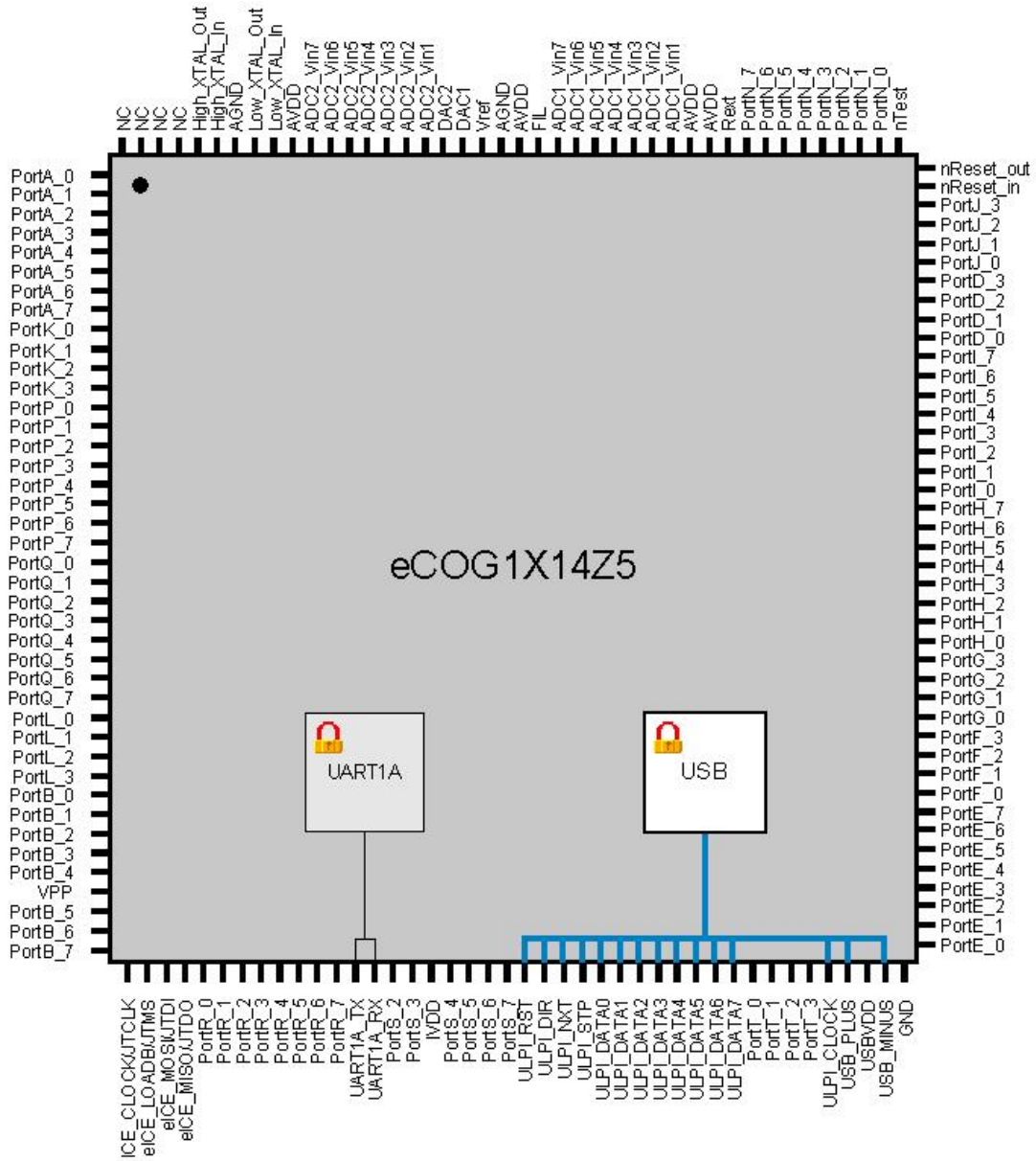


AN053 – eCOG1X USB Hardware Requirements

Version 1.1

This application note describes the external hardware required for use with the USB Peripheral in the eCOG1X microcontroller.



