The evolution of DuinoMite, and the differences between MaxiMite and DuinoMite?

21-OCT-2011

It all started back on the 19th of July 2011, when Don McKenzie from Dontronics sent me an e-mail asking if we at Olimex were interested in modifying our PIC32-PINGUINO to run a Basic interpreter.

Don pointed me to the MaxiMite project (http://geoffg.net/maximite.html) which uses a PIC32 to generate VGA signals, and interfaces with a PS2 keyboard, so that you end up with a single chip, complete BASIC computer system. This project became very popular in Australia after a string of articles in the local Australian Silicon Chip Magazine, but hasn't yet gained much popularity outside Australia. MaxiMite gives you the ability to write small embedded applications without using a PC, compilers, debuggers etc. All you need is the small one PIC computer plus a PS2 keyboard, and a VGA monitor.

Perhaps best of all is that Geoff Graham made the MaxiMite an open source software, and open source hardware project, so you can modify the code or the hardware, and make your own boards and firmware.

We liked the idea straight away, so we built a team inside of Olimex and started working on the schematic. There are three more people that got involved in the development: Don McKenzie and Mick Gulovsen from Dontronics, and Ken Segler of Ken Segler Designs, to help with the firmware.

Don's idea was to make the new board compatible with the popular Arduino (http://www.arduino.cc) shields as there is a vast range of shields developed already, and a lot of open source projects done with them which could be used to build projects with the MaxiMite. The MaxiMite has a 26 pin GPIO connector with 20 IO signals on it, which you can be easily manipulated in Basic by defining their function like: analog input, analog output, digital input, digital output, frequency counter, etc.

To make the MaxiMite with an Arduino shield was fairly easy, as our PIC32-PINGUINO had the same shield layout, and an earlier PIC32 micro, so we already knew which signals we should connect to the Arduino shield layout for maximum compatibility. The problem we saw is that many of these signals were already used for other purposes on the MaxiMite.

Once we decided to run with the project, we had to find a knockout name for it. MaxiMite is an explosive invented by Hudson Maxim (http://www.magnumarchive.com/c/encyclopedia-americana-volume-18/Maximite.html). As we wanted to make it Arduino compatible we tossed around many names and eventually Mick Gulovsen came up with the name DuinoMite. The team fell in love with this name straight away. We now pronounce it "Dynamite", so we have kept the explosion in the name ☺.

When we began to compare the MaxiMite with the PIC32-PINGUINO, we found that the MaxiMite schematic did not use all of the features that the PIC32 could offer, and that we had already implemented in PIC32-PINGUINO, so many changes had to be made to take advantage of these PIC32 features, so they could be used in the new DuinoMite Computer.

Low Power

MaxiMite was not designed to be a low power device. It runs a high frequency crystal so you can either shut down the processor and lose timekeeping functionality, or run it at high speed and consume about 140mA.

DuinoMite was built to work as a hand-held data logger. We used ultra low power voltage regulators, which draw only 1.6 uA current, and the consumption on this board could be as low as 30uA while keeping the RTC functionality, as the board has a 32kHz low frequency crystal.

On the MaxiMite the RC13 PIC32 port where the low frequency crystal should be connected is used for the Boot-loader button, so the MaxiMite can't run on lower frequency without this schematic change. Our

DuinoMite schematic allows the PIC32 to be put into low power mode with a working 32 Khz oscillator, and consuming only 30 uA with all peripherals shut down. To keep the power at minimum we had to add a FET, which will power on and off the SD-card, CAN, and RS232 drivers.

Battery Operation

DuinoMite can run on a Li-Po battery stand alone with some restrictions.

For example, the keyboard can't be used as it requires +5V power and the Li-Po battery allows only 3.7V.

The battery voltage can be measured with the AN2/PB2 ADC. Also the DuinoMite design allows the DuinoMite firmware to be aware when the board runs on battery power supply, by monitoring the USB-FAULT line, which can monitor when there is no USB or external power connected (logic LOW).

Power Supply

We decided to add a Lithium Polimer battery charger and connector for LiPo battery.

Also, as the PIC32-PINGUINO was designed with Industrial applications in mind, we decided to keep the complete power supply from Pinguino i.e. 9-30Vdc input power supply and industrial temperature range -20+85C operation.

The original MaxiMite used a linear LM7805 which can't handle much power dissipation at room temperature.

