Placa de Desenvolvimento para Microcontroladores da família PIC32MX

Rómulo Antão, Luís Terra, Alexandre Mota, Rui Martins

Resumo – **Este artigo apresenta uma placa de desenvolvimento para microcontroladores de 32bit da família PIC32MX da Microchip. São analisados os aspectos mais importantes do seu *core* MIPS, a organização de memória, os seus periféricos assim como as ferramentas de suporte ao desenvolvimento de aplicações para este sistema. A título de comparação, são realizados três testes em que se avaliam os tempos de execução de três algoritmos que fazem uso de capacidades matemáticas e manipulação de dados, utilizando um produto da série PIC32MX e outro da série PIC18, de 8bit. Finalmente, é apresentado também um exemplo de aplicação do microcontrolador, suportado por um Kernel Tempo-Real, destinado ao controlo de um inversor eléctrico.**

Palavras chave – **PIC32MX, 32bit, MIPS, Microcontroladores, Placas de Desenvolvimento, Kernel Tempo-Real**

I. ENQUADRAMENTO

Em praticamente todos os sistemas electrónicos existem unidades microcontroladoras dedicadas à execução de tarefas como a instrumentação, o controlo ou as comunicações em rede. Actualmente, a oferta de microcontroladores presentes no mercado agrupa-se em três grandes categorias, em dispositivos de 8-bit, 16-bit e 32-bit. Ainda que constituam a categoria que geralmente apresenta especificações mais limitadas, os sistemas baseados em arquitecturas de 8-bit continuam com uma forte posição no mercado [1], principalmente devido ao reduzido custo que a sua utilização comporta e por se apresentarem ainda suficientemente adequados à grande parte das aplicações que requerem a integração de uma unidade de controlo para tarefas pouco complexas. No entanto, começa a verificar-se uma tendência de evolução deste mercado, em que o aparecimento de sistemas com elevados requisitos de performance e *troughput* de dados, como dispositivos de controlo em automóveis, automação industrial, descodificadores de vídeo e áudio, *routers*, consolas de jogos ou sistemas que requeiram encriptação de informação justificam a integração de dispositivos com um maior número de bits, que apresentam uma maior eficácia na relação carga de trabalho *vs* consumo energético [1].

Face à crescente generalização da oferta de microcontroladores baseados em arquitecturas de 32-bit, com *cores* de elevada capacidade de processamento, elevada integração de periféricos, elevada capacidade de memória, modos de poupança energética eficientes e à cada vez mais reduzida diferenciação ao nível do preço de venda destes produtos face a outras unidades de 16-bit e 8-bit, torna-se cada vez mais atractiva a opção de os utilizar em projectos tanto a nível académico, como também a nível profissional. Uma arquitectura possuindo elevados recursos tende a levar a uma redução dos custos de desenvolvimento em termos da produtividade da equipa de trabalho uma vez que, ao invés de um engenheiro despender grande parte do tempo a optimizar um programa, reduzindo o número de chamadas encadeadas de funções que, por exemplo, facilmente excedem a capacidade da *stack* do sistema, minimizando o espaço ocupado de modo a ser suportado pela capacidade do microcontrolador ou até reescrever segmentos de código em *assembly* de modo a maximizar a performance de execução do programa, este poderá focar o seu trabalho adicionando funcionalidades inovadoras ao produto que desenvolve.

Não menos importante que as capacidades do hardware do microcontrolador escolhido, são as ferramentas de desenvolvimento disponibilizadas para este. A existência de soluções *out-of-the-box* que apresentem uma curva de iniciação suave à utilização do microcontrolador, bibliotecas de periféricos prontas a utilizar bem como as perspectivas de suporte ao produto por parte do fabricante durante um longo período de tempo, são factores bastante importantes quando se opta por uma determinada plataforma de trabalho.

Uma das companhias responsável pela massificação dos microcontroladores de 8 bit e 16 bit foi a *Microchip Technology Inc.*. Neste momento, grandes companhias como a *NXP*, *ST* e *Texas Instruments*, assumem-se como os grandes *players* no universo dos microcontroladores de 32 bit, baseando as suas arquitecturas em *cores ARM*. No entanto, a *Microchip* lançou-se neste novo mercado nos finais de 2007 apresentando a sua gama de dispositivos PIC32MX que apresentam uma arquitectura que supera largamente as barreiras tradicionais associadas aos microcontroladores de baixo custo existentes no mercado, apresentando um *core* a operar a 80MHz, com design baseado num modelo *reduced instruction set computer* (RISC), que executa um *set* de instruções desenvolvido pela *MIPS Technologies*, bem como uma memória *Flash* com capacidade até 512 KB e memória *SRAM* até 128 KB. De modo a incentivar a migração dos sistemas implementados em plataformas de 16-bit para 32-bit e minimizar o impacto negativo que a alteração da plataforma base de um sistema normalmente acarreta, a série *PIC32MX* manteve a compatibilidade do *pinout*, periféricos e de *software* relativamente à sua família de microcontroladores de 16-bit, sendo adicionado o suporte da nova gama de produtos ao já bem estabelecido ambiente de desenvolvimento *MPLAB*. Esta estabilidade no que toca às ferramentas de trabalho e familiarização com o sistema é em grande parte responsável pela simplicidade de migração para esta arquitectura.