Confidential and Proprietary Information

©Cyan Technology Ltd, 2008

This document contains confidential and proprietary information of Cyan Technology Ltd and is protected by copyright laws. Its receipt or possession does not convey any rights to reproduce, manufacture, use or sell anything based on information contained within this document.

Cyan Technology™, the Cyan Technology logo and Max-eICE™ are trademarks of Cyan Holdings Ltd. CyanIDE® and eCOG® are registered trademarks of Cyan Holdings Ltd. Cyan Technology Ltd recognises other brand and product names as trademarks or registered trademarks of their respective holders.

Any product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Cyan Technology Ltd in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Cyan Technology Ltd shall not be liable for any loss or damage arising from the use of any information in this guide, any error or omission in such information, or any incorrect use of the product.

This product is not designed or intended to be used for on-line control of aircraft, aircraft navigation or communications systems or in air traffic control applications or in the design, construction, operation or maintenance of any nuclear facility, or for any medical use related to either life support equipment or any other life-critical application. Cyan Technology Ltd specifically disclaims any express or implied warranty of fitness for any or all of such uses. Ask your sales representative for details.



Revision History

Version	Date	Notes
V1.0	14/05/2007	First release
V1.1	11/07/2007	Corrected part number in section 5.3.

Contents

1	Introduction	5
2	Glossary	5
3	ULPI Interface.....	6
3.1	VBus Generation	6
4	Internal PHY with OTG Support.....	7
4.1	MAX3355.....	7
4.2	AAT3125.....	7
5	Internal PHY as Host or Peripheral Only	8
5.1	Bus Powered USB Peripheral	8
5.2	Self Powered USB Peripheral	9
5.3	USB Host.....	10
Appendix A	Circuit Diagrams	11
A.1	eCOG1X and Support Components	11
A.2	ULPI and External High-Speed PHY	12
A.3	Internal PHY with OTG and Maxim MAX3355.....	13
A.4	Internal PHY with OTG and Analogic Tech AAT3125	14
A.5	Minimal Configurations	15

1 Introduction

This application note describes the various hardware configuration options available for use with the USB Peripheral in the eCOG1X microcontroller. It also describes the external hardware required for use with these example configurations.

A number of hardware configurations are available for the eCOG1X USB peripheral. The selection of the appropriate one depends on the performance and behaviour required by the application. Different internal USB hardware configurations also require different external hardware support, depending on which functions are required.

2 Glossary

A table of abbreviations used in this document.

ADC	Analogue to Digital Converter
eCOG1	Cyan Technology target microcontroller
USB	Universal Serial Bus

3 ULPI Interface

The ULPI interface option must be used if the USB peripheral is to be used at High Speed (480Mbps). An external PHY device is required for High Speed applications. Page two of the circuit diagrams shows a suggested implementation, using the SMSC USB3300 ULPI PHY.

Note that when acting as a peripheral, the USB Core does not support Low Speed operation (1.5Mbps).

3.1 VBus Generation

In combination with the ULPI PHY, the circuit must be able to supply the 5V VBus signal to the USB when it is acting in Host mode. Sheet two of the circuit diagram shows the example using a MAX5008 charge pump device for this purpose. This generates the VBus voltage from the system 3.3V supply, but is limited to a maximum output current of 125mA.

If a 5V supply is available, then an intelligent power switch such as the National Semiconductor LM3525M-H or the Micrel MIC2025-1 can be used (shown as IC12 on sheet five). In this case, the PWR_EN signal is connected to the CPEN output (pin 3) of the USB3300 and the PWR_FLG signal is connected to the EXTVBUS input (pin 10) of the USB3300.

4 Internal PHY with OTG Support

A number of options exist for supporting the internal USB PHY. The internal PHY supports only the Full Speed (12Mbps) and Low Speed (1.5Mbps) modes of the USB, but can also support the full On-The-Go USB functions with a small amount of external circuitry.

