- Núcleo do ARM Cortex – hardware...

(Registradores, ULA, GPIOs, pipeline, barramentos e interconexão com vários periféricos)

- Conceitos de programação – e.g. ***superloop***, funções, assembly...
- Conceito de Interrupção – programação do NVIC
- Programação de periféricos (GPIO, timers, Systick, PWM...)
- *Comunicação paralela - falamos de barramentos (interfaceamento paralelo)*
- *Comunicação serial*
- *DMA*,
- *ADC e DAC*
- *JTAG, depuração de software, gravação, teste in-circuit*

HOJE:

RTOS - Real-time Operating Systems (Sistemas Operacionais em Tempo Real)

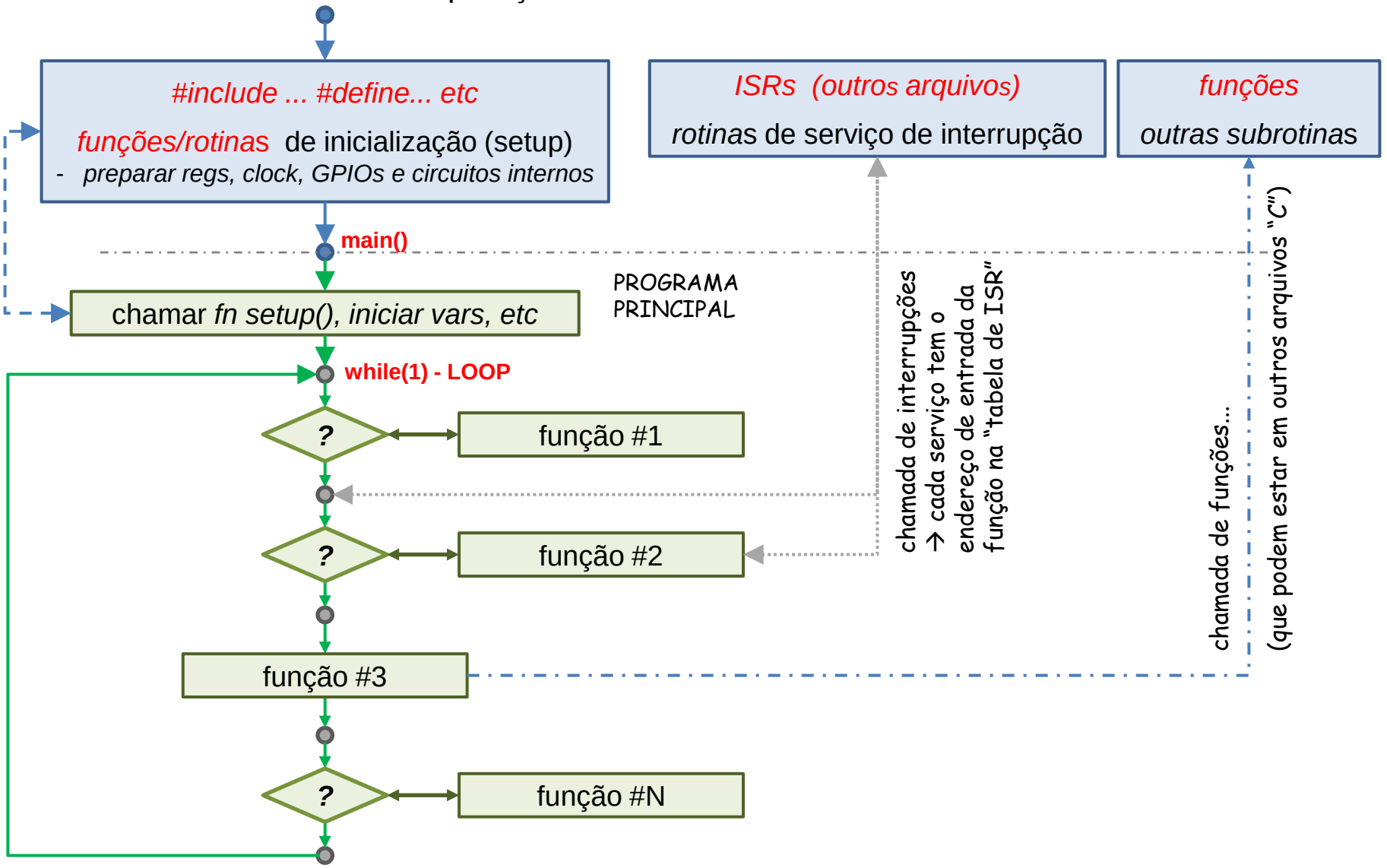
- Há basicamente dois paradigmas de programação embarcada (ou de μControladores)

- ***Superloop*** (*usamos até o momento*): baseado em loop infinito que mantém o μC executando as funções para as quais foi programado, caso contrário o programa seria executado até o final e o sistema para.

- ***RTOS*** (real-time operating sistem) que constitui em um núcleo de programa (kernel) que controla uma agenda de execução de tarefas para a CPU.

- Superloop é normalmente usado em sistemas com poucas tarefas, geralmente em aplicações mais dedicadas. Quando o μC tem que controlar várias tarefas (*ex: varrer teclado, mostrar display, controlar motores, fazer conversões de sinais (ADC) em vários sensores, comunicar serialmente, etc*.) fica difícil controlar a execução de todas as funções.
- Dentro de um superloop, fica especialmente difícil depurar bugs (*debugging*) que possam surgir da interferência de uma função ou tarefa em outra.

➢ Reformulando o modo de operação da CPU... conceito de SUPERLOOP...

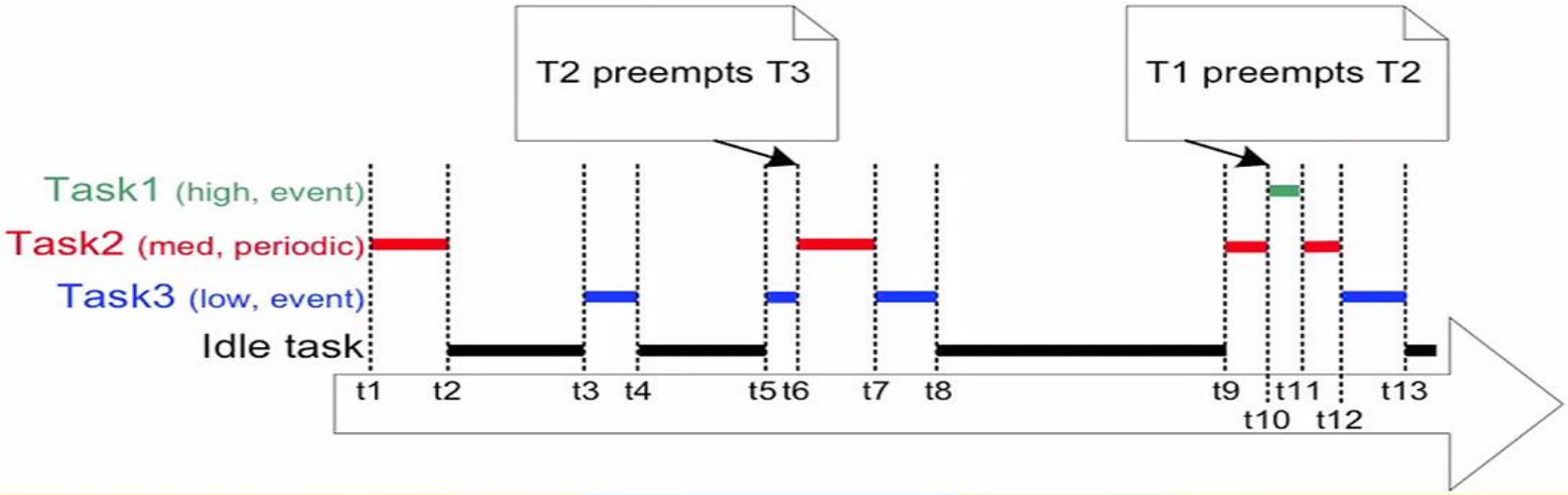


S.O. é uma camada de software opcional que provê serviços de baixo nível para um aplicativo (e.g. gerencia arquivos, acesso ao disco, interface de teclado, display, etc.)

RTOS é um kernel que organiza a execução de tarefas (tasks) por prioridades, controla uso de recursos do sistema, e gerencia o uso racional do tempo de um μP/ μC.

The RT In RTOS

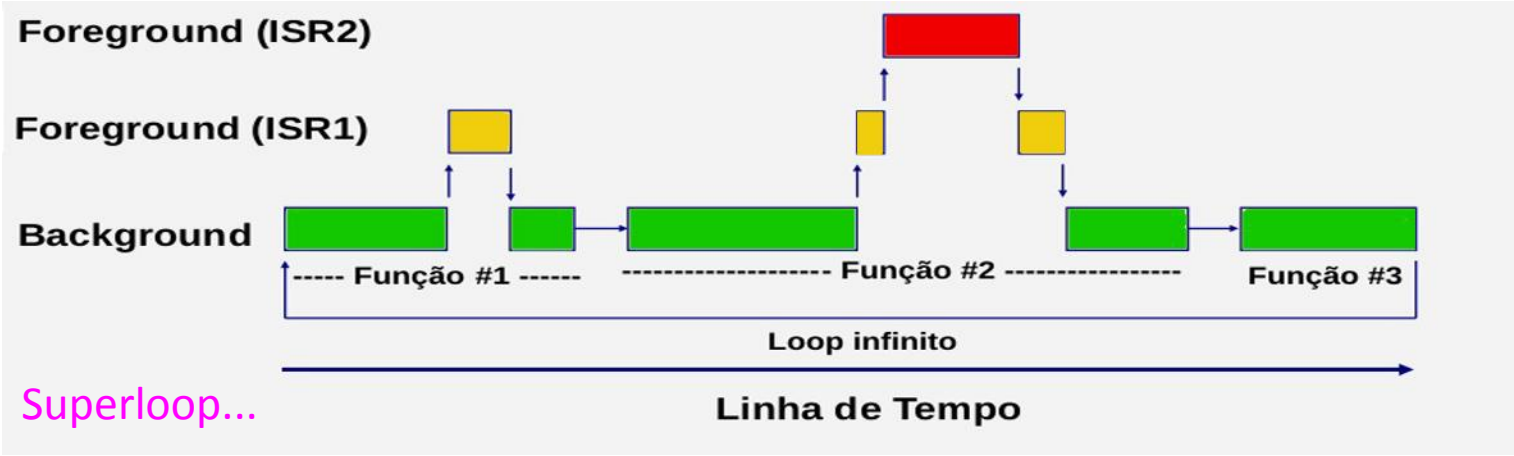
- Deterministic
  - Hard real time – "it absolutely must"
  - Soft real time – "it should"



➢ *Abordagens na construções de software:*

Superloop versus RTOS

```
int main(void)
{
  initHardware();

  while(1)
  {
    fn1();
    fn2();
    fn3();
  }
}

void ISR1(void){
...
}
void ISR2(void){
...
}
```

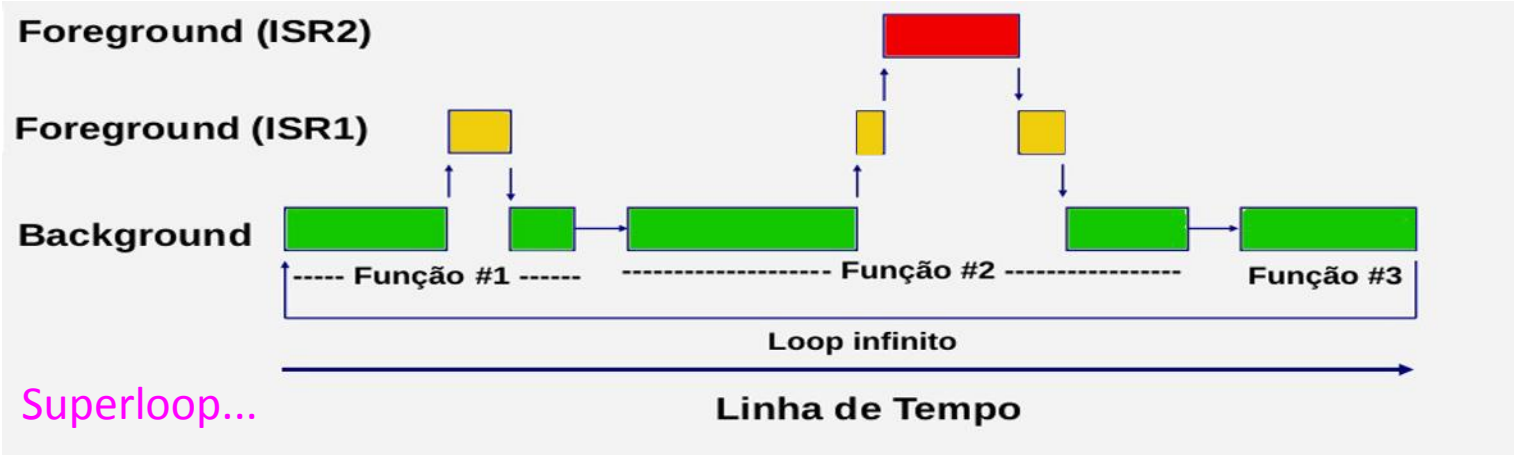


Superloop...

