35: Atualizar lista de pacotes 31](#_Toc473628488)

[Figura 36: Selecionar microcontrolador 32](#_Toc473628489)

[Figura 37: Página inicial do programa STM32CubeMX 33](#_Toc473628490)

[Figura 38: Selecionar o microcontrolador 33](#_Toc473628491)

[Figura 39:Habilitar periférico TIM1 34](#_Toc473628492)

[Figura 40: Desenho do chip no programa 34](#_Toc473628493)

[Figura 41:Mudança de pino (opcional) 35](#_Toc473628494)

[Figura 42: Configuração de clock no STM32CubeMX 35](#_Toc473628495)

[Figura 43: Selecionar configuração de TIM1 36](#_Toc473628496)

[Figura 44: Configuração dos valores armazenados nos registradores 36](#_Toc473628497)

[Figura 45: Configuração do valores de Pulse 37](#_Toc473628498)

[Figura 46: Ícone para gerar o código 38](#_Toc473628499)

[Figura 47: Configurações do projeto 38](#_Toc473628500)

[Figura 48: Ícone para baixar a versão mais recente de firmware 39](#_Toc473628501)

[Figura 49: Criação de um projeto no Eclipse 39](#_Toc473628502)

[Figura 50: Seleção do microcontrolador 40](#_Toc473628503)

[Figura 51: Pastas do projeto 40](#_Toc473628504)

[Figura 52: Plataformas do Projeto (Debug e Release) 40](#_Toc473628505)

[Figura 53: Pastas do Projeto no Project Explorer 41](#_Toc473628506)

[Figura 54: Caminho para acessar bibliotecas a serem ativadas 41](#_Toc473628507)

[Figura 55: Bibliotecas a serem ativadas 42](#_Toc473628508)

[Figura 56: Comandos para ativar as bibliotecas 42](#_Toc473628509)

[Figura 57: Exclude from build 42](#_Toc473628510)

[Figura 58: Ícone para compilação 43](#_Toc473628511)

[Figura 59: Abrir configurações de Debug 44](#_Toc473628512)

[Figura 60: Clicar em GDB OpenOCD Debugging e New 44](#_Toc473628513)

[Figura 61: Configurações de Debug 45](#_Toc473628514)

[Figura 62: Configurações de Debug (continuação) 45](#_Toc473628515)

[Figura 63: Seleção do arquivo .elf 46](#_Toc473628516)

[Figura 64: Seleção do arquivo .elf (continuação) 46](#_Toc473628517)

[Figura 65: Mudar para ambiente de Debug 47](#_Toc473628518)

[Figura 66: Ligando FPU 47](#_Toc473628519)

[Figura 67: Ligando FPU (continuação) 48](#_Toc473628520)

[Figura 68: Relação entre variável interm (eixo y) e AUX (eixo x) 50](#_Toc473628521)

[Figura 69: Cálculo de CCR a partir de Interm 50](#_Toc473628522)

[Figura 70: Fase do seno em interm e CCR 51](#_Toc473628523)

[Figura 71: Período de chaveamento do Timer 51](#_Toc473628524)

[Figura 72: Saída do Timer 51](#_Toc473628525)

[Figura 73: Pino para selecionar a função GPIO\_EXIT7 52](#_Toc473628526)

[Figura 74: Botão para configurações de NVIC 53](#_Toc473628527)

[Figura 75: Configurações de NVIC 53](#_Toc473628528)

[Figura 76: Botão para configurações de GPIO 54](#_Toc473628529)

[Figura 77: Configuração do pino selecionado 55](#_Toc473628530)

Lista de Tabelas

[Tabela 1:Tensões características de alimentação 8](#_Toc473628531)

[Tabela 2: Diferentes operações do ADC para intervalos de tensão de alimentação 9](#_Toc473628532)

[Tabela 3: Valores máximos de corrente 9](#_Toc473628533)

[Tabela 4: Condições gerais de operação 10](#_Toc473628534)

[Tabela 5: Periféricos reservados para APB2 14](#_Toc473628535)

[Tabela 6: Periféricos reservados para APB1 15](#_Toc473628536)

[Tabela 7: Características de cada TIMx 17](#_Toc473628537)

1. Introdução
   1. O que é um microcontrolador?

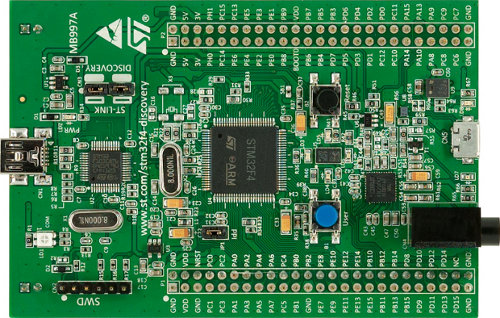
Um micro controlador é um circuito integrado que contém um microprocessador, memórias e periféricos adjuntos, sendo que estes trabalhos em conjunto para processar dados, entradas e saídas. Trata-se de um componente muito utilizado em programação para sistemas embarcados, sendo uma de suas principais aplicações fazer o controle dos sistemas que integra.

* + 1. O kit STM32F4 Discovery e o chip STM32F407VGT6

A família dos micro controladores das linhas STM32F405xx e STM32F407xx é baseada nos processadores de alta performance ARM Cortex-M4 32-bits, específicos para sistemas embarcados, que podem operar em uma frequência de até 168MHz. Tal linha de processadores possui unidade de FPU (Floating Point Unity), e por isso são capazes de fazer operações com todos os tipos de dados. Os micro controladores mencionados têm, portanto, várias aplicações, como controle de motores e inversores.

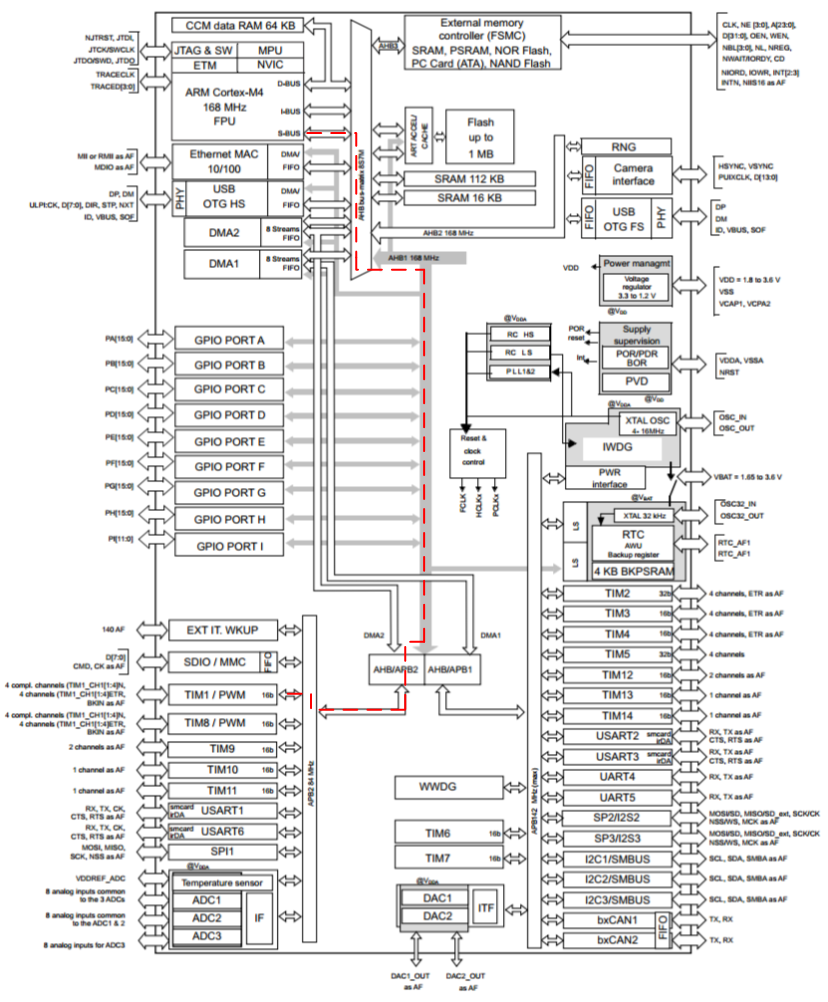
Nesta documentação, apresenta-se o uso do micro controlador STM32F407VGT6 na placa de desenvolvimento STM32F4 Discovery, produzido pela empresa STMicroeletronics. Trata-se de uma placa de desenvolvimento que pode ser programada pelo usuário para as mais diversas funções, mas no caso optou-se por detalhar a geração de PWM (Pulse Width Modulation, ou Modulação por Largura de Pulso). Na *Figura 2*, inclusive, mostra-se em pontilhado a relação de controle entre o clock do processador e o periférico TIM1 (timer), principal componente utilizado para a função mencionada. O funcionando dos periféricos envolvidos nesta relação e a estrutura de clock são melhor caracterizados nos capítulos seguintes.

Figura 1: Kit STM32F4 Discovery



Fonte: manual do usuário do kit STM32F4 Discovery

Figura 2: Estrutura de blocos do microcontrolador STM32F407VG

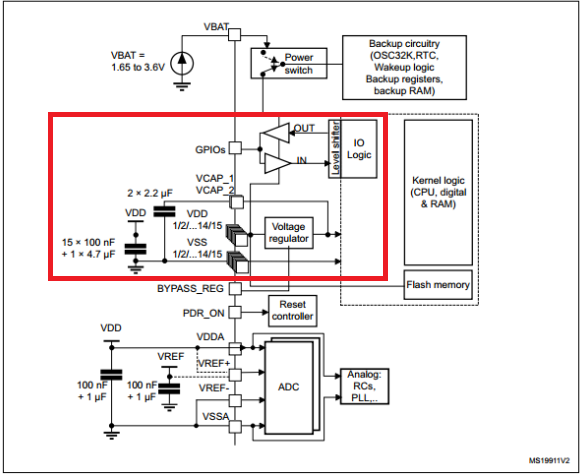


Fonte: Datasheet do microcontrolador STM32F407VG, página 19 [1]

* + 1. Dados Relevantes
       1. Dados
          1. Tensão de alimentação

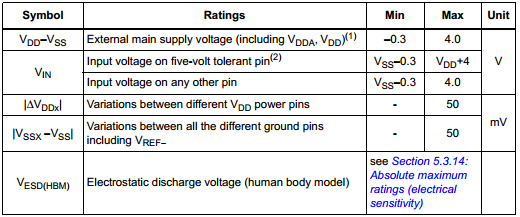
A tensão de alimentação nas entradas VDD é tipicamente 3.3V, mas pode ser ajustada entre 1.8 e 3.6V, se necessário por um regulador de tensão, conforme sinalizado na *Figura 3*. Nela também está indicado o caminho entre a tensão VDD e as portas GPIO. A placa Discovery é alimentada pela porta USO que fornece 5V. Um regulador abaixa esta tensão para 3V para alimentar o micro controlador neste caso.

Figura 3: Esquema de alimentação



Fonte: Datasheet do microcontrolador, página 77 [1]

Tabela 1:Tensões características de alimentação

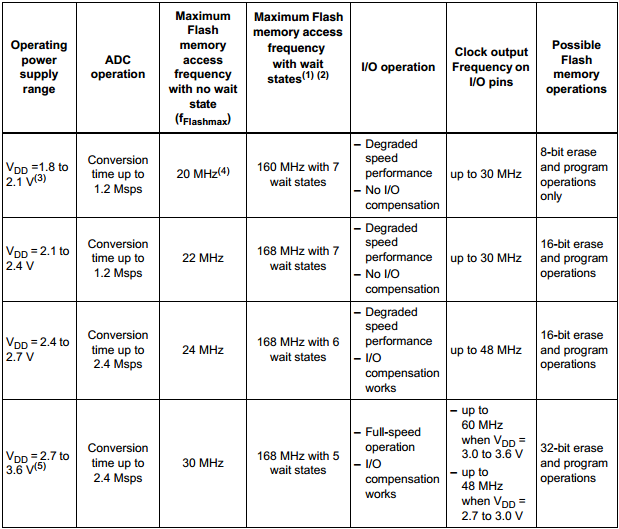


Fonte: Datasheet do microcontrolador, página 78 [1]

* + - * 1. Tensão de entrada do A/D

Ainda em referência à *Figura 3*, percebe-se que a mesma tensão VDD se relaciona à tensão de saída do ADC e também pode variar dentro do intervalo de 1.8V e 3.6V, apesar de sua saída padrão ser 3.3V. Pela *Tabela 2: Diferentes operações do ADC para intervalos de tensão de alimentação*, é possível inferir que a performance da operação do ADC é diferente para valores dentro do intervalo de tensão de alimentação dado.

Tabela 2: Diferentes operações do ADC para intervalos de tensão de alimentação

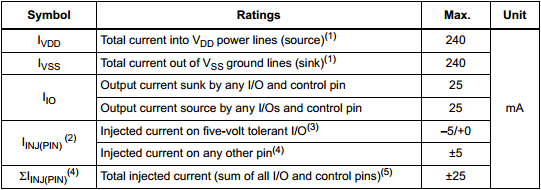


Fonte: Datasheet do microcontrolador, página 81 [1]

* + - * 1. Corrente que flui pelas portas

Pela *Tabela 3*, a corrente máxima que flui por cada porta é de 25mA. A corrente máxima injetada de 5mA por pino, e no total para todos os pinos de 25mA.

Tabela 3: Valores máximos de corrente



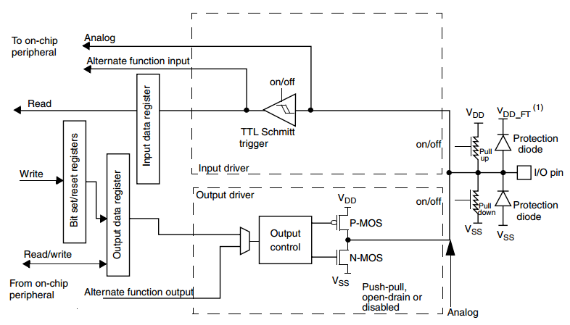
Fonte: Datasheet do microcontrolador, página 79 [1]

* + - * 1. Estrutura do estágio de entrada e saída

A estrutura de estágio de entrada e saída das portas conta com a sua tensão de alimentação (VDD) que, conforme mencionado anteriormente, tem valor padrão de 3,3V mas pode variar entre 1,8V e 3,6V usando-se o regulador de tensão. São utilizados diodos de proteção, tanto entre a porta e VDD quanto entre a porta e o terra (VSS), como também resistores de pull up e pull down que são ligados e desligados via software.

As portas têm função de leitura e escrita. Ambas contam com registradores que armazenam os dados de input e output, respectivamente. Podem ser configuradas como input, output ou como auxiliares para outro periférico (“alternate function”), sendo o último o recurso mais útil para geração de PWM (foco principal desta documentação).

Figura 4: Estrutura de entrada e saída de uma porta I/O

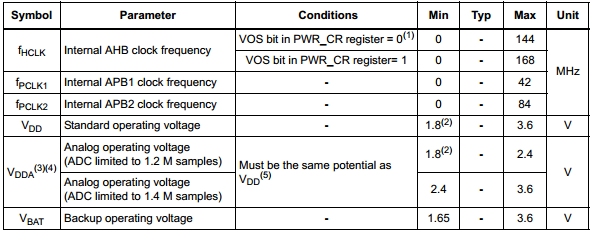


Fonte: Reference Manual, página 270 [2]

* + - * 1. Condições Gerais de Operação

Reiterando, pela tabela de condições gerais de operação (*Tabela 4*), a tensão de saída pelas portas (VDD) vai de 1.8V a 3.6V. A mesma tabela ainda traz informações de clock, cujo funcionamento é retomado e aprofundado no capítulo 1.2 (“Estrutura de Clock”).