Note that when acting as a peripheral, the USB Core does not support Low Speed operation (1.5Mbps).

If the system is required to support USB On-The-Go, where it can be either the Host or the Peripheral, then an interface is required to monitor and supply the VBus signal.

Two example circuits for this are shown in the circuit diagrams. The first on sheet three uses the Maxim MAX3355 device, and the second on sheet four uses the Analogic Tech AAT3125.

4.1 MAX3355

The suggested circuit on sheet three requires a small amount of glue logic in the form of IC6, IC7 and IC8.

The charge pump VCC input (pin 2) can be driven from the 3.3V supply, but the current drive capability is quite limited. See the manufacturer's data sheet for more information.

To reduce power consumption when the USB is not being used, the Status input (Pin 11) can be driven low with a GPIO output from the eCOG1X. If this feature is not required, then the input pin should be tied to 3.3V.

4.2 AAT3125

The suggested circuit on sheet four requires a small amount of glue logic in the form of IC10 and IC11.

Unlike the Maxim part, the Analogic part does not provide an internal pull-up for the ID pin, so this is provided by R18. In addition, basic input protection by is provided by R17.

Power down mode is provided automatically by the EN input.

5 Internal PHY as Host or Peripheral Only

If the eCOG1X is being used with its internal USB PHY only as a host or only as a peripheral, then the interface can be very simple, and these circuits are outlined on the fifth sheet of the circuit diagrams.

Note that when acting as a peripheral the USB Core does not support Low Speed operation (1.5Mbps).

5.1 Bus Powered USB Peripheral

This is the simplest interface where the USB supplies the 5V system power and no monitoring of VBus is required.

Although the USB OTG signals are not used by this interface, they must be placed in a state that allows the USB Core to operate correctly. These signals can be driven internally using the **rg.usb.signals** register. This register allows the OTG signals from the OTG core to be monitored and controlled internally by the application code, without having to bring them out to port pins on the device.

For the Bus Powered Peripheral, the OTG signals need to be set to a fixed state:

OTG Input	State
vbusvalid	Force High
avalid	Force High
bvalid	Force High
sessend	Force Low
iddig	Force High

The register **rg.usb.signals** must be used to force all the signals to their correct states. This can be done with the following code at the start of the application:

```
rg.usb.signals = USB_SIGNALS_VBUSVALID_FORCE_MASK
                | USB_SIGNALS_AVALID_FORCE_MASK
                | USB_SIGNALS_BVALID_FORCE_MASK
                | USB_SIGNALS_SESEND_FORCE_MASK
                | USB_SIGNALS_IDDIG_FORCE_MASK
                | USB_SIGNALS_VBUSVALID_OUT_MASK
                | USB_SIGNALS_AVALID_OUT_MASK
                | USB_SIGNALS_BVALID_OUT_MASK
                | USB_SIGNALS_IDDIG_OUT_MASK;
```

5.2 Self Powered USB Peripheral

In this instance, the system has its own power supply and so must monitor the VBus signal to check when the system is connected to the host.

The status of the VBus signal must be relayed to the core by one of two methods. Either the OTG_vbusvalid and OTG_bvalid signals must be directly driven by the state of the VBus, or they can be driven internally through the register **rg.usb.signals**.

If the signals are to be driven internally through the **rg.usb.signals** register, then VBus can be monitored either with an analogue input to the eCOG1X, or with a GPIO input. Using a GPIO has the disadvantage that the switching level is not as well specified (unless external analogue comparators are used), but one advantage is that the GPIO Interrupt can be used to wake the system from a low power sleep state when it is first connected to the Host.