➢ *Abordagens na construções de software:*

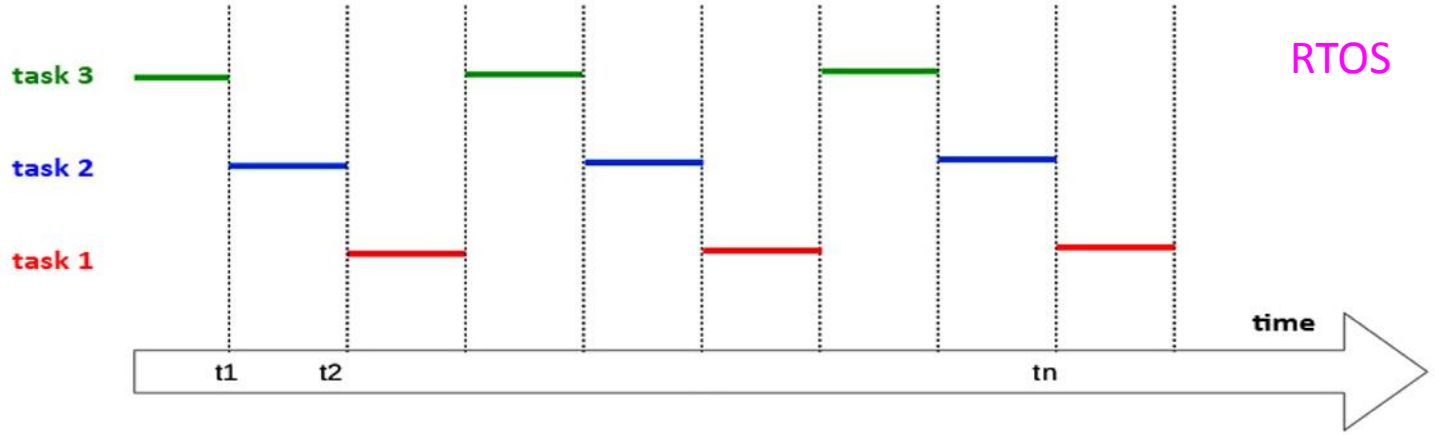
Superloop versus RTOS

```
int main(void)
{
  initHardware();

  while(1)
  {
    fn1();
    fn2();
    fn3();
  }
}

void ISR1(void){
...
}
void ISR2(void){
...
}
```



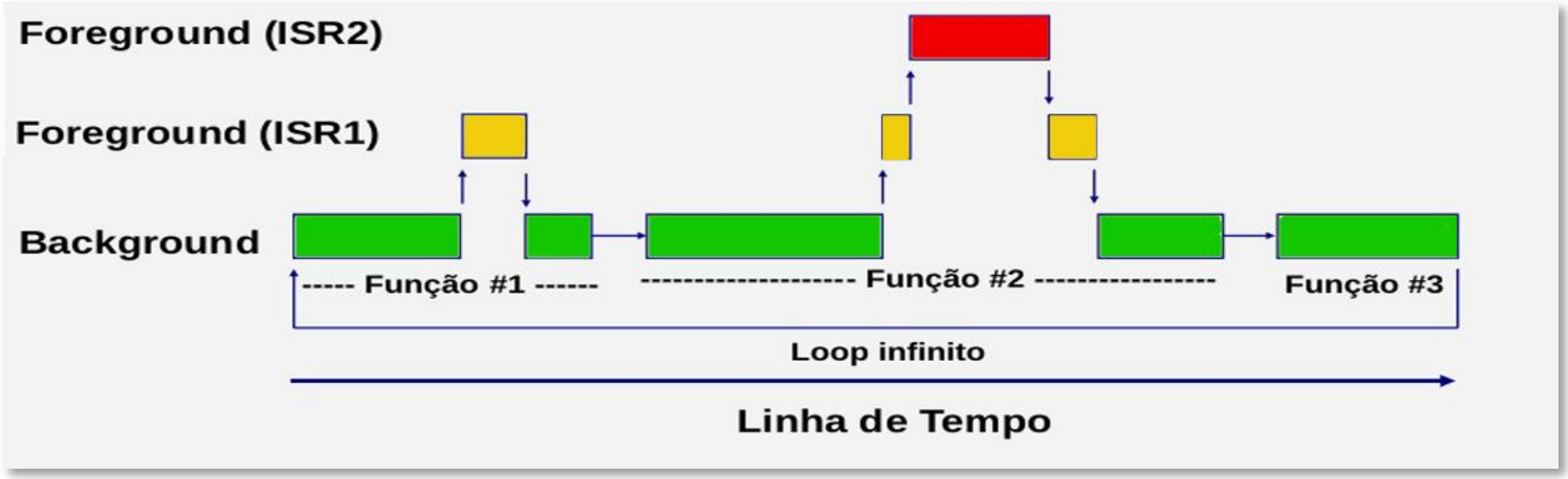
Superloop...

Opção 2:
Utilizar um
Sistema
Operacional
(tempo real)



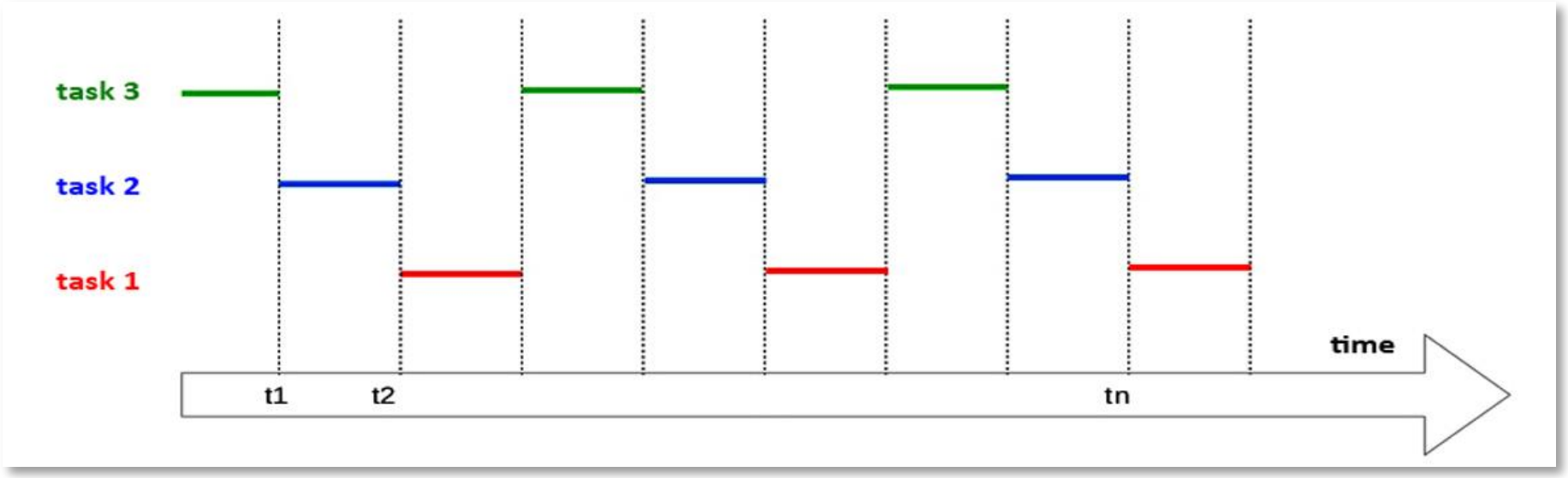
RTOS

1) Criação de um ***superloop***...



Não temos certeza sobre o tempo que a função #2 estará terminada!
Na medida em que a quantidade de funções do embarcado aumenta, fica difícil gerenciar o ***superloop***;

2.a) Usar um ***real-time operating system (RTOS) com tarefas uniformemente distribuídas***...

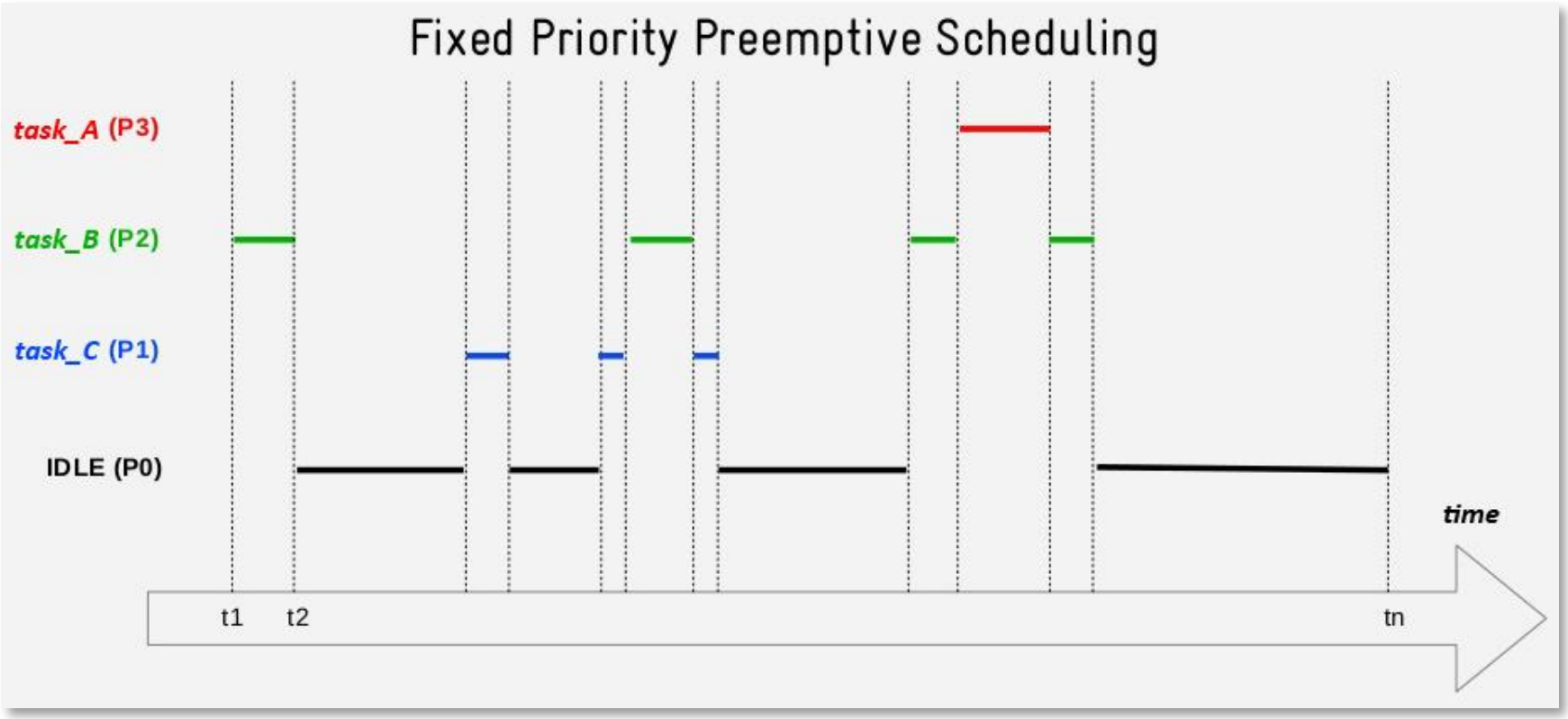


- Divide-se o tempo da CPU (durante cada tempo a CPU processa exclusivamente aquela tarefa);
- Alocam-se tarefas que são encaixadas dentro dessa disponibilidade da CPU;
- Tarefas podem ter níveis de prioridades – assim, certas tarefas podem ser atendidas primeiro;
- Ao findar o tempo reservado para uma tarefa ***Y***, a CPU pode deixar essa tarefa ***Y*** inacabada...

conceito: SCHEDULER (***escalonador***)

