In4073 QR Operations Manual (2011-2012)

Arjan J.C. van Gemund

Embedded Software Lab
Software Technology Department
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

January 2011

1 Introduction

This document describes how to operate the Delft AeroVinci "quad-pilot 2 F.3" Quad-Rotor Aerial Vehicle (QR), of which 6 units are used in the MSc course in4073 Embedded Real-time Systems Lab [1]. The last section contains a list of very important safety guidelines that must be obeyed at all times.

2 QR Overview

The QR has a versatile architecture (Figure 1) that allows it run in different modes of operation (different FPGAs, tethered/wireless flying mode, all explained later). It features an Atmel AtMega1280 microcontroller (hereafter to be called "AtMega") that interfaces to the 4 engines, 3 accelerometers, and 3 gyros. The engines are brushless Robbe motors (Roxxy BL outrunner 2827-34), each of which is individually controlled by AeroVinci's proprietary v2 motor controller. The 4 controllers receive their RPM setpoints from the AtMega. The accelerometers are STMicroelectronics LIS344AL (+/- 3 g full scale range). The gyros are Invensense IDG500 (+/- 500 deg/s FSR). All sensors are filtered to avoid aliasing, allowing up to 1,300 Hz sampling frequency.

The QR also features an FPGA, which is a 100 MHz Trenz TE0300-01BMLP 1600K gates Spartan-3E industrial micromodule (hereafter called "TE0300"). The TE0300 runs the X32 soft core on which the control system is to be programmed. Both AtMega and TE0300 are shown in Figure 1.

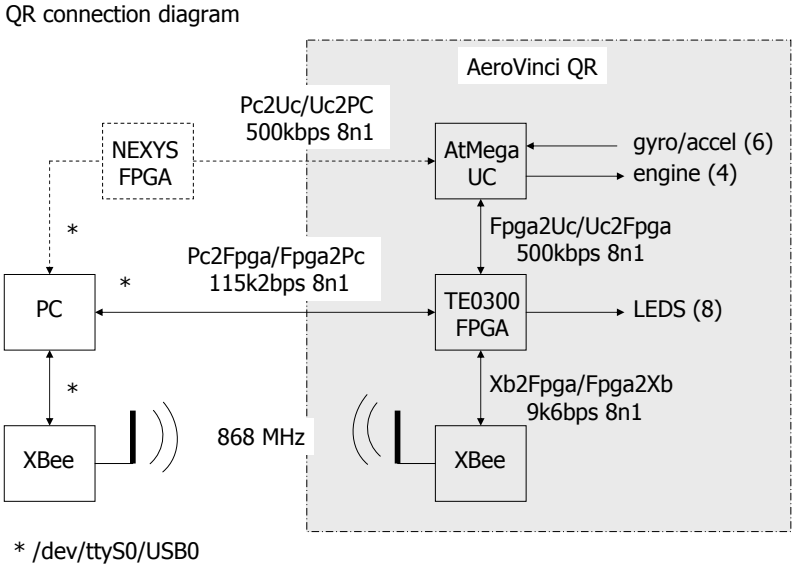

Figure 1: QR Hardware and Interfaces

The AtMega acts as interface between the engines and sensors and the X32 soft core. The QR architecture supports two FPGA modes of operation, i.e.,

- TE0300 mode: TE0300 FPGA on QR, or
- NEXYS mode: NEXYS FPGA board separate from QR.

Up to the 2008-2009 course editions the X32 resided on the external NEXYS board as the previous QR didn't have an on-board FPGA. From the 2009-2010 edition onwards the TE0300 FPGA is used to accommodate the X32, which allows the QR to be flown without a tether (wireless).