O processo de desenvolvimento de sistemas embutidos passa sempre pelas fases de especificação e implementação de soluções, para apresentação de um produto final que

vá de encontro à aplicação a que se destina. Desde a versão inicialmente dimensionada até ao produto final entregue, decorre um processo de investigação e testes em que se efectuam múltiplas alterações e constantes acréscimos ao sistema em desenvolvimento até, finalmente, se obter uma versão pronta para utilização. Durante esta fase, são situações comuns a existência de componentes danificados, erros de dimensionamento ou mesmo a completa alteração das abordagens utilizadas. Em todos estes exemplos, os *developers* necessitam de ser expeditos nas alterações que realizam, sendo mais prioritária a facilidade com que se alteram, testam e validam os sistemas projectados do que a própria dimensão destes, pelo que, a utilização de circuitos integrados com encapsulamento *DIP* é, na maior parte das vezes, a mais adequada. Este encapsulamento permite uma elevada facilidade no que toca à sua montagem e substituição em placas de prototipagem ou mesmo em PCB (*Printed Circuit Board*), recorrendo a *sockets*, permitindo também a realização de *debugging*, directamente sobre o circuito integrado, recorrendo ao osciloscópio com pontas de prova comuns, devido à grande dimensão dos *pads*.

No entanto, os microcontroladores tecnologicamente mais avançados disponíveis à data apresentam unicamente encapsulamentos do tipo *Surface-Mount Devices (SMD)*, como é o caso dos dispositivos da família *PIC32MX*. Assim, de modo a beneficiar dos avanços proporcionados pela arquitectura destes novos microcontroladores, do bom suporte ao nível das ferramentas de trabalho, do apoio de uma forte comunidade de utilizadores, bem como da liberdade proporcionada pelo encapsulamento *DIP* de um circuito integrado, criou-se uma placa de desenvolvimento para facilitar a prototipagem de sistemas electrónicos com base em microcontroladores de 32-bit.

II. ARQUITECTURA DO MICROCONTROLADOR

Os controladores da série *PIC32MX* apresentam-se como complexos *System-on-a-Chip* baseados num *core* da *MIPS* de 32-bit, capaz de operar com uma frequência de relógio até 80MHZ, apresentando diversos módulos integrados dos quais se podem destacar um controlador *DMA* de oito canais, controladores *USB* e *Ethernet*, um barramento matriz que permite comunicação entre os diversos módulos a altas velocidades, um controlador de interrupções de alta performance bem como uma *prefetch cache* de 256 Bytes com vista a reduzir a latência nos acessos à memória *Flash* do sistema. Existem também outros periféricos mais comuns como conversores analógico/digital (*ADC*) de 10-bit, interfaces *SPI*, *I2C* e *UART* para comunicações série. Operando à sua máxima capacidade, este controlador é capaz de oferecer um *throughput* de instruções de 1.56 *Dhrystone MIPS/MHz*, unidade *standard* da indústria para avaliar a performance de processadores e compiladores [2].

Relativamente à sua arquitectura, esta pode ser dividida em quatro blocos funcionais gerais: *core*, memória do sistema, integração do sistema e periféricos, sendo alguns dos módulos apresentados no diagrama da figura 1.

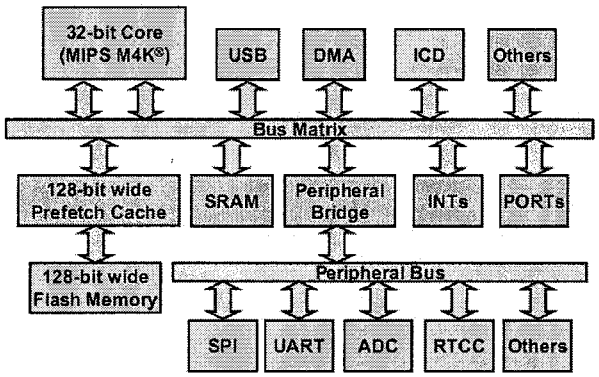

Fig. 1 - Diagrama de Blocos da arquitectura PIC32MX

A. Core

O *core* da série *PIC32MX* é baseado na arquitectura *RISC MIPS32 M4K*, pertencente à gama mais baixa da linha de produtos da *MIPS* de 32-bit. Os *cores* desta gama são bastante costumizáveis aquando da fase de produção, permitindo direccionar este produto para inúmeras aplicações com requisitos diferenciados prevalecendo sempre o melhor compromisso entre performance e o consumo energético/área de silício [3].

A implementação utilizada pela *Microchip*, sumariamente, contempla um *core* com uma frequência de operação limitada a 80MHz, uma unidade de multiplicação/divisão de alta performance (*FMDU*) e 32 registos sombra - uma cópia dos 32 registos de uso geral (*GPR*), que são unicamente utilizados por uma rotina de interrupção de prioridade máxima, de modo a que o atendimento desta não sofra da latência causada pela salvaguarda de todos os registos aquando da comutação de contexto do *core* necessária nestas situações [4].