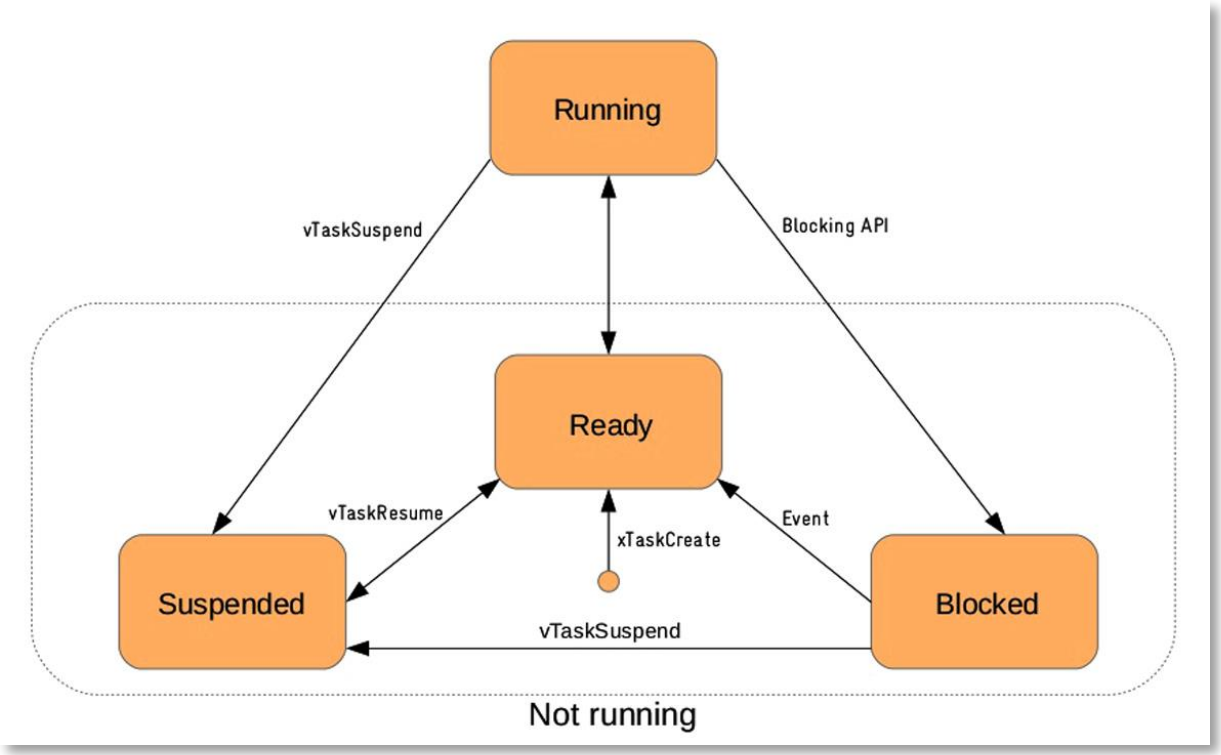
- Aplicativo que supervisiona/controla o uso dos recursos de CPU distribuindo o tempo da CPU entre várias tarefas;
- Cada tarefa tem associada uma prioridade – ajustada pelo programador;
- Cada tarefa pode estar em um estado: running, blocked, suspended, ready;
- Apenas uma tarefa está no estado RUNNING em um determinado tempo;
- (e.g. no FreeRTOS) a tarefa de menor prioridade é a IDLE_task;
- Um escalonador preemptivo sempre seleciona uma tarefa READY e com o MAIOR nível de prioridade para ser executada;

2.b) Usar um ***real-time operating system (RTOS) com tarefas priorizadas***...



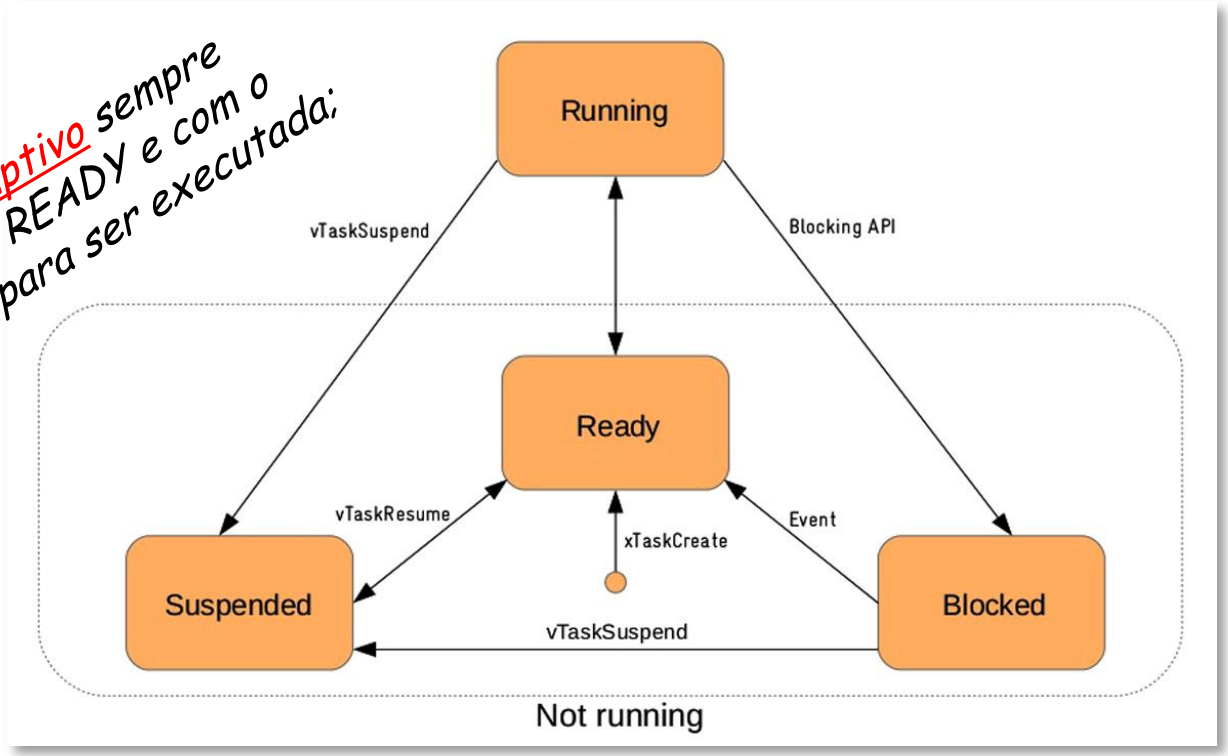
- Exemplo de tarefas com prioridade...
- A tarefa A (P3 - possivelmente uma interrupção) tem prioridade sobre as demais...
- A tarefa B (P2 – pode ser uma *INT* por software) tem prioridade sobre a tarefa C (P1)
- Deve existir uma tarefa IDLE (P0) para onde a CPU sempre retorna quando não tem trabalho!

- Diagrama de estados das tarefas em um RTOS ...



- Uma tarefa está sempre em um desses estados...
- Running – a tarefa está em execução;
- Ready – a tarefa foi terminada
- Blocked – a tarefa X está bloqueada para execução
- Suspended – a task não foi terminada no tempo anterior, está aguardando a CPU atender outras tasks.

- Diagrama de estados das tarefas em um RTOS ...



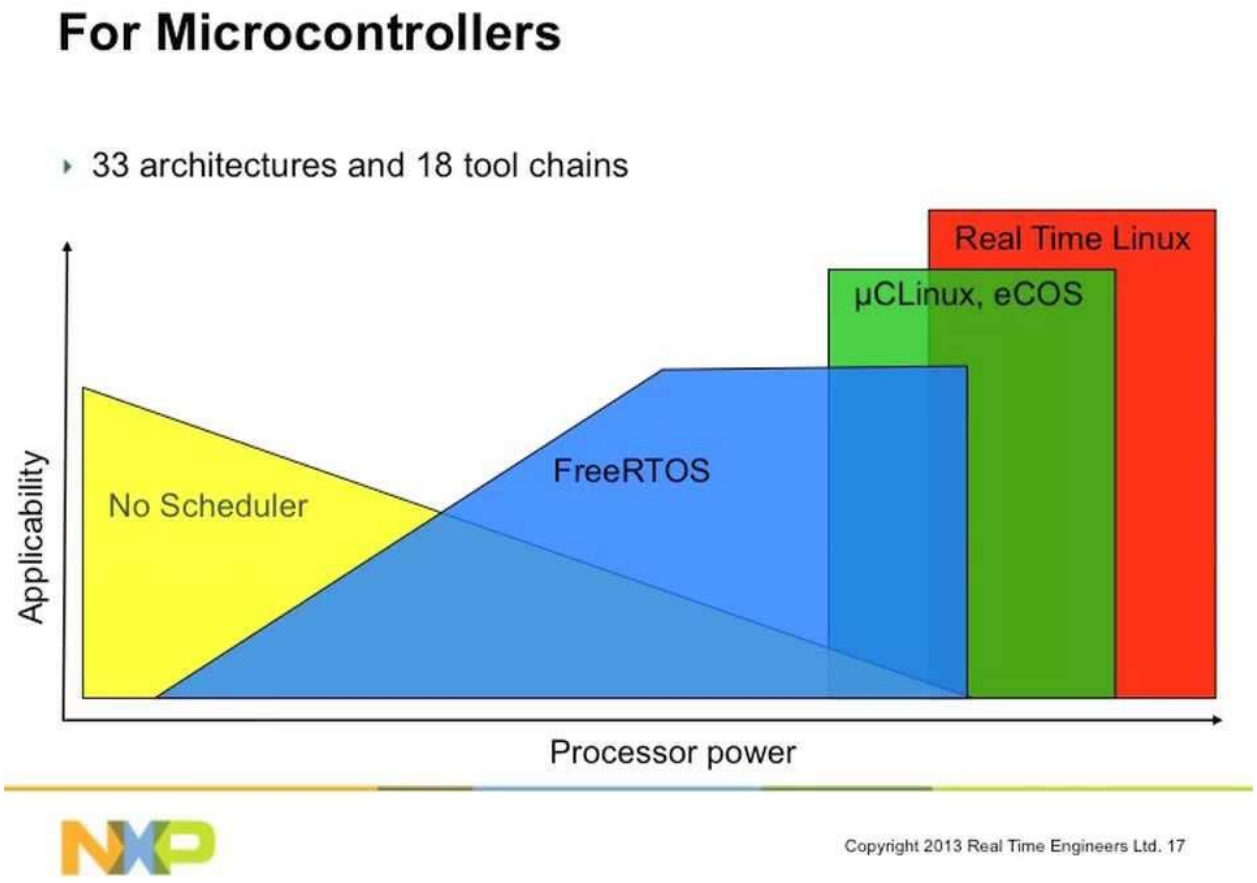
- *Um escalonador preemptivo sempre seleciona uma tarefa READY e com o MAIOR prioridade para ser executada;*

- Uma tarefa está sempre em um desses estados...
- Running – a tarefa está em execução;
- Ready – a tarefa foi terminada
- Blocked – a tarefa X está bloqueada para execução
- Suspended – a task não foi terminada no tempo anterior, está aguardando a CPU atender outras tasks.

S.O. é uma camada de software opcional que provê serviços de baixo nível para um programa (aplicativo). Ex: gerencia arquivos, acesso ao disco, interface de teclado, display, etc. Organiza a execução de múltiplos programas, ou controla o uso racional do tempo de um processador.

RTOS:

- FreeRTOS,
- MQX,
- Keil (RTX)
- QNX
- INTEGRITY
- VxWorks
- QNX Neutrino
- eCOS
- µC/OS-II
- Windows CE
- TI-RTOS



- Como decidir entre usar ou não um RTOS?
  depende da aplicação (complexidade/tamanho) e dos recursos exigidos do μC/ μP

- Como dividir tarefas no software? Devo usar mais/menos tarefas?
  Mais tarefas pode representar: aplicativo mais modular, código mais limpo e melhor manutenção;
  contudo: exige mais RAM, gasta mais tempo da CPU para troca de contexto (tarefa).