In TE0300 mode the X32 softcore resides on the on-QR TE0300 board. The AtMega communicates with the X32 through the duplex serial Fpga2Uc/Uc2Fpga link (RS232, 500 kbps, 8n1). The X32 communicates with the PC through either a wired duplex serial link (Pc2Fpga/Fpga2Pc, TTL, 115.2 kbps, 8n1) or a wireless duplex serial link (Xb2Fpga/Fpga2Xb, 9.6 kbps, 8n1), implemented by an Xbee-PRO 868 OEM RF module (868 MHz), depending on QR flying mode (wired link in tethered mode, wireless link in wireless mode). Since the low data rate of the wireless link prohibits fast uploading of X32 .ce object files, uploading and testing is performed in tethered mode. Only when the control design is fully validated, the QR can be flown in wireless mode.

In NEXYS mode the X32 softcore resides on the external NEXYS board. The AtMega communicates with the X32 through the duplex serial Pc2Uc/Uc2Pc link (500 kbps, 8n1, dotted line in the figure). The X32 communicates with the PC through a standard RS232 duplex serial link (115.2 kbps, 8n1)

Finally, in both FPGA modes the QR features a Pollin Electronics camera set (580 162) that transmits at 2.4 GHz on up to 4 channels (so 4 QRs can have a video link operating simultaneously). This allows a joystick operator a real-time view from the QR to assist him in maneouvering the QR (only permitted when the embedded roll/pitch/yaw controllers are fully validated). The camera can be turned on/off by bit 5 of the LED command sent to the AtMega (through the AtMega interface protocol described later on, via Pc2Uc or Fpga2Uc depending on QR FPGA mode).

The QR features the following switches and connectors:

- main power switch
- QR power inlet
- LiPo battery power outlet, to be connected to QR inlet
- Power inlet, to be connected to external power supply.
- 3×1 pins AtMega serial port (Pc2Uc/Uc2Pc, NEXYS QR link)
- 3×1 pins TE0300 serial port (Pc2Fpga/Fpga2Pc, TE0300 PC link)
- Xbee connector for the Xbee wireless module (Fpga2Xb/Xb2Fpga)
- 6×1 pins JTAG for TE0300 programming via JTAG Parallel cable
- 7×2 pins JTAG for TE0300 programming via Xilinx USB cable IV or equivalent
- 4-switch DIP switch for user configuration purposes
- 2×3 ISP connector for AtMega programming (used for maintainance only)

The TE0300 comes with a standard X32 flashed in ROM. Users may choose to upload a revised X32 design, e.g., extended with additional peripherals for filtering and/or controlling to off-load the X32's C program.

The QR board also accommodates 10 LEDs for diagnostic purposes. The 8 LEDs that are uniformly distributed along the perimeter of the QR PCB can be user-controlled from the TE0300 for application-specific purposes (the LEDS register if programmed via X32). The orange, and green LEDs signal incoming AtMega data communication activity on the Pc2Uc (NEXYS mode), and the Fpga2Uc (TE0300 mode) serial links, respectively. A burning LED signals incoming data whereas a blinking LED represents no data activity. While the AtMega sends sensor data on both links (Uc2Pc, Uc2Fpga) it selects between both links for incoming data (engine commands). When activity is sensed on one of both links, the AtMega automatically selects that link for input ("focus") while subsequently ignoring the other one. This focus is reset when activity has ceased for at least 0.5 s. By this scheme, the QR automatically refocuses itself between TE0300 mode and NEXYS mode without requiring any change in AtMega firmware or dipswitch settings.

3 Operating the QR

The QR is powered up by the on/off switch located on the interface board. The switch connects the engines and electronics to the on-board 3-cell 3000 mAh LiPo battery (Lithium Polymer). The AtMega will go through a startup sequence which includes initializing the 4 microcontrollers that control the brushless engines. This involves a short autonomous rotation of the 4 engines in sequence. After this, the QR is ready to be operated.

3.1 Power

In order to allow wireless flight, the LiPo is used which provides some 10 mins of power before the battery is depleted. In view of the limited battery capacity, most of the time the QR will be operated in tethered mode, where the QR is connected to an external DC power supply (PS) through a 2-wire power cable. The PS acts both as a source of power to the QR engines and electronics, and to constantly recharge the battery, who typically provides most of the power when the engines are running (the engines consume more than 99% of the QR power).