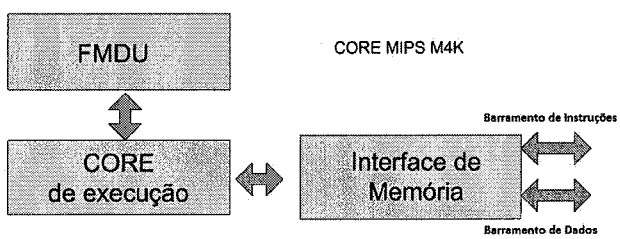

Fig. 2 - Diagrama de Blocos do *Core MIPS M4K*

O *core MIPS M4K* é baseado no modelo *Harvard*, possuindo dois barramentos independentes, dedicados ao *instruction fetch* e às operações *load-store* separadamente, possibilitando a execução de uma instrução e uma transferência de dados paralelamente a cada ciclo de relógio. Cada um destes barramentos apresenta 32-bit de dimensão, permitindo, assim, um maior *throughput* do sistema do que seria atingível no caso de se implementar um único barramento de instruções e dados partilhado. As instruções lidas pelo CPU são executadas em 5 etapas (*Instruction Stage, Execution Stage, Memory Stage, Align Stage, Writeback Stage*) recorrendo a um *pipeline* de 5 níveis. Desta forma,

torna-se possível executar, mediante certas condições, uma instrução por cada ciclo de relógio. No entanto, muitas vezes existe dependência de registos entre múltiplas instruções existentes no *pipeline*, situação conhecida como *interlock*, que implica o atraso da execução das restantes instruções que não podem aceder ao registo pretendido [5].

O *core* apresentado é uma arquitectura do tipo *Load-Store* com 32 registos de uso geral (*GPR*), registos esses que, para além do uso comum enquanto armazenadores de operandos das instruções, contadores e endereços necessários à operação do CPU, possibilitam ao utilizador salvaguardar valores frequentemente utilizados por forma a reduzir o *overhead* das operações *load-store*, tornando o sistema mais rápido na realização de determinadas tarefas. Esta funcionalidade pode apresentar vantagens em aplicações relacionadas com o processamento de sinal em que, por exemplo, a rápida disponibilidade dos coeficientes de filtros resulta num aumento do desempenho global da tarefa em questão.

Ao nível do *core* existe ainda uma *FMDU, Fast-Multiply-Divide Unit*, que suporta diferentes tipos de instruções, como multiplicações com sinal, sem sinal, multiplicação-acumulação e divisão. Esta unidade é então responsável pela execução das instruções matemáticas, permitindo que o *core* efectue o *fetch* das instruções seguintes que não recorram à *FMDU* paralelamente. A *FMDU* existente nesta arquitectura é capaz de executar num único ciclo de relógio multiplicações 16X16-bit e 32X16-bit e necessita de dois ciclos para operações 32X32-bit. As operações de divisão são implementadas com um algoritmo iterativo de 1-bit por ciclo de relógio que podem durar de 11 a 34 ciclos de relógio, dependendo da dimensão dos operandos [6], [3].

Por defeito, o *PIC32MX* executa instruções de 32-bit mas, no entanto, para aplicações em que o tamanho do código é crítico, existe a possibilidade de executar instruções *MIPS16e*, instruções de 16-bit, que permitem obter uma redução de 40% no espaço ocupado pelo código, comparativamente ao uso de instruções de 32-bit. Além disso, como o *fetch* de instruções é realizado em blocos de 32-bit, existe uma maior poupança de energia no sistema pois apenas é necessário aceder à memória a cada duas instruções executadas [6].

B. Memória do Sistema

No que toca à memória, a série *PIC32MX* disponibiliza uma memória *Flash* com capacidade variável entre 32KByte e 512KByte, bem como uma memoria *Static Random Acess Memory* (*SRAM*), com tamanho entre 8Kbit e 128Kbit, de alto desempenho. Relativamente ao mapeamento desta, os sistemas desta gama de microcontroladores baseiam-se num modelo de memória unificado, ou seja, instruções e de dados residem num espaço de endereçamento único, mas em gamas distintas da memória. Além disso, é também possível executar instruções a partir de uma parte da memória *SRAM* do sistema devido à existência de uma configuração de barramento conhecida como *Bus Matrix*, que permite estabelecer comunicação entre os barramentos da memória *SRAM* e memória *Flash* com o *core* do sistema [5].

Uma das grandes limitações dos processadores em sistemas embutidos prende-se com a performance das memória *Flash* utilizadas, uma vez que, memórias rápidas implicam um consumo energético também superior. De forma a serem cumpridos os tempos necessários para executar correctamente leituras e escritas por parte de *cores* a funcionar a frequências acima das velocidades da *Flash*, é necessária a inclusão de *wait-states*, período durante o qual o *core* aguarda pelo acesso à memória. Os projectistas deste controlador conseguiram balancear a performance do processador com o baixo custo da memória, desacoplando o barramento do core do *PIC32MX* da memória *Flash* pela inclusão de uma *Prefetch Cache*, pequeno mas rápido bloco de memória com 256 bytes de capacidade, organizado em blocos de 128-bit dispostos em 16 linhas, que tem como função armazenar instruções quando estas são acedidas assim como realizar o *prefetch* de instruções antes destas serem efectivamente necessárias. É também possível reservar até quatro linhas deste bloco para realização do *fetch* de dados. Assim, apenas existe uma redução de performance do sistema devido a acessos à memória quando ocorrem *cache-miss* e o *core* tem de aguardar pela disponibilização da instrução necessária com uma latência superior. Na figura 3, é possível observar um diagrama representativo da evolução da performance do *core* com o aumento da velocidade de relógio do *core*. As quebras de performance resultam da necessidade de inclusão de *wait-states* no acesso à memória e são atenuadas pela utilização da *prefect cache* [5].