Tabela 4: Condições gerais de operação



Fonte: Datasheet do microcontrolador, página 79 [1]

* 1. Estrutura de Clock

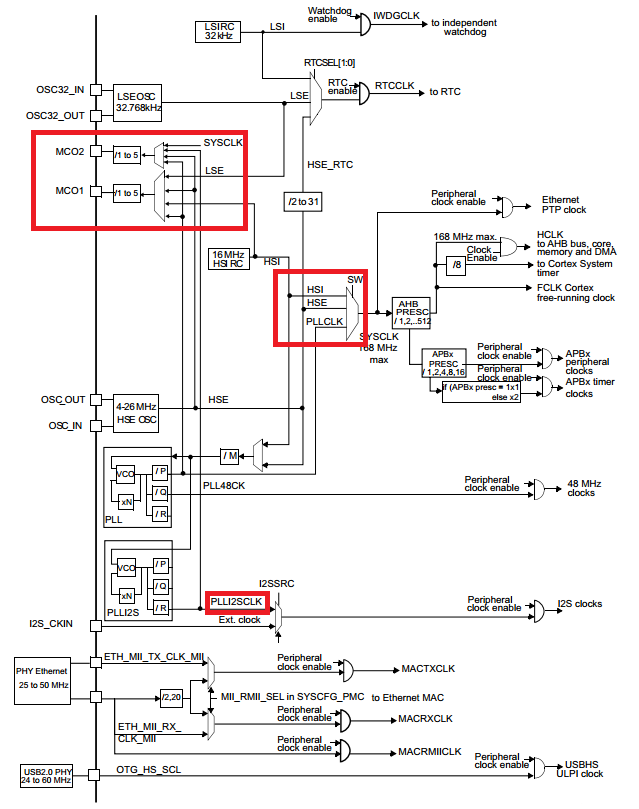
No que se refere ao funcionamento de timers (principal escopo desta documentação), existem três fontes que podem ser usadas para criar o clock do sistema:

* Oscilador HSI (16MHz);
* Oscilador HSE (de acordo com o cristal instalado na placa, no caso da placa Discovery é 8MHz);
* Clock por PLL.

O chip tem estrutura de clock para todas as aplicações que oferece. Em termos de habilitação de timers (que é o assunto principal deste manual), é usado um MUX para ativar a fonte necessária para a frequência de operação escolhida por software, conforme sinalizado na figura do diagrama da estrutura de clock (*Figura 5: Estrutura de Clock*. A partir desta seleção, é gerado o sinal SYSCLK (que alimenta o núcleo de processamento Cortex).

É interessante notar também que os pinos do tipo MCO (Microcontroller Clock Output) podem ser usados como saídas por onde passam os sinais de clock, oriundos de diferentes fontes. Há duas opções no caso: o MCO1, que pode ter como saída os sinais de HSE, HSI, PLL e LSE (sinal de fonte externa, a partir de um oscilador, de menor velocidade); e o MCO2, que pode ter como sinal de saída o clock oriundo das fontes HSE, PLL, SYSCLK e PLLI2S (sinal indicado no diagrama de blocos).

Figura 5: Estrutura de Clock



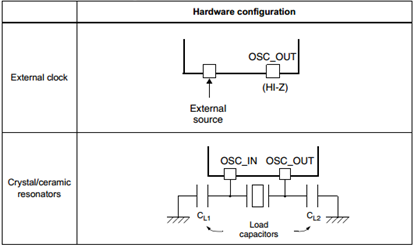
Fonte: Reference Manual, página 216 [2]

1.2.1. Geração Interna/Externa

* Gerador Externo (HSE)

O Gerador Externo pode ser gerado a partir de duas fontes: um ressonador cerâmico/cristalino externo ou um outro clock externo provido pelo usuário. Ao optar pelo ressonador, todos os componentes usados na montagem (o próprio ressonador e capacitores) devem ser colocados o mais próximos o possível do pino do oscilador para que interferências de ruídos sejam evitadas.

Figura 6: Alimentação dos clocks HSE/LSE



Fonte: Reference Manual [2]

* Gerador Interno (HSI)

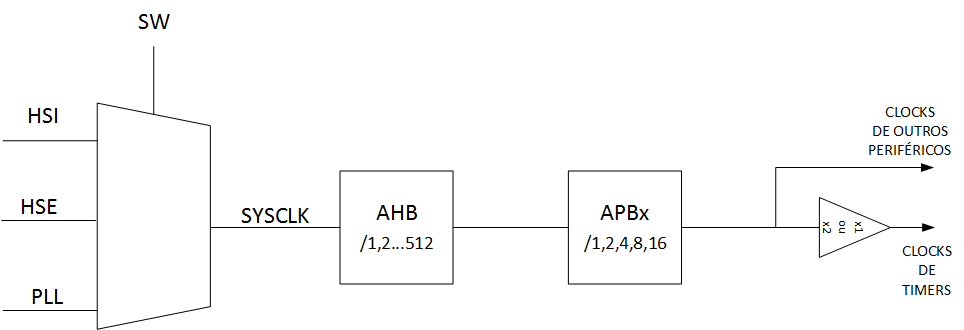
O clock HSI é gerado a partir de um oscilador RC interno de 16MHz. Ele tem a vantagem de ter custo menor por não precisar de componentes extras e também um tempo de inicialização menor do que o HSE, mas sua frequência é menos precisa do que deste.

1.2.2. Divisão nos Barramentos dos Periféricos

Feita a seleção da fonte a ser usada para gerar o clock (HSI, HSE ou PLL), o sinal (SYSCLK) passa por um fator AHB, que pode dividir a frequência por valores de 1 a 512.

Passado pelo AHB, o sinal resultante passa por um novo fator prescaler APBx, agora podendo ser dividido por 1, 2, 4, 8 e 16 dependendo da frequência de operação escolhida. O resultado do sinal após o fator APBx será a frequência de atualização do timer, podendo este ser multiplicado por 2 caso sinalizado.

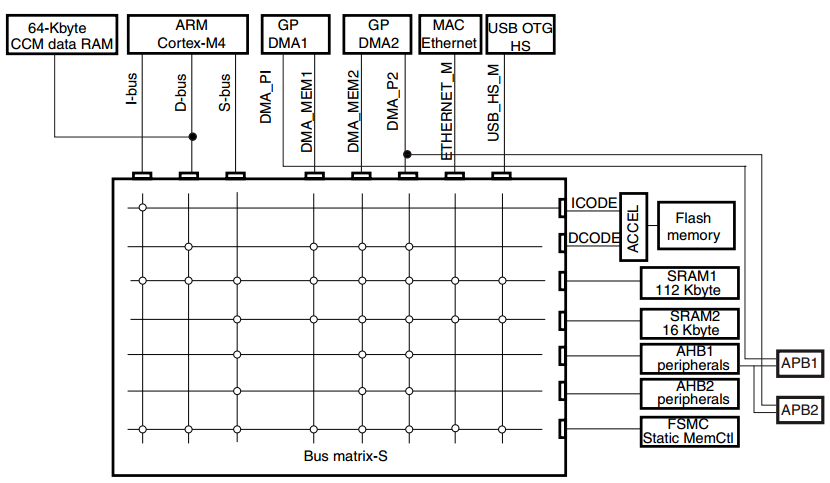
Figura 7: Diagrama de Blocos da divisão do barramento dos periféricos



Fonte: Próprios Autores

A *Figura 8* mostra a matriz de conexão via bus entre o processador ARM Cortex-M4 e o fator AHB. A escolha de uso entre os fatores APB1 ou APB2 depende de qual periférico está sendo usado, conforme a *Tabela 5*. Se o periférico TIM1 está sendo usado, por exemplo, utiliza-se o fator APB2.

Figura 8: Matriz de Barramentos



Fonte: datasheet do microcontrolador, página 22 [1]

Tabela 5: Periféricos reservados para APB2

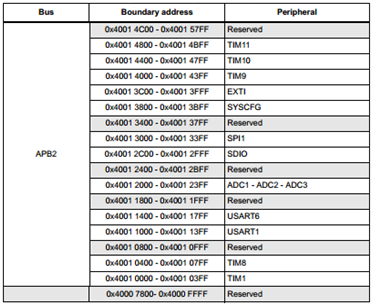
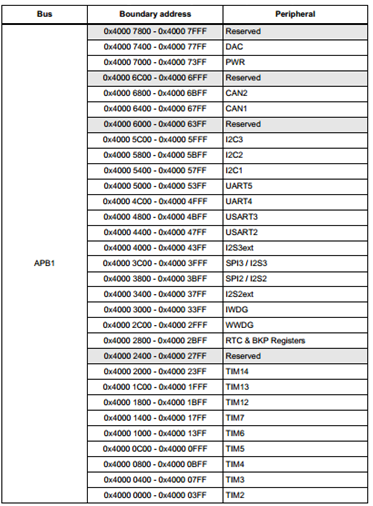


Tabela 6: Periféricos reservados para APB1



1.2.3. PLL e aplicação para multiplicação de clock

A partir das fontes disponíveis pode-se gerar um clock de 16 ou 8 MHz. No entanto, frequências mais elevadas são necessárias para operação na maioria dos casos. Para tanto, usa-se o PLL para multiplicar a frequência do HSE ou HSI para obter o valor desejado.

A utilização do PLL é descrita na seção 1.2 desta documentação (“Estrutura de Clock”).

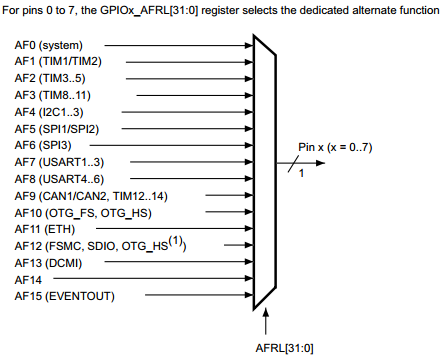
1.3.1. GPIO

Os General-purpose input/outputs (GPIOs) são pinos do tipo I/O que podem ser configurados por software como output, input ou auxiliar de função para um outro periférico (“alternate function”/AF).

Quando ligados como output, podem funcionar como push-pull ou coletor aberto, podendo ou não ser ligados a um resistor pull-down ou pull-up. Como input, funcionam “floating” (“flutuante”), também podendo ser ligado a um resistor de pull-up ou pull-down.

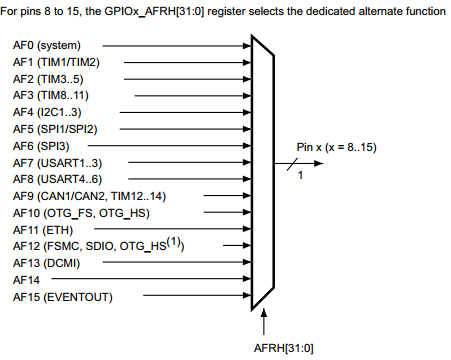
Cada um dos pinos é ligado aos periféricos do microcontrolador a um multiplexador, o qual permite a operação de uma função por vez associada a cada pino, de modo a evitar conflitos no funcionamento de periféricos. Cada multiplexador tem 16 “alternate functions” (AF0 até AF15), sendo AF0 a função de reset, AF1 a AF13 o mapa de periféricos e AF15 ligada à FPU do processador. É possível ver esta configuração na *Figura 9: Seleção de Alternate Functions*, mas nota-se que há dois multiplexadores: um para os módulos de 0 a 7 e outro para os módulos de 8 a 15, sendo, portanto, usados dois registradores diferentes para a escolha da AF.

Figura 9: Seleção de Alternate Functions (pinos 0 a 7)



Fonte: Reference Manual, página 274 [2]

Figura 10: Seleção de Alternate Functions (pinos 8 a 15)



Fonte: Reference Manual, página 274 [2]

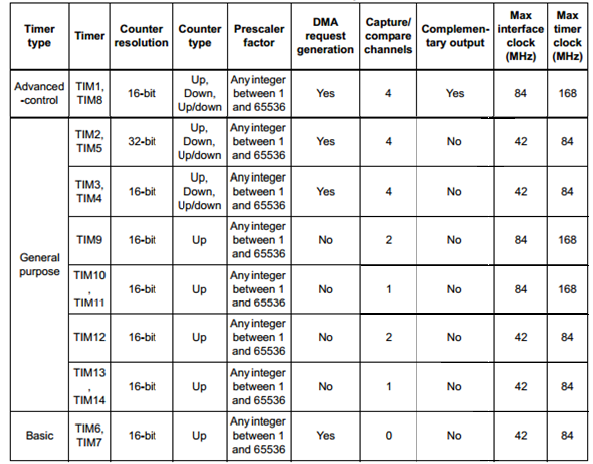
1.3.2. TIM

O periférico TIM consiste basicamente em um contador (timer) de 16 bits com refresh automático, podendo ser adaptado por um prescaler programável. Existem ao todo 14 tipos de periférico TIM, sendo estes (divididos por tipo):

* Advanced-Control (controle avançado em tradução livre) - TIM1 e TIM8, especificados na tabela x;
* General Purpose (aplicações gerais em tradução livre) - TIM2, TIM3, TIM4, TIM5, TIM9, TIM10, TIM11, TIM12, TIM13, TIM14 - especificados na tabela x;
* Basic (básicos, em tradução livre) - TIM6 e TIM7 - especificados na tabela x;

Todos os periféricos TIM têm função de PWM, excetuando-se o TIM6 e TIM7. Para aplicação em conversores trifásicos, geralmente opta-se pelos timers de controle avançado, pois eles possuem os recursos necessários para tal. Portanto, descreve-se mais detalhadamente o TIM1, uma vez que ele é mais adequado para essa função, sendo que os outros operam de maneira semelhante dentro de suas particularidades. Caso o leitor queira usar outro timer com função PWM, basta atentar-se para os intervalos de valores que podem ser armazenados nos registradores (melhor especificado na *Tabela 7*), nas interrupções disponíveis e no número de canais disponíveis pelo TIM que se deseja usar.

Tabela 7: Características de cada TIMx



O TIM1 pode ser usado para aplicações com input, como medida da largura de um pulso em sua entrada, como também geração de saídas (output), dentre elas PWM, PWM com deadtime e output-compare, sendo que este documento especifica-se em seu funcionamento no modo PWM. Dito isto, o TIM1 possui as seguintes partes:

* Contador up/down de 16 bits com refresh automático;
* Prescaler de 16 bits programável, que permite divide a frequência do clock do contador por qualquer fator entre 1 e 65536;
* Quatro canais independentes, que podem exercer função de geração de PWM, captura de entrada, comparação de saída, geração de saída de apenas um pulso, e que tem saídas “normais” e negadas;
* Geração de sinais de saída com dead-time, entre outros.

O contador pode funcionar nos modos up, up-down e down. Para tal, o timer tem vários registradores que são usados para configurar o seu funcionamento. Nesta documentação, optou-se por mencionar apenas os principais registradores: counter register (TIMx\_CNT), Prescaler register (TIMx\_PSC), Autoreload register (TIMx\_ARR), Repetition Counter register (TIMx\_RCR) e TIMx\_CCRx, cujos funcionamentos estão melhor descritos adiante. Se o leitor desejar um aprofundamento sobre os demais registradores, recomenda-se a leitura do capítulo 17 do Reference Manual, “Advanced Control Timers (TIM1&TIM8)”.

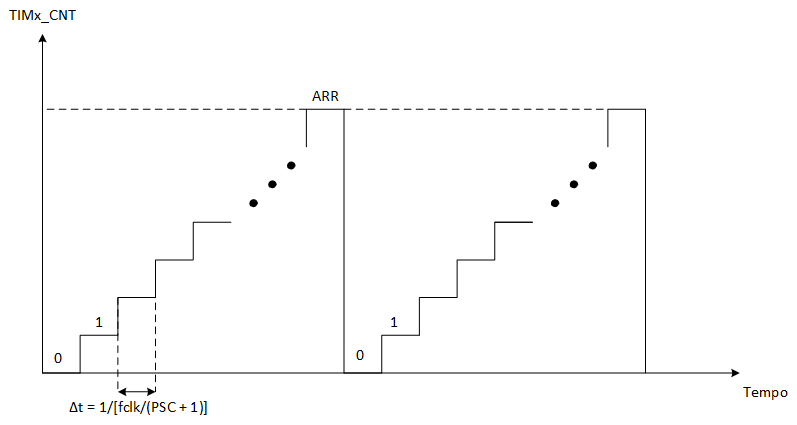
1.3.2.1. Tipos de contador

O periférico TIM1 pode funcionar como três tipos de contador: up, down (edge-aligned) e up-down (center-alined). A frequência de atualização do contador é resultado da divisão do sinal CK\_PSC de frequência , provindo dos barramentos na estrutura de clock (descrita no item 1.2.2) pelo valor PSC + 1, sendo PSC o valor escolhido como prescaler pelo usuário e armazenado no registrador TIMx\_PSC. Ou seja, o contador é atualizado numa frequência de .

Além disso, o número armazenado no registrador TIMx\_ARR, também definido pelo usuário, é o que determina o valor máximo do contador, o qual portanto vai de 0 até ARR. Os valores de contagem são armazenados e atualizados no registrador TIMx\_CNT.

Como o contador é atualizado no número de vezes definido pelo valor de ARR, a frequência de chaveamento é definida por . Para ilustrar isso, fez-se a *Figura 11*, que mostra a evolução do valor do timer com as suas atualizações, para o caso do modo up.