- Pense o projeto na forma de tarefas. Você deve pensar que os requisitos do projeto serão distribuídos entre tarefas;

- Analise as prioridades das tarefas. Funções com prioridades diferentes devem ser executadas por tarefas diferentes;

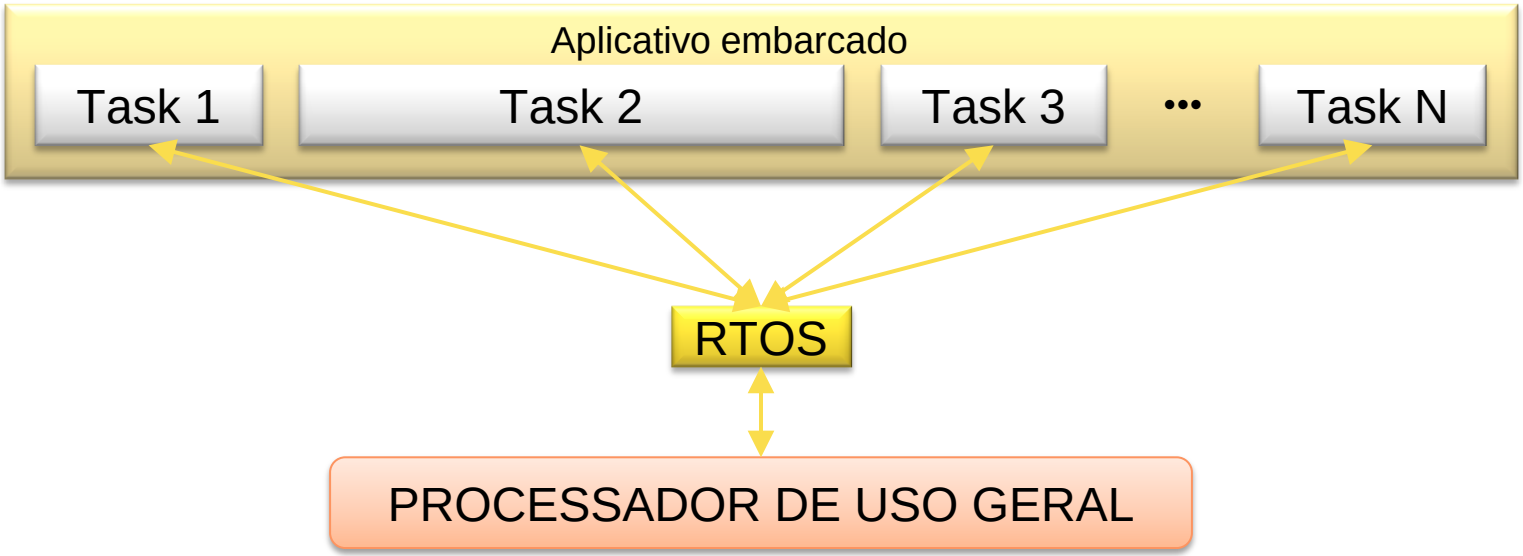
- Atividades que podem ser paralelizadas, ou funções periódicas, devem ser implementadas em tarefas diferentes;

- Uma tarefa para cada dispositivo de hardware compartilhado por várias funções no sistema;

- Provavelmente, cada interrupção do sistema terá uma (ou mais) trarefa;

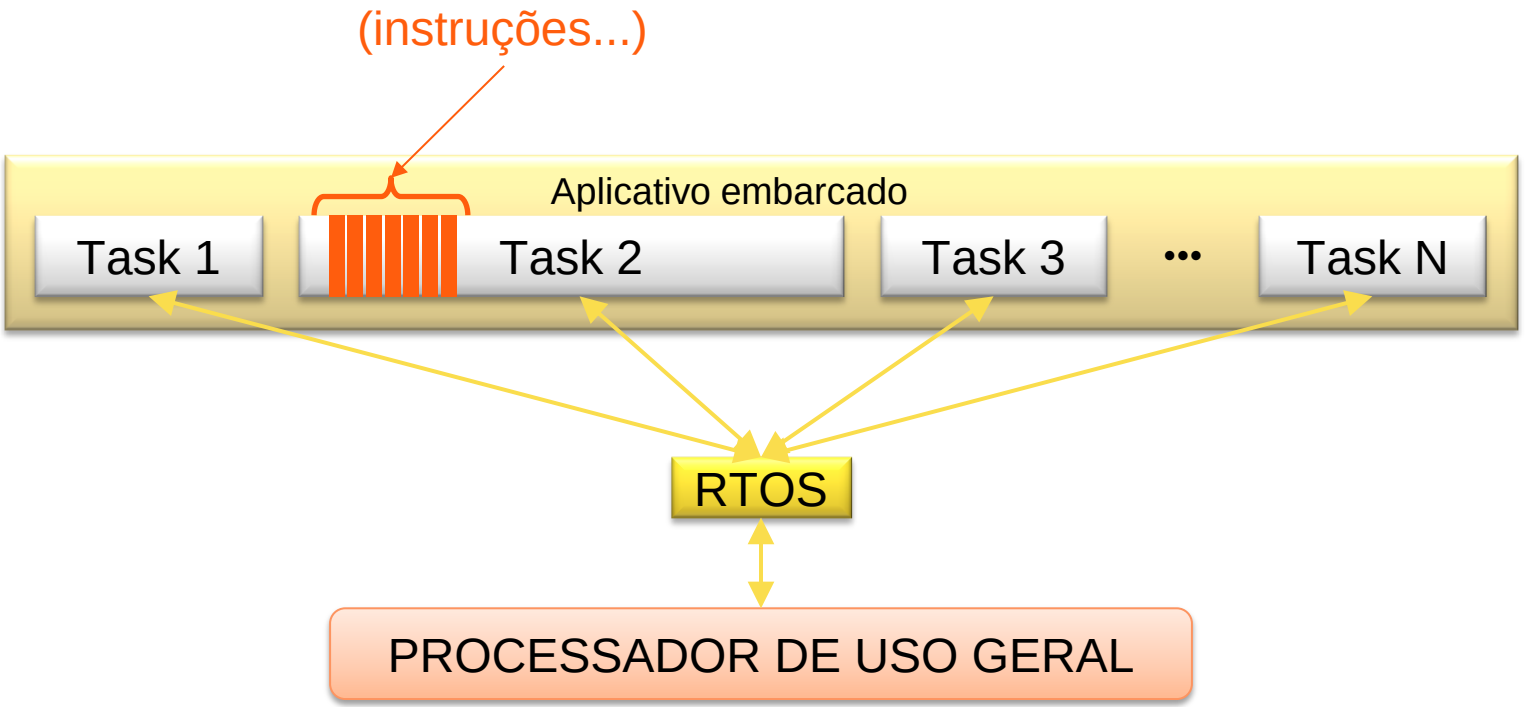
→ RTOS remete a uma nova dinâmica (forma) de pensar o projeto com μP de uso geral.

→ Trata-se de uma camada (aplicativo) para gerenciar o compartilhamento temporal do processador (que por sua vez já compartilha temporalmente os recursos de hardware – ULA, registros, etc – entre as várias instruções que este processador genérico executa).



→ RTOS remete a uma nova dinâmica (forma) de pensar o projeto com μP de uso geral.

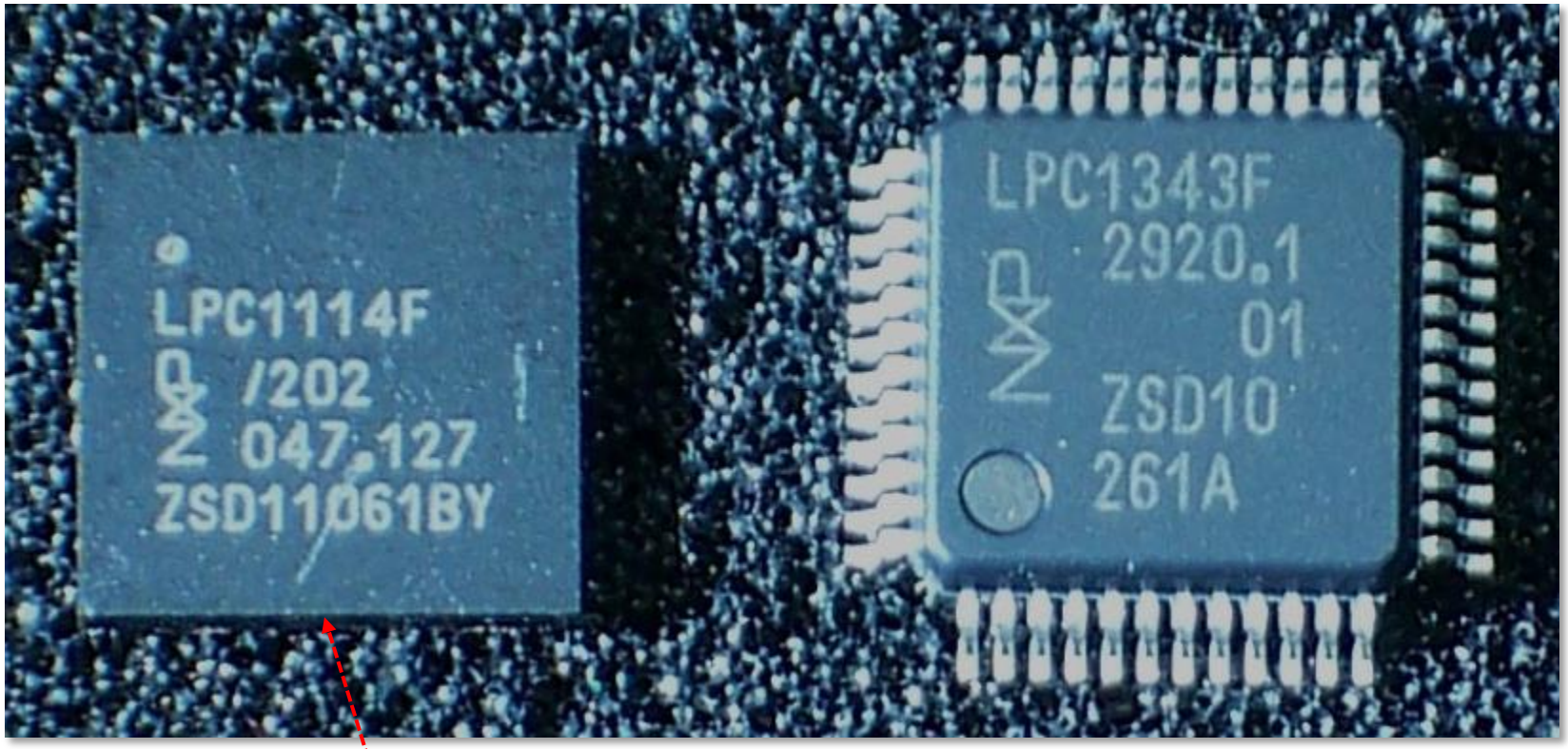
→ Trata-se de uma camada (aplicativo) para gerenciar o compartilhamento temporal do processador (que por sua vez já compartilha temporalmente os recursos de hardware – ULA, registros, etc – entre as várias instruções que este processador genérico executa).



RTOS (detalhes) e modo de programação em RTOS foge ao escopo do curso de Sistemas Microprocessados.

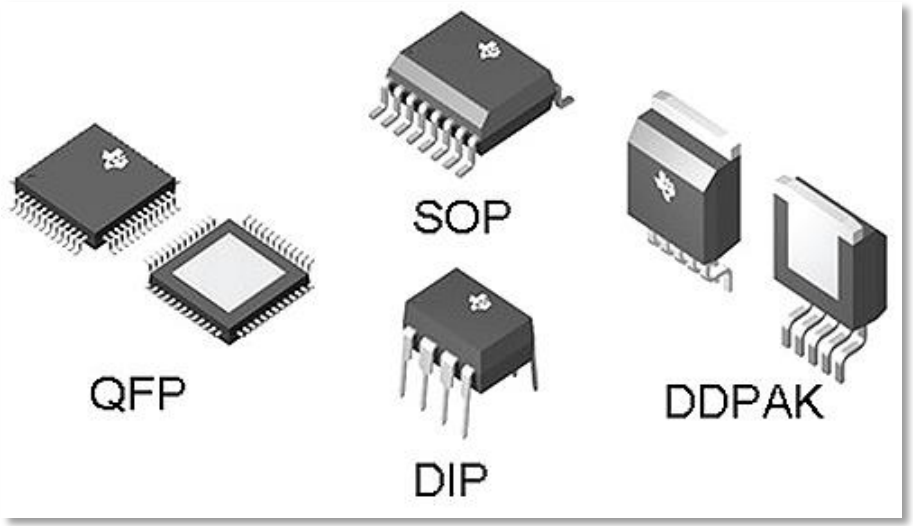
Este assunto deve ser abordado em:
Programação de Software Embarcado;
e
Aplicação de Microcontroladores.

- Dois CIs da NXP (ambos ARM CORTEX)