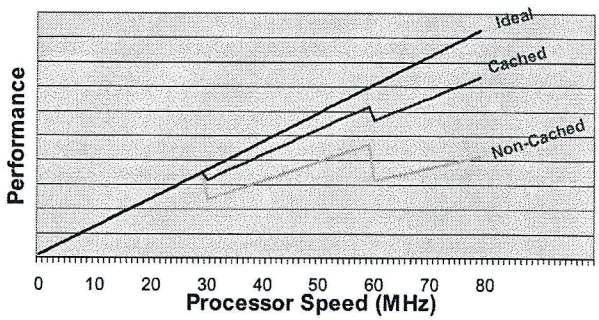

Fig. 3 - Efeito da *Prefetch Cache* no desempenho do sistema

C. Bus Matrix

Para haver comunicação entre os diversos módulos do microcontrolador, torna-se necessária a existência de um barramento partilhado para as trocas de dados. O *Bus Matrix* surge como a estrutura capaz de descodificar uma gama de endereços para mapear o alvo das transferências, *Flash*, *SRAM* ou Periféricos, permitindo assim estabelecerem-se as comunicações entre dispositivos *Master* e *Slave* existentes no barramento. Sendo a arquitectura analisada um sistema que poderá gerar e processar elevadas quantidades de dados a transferir, seria ineficiente todos os eventos entre os diferentes módulos que pretendem estabelecer comunicação estarem dependentes da permissão do *core*. Um dispositivo *Master* diferencia-se de um *Slave* pelo facto

de poder ser iniciador de comunicações, realizando assim, autonomamente, pedidos de leitura e escrita de dados dos periféricos e outros módulos internos do microcontrolador. Actualmente, apenas o *core* e os módulos *In Circuit Debugger*, *USB* e controlador *DMA* são *Master*. Devido à existência de múltiplos *Masters* no barramento, em muitas ocasiões estes iniciarão trocas de informação simultaneamente no barramento. Não permitir esta situação seria ineficiente do ponto de vista do *troughput* de dados pelo que uma das funcionalidades do *Bus Matrix* é funcionar segundo o principio de um *Switch Ethernet*, permitindo a existência de múltiplos fluxos de dados simultaneamente, como se encontra representado na figura 4. Da existência de vários *Masters* decorre também a possibilidade de, num mesmo instante, ocorrerem várias tentativas de acesso ao mesmo *Target*. Nessa situação, o *Bus Matrix* encarrega-se de resolver o conflito utilizando um sistema de prioridades para o desempate, definido aquando da sua configuração [5].

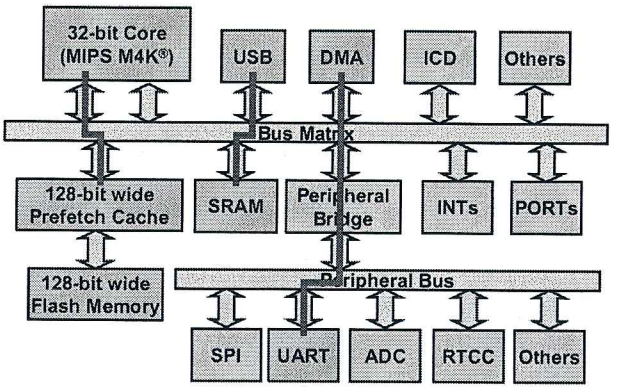

Fig. 4 - Acesso concorrente ao barramento segundo o conceito *Bus Matrix*

III. PLACA DE DESENVOLVIMENTO

Com o intuito de facilitar o desenvolvimento de aplicações com base em dispositivos da família *PIC32MX*, criou-se uma placa de desenvolvimento suportada pelo microcontrolador *PIC32MX340F512H*, disponível na versão com encapsulamento *TQFP* de 64 pinos. Normalmente, para utilização de circuitos integrados com encapsulamentos *SMD* na prototipagem de sistemas, é comum criarem-se simples *breakout boards* que facilitam a rápida integração e substituição destes componentes numa montagem de teste. O conceito subjacente à placa desenvolvida é semelhante mas, no presente caso, este foi elevado a um nível superior, de modo a criar um pequeno módulo que, para além de facilitar o acesso a todos os pinos de I/O, inclui os componentes que asseguram a operacionalidade base do microcontrolador, como a sua correcta alimentação, um interruptor de *reset*, um conector para interface *JTAG*, útil para a realização de *debug*, um cristal de 8MHZ que constitui o oscilador primário externo, um cristal secundário de 32.768kHz utilizado nos módulos de operação de baixo consumo e também permite o funcionamento do relógio e calendário internos do controlador, e *leds* de sinalização das tensões de alimentação. Para se iniciar a utilização do microcontrolador é então apenas necessário alimentar o módulo, tipicamente a 5V, sendo a sua integração com outros componentes bastante facilitada, por apresentar uma estrutura baseada em dois barramentos de 28 pinos paralelos, compatíveis com o espaçamento dos encaixes das *breadboards*, usualmente utilizadas para prototipagem. Da mesma forma, este sistema também é passível de funcionar como um módulo *piggyback*, de modo a integrar sistemas modularmente desenvolvidos.