Figura 11: Operação do contador e frequência de chaveamento



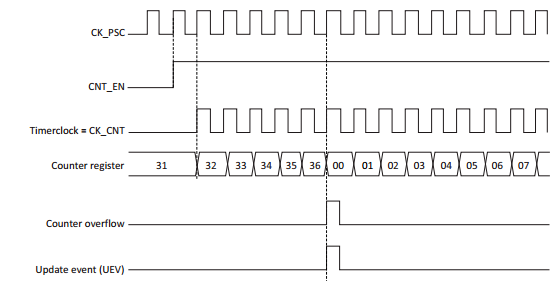
Fonte: Próprio Autor

1. Edge Aligned
   1. Modo Up

Neste modo, o contador vai de 0 até o valor armazenado em TIMx\_ARR, partindo novamente do 0 após isso ocorrer (evento de overflow). Caso o registrador TIMx\_RCR não esteja sendo usado, é criada uma atualização (update event - UEV) a cada overflow. Do contrário, a atualização ocorre no número de ciclos (TIMx\_RCR + 1).

Na figura x, o sinal CK\_PSC é o clock usado pelo processador, CNT\_EN é o sinal de enable do contador e CK\_CNT/Timerclock é o clock do timer. Como não foi escolhido um valor de prescaler (PSC = 0), o clock do timer é o mesmo do barramento. O Counter register equivale ao registrador TIMx\_CNT e armazena os valores do contador e é atualizado na borda de subida do clock do contador. Como ARR foi ativado em 36, quando o counter register atinge 36, a contagem é recomeçada. Nota-se também que o registrador TIMx\_RCR não está sendo usado.

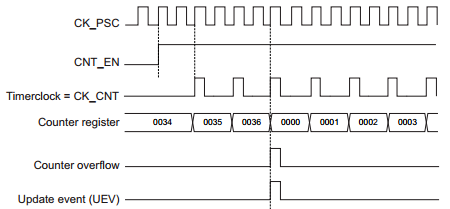
Figura 12: Sinais referentes ao contador (PSC = 0)



Fonte: Reference manual, página 520 [2]

Na *Figura 13*, mostra-se um exemplo em que o sinal de clock do processador (CK\_PSC) é dividido por 2 (PSC = 1), e portanto o sinal de clock do contador (TimerClock) tem sua frequência também dividida por 2.

Figura 13: Sinais referentes ao contador (PSC = 1)



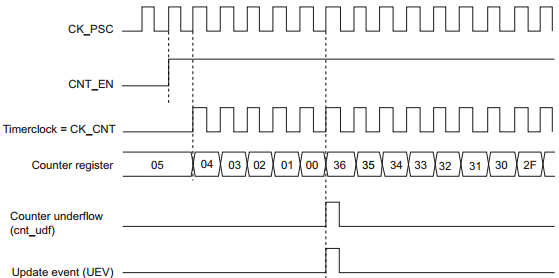
Fonte: Reference Manual, página 521 [2]

* 1. Modo Down

O contador no modo down funciona analogamente ao contador no modo up, com a diferença de que a contagem é decrescente, começando no valor de ARR e indo até 0. Quando o contador chega em 0 (evento de underflow), a contagem é recomeçada a partir de ARR.

O funcionamento do clock do contador (Timerclock) também é análogo ao modo Up no que se refere ao valor escolhido de PSC. Ou seja, se PSC = 0, o sinal TimerClock é o mesmo de CK\_PSC desde que o timer esteja tivado (CNT\_EN = HIGH). O mesmo vale para o registrador TIMx\_RCR.

Figura 14: Sinais referentes ao contador (PSC = 0)

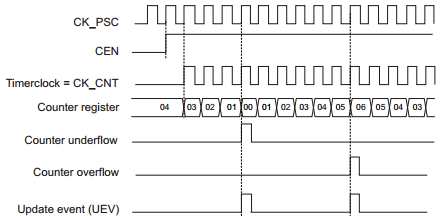


Fonte: Reference manual, página 524 [2]

1. Center Aligned
   1. Modo Up-Down

No modo center-aligned, o contador vai de 0 até o valor de ARR - 1, gera um evento de overflow; depois decresce de ARR até 0, gerando um evento de underflow. O funcionamento quanto ao clock interno e o valor de PSC é análogo ao procedimento no modo up.

Figura 15: Sinais referentes ao funcionamento do contador (PSC = 0)



Fonte: Reference Manual, página 527 [2]

O registrador TIMx\_RCR (Repetition Counter) serve para configurar a geração das atualizações (Update Events-UEVs). Se este registrador não for configurado (TIMx\_RCR = 0), o timer por padrão gera uma atualização a cada ciclo do contador. Se TIMx\_RCR = 1, ele gera uma atualização a cada dois ciclos do timer, se TIMx\_RCR = 2, ele gera uma atualização a cada três ciclos do timer. Ou seja, as atualizações são geradas a n + 1 ciclos, em que n é o valor armazenado no registrador TIMx\_RCR.

Para as aplicações deste manual, usou-se TIMx\_RCR = 0 por não ser necessário fazer para outro valor.

1.3.2.2. Modo PWM

O Modo PWM do TIM1 pode ser usado para gerar sinais modulados por largura de pulso, de frequência ligada ao valor armazenado em TIMx\_ARR e com duty cycle definido pelo valor armazenado no registrador TIMx\_CCRx, sempre através de uma comparação.

Tal comparação vai dependender do modo de operação do contador, dentre duas possibilidades: edge aligned ou center aligned, previamente definidas no setor 1.3.2.1 (“Tipos de Contador”)

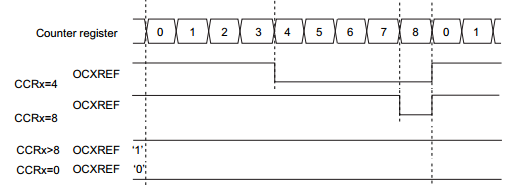
1. Edge Aligned

No caso edge aligned, a comparação depende do modo de operação do contador:

* TIMX\_CNT TIMx\_CCRx com o contador no modo up;
* TIMx\_CNT TIMx\_CCRx com o contador no modo down;

Um exemplo do funcionamento para o modo up é descrito na *Figura 16*, com ARR = 8 e diferentes valores de CCRx. O sinal OCXREF é o resultado da comparação sendo analisada.

Figura 16: Funcionamento do TIM1 no modo PWM edge-aligned com ARR = 8



Fonte: Reference Manual, página 542 [2]

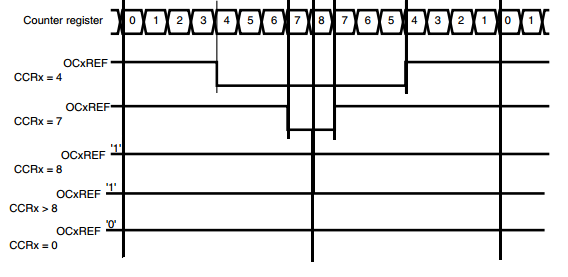
Conforme a *Figura 16*, percebe-se que o valor da saída permanece em HIGH enquanto o valor de CNT (Counter Register na figura) é menor que CCRx, tornando-se LOW quando passa do valor de CCRx. Para o caso em que CCRx = 0, a saída é sempre LOW pois a condição verificada é sempre verdadeira (TIMX\_CNT TIMx\_CCRx), e para o caso em que CCRx>ARR, a saída é sempre HIGH pois a mesma condição nunca é verdadeira.

Para o contador no modo down, o funcionamento é o dual: a saída é HIGH quando a condição verificada (TIMx\_CNT TIMx\_CCRx) é verdadeira, e LOW para quando é falsa.

1. Center Aligned

O modo center aligned (contador tipo up-down) funciona basicamente como uma extensão do modo up. A condição testada é TIMX\_CNT TIMx\_CCRx: quando verdadeira a saída é HIGH, e quando falsa a saída é LOW. A *Figura 17* exemplifica esse funcionamento para ARR = 8 e diferentes valores em CCRx.

Figura 17: Funcionamento do TIM1 no Modo PWM center-aligned com ARR = 8



Fonte: Reference Manual, página 543 [2]

1.3.3. UART

1.3.3.1 Definição

O que é UART

Explicar o frame de comunicação (estrutura do que é passado) com ilustração

1.3.3.2 Implementação

a) Explicar modo polling

b) Modo interrupção

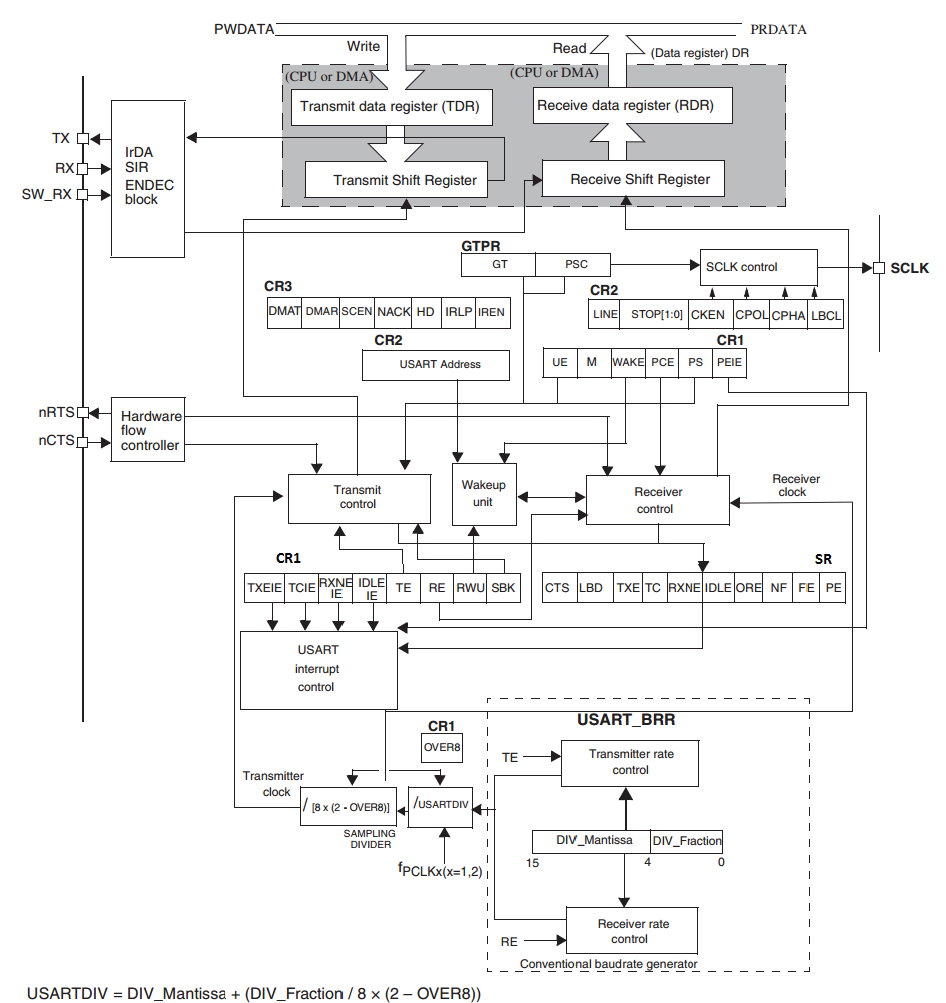
c) Modo DMA

Em cada item incluir: explicação teórica.

Configuração no Cube, exemplo no código fonte (colocar em anexo)

O Universal Synchronous Asynchronous Receiver Transmitter (USART) permite a troca de dados com equipamentos externos através de diversas taxas de transmissões (“baud rates”). Ele permite comunicação de uma via só, como também de duas vias. Para tal, ele conta com dois módulos: um de leitura (Receive Data In - RX) e um de envio/transmissão de dados (TX – Transmit Data Out).

Figura 18: Diagrama de Blocos do Periférico USARTDiagrama de Blocos do Periférico USART



Fonte: Reference Manual, página 972 [2]

1.3.4. ADC

Capítulo em desenvolvimento.

1.3.5. DAC

Capítulo em desenvolvimento.

1.4 Processador

O processador usado no microcontrolador STM32F407VGT6 é o ARM Cortex-M4. Uma de suas vantagens é ter uma Unidade de Ponto Flutuante, ou Floating Point Unit (FPU). Ela permite a conversão de variáveis em ponto fixo para ponto flutuante, com precisão simples (variáveis tipo float) e dentro do padrão IEEE 754.

O processador conta ainda com o Nested Vector Interrupt Controller (NVIC), responsável pela parte de interrupções e eventos.

1.4.1. Diferença entre ponto fixo e ponto flutuante

A representação de números em ponto fixo é assim denominada por ter a posição da vírgula predeterminada, onde a vírgula é utilizada para separar os algarismos inteiros dos fracionários. Em computadores, a notação binária em ponto fixo normalmente é utilizada em complemento de dois. Uma das limitações desta representação é a faixa fixa de números que podem ser representados, problema cuja solução pode ser utilizar a representação de números em ponto flutuante.

A aritmética de ponto flutuante representa os números nas seguintes partes principais: sinal, mantissa e a base com seu respectivo expoente. Pelo padrão IEEE 754, o número deve estar representado como SM\*, onde ‘S’ é o sinal, ‘M’ é a mantissa (1M2) e ‘***e***’é o expoente. Se a representação é em bits, deve-se seguir o seguinte padrão:

Figura 19: Representação de um número em ponto flutuante

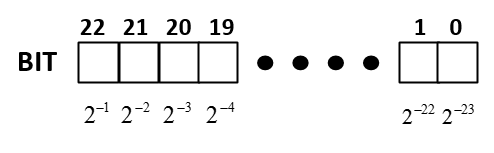


Fonte: Próprios autores

O sinal é sempre representado por um bit: 0 para positivo e 1 para negativo. O expoente é composto por 8 bits e a mantissa por 23 bits. Para converter um sinal decimal para ponto flutuante, basta seguir os seguintes passos:

1. Representar o sinal para a forma SM\*2^***e***, mencionada anteriormente;
2. Definir o bit do sinal (0 para negativo e 1 para positivo);
3. Fazer o cálculo ‘***e***’ + 127. O resultado é o número que deve ser representado em binário na parte “expoente” da representação;
4. A mantissa é preenchida conforme o valor de M com 1M2. Para M = 1, todos os bits da mantissa são ‘0’. Caso contrário, os bits devem ser preenchidos conforme as equivalências:

Figura 20: Estrutura da mantissa



Fonte: Próprios Autores

Como o valor do expoente pode ir de -126 a +128, existe, portanto, na representação por ponto flutuante uma faixa enorme de valores que podem ser representados.

Se o leitor desejar fazer a conversão de um número decimal específico, além do procedimento apresentado é possível utilizar um conversor, como por exemplo o da referência [3], atentando-se apenas se ele utiliza o padrão IEEE 754.

1.4.2. Interrupções

Para o processador, uma interrupção é uma função em que ele suspende o seu funcionando quando ocorre um evento específico. Assim ele executa uma rotina específica chamada Interrupt Handler (tratador de interrupção). Nela os comandos necessários para a interrupção devem ser executados e depois o processador retoma o seu funcionamento a partir de onde ele parou. O Nested Vector Interrupt Controller (NVIC) é responsável por tais interrupções, tendo as seguintes características:

* 82 entradas para interrupções internas ou externas;
* 16 níveis de prioridade programáveis;
* Baixa latência para lidar com interrupção e exceções;
* Implementação de registradores de controle do sistema.

As aplicações das interrupções são diversas, tais como detecção de bordas de subida ou descida, eventos de overflow ou underflow de contadores, acionamento de botões, entre outros.

2. Programação

Neste capítulo são descritos os principais aspectos referentes à programação propriamente dita com os microcontroladores STM32, no caso usando a placa de desenvolvimento Discovery.

2.1. STM32CubeMX

O STM32CubeMX é um software desenvolvido pela própria STMicroeletronics a fim de auxiliar o trabalho da programação de seus microcontroladores da linha STM32F4. Ele pode ser usado para gerar um código inicial com configurações gerais. A sua principal vantagem é permitir tais configurações de maneira gráfica, possibilitando melhor visualização da estrutura de clock, relação entre os pinos e os periféricos, etc. Ele pode ser baixado via website da ST [4].