For the Self Powered Peripheral, the OTG Signals need to be driven as follows:

OTG Input	State
vbusvalid	VBus > 4.75V
avalid	Force High
bvalid	VBus > 4.00V
sessend	Force Low
iddig	Force High

If the OTG signals are driven directly, then the register **rg.usb.signals** must be used to force all the other signals to their correct states. This can be done with the following code at the start of the application:

```
rg.usb.signals = USB_SIGNALS_AVALID_FORCE_MASK
                | USB_SIGNALS_SESSEND_FORCE_MASK
                | USB_SIGNALS_IDDIG_FORCE_MASK
                | USB_SIGNALS_AVALID_OUT_MASK
                | USB_SIGNALS_IDDIG_OUT_MASK;
```

If the OTG signals are driven internally through the register **rg.usb.signals**, then it must be used to force *all* the signals to their correct states. This can be done with the following code at the start of the application:

```
rg.usb.signals = USB_SIGNALS_VBUSVALID_FORCE_MASK
                | USB_SIGNALS_AVALID_FORCE_MASK
                | USB_SIGNALS_BVALID_FORCE_MASK
                | USB_SIGNALS_SESSEND_FORCE_MASK
                | USB_SIGNALS_IDDIG_FORCE_MASK
                | USB_SIGNALS_AVALID_OUT_MASK
                | USB_SIGNALS_IDDIG_OUT_MASK;
```

The states of the vbusvalid_out and bvalid_out signals can then be controlled internally through the **fd.usb.signals.vbusvalid_out** and **fd.usb.signals.bvalid_out** bit field definitions.

5.3 USB Host

When the system is being used as a USB Host, then it must supply the VBus for the USB.

If a 5V supply is available in the system, then VBus can be powered continuously via a passive self-resetting fuse (such as a Raychem Polyswitch or similar) or through an active switch (such as the National Semiconductor LM3525M-H or Micrel MIC2025-1 as shown). The active switch has the advantage that the power consumption of the system can be controlled more carefully than with a permanent connection.

If a 5V supply is not available, then a charge pump can be used to generate the 5V VBus output from the 3.3V supply. An example of this is the Maxim MAX5008 device, shown on sheet two of the circuit diagrams.

With the active switch or charge pump device, the PWR_EN pin should be driven according to the state of the OTG_drvvbus signal. This can be done either by connecting it directly to the OTG_drvvbus signal, or by monitoring the state of the **fd.usb.signals.drsvbus_in** field and then driving the PWR_EN with a GPIO output.

VBus should be monitored and its state relayed to the USB core by one of two methods. Either the OTG_vbusvalid and OTG_avalid signals can be directly driven by the state of the VBus, or they can be driven internally through the register **rg.usb.signals**. If the signals are to be driven internally through the **rg.usb.signals** register, then VBus can be monitored either with an analogue input to the eCOG1X, or with a GPIO input. External analogue comparators may be used with a GPIO input for higher accuracy level sensing on the VBus signal if required.

For the USB Host, the OTG Signals need to be driven as follows:

OTG Input	State
vbusvalid	VBus > 4.75V
avalid	VBus > 2.00V
bvalid	Force Low
sessend	Force Low
iddig	Force Low

If the OTG signals are driven directly, then the register **rg.usb.signals** must be used to force all the other signals to their correct states. This can be done with the following code at the start of the application:

```
rg.usb.signals = USB_SIGNALS_BVALID_FORCE_MASK
                | USB_SIGNALS_SESEND_FORCE_MASK
                | USB_SIGNALS_IDDIG_FORCE_MASK;
```