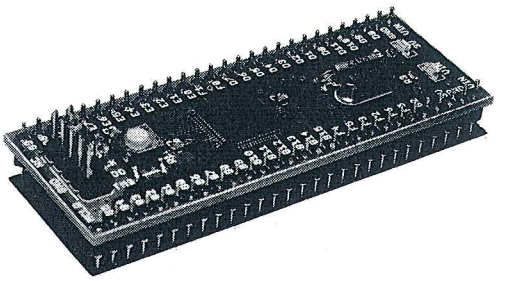

Fig. 5 - Placa de desenvolvimento PIC32UA

Podem-se assim resumir as principais características da placa de desenvolvimento apresentada:

- Controlador PIC32MX340F512H
- Cristal primário de 8MHz
- Cristal secundário de 32.768kHz (RTC)
- Regulador 3.3V *Low Dropout*
- Dois *led* de estado das tensões de alimentação
- Interruptor de *Reset*
- Conector *JTAG* para programação/ *debug*
- 56 Pinos com pitch de 2.54mm
- 50 Pinos de I/O utilizáveis
- Tensão de alimentação entre 4V e 6.47V

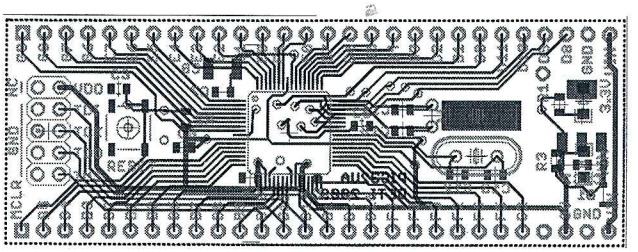

Fig. 6 - Layout da Placa de desenvolvimento PIC32UA

O protótipo apresentado na figura 5 é o resultado do *layout*, exibido na figura 6, criado utilizando a versão livre do programa *CadSoft EAGLE* [7], uma ferramenta simples mas bastante completa que permite produzir esquemas eléctricos e desenhar placas de circuito impresso.

Ao longo do desenvolvimento do *layout*, até ser obtida uma versão final, tiveram-se em conta diversas considerações de modo a optimizar a disposição dos pinos do microcontrolador, minimizar o comprimento das pistas, reduzir o número de vias utilizado, cumprir com os requerimentos energéticos, bem como a distribuição do plano de massa do sistema. O ponto de partida foi a escolha dos componentes a utilizar. Relativamente ao microcontrolador,

a opção recaiu sobre o *PIC32MX340F512H*, uma versão que, à altura da escolha, apresentava as especificações mais avançadas da família. Além disso, de modo obter uma placa de desenvolvimento de dimensões reduzidas, optou-se pela utilização de uma versão de 64 pinos. A nível funcional, o design desenvolvido é compatível com todas as versões de 64 pinos de microcontroladores desta família. Actualmente, existe já uma nova placa que utiliza o microcontrolador *PIC32MX795*, uma versão que se distingue da anteriormente enunciada por apresentar uma maior capacidade de memória *SRAM* e *Flash*, suporte para protocolo *CAN* e interface *USB*.

Relativamente à alimentação do sistema, o consumo de corrente global realizado pelo microcontroloador e *leds* é bastante reduzido. Os dois *leds* consomem cerca de 40mA em conjunto - valor que pode ser bastante melhorado utilizando *leds* de alto rendimento, o microcontrolador operando a 80MHz e executando código a partir da memória *Static RAM*, configuração onde os requerimentos energéticos são superiores, apresenta um consumo típico de 55mA e a corrente máxima que é possível fornecer pelo conjunto de todos os portos de I/O é 200mA. [4] Para alimentar a placa de desenvolvimento, escolheu-se um regulador *Low-Dropout* MCP1825S com uma tensão de saída de 3.3V, capaz de fornecer 500mA ao sistema, garantindo-se ainda alguma margem de segurança, de modo a evitar a ocorrência de fenómenos de *brown-out*. O *package* escolhido, *SOT-223*, é de reduzidas dimensões e apresenta um *pad* exposto com uma área significativa conectado à massa do regulador, podendo assim ser ligado ao plano de massa da placa, de modo a aumentar a sua capacidade de dissipação.

O comprimento das pistas de interligação dos pinos e o número de vias foi também estudado optando-se por não agrupar os portos do sistema no *layout* da placa, perdendo-se uma optimização funcional, mas obtendo-se um plano de massa mais uniforme na parte inferior da placa, que garante uma maior imunidade ao ruído electromagnético e permite reduzir os fenómenos de *cross-talk* causado pela disposição paralela das pistas do sistema [8]. Foram também colocados condensadores de desacoplamento na proximidade dos pinos de alimentação do controlador de modo a evitar que eventuais ruídos na alimentação da placa afectem o funcionamento deste.

Finalmente, foi incluído um *header* na placa que permite utilizar *debuggers JTAG* para análise do sistema durante o seu funcionamento.