2.2. GNU ARM Eclipse

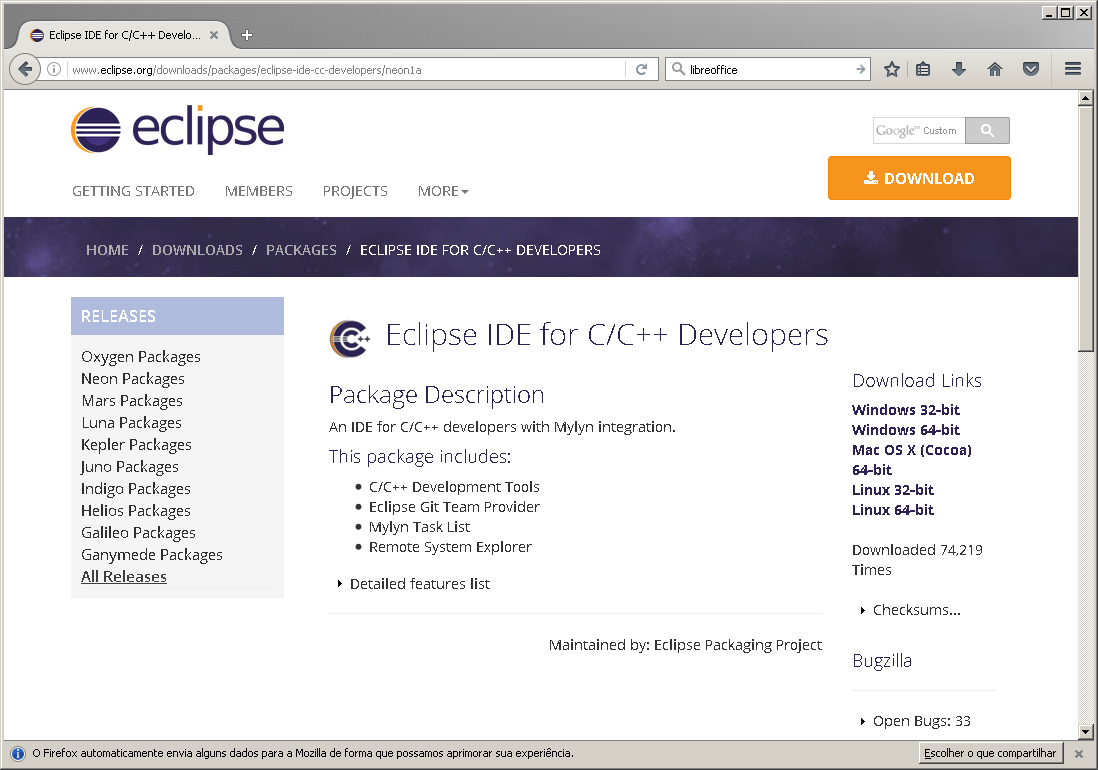
2.2.1. Instalação do Toolchain

0- Instale a última versão do Java a partir de<http://www.java.com/download>. Prestem atenção que o instalador automático baixa a versão de 32 bits mesmo que o computador seja de 64 bits. Na dúvida use o instalador manual. Se escolher 64 bits, baixe todos os softwares para 64 bits e vice-versa.

1 - Instalar o Eclipse IDE for C/C++ Developers

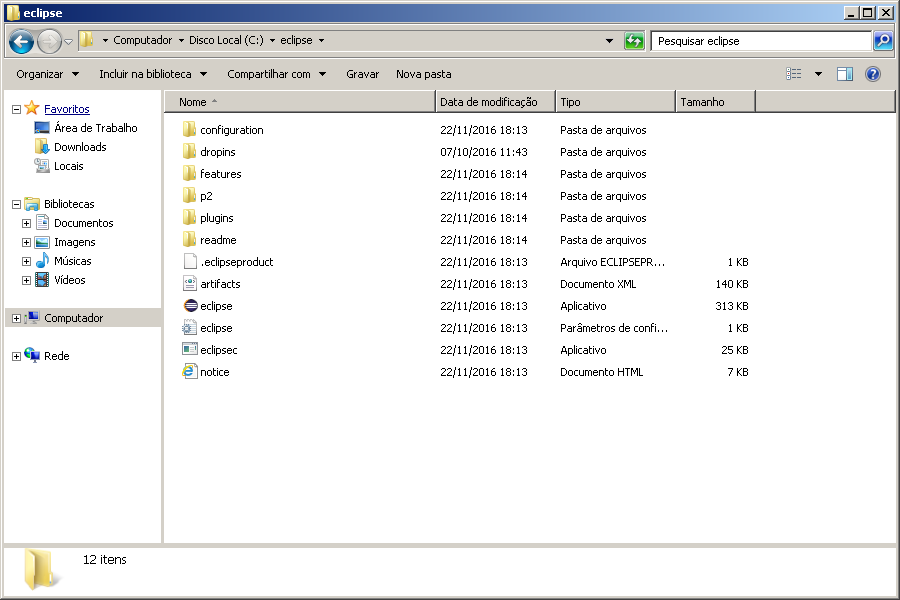
1. Baixe a versão correspondente em<http://www.eclipse.org/downloads/packages/eclipse-ide-cc-developers/neon1a>

Figura 21: Página da web para baixar a IDE



1. Para instalar, extraia os conteúdos do arquivo .zip baixado em C:\eclipse

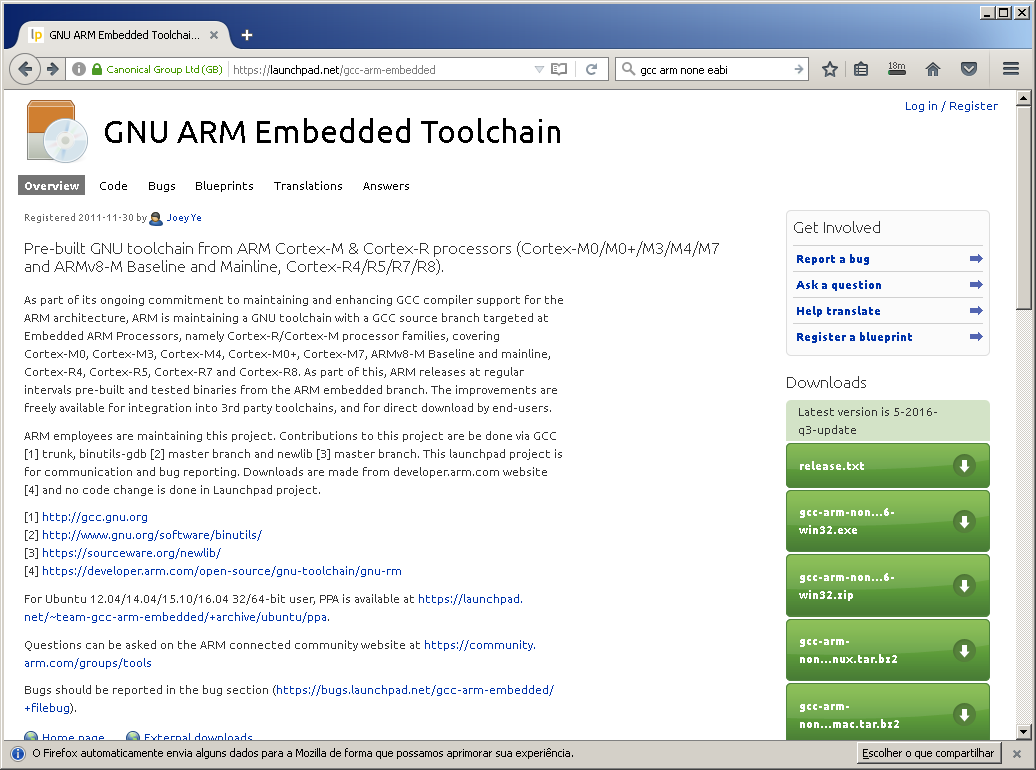
Figura 22: Pasta compactada com os arquivos baixados



2- Instalar o compilador GCC para ARM

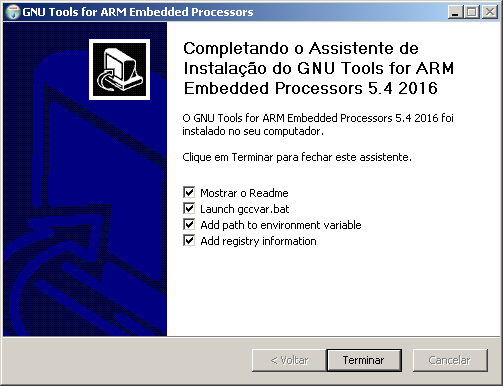
1. Faça download do instalador em<https://launchpad.net/gcc-arm-embedded>

Figura 23:Página da Web para baixar o compilador



1. Use as configurações-padrão para instalar. Na última etapa marque a opção "Add path to environment variable"

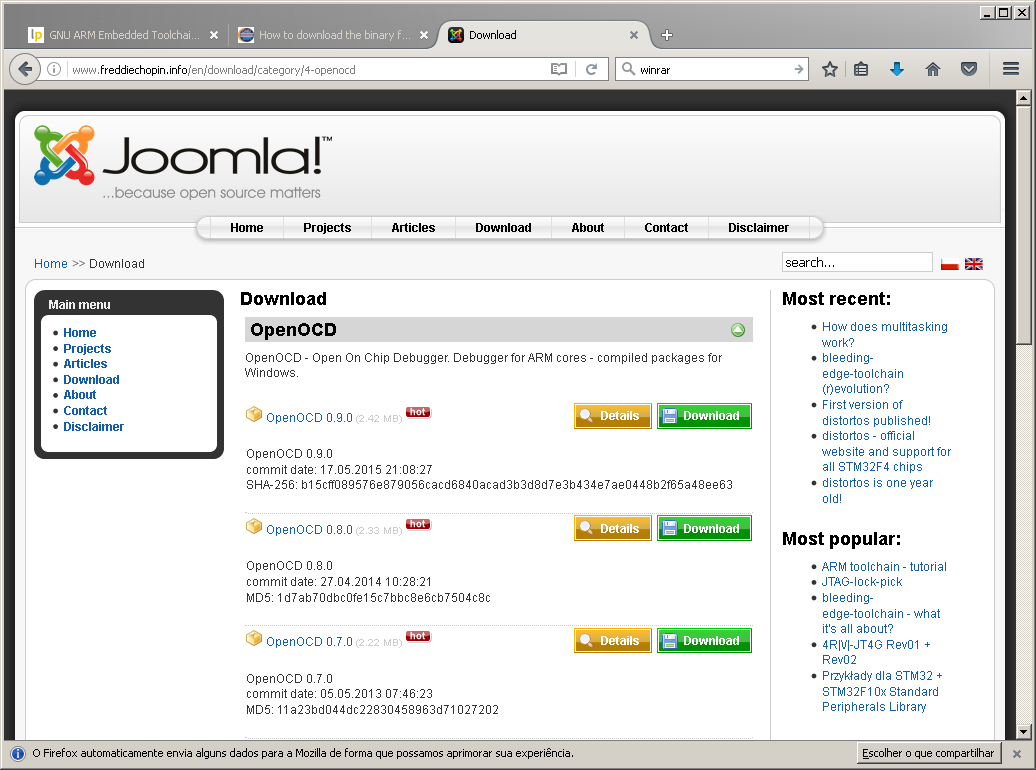
Figura 24: Configurações recomendadas de instalação



3- Instale a interface de depuração OpenOCD

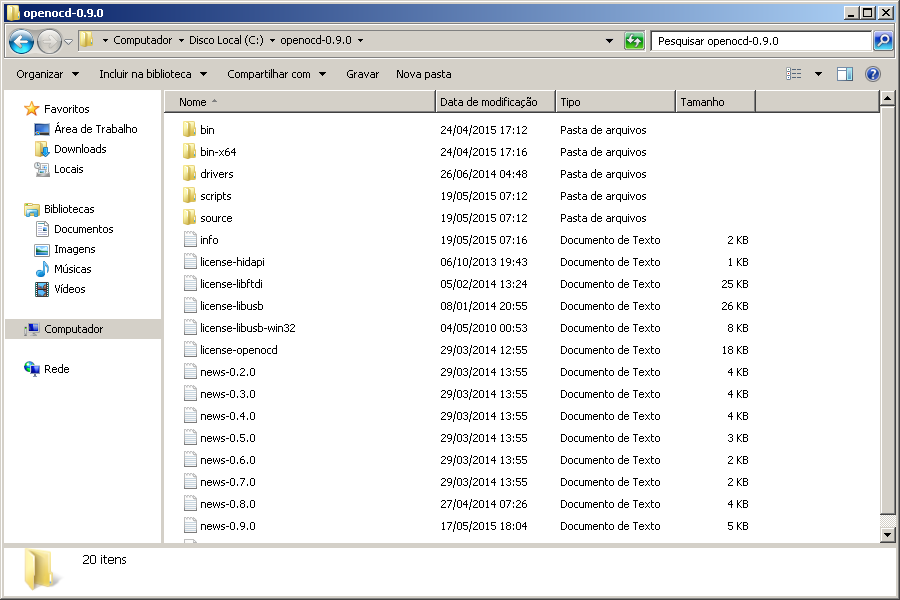
1. Baixe a última versão do site<http://www.freddiechopin.info/en/download/category/4-openocd>

Figura 25: Website para baixar interface de depuração OPENOCD



1. Extraia o conteúdo do arquivo .7z baixado para C:\openocd-0.9.0

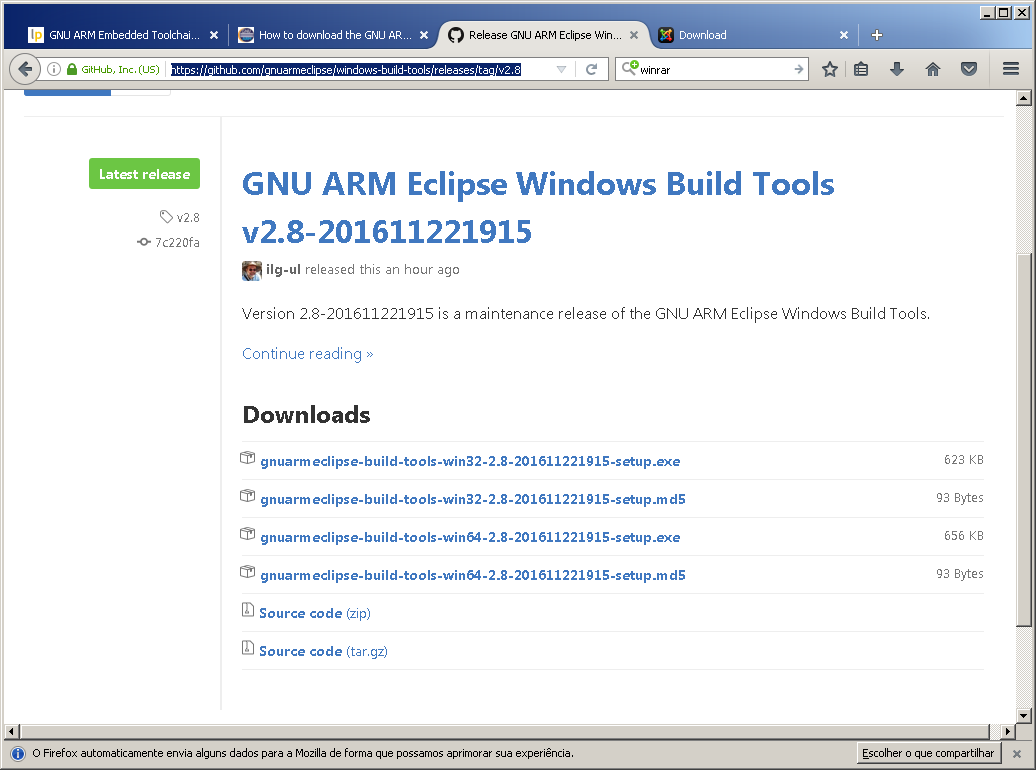
Figura 26: Pasta compactada baixada



4- Instale o Windows Build Tools

1. Faça download a partir de<https://github.com/gnuarmeclipse/windows-build-tools/releases/tag/v2.8>

Figura 27: Website para baixar o Windows Building Tools

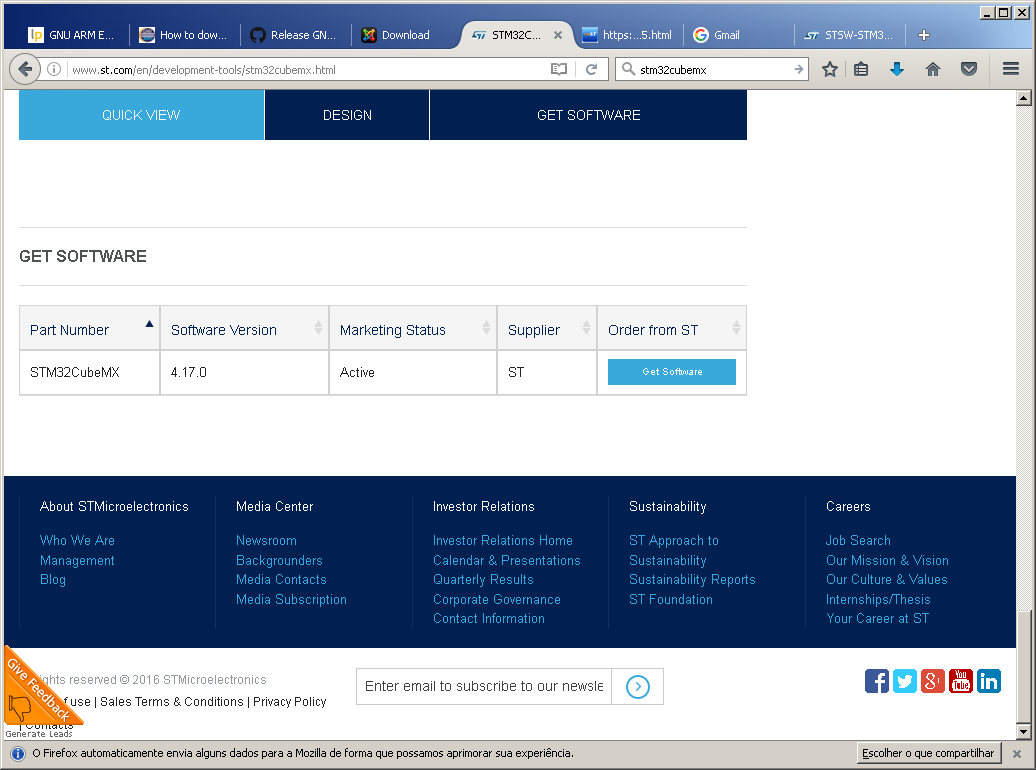


b) Execute o instalador e use as configurações-padrão.