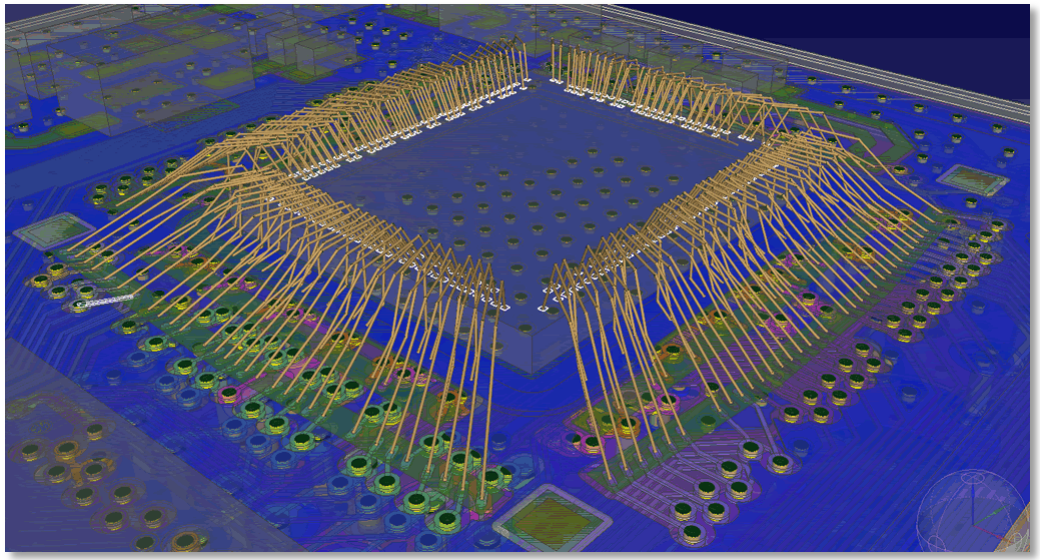


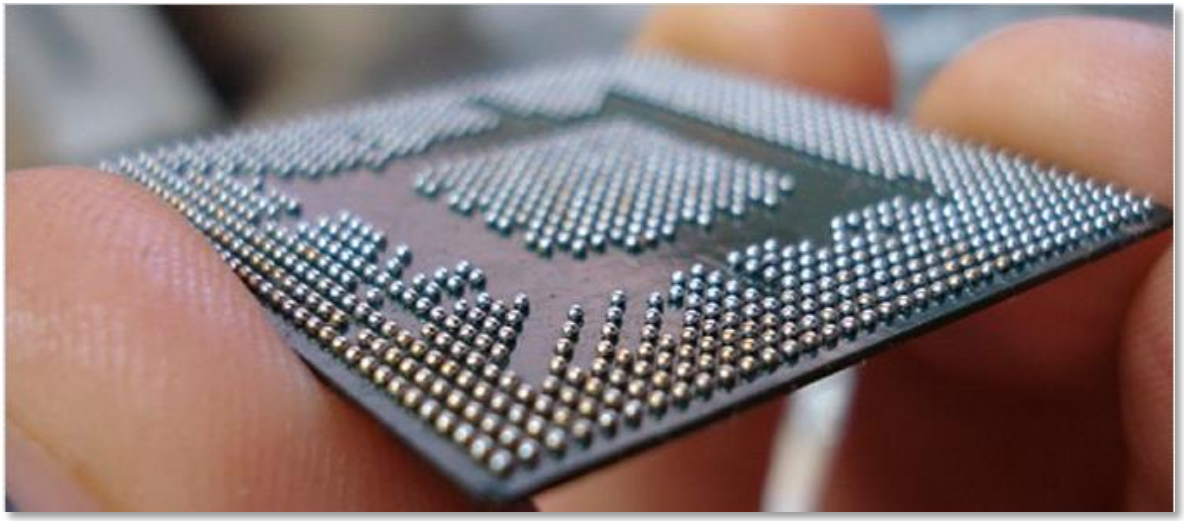
(este chip é BGA – Ball Gate Array)

- Tecnologia BGA cria circuitos (PCBs) mais compactos, mas dificulta encontrar erros!

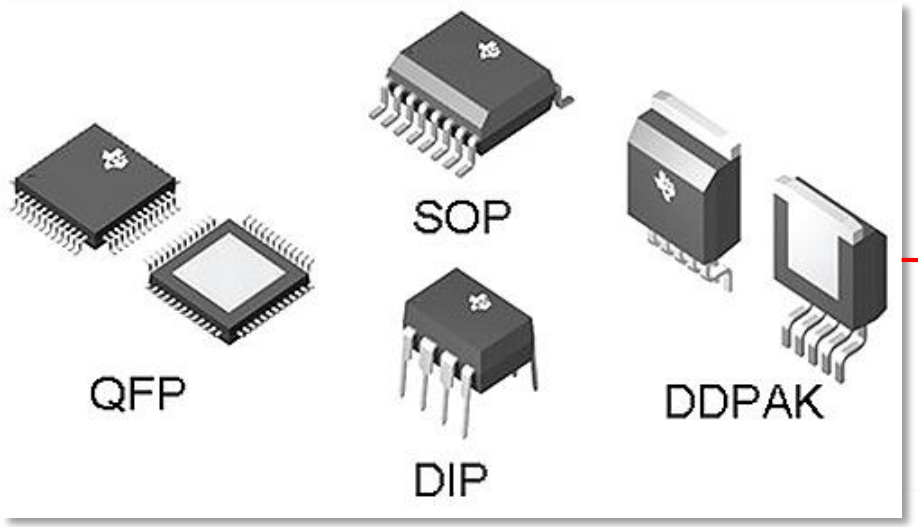


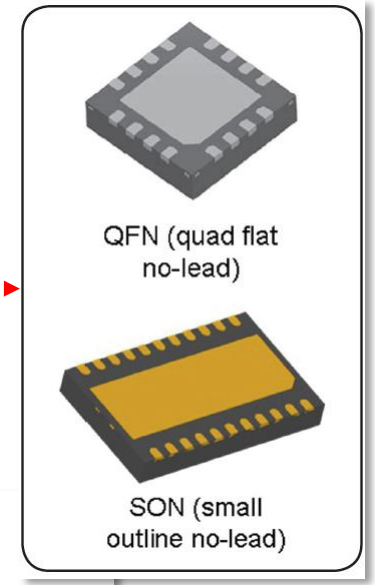
Os circuitos integrados cada vez mais possuem conexões com o mundo externo, isso fez surgir os circuitos BGA...

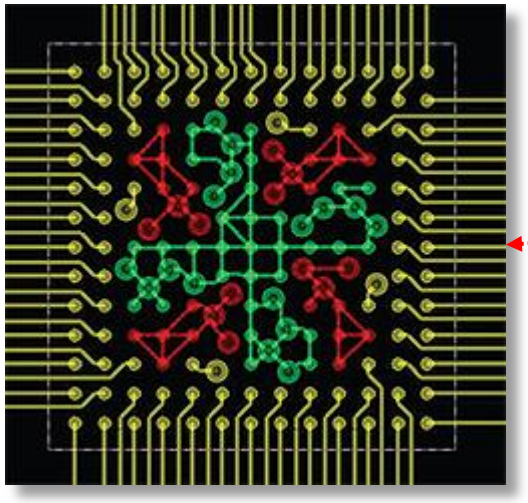


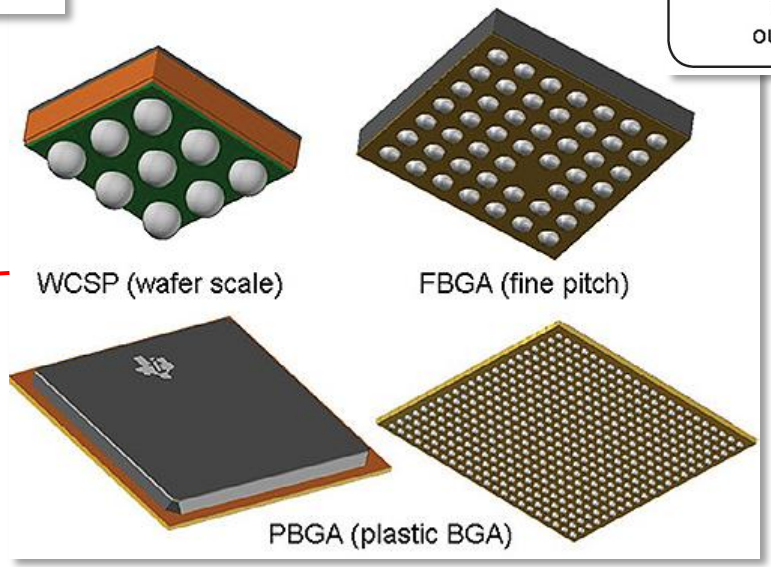


- Tecnologia BGA cria circuitos (PCBs) mais compactos, mas dificulta encontrar erros!





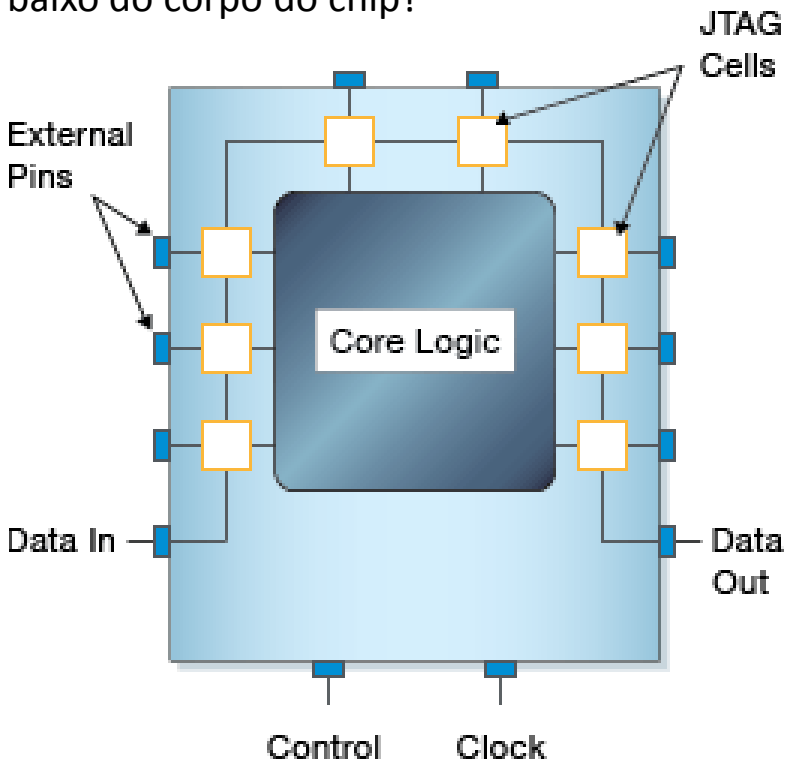




PCBs com trilhas escondidas...

TESTAR placas de circuito impresso (PCB) tem sido uma preocupação desde a década de 1980!
Como testar placas cujas trilhas estão escondidas em baixo do corpo do chip?

IEEE criou um grupo chamado JTAG
Joint
Test
Action
Group



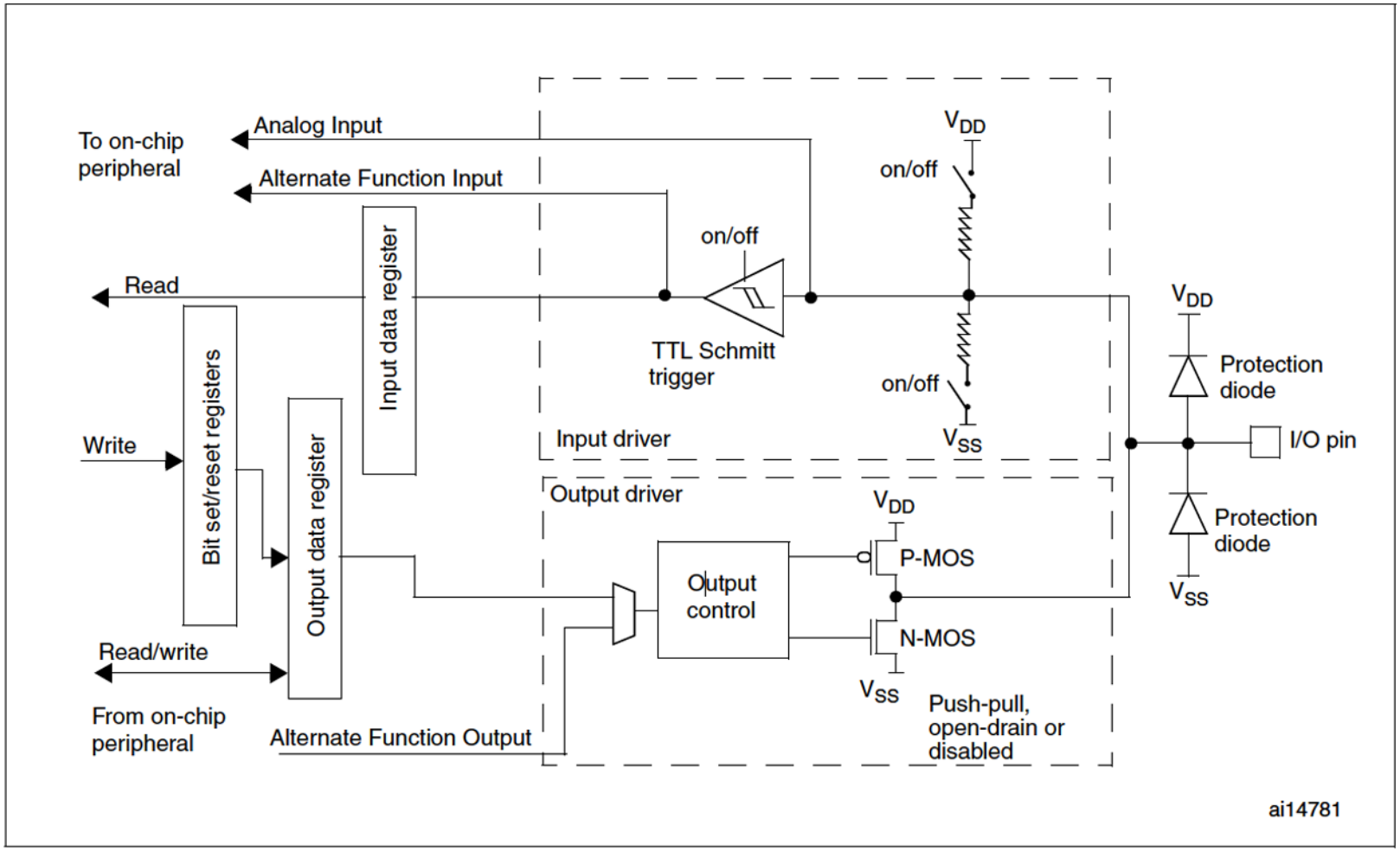
A IDEIA:
Criar uma ***interface*** em cada pino dos CIs...
(Boundary Scan)

Esta interface é a JTAG cell, controlada por uma FSM.