If you look at the DuinoMite-Mega schematic:

http://www.olimex.com/dev/DUINO/DUINOMITE/DUINOMITE-MEGA-REV-A.pdf

it seems quite complicated and unnecessary, but it allows the board to be powered from 3 sources – External Power Supply / USB / Battery. Switching between the power sources is done automatically.

If you have no external power and no USB connected, the board takes power from the battery.

If USB is connected, the power automatically switches to the USB, and if the battery is not charged at 100%, it will start charging

If external power is applied, the power will be sourced from the external connector.

On the MaxiMite you have a jumper to select the power source, either USB or external supply, and when you alter the jumper from one to other, the MaxiMite will power down.

NOTE!! The DuinoMite-Mini board only supports +5VDC External Power Supply. Of course, it can also be powered by USB. This was to keep costs down for an entry level (or student model) DuinoMite. A cheap source of external Power Supply is an AC to USB +5V Power Supply, which can be found for under $2 on Ebay, and is used for iPODs, cameras, eReaders, etc.

RS232

For some reason the MaxiMite design has no real UART on the GPIO pins, although this is a very common interface in the embedded world.

Most of the peripheral devices such as GSM, GPS, Bluetooth, even Camera modules, use UARTs to transmit and receive data. The PIC32 has 6 hardware UARTs available with interrupts, buffering, DMA etc., which

normally releases the CPU from any overhead time to handle it, but in the MaxiMite the UART has to be achieved by bit-banging, and GPIO polling, as none of the hardware UARTs are available on the GPIO pins.

This just adds unnecessary overhead to the CPU as it polls GPIO lines instead of doing something useful, which effectively slows down the code execution. We had to fix this in our DuinoMite design. As the PIC32 multiplexes the UART lines with other ports it was not possible to use all 6 UARTS on the DuinoMite, as some ports were used for the VGA generation, and also shared with UART functionality.

DuinoMite offers 3 UARTs for users:

UART5 goes directly to the RS-232 driver chip and RS-232 connector, (Mega Version Only) and is also available on pins D0 and D1 of the Arduino Shield and the GPIO. Both of these signals are fully isolated with un-populated resistor positions R2, and R3. If you wish to connect UART5 to the GPIO and the shield, then you must populate these positions with zero Ohm resistors, or shorting links.

UART1 is available on pins D11 and D12 of the Arduino Shield, and also the GPIO connector.

UART2 goes directly to the UEXT connector, which interfaces to our range of external modules. (see UEXT later in this document)

USB-OTG

The PIC32 has USB On-The-Go which allows the PIC32 to act not only as a USB Device, but also as a USB Host interface. This means when you initialize the USB as host you can use peripherals such as: USB mouse, USB keyboard, USB camera, USB printers, USB Bluetooth, WiFi modules etc.

Of course all of these devices need proper drivers to be implemented, but in the original MaxiMite this USB-OTG feature is not available although the PIC32 has it.

We have this in the PIC32-PINGUINO so we decided to keep it in our DuinoMite schematic too. A good application for USB-OTG is that it allows Android devices to be connected to the PIC32 and DuinoMite, and to use the Android ADK to interface to sensors, etc. with it.

With the board drawing about 30uA in low power mode, building Android phone accessories can be very easy to do with DuinoMite.

Here are few videos which you can look at to see what you can do with Android device and DuinoMite:
http://www.youtube.com/watch?v=SXyc8uE5YuI
http://www.youtube.com/watch?feature=player_embedded&v=CJ0j_vZ4AeM
When you consider that the DuinoMite is low power, you can easily make hand-held sensor accessories to any Android device.

Buttons

We decided to add a User button with noise filtering (remember, we designed this board to work in industrial applications, and in noisy environments). This same User button in conjunction with the reset button, allows the board to go into the Bootloader mode, for updating the firmware.

Noise Immunity

DuinoMite was built to work in Industrial environments. The USB, reset, user button, SD card and all peripheries were carefully designed to be `noise immune'. MaxiMite has no protection for the USB, buttons, etc., so it should work OK at home, but may not work as well in industrial or noisy environments.

VGA/composite Video

The original MaxiMite has a jumper selector to manually configure the output mode. Ken Segler came up with a proposal to use the monitor itself as a means of selecting the output. The VGA connector has multiple grounds, so we have used two pins as a jumper select. The DuinoMite firmware can recognize if a VGA monitor is attached and start generating the correct output. If nothing is attached to VGA connector, it automatically switches to composite video output mode.