IV. SOFTWARE DE SUPORTE

Uma das vantagens em utilizar os microcontroladores da família *PIC32MX*, da *Microchip*, prende-se com o bom suporte a nível de ferramentas de desenvolvimento e ao facto de existirem versões *freeware* para estudantes dos compiladores necessários que, comparativamente às versões comerciais, apresentam apenas ligeiras limitações ao nível das funcionalidades disponibilizadas, sem comprometer, no entanto, a utilização das avançadas capacidades do microcontrolador.

A base de trabalho facultada pela *Microchip*, o ambiente de desenvolvimento integrado (IDE) *MPLAB*, apresenta-se como uma plataforma na qual é possível para além de produzir o código do sistema a desenvolver, proceder à sua compilação, efectuar a programação do microcontrolador, realizar o *debug* do sistema directamente sobre o microcontrolador, e por último possibilitar também a emulação do microcontrolador, tal como a interacção dos seus portos de I/O [9].

Relativamente aos compiladores suportados, a *Microchip* possui um compilador para a linguagem de programação C, o *MPLAB C32*, baseado em *GCC* [10] e disponibilizado em versões *Lite*, com limitações ao nível da optimização de código, em versão académica, unicamente limitada na dimensão do código gerado, que tem um tamanho máximo de 64 KB e em versões comerciais, totalmente funcionais. Na versão académica, o facto de ser possível utilizar todos os níveis de optimização de código, permite explorar melhor a limitação imposta ao tamanho máximo do código compilado. Para além de permitirem uma redução no tamanho do código, as optimizações realizadas pelo compilador visam minimizar o tempo de execução deste através de técnicas como a expansão de ciclos, de modo a tomar maior partido das capacidades de cache do microcontrolador, realizar o *inline* de funções ou efectuar o alinhamento dos endereços das instruções em potências de dois para melhorar a performance do sistema em casos de execução condicional. Juntamente com o compilador, são também facultadas bibliotecas matemáticas e de processamento de sinal de alta performance, cujo desempenho tem vindo a ser optimizado a cada actualização destas ferramentas, bem como bibliotecas bastante úteis para utilização de todos os periféricos integrados nos microcontroladores da série *PIC32MX*, como sejam *stacks TCP/IP* e *USB*, *bootloaders* para as interfaces série e *USB* juntamente com as interfaces de carregamento do código para o sistema operativo *Microsoft Windows* [10], [11]. Mais recentemente tornou-se disponível no mercado um novo compilador multi-plataforma *HI-TECH C PRO for the PIC32 MCU Family* produzido pela *Hi-Tech Software*, que também apresenta versões comerciais, com optimizações de código avançadas, e uma versão livre, de funcionalidades mais comedidas ao nível das optimizações mas com total suporte da plataforma de hardware do microcontrolador [12].

No que toca às ferramentas de programação e *debugging*, existem várias opções baseadas em hardware facultadas pela *Microchip*. Os microcontroladores das séries *PIC32MX3xx* e *PIC32MX4xx* são suportados pelos debuggers *ICD2* e *MPLAB REAL ICE*, produtos já existentes no mercado à altura da disponibilização destes microcontroladores de 32-bit. A distinção destes produtos prende-se com as velocidades máximas de *debugging*, a capacidade de introdução de *breakpoints*, cronometragem dos tempos de execução, capacidade de realizar o *trace* dinâmico de variáveis do sistema, bem como a possibilidade realizar uma análise do sistema ao nível lógico de diferentes sinais. Estas ferramentas auxiliares são bastante vantajosas na medida que permitem analisar o comportamento do microcontrolador enquanto este se encontra em pleno funcionamento, conectado ao *hardware* a controlar [9].

O ambiente *MPLAB* faculta também a possibilidade de realizar testes ao código desenvolvido emulando o *hardware* do microcontrolador bem como as interacções com sinais e alguns dispositivos, utilizando a ferramenta *MPLAB SIM*. A pequena diferença relativamente aos *debuggers* baseados em *hardware* prende-se com o facto destes, quando se encontram a correr o código à velocidade máxima, não permitirem a monitorização de todos os registos e memória em tempo real, contrariamente ao simulador, cuja performance apenas se encontra limitada pelas capacidades do computador onde corre a simulação e tem sempre disponível toda a informação relativa ao estado do microcontrolador [9].

A utilização conjunta destas ferramentas constitui assim uma forte plataforma de desenvolvimento, bastante flexível na relação custo/funcionalidades, que permite abranger um vasto grupo de utilizadores, que pretendam ou não beneficiar dos recursos mais avançados de desenvolvimento, que apresentem um maior ou menor conhecimento da arquitectura *software* e *hardware* em questão. Um factor que pode constituir um entrave para utilizadores de sistemas operativos *Linux* e *MacOS* prende-se com o facto de toda esta plataforma de *software* e *hardware* de *debugging* baseada no ambiente *MPLAB* ser apenas suportada por sistemas operativos *Microsoft Windows*.