Joint Test Action Group (JTAG) foi um grupo criado em 1985 para desenvolver métodos para testar circuitos impressos depois de fabricados.
Em 1990, o IEEE(*Institute of Electrical and Electronics Engineers*) criou a norma IEEE 1.149,1-1990 intitulado "*Standard Test Access Port and Boundary-Scan Architecture*" para homologar os processos de testes. (*wikipedia*)

- (*recordando...*) Como μControladores têm mais funções que pinos de I/O os fabricantes atribuem mais de uma função a cada pino. A funcionalidade do pino deve ser programada.

Figure 13. Basic structure of a standard I/O port bit

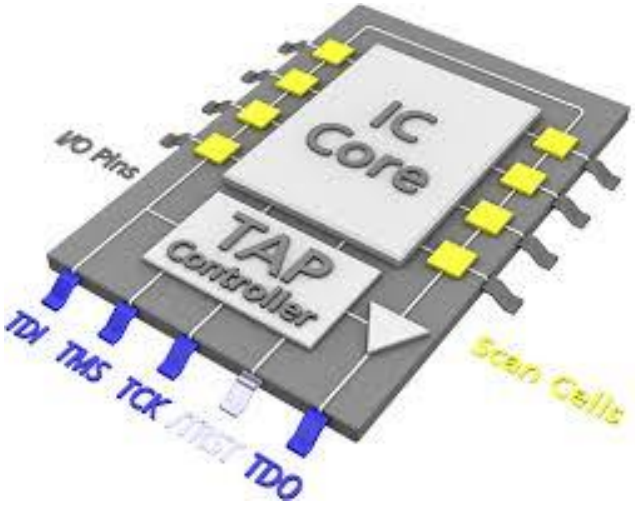


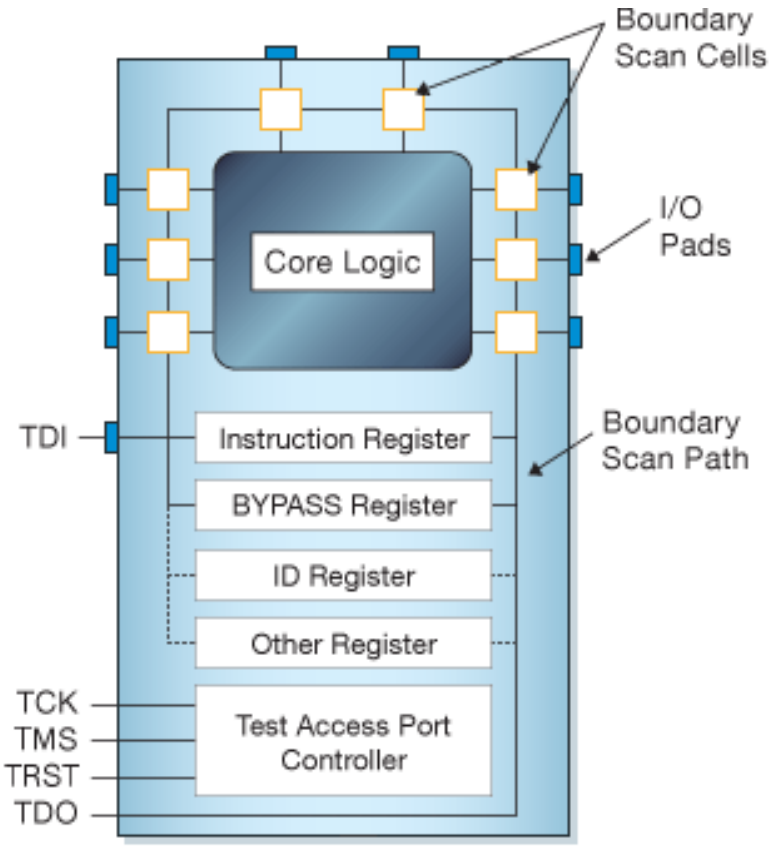
JTAG – utiliza um conjunto de fios para controlar uma FSM

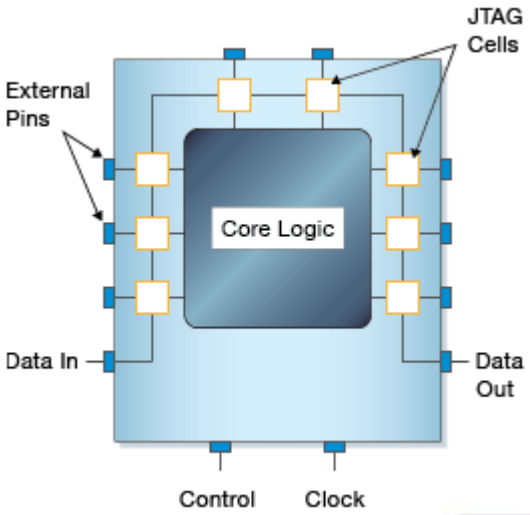
→ que conecta ou desconecta cada pino do CI do mundo exterior;

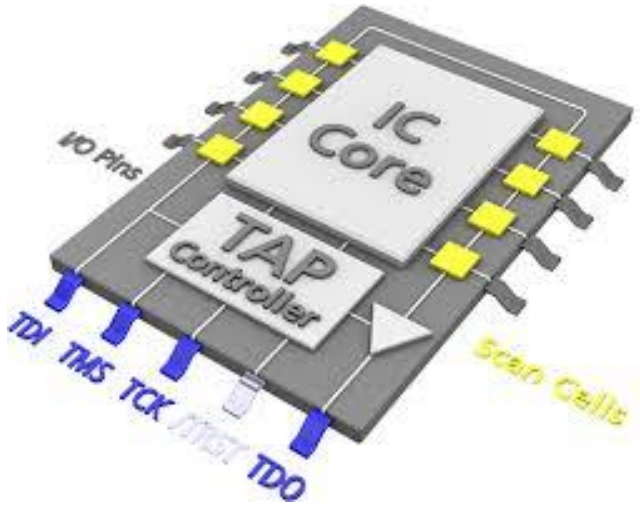
→ Pode então excitar internamente e externamente cada pino.

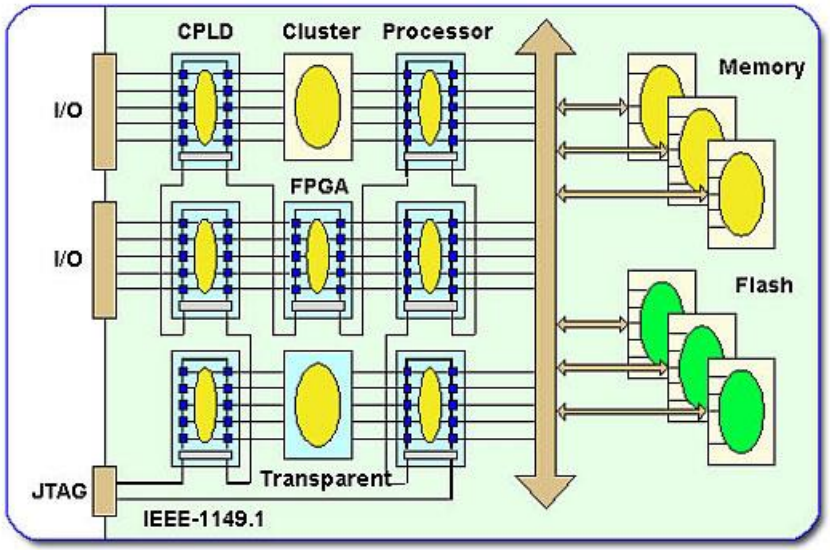
→ Pode ler a resposta (interna e externa) de qualquer pino desconectado do núcleo ou da PCB.





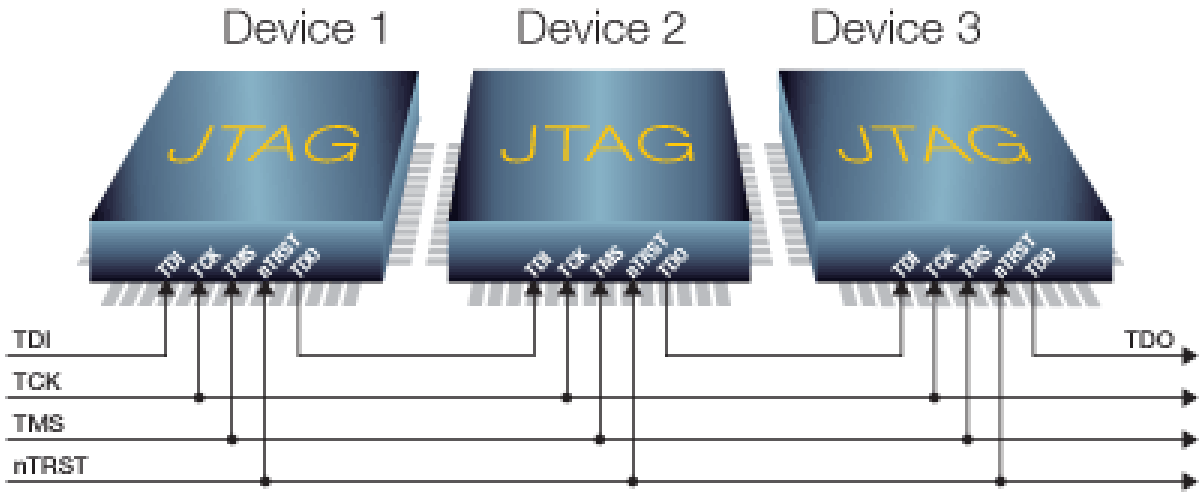






Os CIs que possuem interface JTAG são interligados em cascata, formando uma cadeia. Alguns CIs da placa não terão interface JTAG, mas de alguma forma estão conectados com algum pino JTAG.

JTAG – Os CIs interligados em cascata (CHAIN) são conectados a um conector JTAG único para a PCB. Isso forma uma rede que permite excitar/ler qualquer pino (externo) ou programar/excitar qualquer sinal (interno) do núcleo dos circuitos integrados da placa.



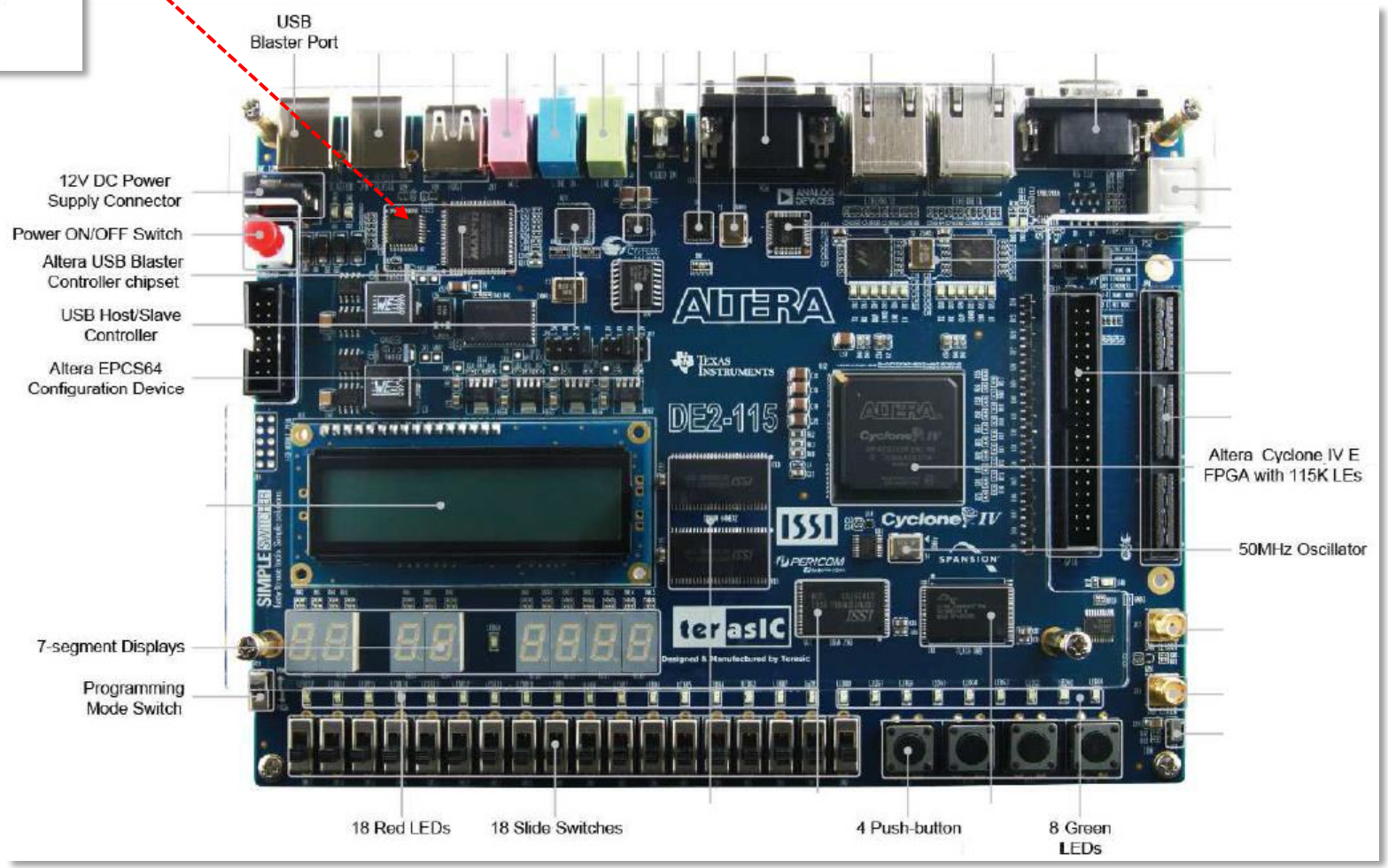
ver vídeo...