CAN – (DuinoMite Mega Only)

Controller Area Network (CAN or CAN-bus) is a vehicle bus standard designed to allow micro-controllers and devices to communicate with each other within a vehicle, and without a host computer.

MaxiMite uses the PIC32MX795 which has CAN, but MaxiMite doesn't make use of it. CAN is a very useful interface, it's the de-facto standard for the automotive bus applications, so by having CAN it would be possible to connect to your car and read all of the data sensors for speed, temperatures, fuel consumption, etc . This video can give you rough idea what you can do with CAN and DuinoMite.
http://www.youtube.com/watch?v=PbA_bOO2mMw
Ken is working on firmware which will allow you to see CAN as a COM port on DuinoMite basic so you can use INPUT #5 and PRINT #5 to receive and send CAN messages.

Being a robust and noise immune protocol, CAN is used not only in automotive but also in industrial robot applications – For more information see the following links
http://en.wikipedia.org/wiki/DeviceNet
http://en.wikipedia.org/wiki/CANopen

UEXT

This is our universal connector we developed here at Olimex, to allow external modules to be connected. The list of modules currently available can be found at:
http://www.olimex.com/dev/OTHER/UEXT.pdf
One of the most interesting modules for DuinoMite users may be the RS485 module which allows a UART to be connected to a RS485 network using the MOD-IO module, which will allow, virtually, an unlimited number of relays, digital opto-isolated inputs, and analog inputs to be cascaded, all fully supported in MaxiMite/ DuinoMite Basic. Ken is working on firmware where you will be able to access MOD-IO inputs and outputs with Basic commands.

Having all these enhancements, we simply couldn't keep the original MaxiMite schematic and port assignments the same, so here is brief summary of the GPIO port differences between the MaxiMite and the DuinoMite:

Both the MaxiMite and the DuinoMite have 20 GPIOs (General Purpose I/O Pins) which can be accessed with PIN() in basic, but there are some distinct differences in the way these GPIOs can be configured between the MaxiMite and DuinoMite.

Refer to the following GPIO Table:

SetPin #	I/O Type	Maximite GPIO Pin()	DuinoMite GPIO Pin()
1	**Analog Input**	**1-10**	**1-7,19,20**
2	**Digital Input**	**1-20; (11-20 are 5V tolerant)**	**1-10, 12-20; (8-10, 13-18 are 5V tolerant)**
3	**Frequency Input**	**11-14**	**5-7**
4	**Period Input**	**11-14**	**5-7**
5	**Counting Input**	**11-14**	**5-7**
6	**Interrupt (L-H)**	**1,2,10-14**	**1,2,5-7**
7	**Interrupt (H-L)**	**1,2,10-14**	**1,2,5-7**
8	**Digital Output**	**1-20**	**1-20**
9	**Digital Output (+5V Open Collector)**	**11-20**	**8-10,13-18**

Also special care should be taken when using DuinoMite GPIOs 7-10,19,20 as these are shared with the SD card and VGA signals. DuinoMite firmware has provision for the VIDEO ON/OFF command which will shut down the VGA signal generation to lower the power, and in this case the multiplexed GPIOs could be used. Same applies to the SD card multiplexed pins.

What does this mean? When you want to move code from the MaxiMite to the DuinoMite, you should take care to re-map the GPIO pins you use so that they have the functionality you require. For instance if you use frequency input on the MaxiMite PIN(11) you can't use the same code but should modify it to use PIN(5) on the DuinoMite for instance, and vice versa. Code which is written for the DuinoMite and does not use some of the special features such as CAN, RS232, Low power mode, USB-OTG may be ported to work on the MaxiMite if the GPIOs are re-mapped.

The PINGUINO connection (http://www.pinguino.cc) – DuinoMite is very similar to the PIC32-PINGUINO as functionality, and the support for this board in Pinguino IDE will be pretty easy to implement, so the people that use this board will have a choice – to program it in Basic, or to program it in an Arduino like language, or to program it directly in C as there is a C compiler on the back-end of Pinguino environment.

Another option is to develop in Assembler, or with a C32 compiler via MPLAB and PIC-KIT3

So in the near future, DuinoMite will give you these 4 different programming options:

- MaxiMite basic language extended with additional Basic commands which exploit the new features of DuinoMite offers over MaxiMite.
- Pinguino IDE development via Arduino like sketches
- Pinguino IDE development in C language
- MPLAB and C32 development and PIC-KIT3 as programmer