V. CASE STUDIES

A. Comparação com o PIC18F458

De modo a avaliar comparativamente todos os avanços desta arquitectura apresentada, realizou-se um pequeno teste com dois microcontroladores da *Microchip*, um *PIC32MX340F512H*, que apresenta uma arquitectura de 32-bit, com um *core* a operar a 80MHz e um *PIC18F458*, arquitectura de 8-bit, funcionando a 40MHz. Compilou-se o código de teste utilizando o compilador *PICC-18 V8.30* da *HT-Soft* para o *PIC18*, e o compilador *MPLAB* C32 V1.05 para o *PIC32MX*, sem quaisquer optimizações para as duas plataformas. O programa de teste consiste em calcular o 24° elemento da série de Fibonacci, o máximo número da série representável numa variável de 16-bit, calcular os elementos de um triângulo de Pascal com 10 linhas, sendo este teste limitado pelas restrições de memória do *PIC18*, e finalmente calcular o resultado da operação *sin(0.123)*. Estes códigos, embora bastante simples, exemplificam situações de execução em *loop*, manipulação de ponteiros e o uso de *libraries* matemáticas para cálculos em vírgula flutuante, constituindo assim uma boa base de comparação da performance destes sistemas. Medindo os tempos de execução, recorrendo aos *timers* dos sistemas, obtiveram-se os resultados apresentados na tabela I.

Teste	PIC18F458	PIC32MX340
	μs	μs
Série de Fibonacci	113	3.38
Triângulo de Pascal	681.6	16.16
sin(0.123)	442	3.24

TABELA I
COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO EM DIFERENTES TAREFAS

Confirmando o esperado à partida devido à superioridade da arquitectura do *PIC32MX*, verificam-se tempos de execução bastante inferiores neste sistema comparativamente ao *PIC18*. Estes resultados são, no entanto, meramente indicativos uma vez que existem nesta situação factores inerentes ao código *assembly* produzido pelos compiladores. No que toca aos preços de mercado destes produtos, em Maio de 2010, o *PIC32MX340F512H* apresenta um custo de €6,74 enquanto o *PIC18F458* um custo de €6,61, que vai de encontro com a análise previamente realizada neste artigo, em que a relação performance *vs* custo tende notoriamente para o microcontrolador de 32-bit.

B. Aplicação baseada num Sistema Operativo Tempo-Real (RTOS)

Acompanhando a evolução dos microcontroladores, que apresentam arquitecturas cada vez mais avançadas e capacidades de memória mais elevadas, os engenheiros de sistemas embutidos começam a ter disponíveis para os dispositivos de custo mais reduzido sistemas operativos, que permitem o desenvolvimento de complexas aplicações segundo um modelo multi-tarefa, à semelhança do que há muito se pratica ao nível dos computadores pessoais. Esta evolução do modelo de programação em microcontroladores, que até muito recentemente era baseado numa estrutura monolítica, em que se recorria a rotinas de interrupção temporizadas para executar acções com requisitos temporais, permite ao utilizador uma maior abstracção dos detalhes complexos relativos ao escalonamento temporal de acções, que passa a ser da responsabilidade do sistema operativo e promove um maior particionamento do software, que facilita o teste, a divisão do trabalho em equipas e reutilização de código entre plataformas.

No seguimento desta filosofia, surge o *kernel* Tempo-Real *FreeRTOS*, um sistema multi-tarefa e multi-plataforma desenvolvido para ser simples, pequeno e fácil de utilizar baseado em *ticks*, unidade de tempo base da qual, todos os eventos do sistema, são múltiplos inteiros. Este *kernel* não impõe restrições ao número de tarefas que o sistema pode apresentar, realizando o escalonamento destas com base um mecanismo de prioridades fixas, sendo opcional a existência ou não de preempção entre estas. Embora nestas arquitecturas de *software* um programa se apresente segmentado em tarefas, na maioria das situações existem relações de dependência entre estas, como precedências de execução, partilha de recursos e de dados, sendo portanto facultada pelo *FreeRTOS*, uma Interface de Programação de Aplicativos (API) que inclui ferramentas de sincronismo e comunicação entre tarefas ou entre tarefas e interrupções, como filas de mensagens, semáforos e *mutexes*. Devido ao facto de ser possível a existência de preempção entre as tarefas, estes recursos são importantes também para evitar a existência de condições de corrida no acesso a estes, que degeneram numa execução inconsistente dos programas. Por ser escrito maioritariamente em C e ser uma ferramenta *open-source*, o *kernel FreeRTOS* apresenta elevada portabilidade e escalabilidade, sendo compatível com microcontroladores de fornecedores, para além da *Microchip*, como a *ATMEL*, a *NXP*, a *Xilinx*, entre outros [13].

C. Aplicação Prática

No âmbito da dissertação de mestrado *Inversor Eléctrico para Sistemas de Microgeração*, realizada no DETI no ano lectivo 2009/2010, pretende-se utilizar o potêncial do *hardware* da placa de desenvolvimento previamente apresentada, bem como das capacidades do *kernel freeRTOS*, desenvolvendo uma aplicação que realize o controlo de um inversor DC-AC. Entre as diversas tarefas implementadas no sistema, destacam-se o controlo da forma de onda da corrente de saída, activação dos mecanismos de protecção do sistema, regulação das diferentes fontes de alimentação do circuito, bem como implementar uma interface de comunicação com o operador para monitorizar a resposta do sistema e proceder à calibração dos controladores.