- Já usamos JTAG para gravar FPGA !!! (para isso usamos um adaptador USB → JTAG

✧ JTAG – Joint Test Action Group

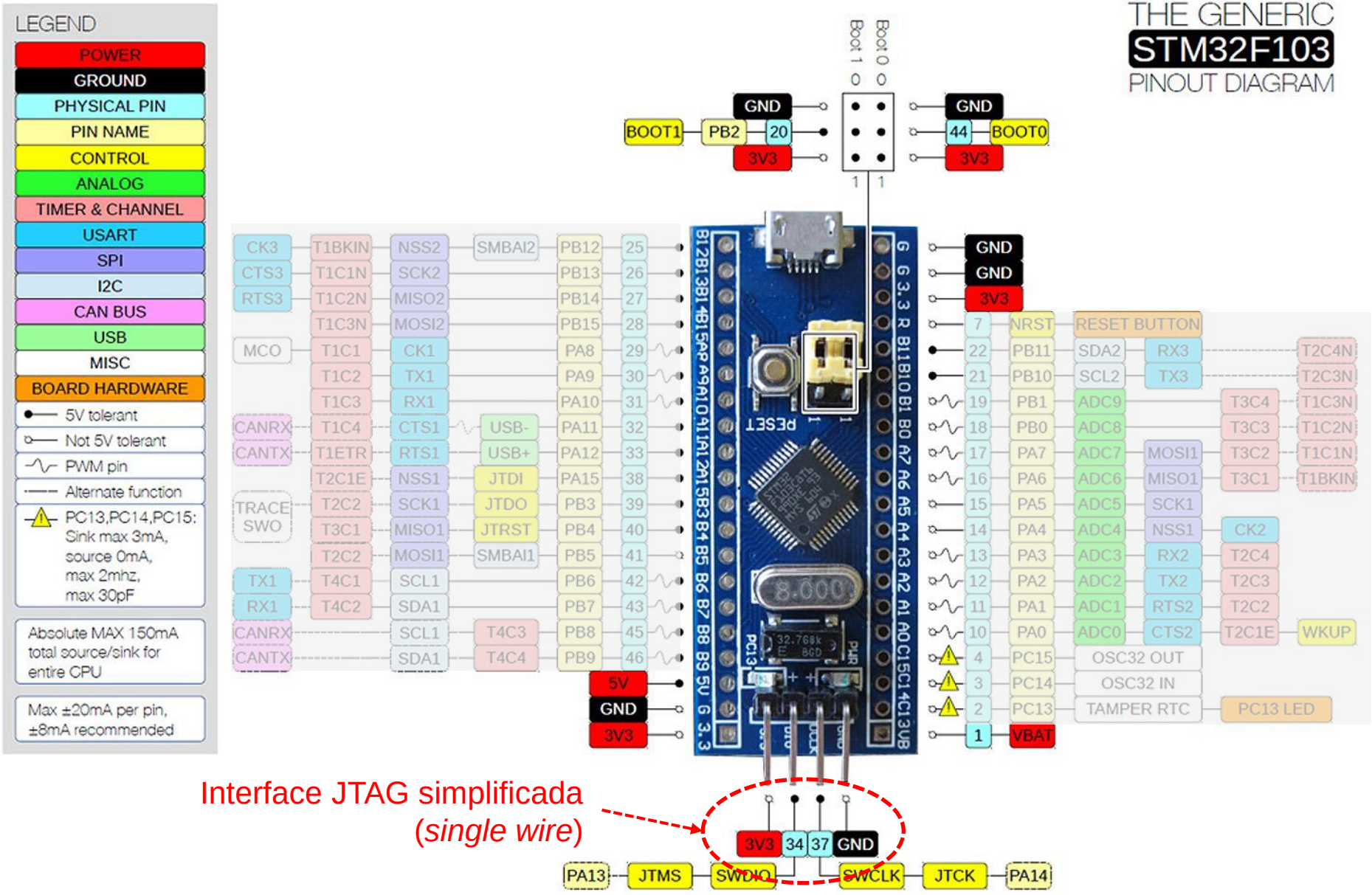
João Ranhel

- Nas placas FPGA DE2-115 o USB-Blaster já está embutido na placa



✧ JTAG – Joint Test Action Group

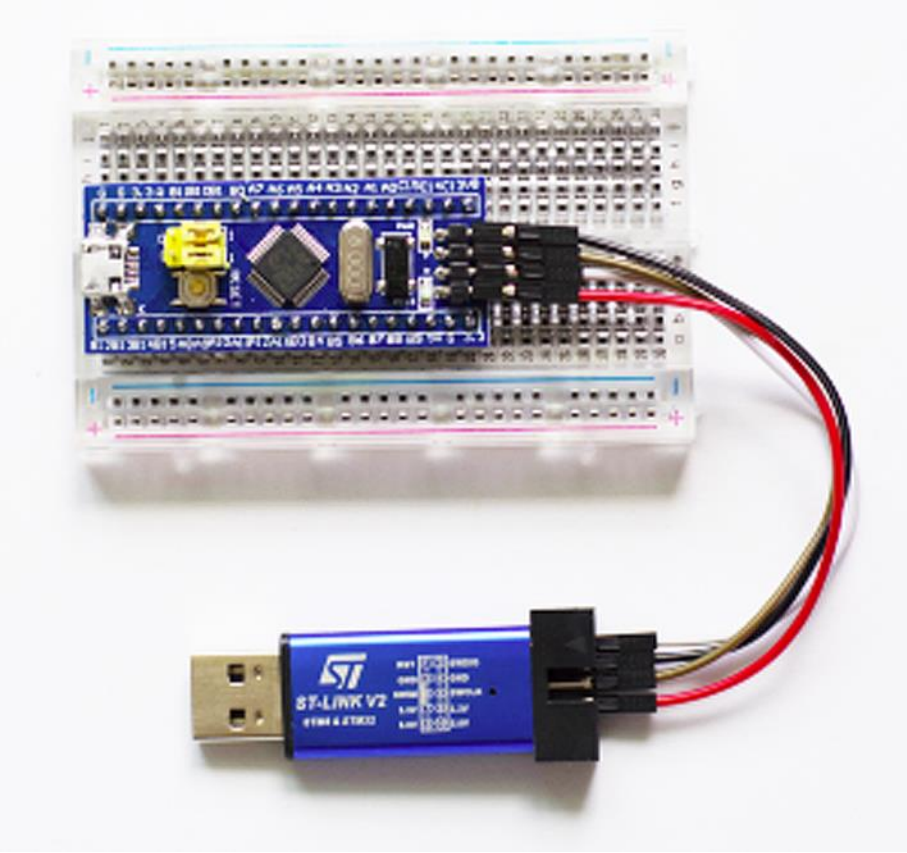
João Ranhel



✧ JTAG – Joint Test Action Group

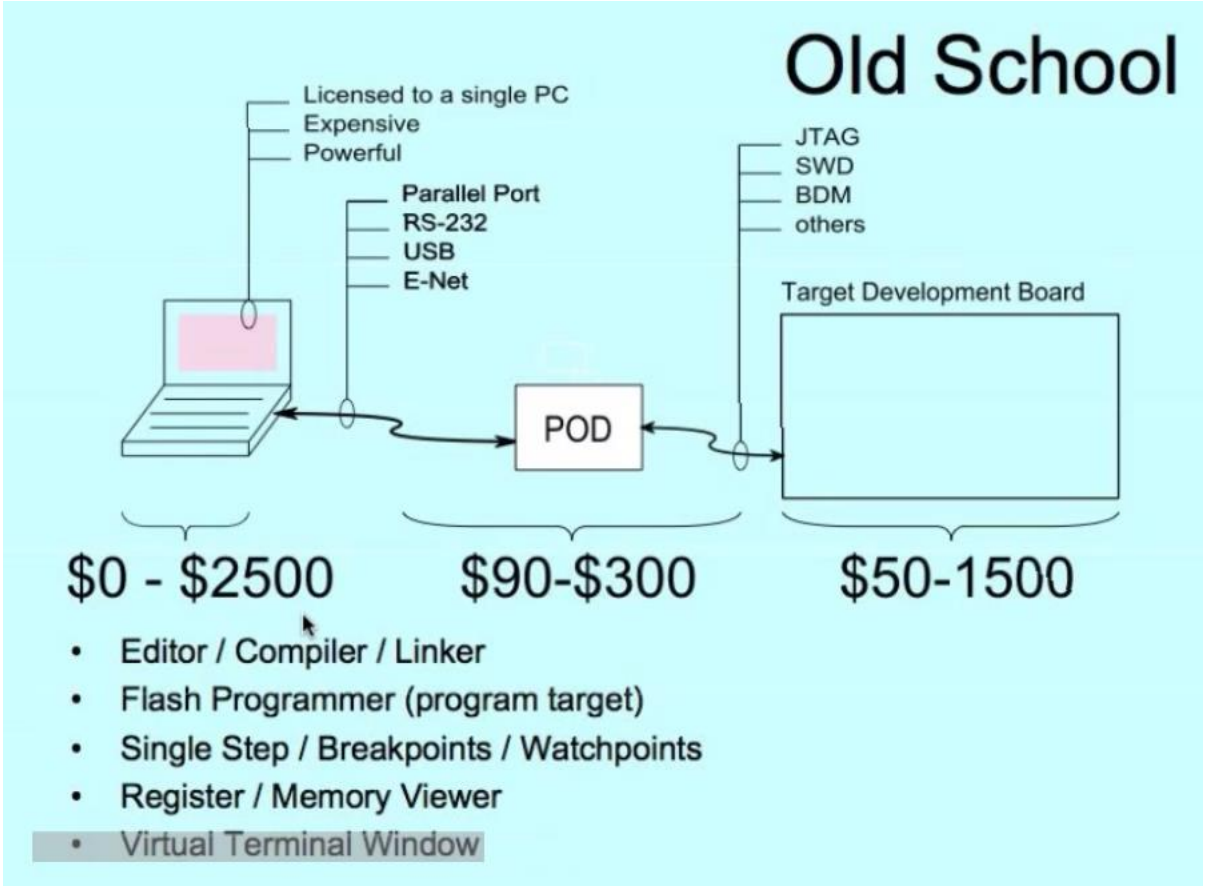
João Ranhel

- Nas placas STM32F103C8 o JTAG é ativado pela interface SW.