5- Instale o STM32CubeMX

1. Baixe o instalador a partir de<http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-configurators-and-code-generators/stm32cubemx.html> (a validação do seu email pode ser necessária)

Figura 28: Website para baixar o programa STM32CubeMX



b) Use as configurações-padrão para a instalação

c) Abra o programa STM32CubeMX e clique em Help/Install New Libraries

d) Selecione o pacote de firmware para o microcontrolador desejado e clique em Install Now (pode demorar um pouco).

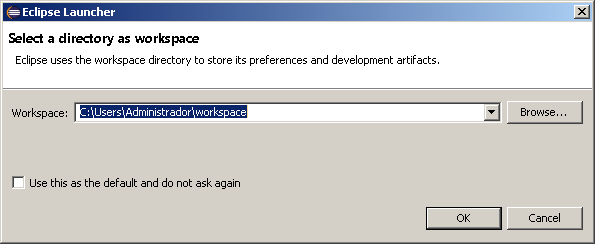
Figura 29: Seleção do pacote de firmware



6- Instale o plugin do GNU ARM para Eclipse

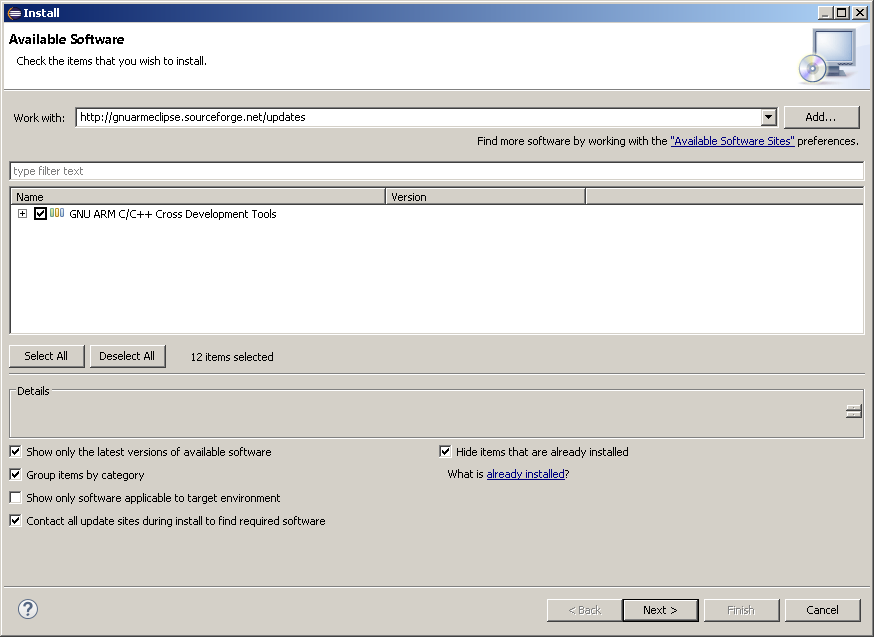
1. Abra o Eclipse (execute C:\eclipse\eclipse.exe) e use o workspace padrão

Figura 30: Seleção do diretório para workspace



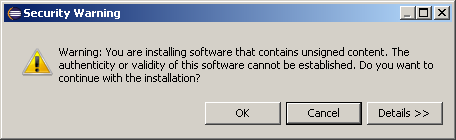
1. Clique em Help/Install New Software e cole<http://gnuarmeclipse.sourceforge.net/updates> no campo Work With e aperte enter. Depois marque a opção GNU ARM C/C++ Cross Development Tools. Prossiga com a instalação.

Figura 31:Software para a instalação



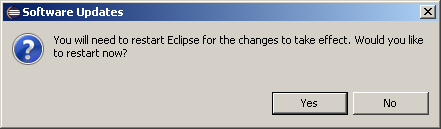
1. Confirme a instalação de software não-assinado

Figura 32: Confirmação de instalação de software não assinado



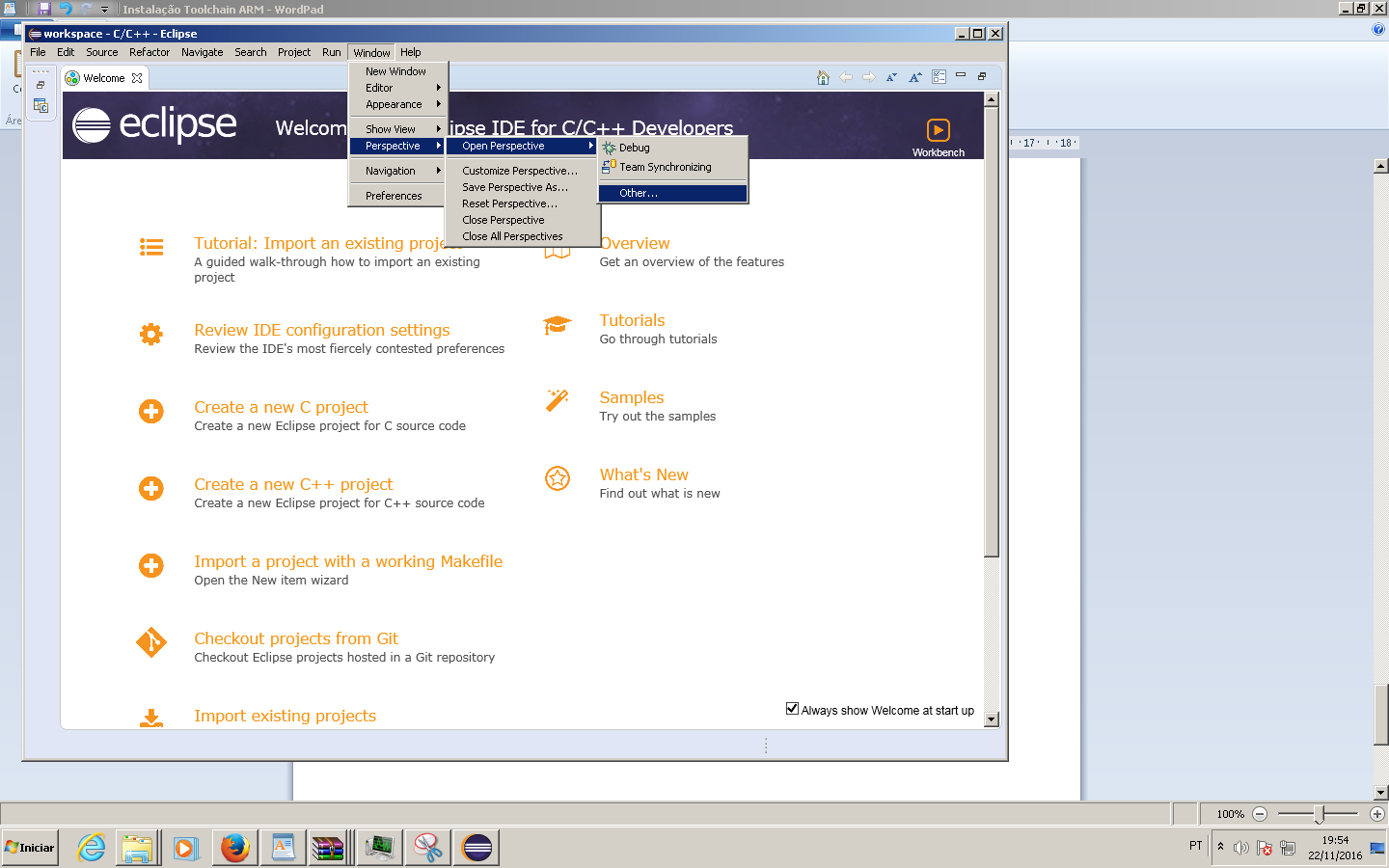
1. Reinicie o Eclipse

Figura 33: Reiniciar o Eclipse



e)Clique em Window/Perspective/Open Perspective/Other

Figura 34: Abrir uma nova Perspectiva



1. Selecione a opção Packs

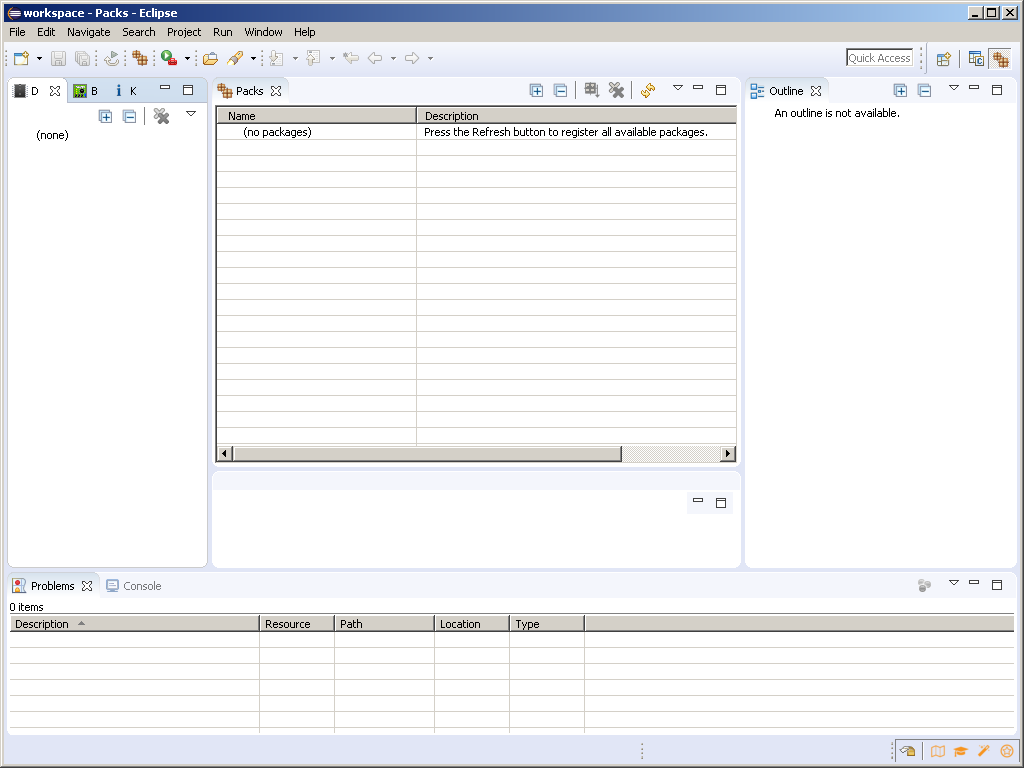
Figura 35: Selecionar opção Packs



g) Se necessário clique em Workbench

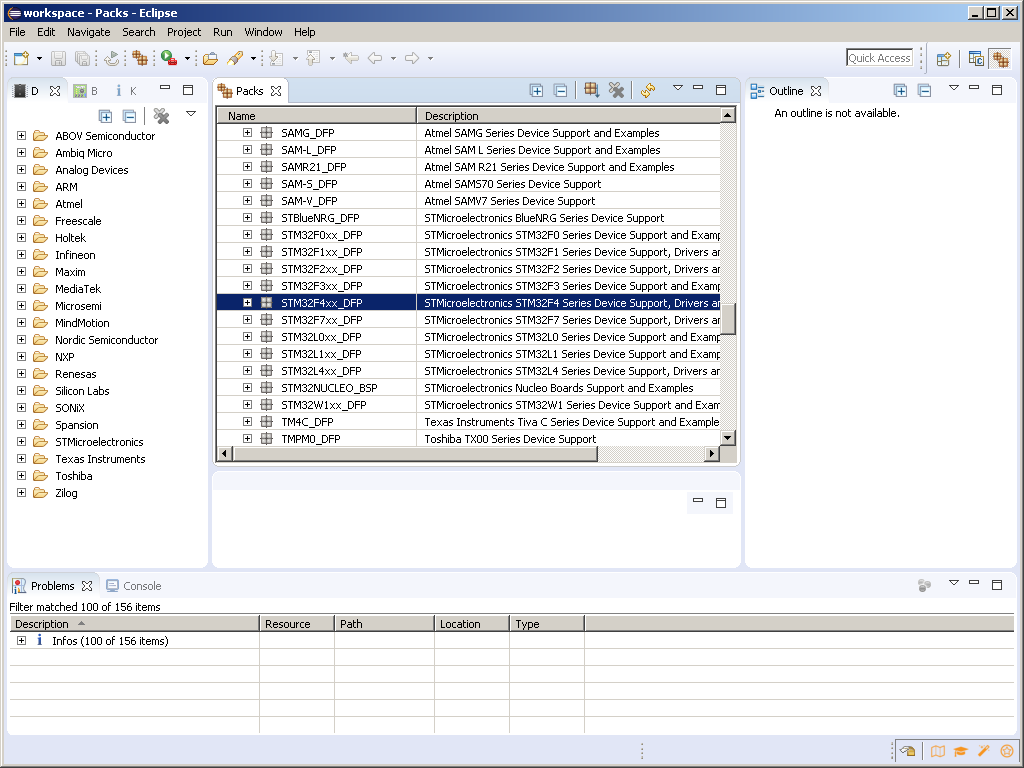
h) Atualize a lista de pacotes clicando em Refresh

Figura 36: Atualizar lista de pacotes



i) Selecione a opção referente ao microcontrolador desejado e clique com o botão direito nele e selecione a opção instalar (a operação é realizada em background e o status aparece na parte inferior da tela)

Figura 37: Selecionar microcontrolador



7) Se o computador não reconhecer a placa Discovery, baixe o driver a partir de http://www.st.com/en/embedded-software/stsw-link009.html

3. Geração de PWM

Existem mais de uma maneira para gerar de um sinal de PWM usando o microcontrolador do kit STM32F4-Discovery. Uma dessas maneiras é usando um contador em modo up, provido pelo periférico TIM1. Neste capítulo será descrito como gerar um sinal de PWM com **duty cycle fixo**. Para modulação senoidal, ver capítulo 4.

Antes de iniciar a programação, é necessário determinar a frequência de chaveamento do contador, no caso utilizando o modo up, conforme a fórmula (1).

(1)

Para este tutorial, escolheu-se o valor de frequência de chaveamento de 10kHz.

1. Abrir um novo projeto no programa STM32CubeMX compatível com o microcontrolador usado

No caso da placa Discovery, utiliza-se o microcontrolador STM32F407VGTx.