O inversor desenvolvido foi especificado para operar ligado directamente à rede de distribuição eléctrica, possibilitando a venda de energia ao operador desta, produzida por fontes renováveis como panéis fotovoltaicos ou geradores eólicos de baixa potência. Como é um dispositivo que se encontra ligado à rede eléctrica, e sendo esta um meio partilhado, a qualidade da corrente injectada pelo inversor na rede deverá respeitar normas de qualidade, nomeadamente ao nível da distorção harmónica máxima que esta pode apresentar [14], bem como apresentar um desfasamento mínimo com a onda de tensão de modo a maximizar a potência activa entregue. Desta forma, o controlo de corrente deverá ser capaz de gerar uma corrente sinusoidal, com a frequência de 50Hz, em fase com a tensão. Nos testes realizados, o inversor foi configurado para fornecer uma corrente de saída de 2A *rms*. Os resultados desta configuração encontam-se apresentados na figura 7, onde são exibidas as formas de onda de corrente e tensão no ponto de interface entre o inversor e a rede eléctrica.

Nas tabelas II e III são apresentados os resultados obtidos nestas condições de operação.

Estes resultados atestam o correcto funcionamento do inversor nas condições enunciadas.

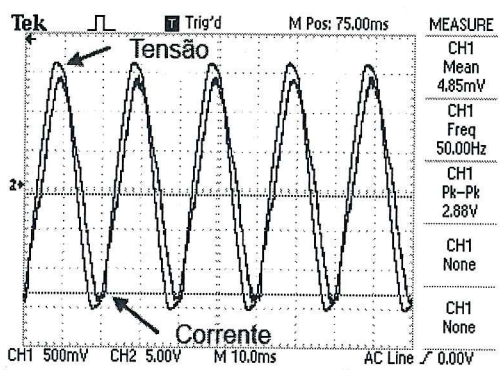

Fig. 7 - Formas de onda da tensão e corrente no ponto de interface com a rede eléctrica

VI. CONCLUSÃO

Os avanços proporcionados pela arquitectura PIC32MX prometem alterar o panorama dos microcontroladores utilizados em sistemas embutidos de baixo custo, por proporcionarem aos utilizadores vantagens fundamentalmente ao nível da performance e elevada disponibilidade de recursos, como a memória e integração de periféricos. O seu baixo custo, o bom suporte ao nível de ferramentas de trabalho e semelhança do modelo de programação relativamente a outros produtos da *Microchip*, tem impulsionado o aumento da cota de mercado dos microcontroladores de 32bit da família PIC32MX. Desta forma, tanto em meio académico como empresarial, esta plataforma assume-se cada vez mais como uma alternativa aos microcontroladores de 16bit, principalmente em aplicações embutidas que apresentem um grande *throughput* de dados e seja necessário capacidade de processamento, como em sistemas de automação com integração em redes de comunicação.

Entrada				Saída	
$I_{out\ esperado}$	I_{in}	V_{in}	P_{in}	I_{out}	P_{out}
A	A	V	W	A	W
2.0	19.860	26.70	530.26	2.10	484.37

TABELA II

DADOS RELATIVOS À OPERAÇÃO DO INVERSOR, PARA UMA ONDA DE CORRENTE DE REFERÊNCIA DE 2A

Factor Potência	THD	Rendimento
	%	%
0.94	0.54	85.8

TABELA III

PARÂMETROS DE QUALIDADE DA OPERAÇÃO DO INVERSOR

REFERÊNCIAS

[1] "Bulk of MCU Revenue Moving to 16-bit and 32-bit", www.instat.com/newmk.asp?ID=1494.

[2] Alan R. Weiss, "Drystone Benchmark, History, Analysis, Scores and Recomendations", www.johnloomis.org/NiosII/dhrystone/ECLDhrystoneWhitePaper.pdf, 2002.

[3] Inc. Berkley Design Technology, "An Independent Analysis of the mips Technologies MIPS32 M4K Sythesizable Processor Core", 2007.

[4] "PIC32MX Family Reference Manual, Microchip", ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX_FRM_Sect01_Intro_61127C.pdf.

[5] "PIC32 Web Seminars", http://techtrain.microchip.com/webseminars/QuickList.aspx.

[6] "MIPS32 M4K Core Datasheet", www.mips.com/media/files/MD00247-2B-M4K-DTS-02.01.pdf.

[7] "Cadsof EAGLE", www.cadsoft.de/.

[8] "Maxim Application Note 716 - Proper Layout and Component Selection Controls EMI", http://pdfserv.maxim-ic.com/en/an/AN716.pdf.

[9] "MPLAB IDE User's Guide with MPLAB Editor and MPLAB SIM Simulator", ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_User_Guide_51519c.pdf.

[10] "MPLAB C Compiler for PIC32 MCUs User's Guide", ww1.microchip.com/downloads/en/DeviceDoc/51686B.pdf.

[11] Lucio Di Jasio, *Programming 32-bit Microcontrollers in C - Exploring the PIC32*, Newnes, 2008.

[12] "Hitech-C compilers by Microchip Technology", www.htsoft.com/.

[13] "FreeRTOS Features Overview", www.freertos.org.

[14] "IEEE 519-1992 Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems", IEEE Industry Applications Society / Power Engineering Society, 1992.