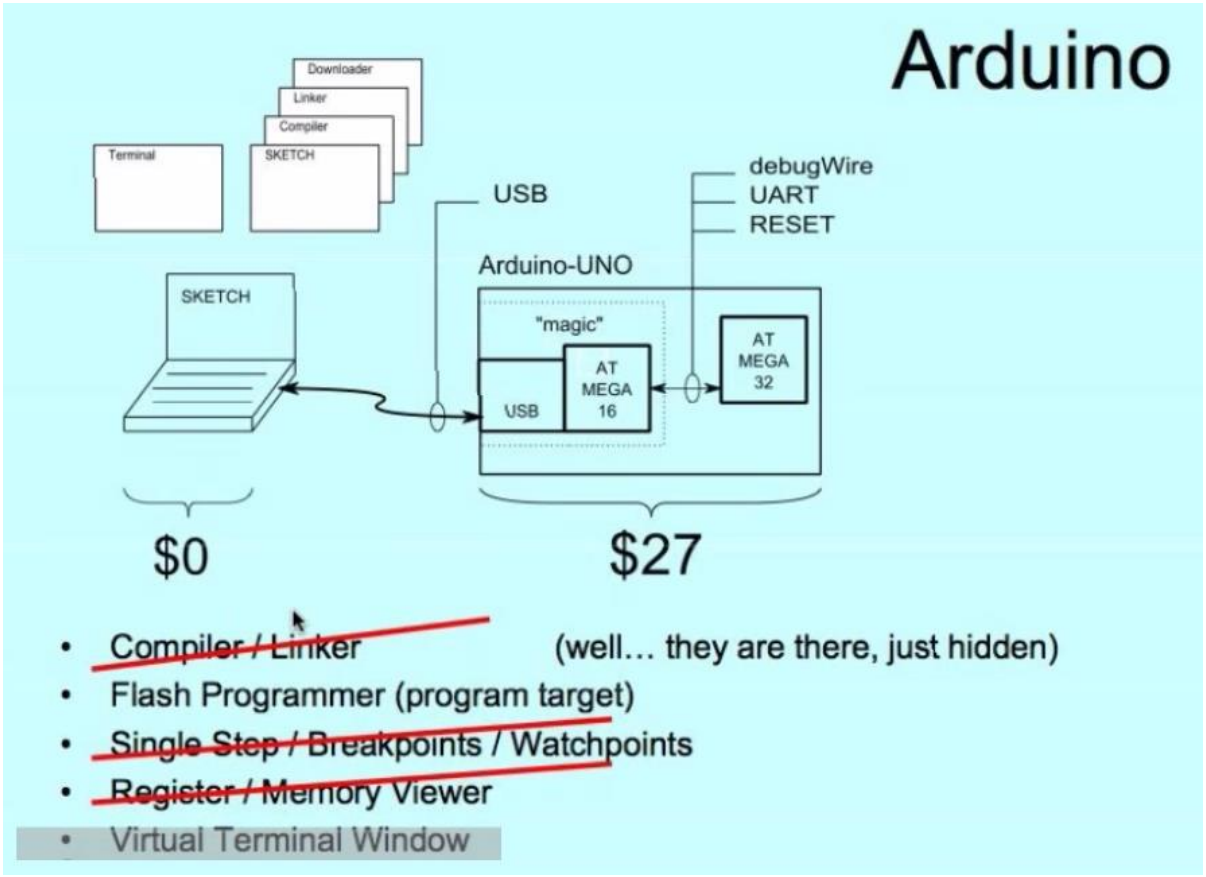
- O ST-LINK V2 é um adaptador que transforma USB em JTAG (SW).

- As ferramentas de desenvolvimento ***evoluíram*** com o passar dos tempos (esta era a "old-school"



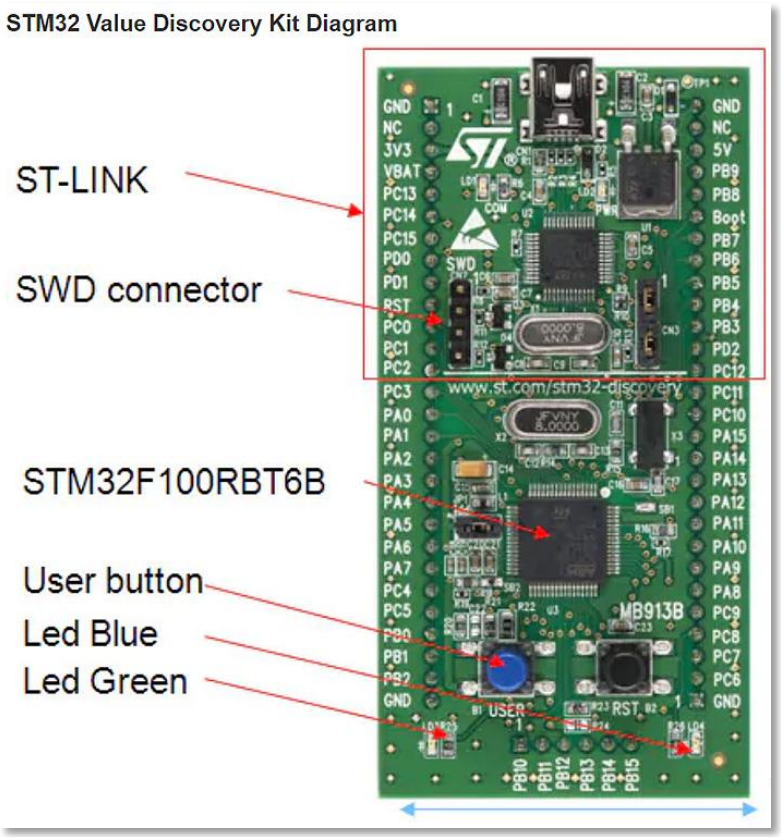
Com a tecnologia antiga, tínhamos que comprar um POD para interfacear uma placa target... E às vezes uma placa para emular um processador (ambos caros)

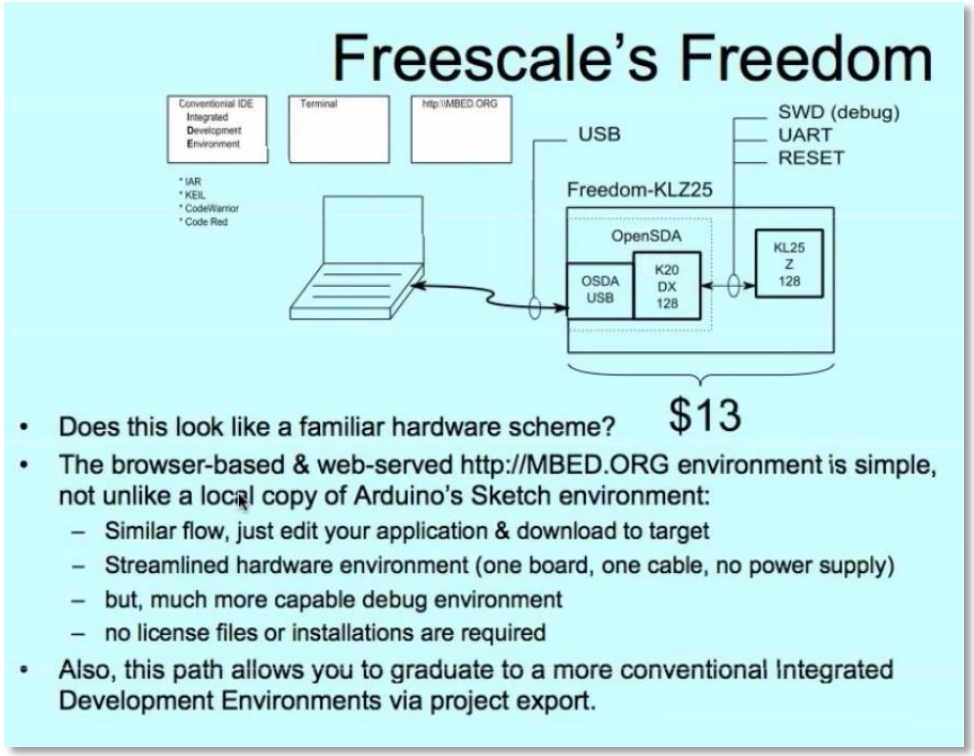
- As ferramentas de desenvolvimento ***evoluíram*** com o passar dos tempos (esta era a "old-school"



Arduino quebrou barreiras – muita gente que hoje tem acesso à embarcados começou por ter uma placa portátil que não requeria POD e pode ser programada/testada direta do PC !!!

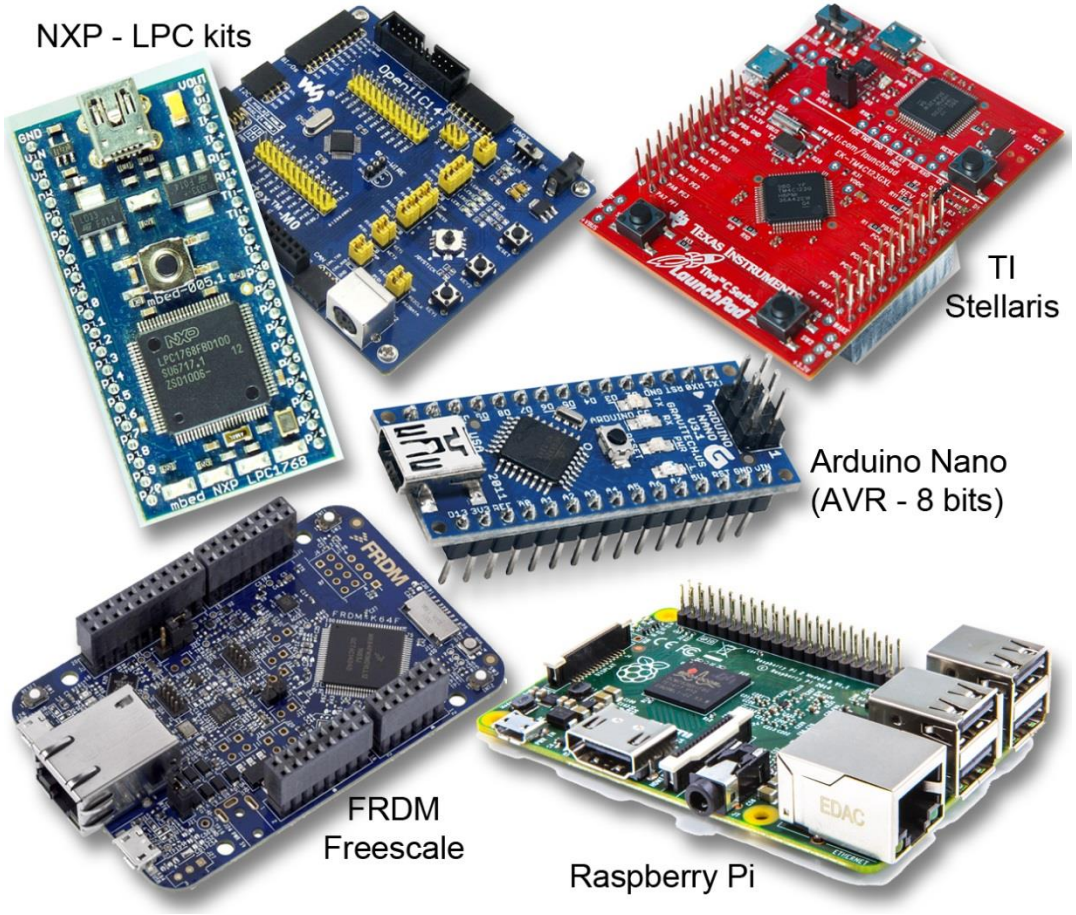
- As ferramentas de desenvolvimento ***evoluíram*** com o passar dos tempos (esta era a "old-school"
- 2 exemplo: placas que já possuem adaptadores JTAG embutidas...





No caso da STM32 Discovery o ST-LINK já está embutido na placa (eliminando o ST-LINK externo)
A *Freescale* (NXP) criou uma interface aberta (OpenSDA) que faz o papel do adaptador USB-JTAG.

As ferramentas de desenvolvimento de software (IDE – Integrated Design Environment) permitem acesso ao JTAG da maioria dos μP/ μC – desde que ele tenha uma interface... (*ex: uma interface via USB com um PC, como o USB blaster da Altera - FPGA*). Mas existem diversos kits diferentes !!!



Um mar de siglas... ***interfaces físicas/hardware***

Estas são algumas das siglas que encontramos como portas/sinais de debugging para ARM Cortex

CMSIS - Cortex Microcontroller Software Interface Standard

Cortex Microcontroller Software Interface Standard (CMSIS), padronizado pela ARM, é uma camada de abstração de hardware para a série do processador Cortex-M e especifica as interfaces do depurador – é independente de fornecedor (nota: Cortex M = μControladores).

CMSIS–DAP - Cortex Microcontroller Coresight Debug Access Port (DAP)

CMSIS-DAP fornece uma maneira padronizada para acessar a porta de debug de um μC ARM Cortex via USB. É geralmente ***implementado como um chip de interface on-board***, ligando a placa de desenvolvimento via USB diretamente com um depurador no PC host. O degugger pode usar uma porta JTAG (Joint Teste Action Group) ou SWD (Serial Wire Debug) do dispositivo de destino para aceder ao Coresight DAP.

Joint Test Action Group – ou Boundary Scan - (JTAG)

Conjunto de fios (interface física + máquina de estados) que controla cada pino físico de entrada/saída do circuito integrado, permitindo isolar o core ou os pinos de saída. Usado para ***testes*** de placas, para ***gravação*** de memória in-circuit, e para ***depuração***.

Serial Wire Debug (SWD)

SWD está disponível como parte do Coresight Debug Access Port (DAP), e fornece uma porta de depuração de apenas 2 pinos, alto desempenho, e uma alternativa para a interface JTAG.

Background debug mode (BDM) interface

Interface eletrônica que permite a depuração '***in-circuit***' de μC ARM Cortex da Freescale/NXP. Exige ***um*** fio e um circuito especializado no sistema a ser depurado, mas nenhum hardware especial é necessário no hospedeiro; apenas um pino de I/O biidirectional. A interface permite a um host gerir e consultar um μC alvo.

(existem alguns padrões de interface para depuração/gravação... veremos o OpenSDA)

Funções da interface JTAG:

1) ***Teste das placas*** de circuito impresso (depois de soldados os chips)

2) ***Programação*** de chips já soldados na placa (in-circuit programming)

3) ***Depuração (debugging)*** de software e hardware na placa.

Para quem tiver espírito empreendedor: ***boundary scann (JTAG)*** está gerando muitas oportunidades em qualquer das três aplicações acima!

Obrigado...

Até a próxima aula.