Figura 38: Página inicial do programa STM32CubeMX

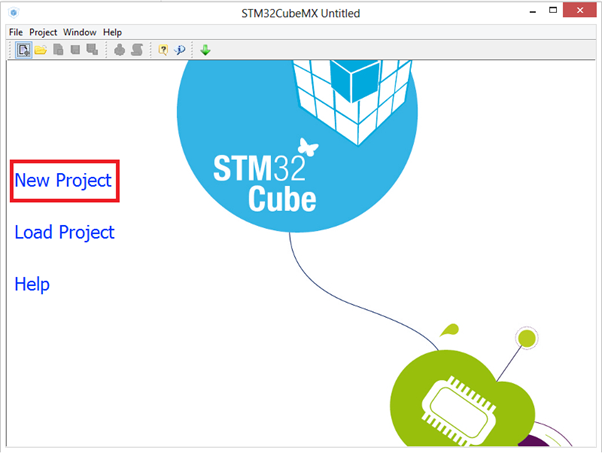
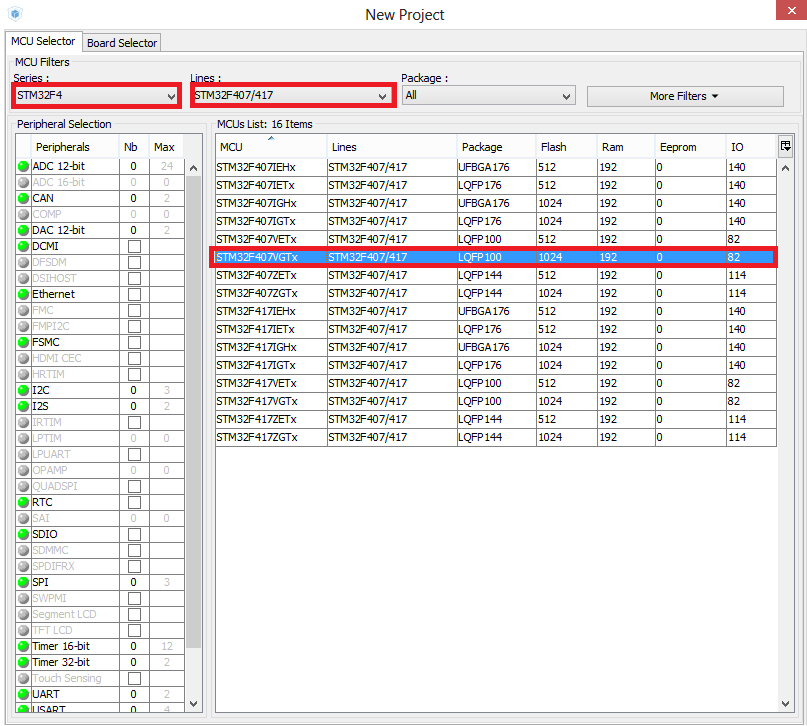


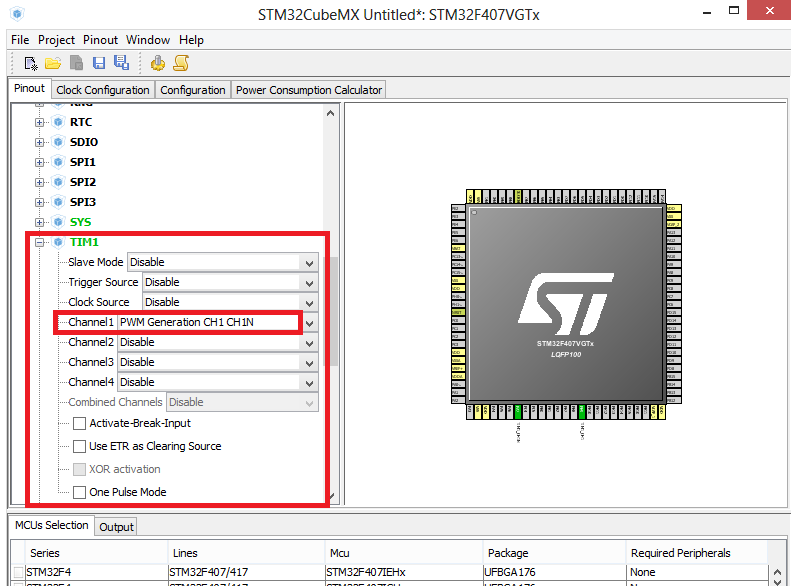
Figura 39: Selecionar o microcontrolador



1. Habilitar o periférico TIM1

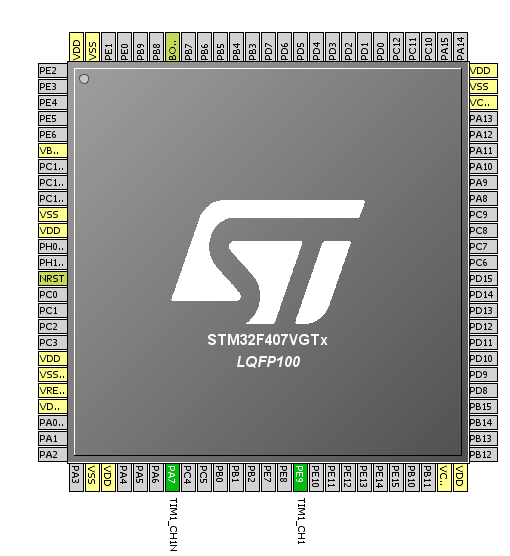
Na aba “pinout”, habilitar o periférico TIM1, sendo necessário ligar apenas um de seus canais desejados de acordo com o número de braços do inversor, no modo PWM. É possível escolher se deseja habilitar a saída normal de cada canal, a sua saída negada ou ambas.

Figura 40:Habilitar periférico TIM1



Notar no desenho do chip ao lado o nome dos pinos que foram habilitados como saídas do timer.

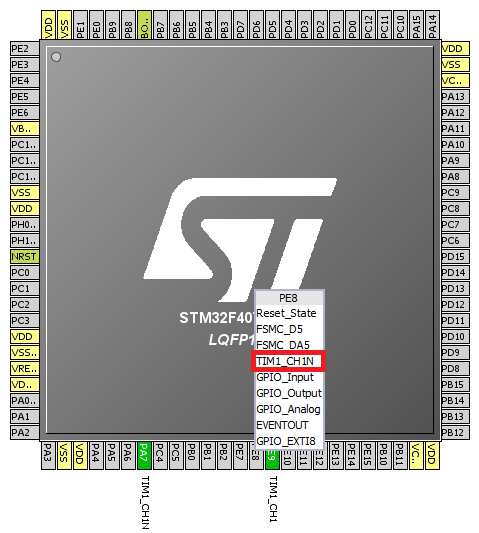
Figura 41: Desenho do chip no programa



1. Mudar os pinos de saída do timer (opcional)

Caso se deseje mudar a seleção feita pelo próprio programa dos pinos de saída do timer, é possível escolher os pinos para tal desde que eles tenham a configuração para funcionar como tal. Por exemplo, para mudar a saída negada do canal 1 (TIM1\_CH1N), escolheu-se o pino PE8 e verificou-se que ele pode funcionar para esta aplicação. Para ver quais saídas cada pino oferece, basta clicar em cima dele. Uma outra maneira de ver quais pinos também têm a opção de TIM1\_CH1N é clicar a tecla CTRL e selecionar o pino PA7 (configurado como TIM1\_CH1N).

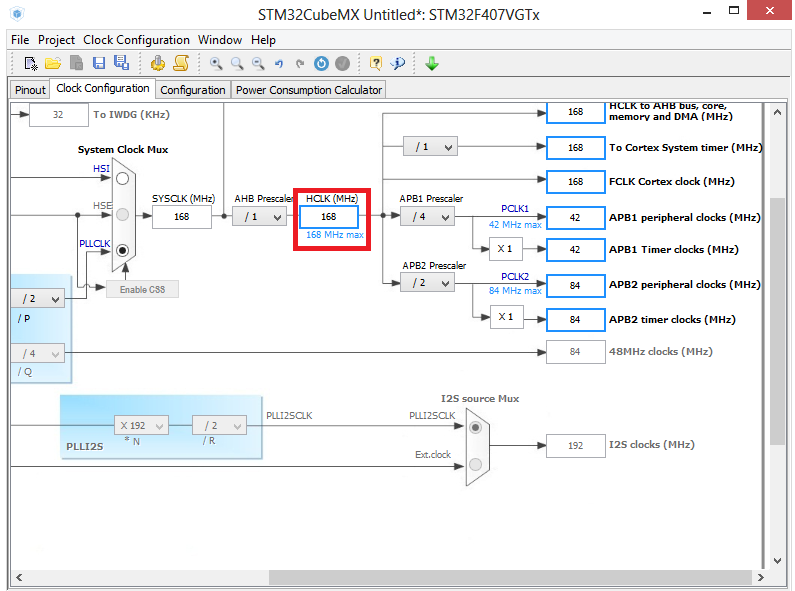
Figura 42:Mudança de pino (opcional)



1. Definir o valor de no programa STM32CubeMX

Para definir a frequência de atualização do contador, define-se HCLK na aba “clock configuration” digitando o valor e pressionando Enter. No caso optou-se por operar com a frequência máxima permitida pelo microcontrolador (168MHz).

Figura 43: Configuração de clock no STM32CubeMX



1. Definir os valores de PSC e ARR

Para uma frequência de chaveamento de 10kHz e =168MHz, escolheu-se o valores de ARR = 168 e PSC = 99 (valores escolhidos conforme a fórmula 1, fornecida no início do capítulo). Para configurar estes valores, basta abrir a aba “configuration”, clicar no botão “TIM1” na parte “control”, o qual abrirá uma janela de título “TIM1 configuration”. Aberta a janela, escrever os valores de PSC e ARR respectivamente nos campos Prescaler e Counter Period. Configurar também o contador no modo up. Neste caso optou-se por não utilizar o registrador RCR (descrito melhor no capítulo 1.3.2 - “TIM1”). Reescrevendo a fórmula (1):

Figura 44: Selecionar configuração de TIM1

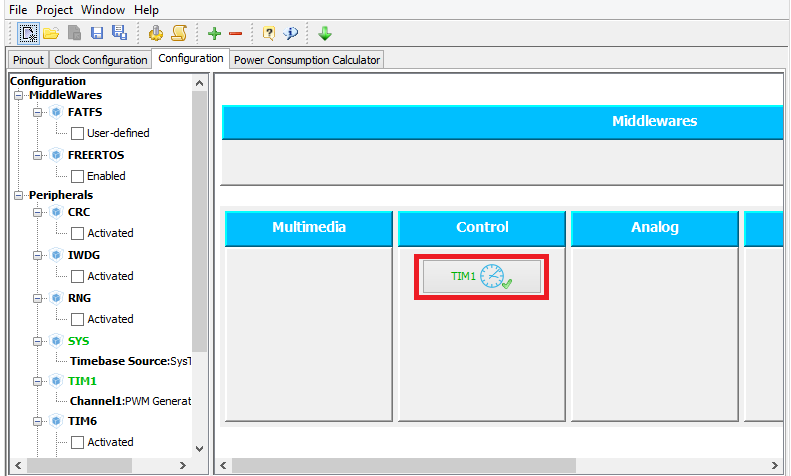
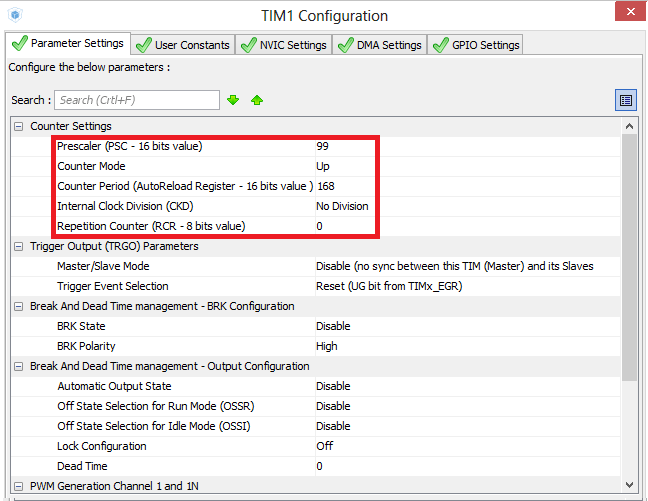


Figura 45: Configuração dos valores armazenados nos registradores



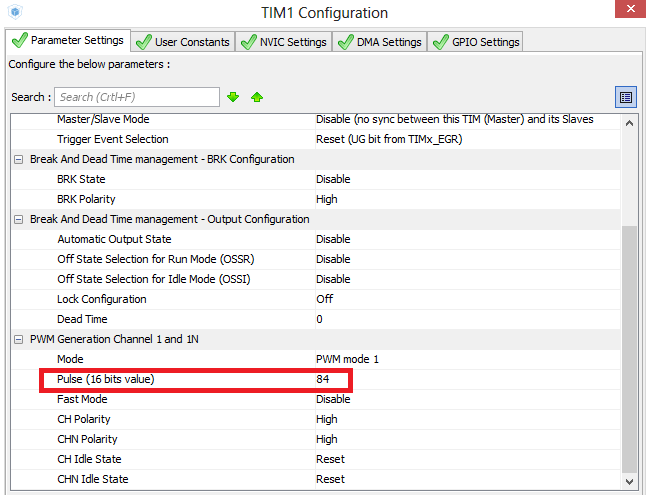
1. Definir o duty cycle

Definido o duty cycle da onda, deve-se definir um valor de 0 a ARR que será usado para satisfazer a comparação definida também no capítulo 1.3.2 - “TIM1”, usando a fórmula (2):

(2)

Para um duty cycle de 50%, por exemplo, o valor é 84. Ele deve ser armazenado no campo “Pulse”, ainda na aba “Parameter Settings” na janela “TIM1 Configuration”.

Figura 46: Configuração do valores de Pulse



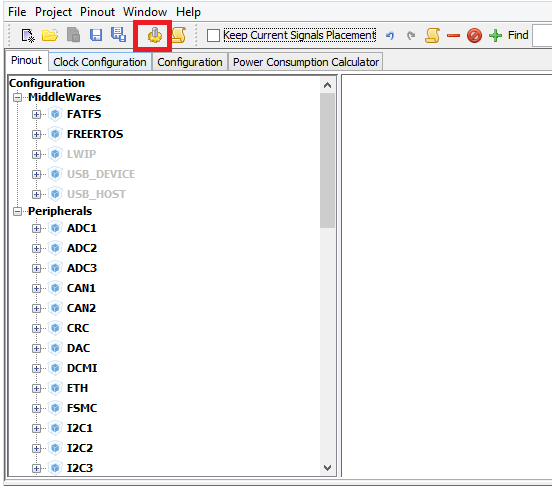
1. Gerar o código

Feitas estas configurações, deve-se gerar o código fonte. Para isso, clicar no ícone no formato de uma engrenagem em volta de um eixo (*Figura 46*) na parte superior da tela. Em seguida será aberta uma janela de título “project settings” e nela devem ser escolhidos o nome do projeto e um diretório em que o código gerado será salvo.

É importante também que a toolchain correta seja selecionada (no caso do GNUARM pode-se escolher EWARM) e que o pacote de firmware referente ao MCU utilizado esteja instalado. Caso não haja nada no campo referente ao firmware, é necessário baixar o pacote mais recente disponível, e para tal basta clicar no ícone de seta na *Figura 48*, o qual mostrará tudo que é possível ser baixado (tanto versões de software quanto de firmware).

Após gerar o código, abrir o diretório que o contém (normalmente o próprio programa redireciona o usuário para a pasta criada).

Figura 47: Ícone para gerar o código



Fonte: Screenshot

Figura 48: Configurações do projeto

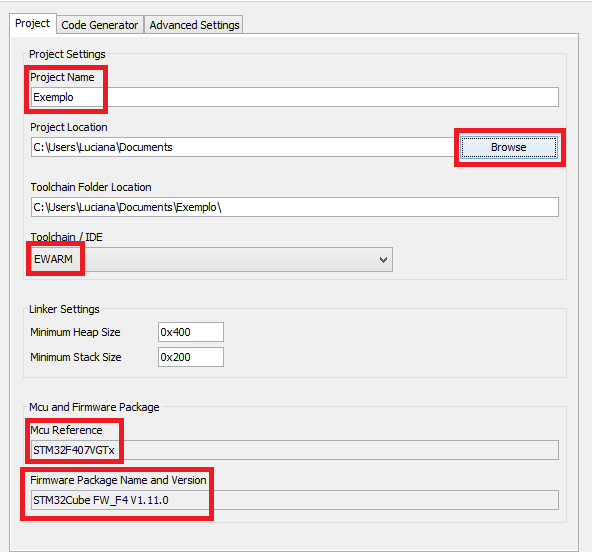
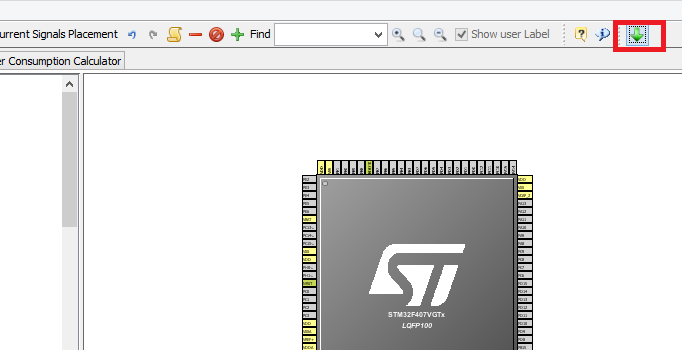


Figura 49: Ícone para baixar a versão mais recente de firmware



Fonte: Screenshot

1. Abrir um projeto no Eclipse

Gerado o código no STM32F4CubeMX, é necessário criar um projeto na IDE Eclipse que seja compatível com o microcontrolador usado. Para tal, após abrir o programa, clicar em FILE->New->C project. Será aberta uma janela de título “C Project”, e devem ser escolhidos o nome do projeto, o seu tipo (STM32F4xx C/C++ Project) e a sua toolchain (Cross ARM GCC). Feito isto, clicar em Next e na nova janela (“Target Processor Settings”) selecionar o tipo do chip e conteúdo vazio.