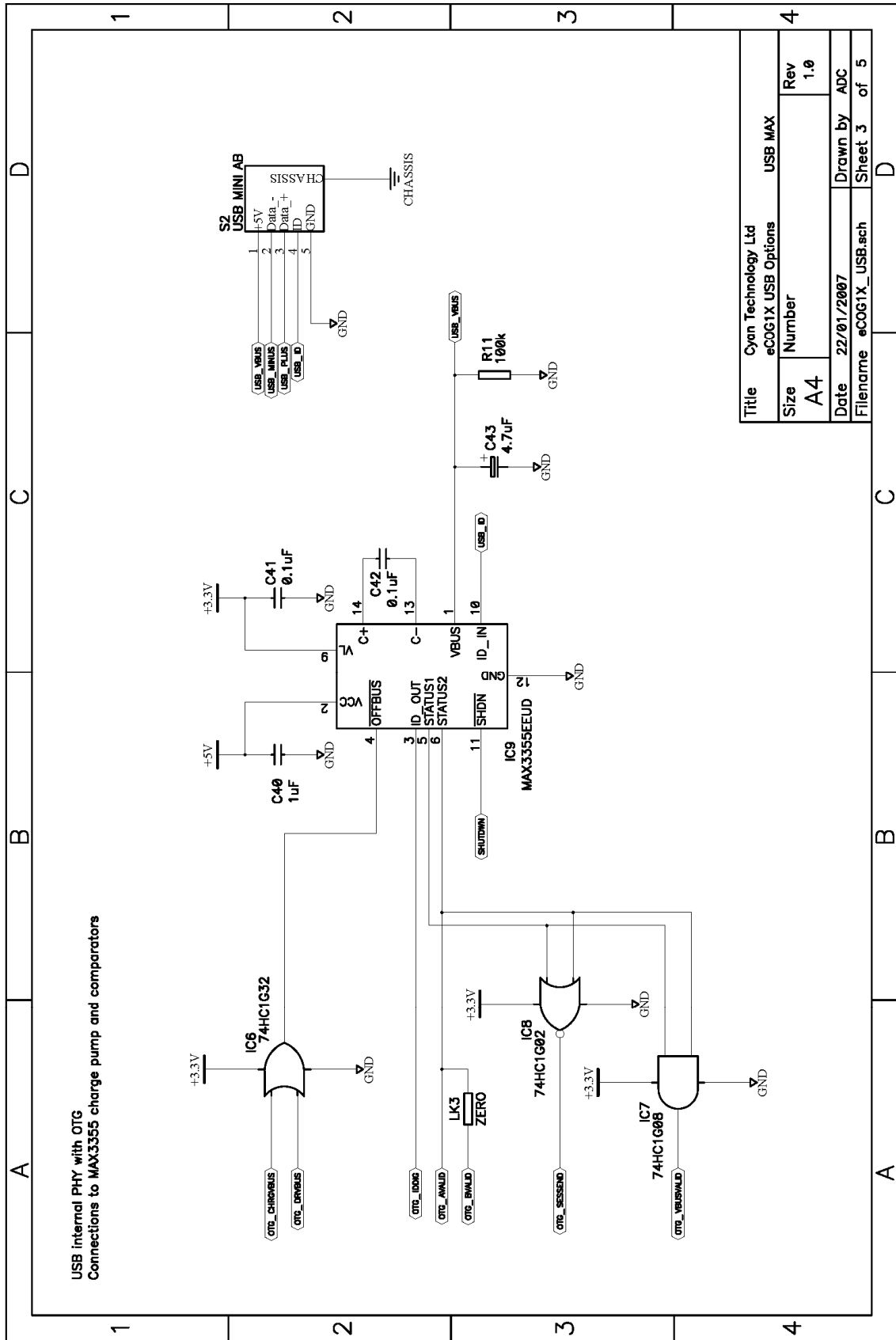
If the OTG signals are driven internally through the register **rg.usb.signals**, then it must be used to force *all* the signals to their correct states. This can be done with the following code at the start of the application:

```
rg.usb.signals = USB_SIGNALS_VBUSVALID_FORCE_MASK
                | USB_SIGNALS_AVALID_FORCE_MASK
                | USB_SIGNALS_BVALID_FORCE_MASK
                | USB_SIGNALS_SESEND_FORCE_MASK
                | USB_SIGNALS_IDDIG_FORCE_MASK;
```

The states of the vbusvalid_out and bvalid_out signals can then be controlled through the **fd.usb.signals.vbusvalid_out** and **fd.usb.signals.avalid_out** bit field definitions.

Additionally, the PWR_FLG signal can be monitored with a GPIO input to provide fault information to the eCOG1X.

A.3 Internal PHY with OTG and Maxim MAX3355



A.5 Minimal Configurations