Clicar novamente em Next e na nova janela (“Folder Settings”) deixar os nomes padrões que o programa criou para cada pasta. Clicar em Next de novo e na janela redirecionada (“Select Configurations”) certificar-se que as plataformas de Debug e Release estão ambas marcadas. Clicar em Next novamente e depois em Finish.

Figura 50: Criação de um projeto no Eclipse

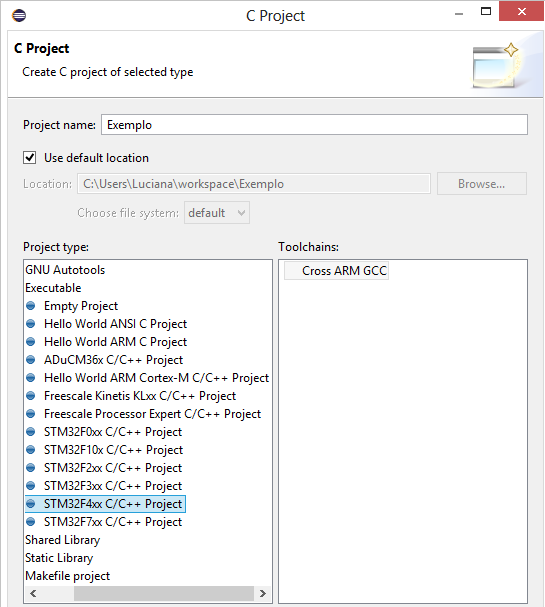


Figura 51: Seleção do microcontrolador

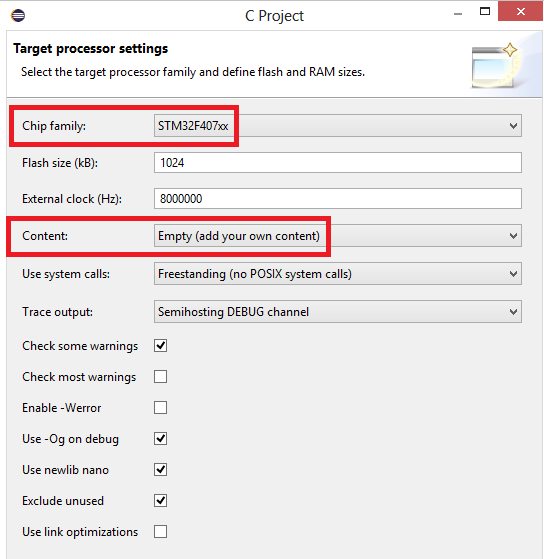


Figura 52: Pastas do projeto

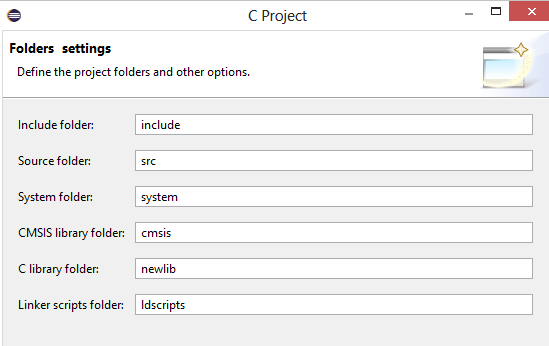
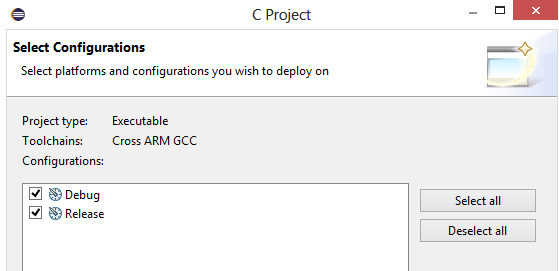


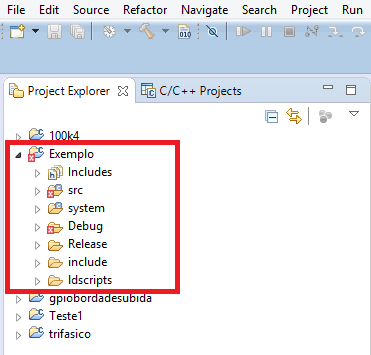
Figura 53: Plataformas do Projeto (Debug e Release)



1. Copiar os arquivos src e include

Criado o projeto, é possível ver suas pastas criadas no canto esquerdo, na aba “Project Explorer” (juntamente com outros projetos criados anteriormente).

Figura 54: Pastas do Projeto no Project Explorer



Agora é necessário copiar os arquivos das pastas “Src” e “Inc” criadas anteriormente no projeto do programa STM32F4CubeMX, para as pastas “src” e “include” respectivamente. Para isto, basta copiar os arquivos no diretório escolhido para o código gerado pelo STM32CubeMX e colar nas pastas no Project Explorer (talvez seja preciso subscrever alguns arquivos).

1. Ligar os canais do TIM1 escolhidos

Para habilitar os canais escolhidos do TIM1, é necessário acrescentar as seguintes linhas de código dentro da função MX\_TIM1\_INIT do arquivo main.c (pasta “Src”):

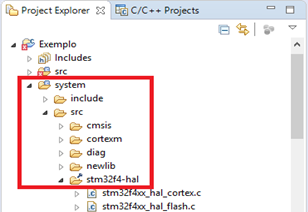
HAL\_TIM\_PWM\_Start\_IT(&htim1, TIM\_CHANNEL\_1;

HAL\_TIMEx\_PWMN\_Start(&htim1, TIM\_CHANNEL\_1);

1. Ativar as bibliotecas stm32f4xx\_hal\_tim.c e stm32f4xx\_hal\_timex.c

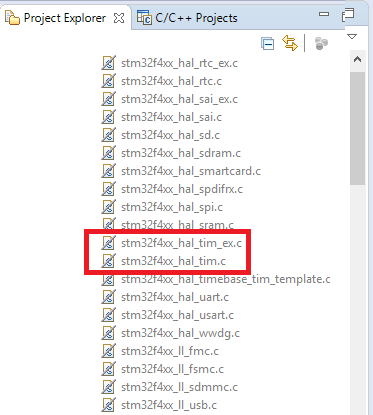
Na aba Project Explorer, abrir a pasta de bibliotecas, seguindo o caminho: system->src->stm32f4-hal. Nela, encontram-se todas as bibliotecas disponíveis (ativadas e não ativadas).

Figura 55: Caminho para acessar bibliotecas a serem ativadas



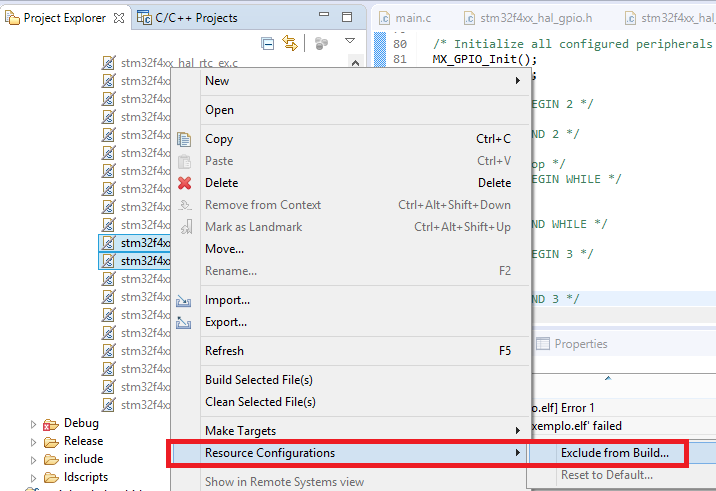
Na lista de bibliotecas, basta selecionar stm32f4xx\_hal\_tim.c e stm32f4\_hal\_timex.c, que em primeira instância deverão estar desativadas (coloração da fonte mais clara).

Figura 56: Bibliotecas a serem ativadas



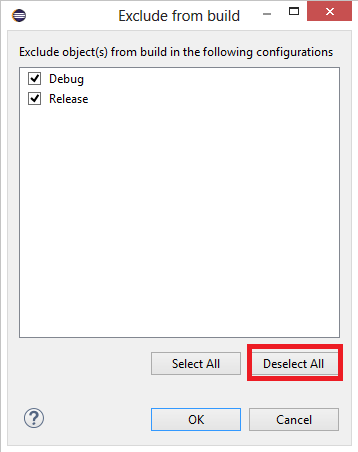
Após localizar as bibliotecas, selecioná-las e clicar com o botão direito do mouse. Depois, em “Resource Configurations”, clicar em “Exclude from Build”.

Figura 57: Comandos para ativar as bibliotecas



Feito isto, deve aparecer uma janela de título “Exclude from Build”. Nela, clicar em Deselect All e depois em OK.

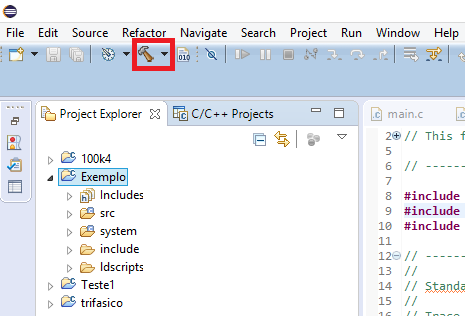
Figura 58: Exclude from build



1. Compilar

Para compilar, basta clicar no botão cujo ícone é o desenho de um martelo.

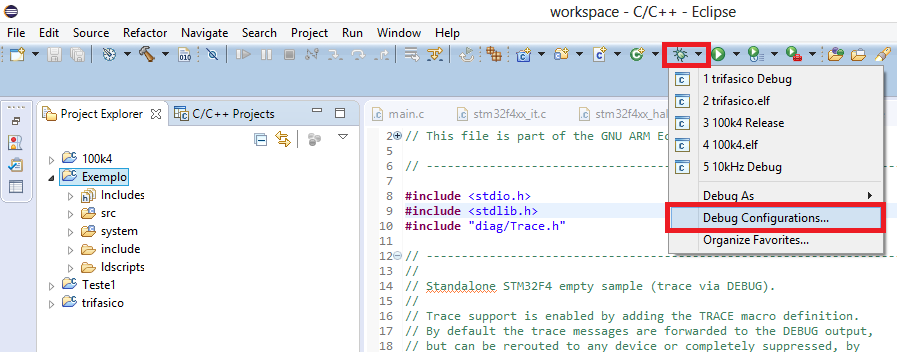
Figura 59: Ícone para compilação



1. Abrir configurações de debug

Se a compilação do programa foi feita com sucesso, é preciso passar para o ambiente de Debug. Para tal, clicar na seta para baixo logo ao lado do ícone cujo desenho é de um inseto, e clicar em Debug Configurations.

Figura 60: Abrir configurações de Debug



Abrir o menu “Debug Configurations”. Clicar em “GDB OPEN OCD Debugging” na aba esquerda da janela e clicar em New. Isto deve criar uma nova configuração de Debug com um nome padrão “<nome do projeto> Debug”. Neste caso, por exemplo, como o nome do projeto era “Exemplo”, ficou como “Exemplo Debug”. Feito isto, na aba “Debugger”, acrescentar “-f board\stm32f4discovery.cfg” em “Config Options”. Depois clicar em “Apply”.

Figura 61: Clicar em GDB OpenOCD Debugging e New

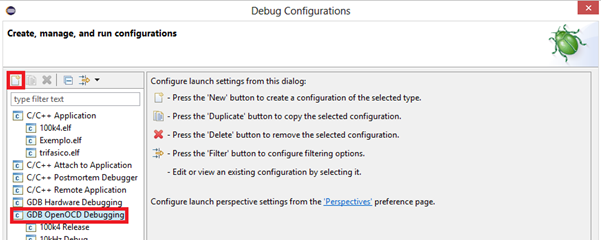


Figura 62: Configurações de Debug

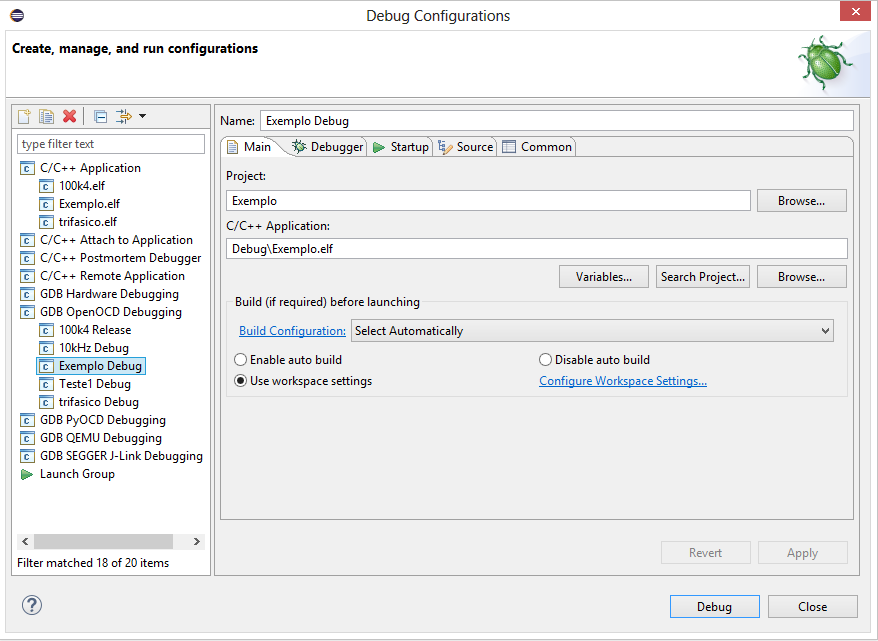
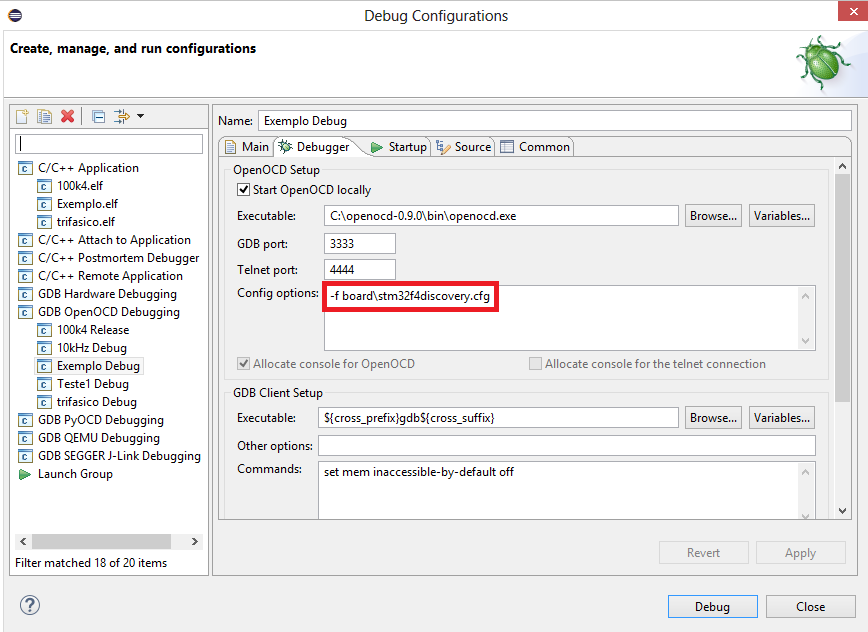


Figura 63: Configurações de Debug (continuação)



Se na aba Main o campo C/C++ Application estiver em branco, o que ocorre se o Debug for iniciado antes da compilação, clicar em “Search Project” e selecionar o arquivo .elf disponível.

Figura 64: Seleção do arquivo .elf

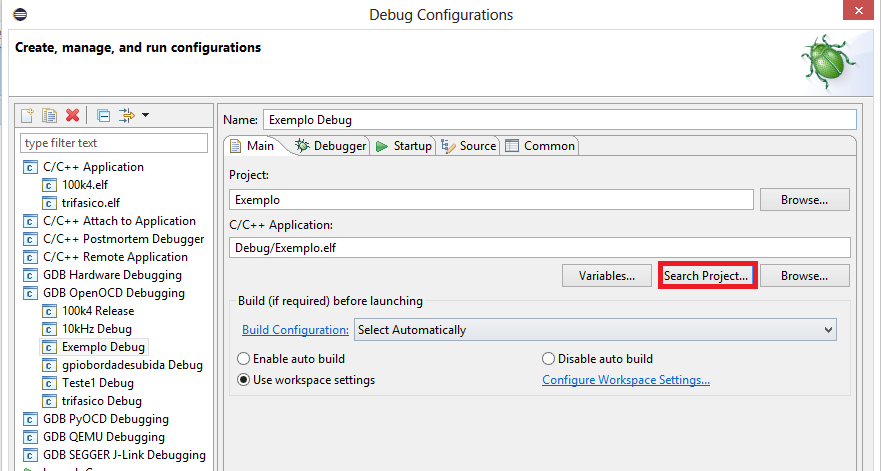
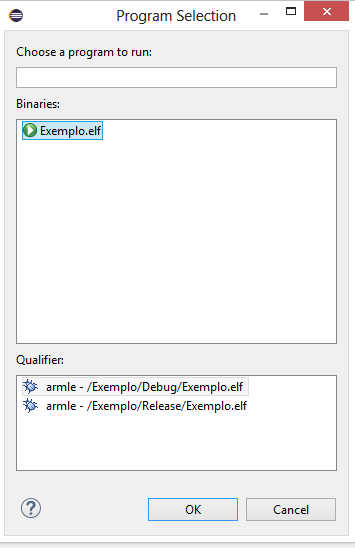


Figura 65: Seleção do arquivo .elf (continuação)



Em seguida, clicar em Debug. Se o Debug não começar automaticamente, basta clicar novamente no mesmo ícone da figura 58.

1. Rodar o programa

Feito o Debug com sucesso, o Eclipse deve abrir o ambiente de Debug automaticamente. Se isto não ocorrer, é possível acessar tal ambiente clicando no ícone parecido com o de Debug (figura 58) no canto superior direito da tela.

Figura 66: Mudar para ambiente de Debug



Acessado o ambiente, clicar em Resume. O programa deve funcionar se todos os passos anteriores forem executados com sucesso.

4. Modulação Senoidal e Aplicações a Conversores Estáticos

Para criação de PWM com modulação senoidal, o procedimento é análogo ao do capítulo 3, mas com algumas particularidades.

1. Ao criar o projeto no STMCubeMX, na aba PROJECT SETTINGS, selecionar a caixa que permite fazer interrupções.
2. Após abrir o Eclipse, habilitar a FPU
   1. Clicar com o botão direito do mouse em cima do nome do projeto no Project Explorer
   2. Clicar em Properties

Figura 67: Ligando FPU

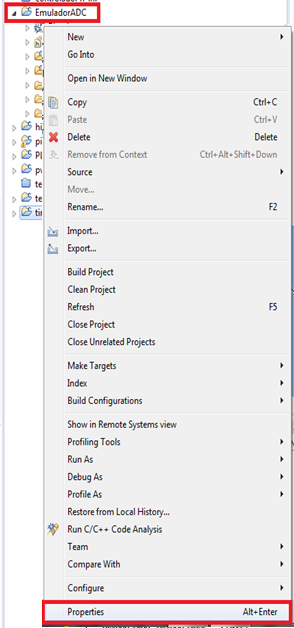
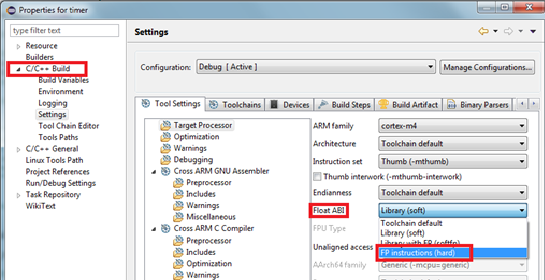
* 
  1. Ir em C/C++ Build -> Settings. Em Float ABI, selecionar FP Instructions (hard).

Figura 68: Ligando FPU (continuação)



1. Ao abrir o eclipse e após copiar os arquivos e habilitar a saída do TIMER no arquivo main.c, abrir o arquivo stm32f4xx\_it.c, que também se encontra na pasta “src”.
   1. No mesmo arquivo, importar a biblioteca <fastmath.h> escrevendo o seguinte comando no início do arquivo;

#include <festmath.h>;

* 1. Declarar, antes da função TIM1\_CC\_IRQHandler, as variáveis aux (uint32\_t) e interm (float) utilizando os seguintes comandos:

float interm;

uint32\_t aux;

* 1. Na função TIM1\_CC\_IRQHandler, copiar o seguinte procedimento:

1. aux = aux + 1;
2. if (aux > TIM1->ARR){
3. aux = 0;
4. }
5. interm = sinf(F\*2\*3.1416\*(float)aux/(float)TIM1->ARR);
6. TIM1->CCR1 = (TIM1->ARR + (uint32\_t)((float)TIM1->ARR\*interm))/2;

O Handler é uma função de manejo de interrupções feitas pelo NVIC. No caso, ele está sendo utilizado para monitorar o funcionamento da modulação senoidal, feita através da iteração acima.

A variável aux serve como contador em modo Up, sendo zerada toda vez que atingir o valor de ARR (linha 3). A variável interm é utilizada para calcular o valor da onda senoidal, de acordo com a iteração definida por aux. Para tal utiliza-se a função sinf, que retorna valores float. Para esta onda, deve-se também definir o valor de sua frequência no lugar da variável F no código fonte (linha 5).

O valor de CCR1 é comparado ao de CNT (TIMX\_CNT TIMx\_CCRx), conforme mencionado no capítulo 1.3.1 (“TIM”). Se a comparação for verdadeira, a saída do timer é HIGH e do contrário é LOW. Como o valor de CCR1 é senoidal, a modulação da saída também é senoidal.

Na *Figura 68* e Figura *69*, tem-se um exemplo de cálculo para as variáveis interm e CCR1. No caso escolheu-se ARR = 100 e F = 1. Percebe-se assim que a senóide calculada na variável interm tem frequência de 1Hz e seu módulo vai de -1 até +1. É importante notar que a o cálculo do seno segue a fase exibida no gráfico da Figura *70*, que vai de 0 até 2π (6,2832).

Para se ter uma ideia melhor do resultado da comparação TIMX\_CNT TIMx\_CCRx, estabeleceu-se um período de chaveamento para o Timer (Figura *71*) e fez-se uma imagem ilustrativa do que seria aproximadamente a saída do periférico (Figura *72*)

Figura 69: Relação entre variável interm (eixo y) e AUX (eixo x)

Figura 70: Cálculo de CCR a partir de Interm

Figura 71: Fase do seno em interm e CCR

Figura 72: Período de chaveamento do Timer

Figura 73: Saída do Timer

5. Comunicação

Capítulo em desenvolvimento.

6. Aquisição de Dados

Capítulo em desenvolvimento.

7. GPIO

7.1 Comandos de leitura e escrita

Através de configuração de software, é possível ativar portas GPIO como alto (HIGH) ou baixo (LOW), por meio da função WritePin. Existe também a função ReadPin, que lê e interpreta o valor lógico de um pino e devolve um valor dentre 0 (LOW) e 1 (HIGH). Um exemplo desses comandos seriam:

* HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_SET);

Este comando ativa em alto (HIGH) o pino PB2 da placa Discovery.

* HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_2, GPIO\_PIN\_RESET);

Este comando ativa em baixo (LOW) o pino PB2 da placa Discovery.

* {variável}zeroouum=HAL\_GPIO\_ReadPin(GPIOx, GPIO\_PIN\_1);

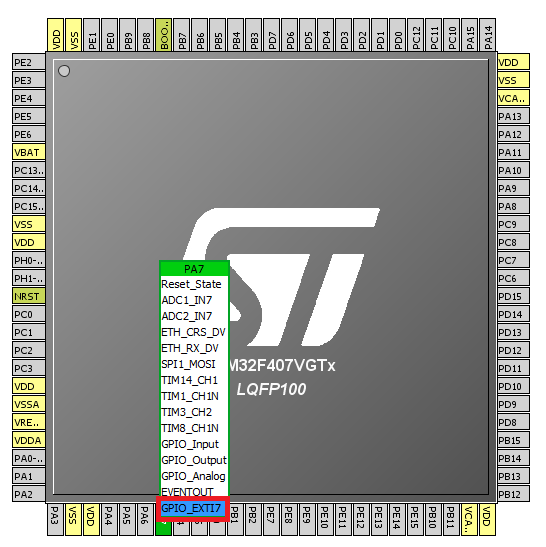
Este comando lê e retorna o valor lido no pino escolhido.

7.2 Detecção de borda de subida e descida

Para ativar um pino como detector de borda de subida ou descida, basta configurá-lo via o programa STM32F4CubeMX. Seguem os passos:

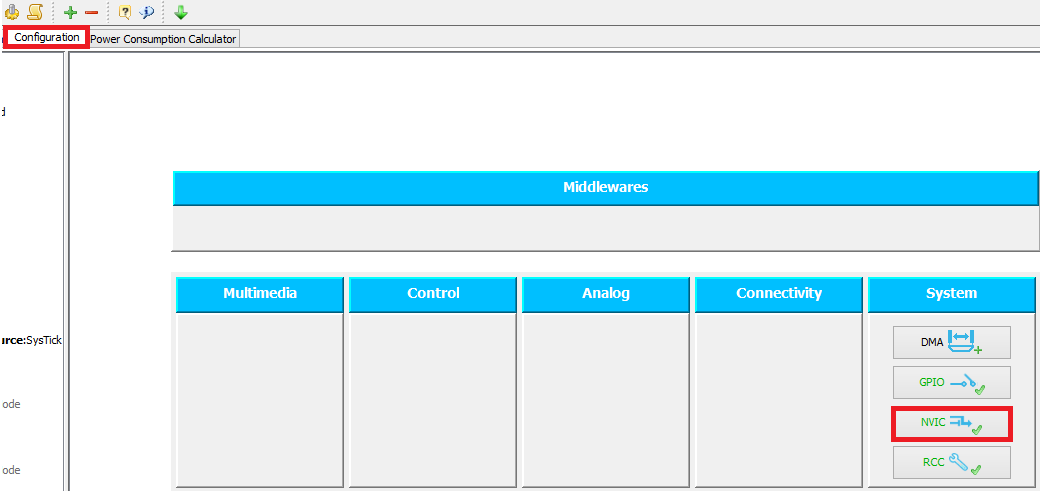
1. Criar um projeto no programa STM32F4CubeMX (similar aos primeiros passos do capítulo de geração de PWM);
2. Escolher o pino e ativar a função GPIO\_EXIT7 (no caso foi escolhido o pino PA7);

Figura 74: Pino para selecionar a função GPIO\_EXIT7



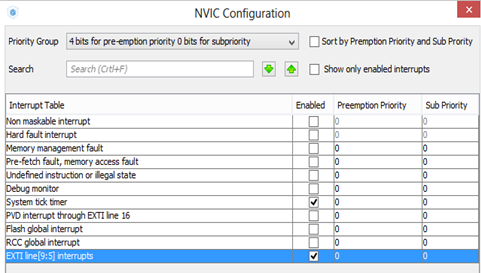
1. Na aba “Configuration”, clicar em NVIC, o que deve abrir uma janela de título “NViC CONFIGURATION”;

Figura 75: Botão para configurações de NVIC



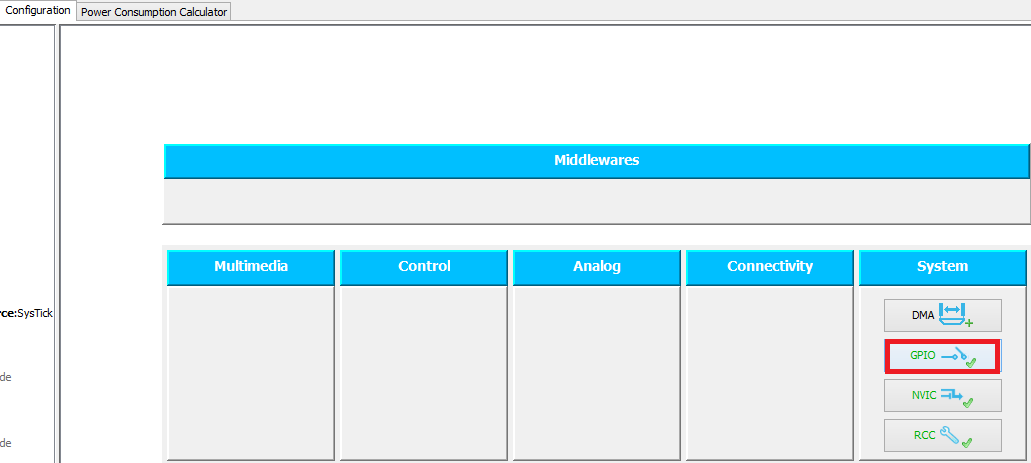
1. Na janela “NVIC CONFIGURATION”, selecionar a opção “EXTI lines[9:5] interrupts” e clicar em OK;

Figura 76: Configurações de NVIC



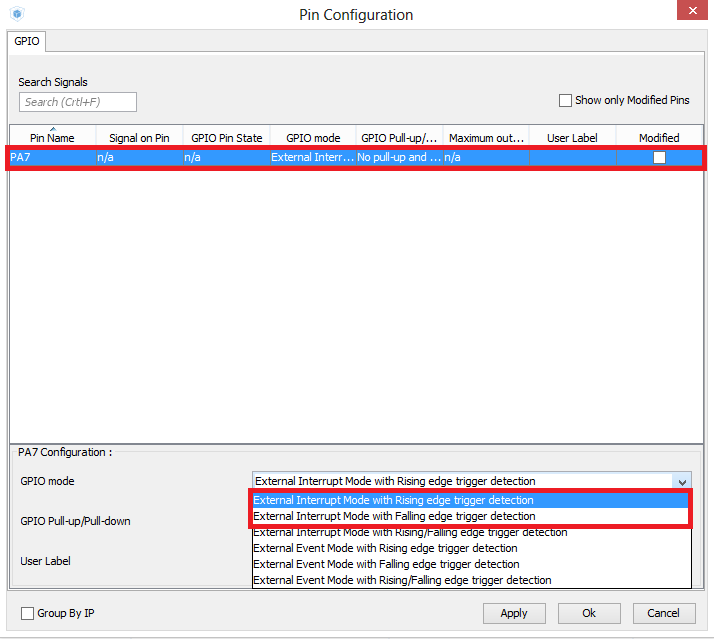
1. Voltando na aba “Configuration”, clicar em GPIO, que deve abrir uma janela de título “Pin Configuration”;

Figura 77: Botão para configurações de GPIO



1. Na janela “Pin Configuration”, clicar em cima do pino selecionado no passo 2 na listagem que aparece e selecionar o modo em que se deseja operá-lo (detector de borda de subida ou descida) em GPIO mode;

Figura 78: Configuração do pino selecionado



1. Gerar o código.
2. Após criar um projeto no Eclipse e copiar o código gerado, é possível editar a função EXTI9\_5\_IRQHandler no arquivo stm32f4xx\_it.c na pasta src.

# 7.Bibliografia

|  |  |
| --- | --- |
| [1] | Datasheet - STM32F405XX/STM32F407XX. |
| [2] | Reference Manual - STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced ARM®-based 32-bit MCUs. |
| [3] | IEEE 754 Converter, visitado em Dezembro de 2016. <https://www.h-schmidt.net/FloatConverter/IEEE754.html>.. |
| [4] | Website para baixar o software STM32F4CubeMX, visitado em Dezembro de 2016, <http://www.st.com/en/development-tools/stm32cubemx.html>. |
| [5] | ARM Cortex M4 Processor - Technical Reference Manual. |