The QR can be operated in tethered mode or wireless mode.

- **Tethered Mode**
  Before operating the QR the PS must be turned on and must be connected to the QR terminals labeled "12.6 V MAX" (polarity-protected). The PS voltage must be close to 12.6 V (the nominal voltage of the LiPo battery when fully charged). The voltage must be adjusted such that the current flow to the LiPo is approximately 10 mA *when the battery is full.* When the engines run this current will increase as the PS will provide part of the current consumed by the engines. After the QR engines have run for a while the LiPo will have become partially discharged as the PS cannot provide *all* the current the engines need. As a result, after the engines have halted the PS current flow will not (yet) return to 10 mA but will stay at a higher level as the LiPo has (partially) been recharged and are consequently recharging. After a possibly long time, the current will drop to the original, 10 mA level as the LiPo becomes fully charged. NOTE: *Never* supply more voltage than (1) 12.6V, or (2) under *flying* conditions the PS current exceeds 5 A, or (3) such that *under zero engine RPM* conditions the PS current exceeds 500 mA. A LiPo is a potentially explosive device when overloaded and needs to be *handled with care* not to ignite! When a LiPo has been overcharged or undercharged it will be irreparably damaged, and can often become internally pressurized. *If the battery feels puffy or starts to balloon then immediately disconnect the battery from the QR and store it at a safe location for handling by the TAs!*

- **Wireless Mode**
  In wireless mode the PS is decoupled from the QR and the LiPo is the only source of energy. In order to prevent the LiPo from being too much depleted (which entails a very lengthy recharging period) the QR should not be flown for more than 5 mins before fully recharging. NOTE: too much draining of the LiPo can cause *considerable* damage to the battery.

3.2 Engines

The engines are brushless motors. Unlike ordinary motors that receive their electrical power through brushes, the brushless motors virtually don't wear, and provide much more efficiency. However, there is a trade-off. Controlling brushless motors is less straightforward than ordinary motors, and requires careful timing of the currents that flow into the various coils within the motor. Motor control is performed by the dedicated microcontrollers located close to each engine. While these microcontrollers relieve the main Atmel from generating all the time-sensitive signals, they cannot shield the programmer from one major issue: *stalling.* If the engines are ramped-up too fast, actual rotor RPM cannot keep up with the RPM setpoint, which may cause the engines to stall. This problem only occurs for low RPM as the controllers are optimized for the high RPM regimen where flight stability control is important. Consequently, the user is advised to slowly ramp up engine RPM until flight-level RPM is reached. Furthermore, when the RPM is controlled by software the RPM setpoint must not drop to a low level for the same reason. Thus, the software must clip its output to a lower bound (e.g., 100, see next section). Another reason for applying the lower bound is that in an emergency the engines are not to fully shut down as the QR will then crash land. Maintaining a low (and relatively safe) RPM will allow for a much softer landing, avoiding damage to the QR.

3.3 AtMega Interface Protocol

As mentioned earlier, in TE0300 and NEXYS mode the ATMega is connected to the TE0300 and the NEXYS, respectively. In the QR circuit diagram this link is denoted Fpga2Uc/Uc2Fpga (serial TTL) and Pc2Uc/Uc2Pc (serial RS232), respectively.

The interface is implemented by a duplex serial link operating at 500,000 bps using an 8-bits, no parity, 1 stop bit (8n1) protocol at the byte level. In TE0300 mode the green LED signals activity on the Fpga2Uc link, whereas in NEXYS mode the orange LED signals activity on the Pc2Uc link. When either link is operational the corresponding LED burns continuously, signalling that valid data is being received by the AtMega.

3.3.1 Sensors

At a frequency rate of approximately 1,270 Hz ($\approx$ 0.7 ms) the AtMega samples the sensors and also transmits to the FPGAs the 6 sensor values and battery voltage reading on both serial links Uc2Pc and Uc2Fpga, packaged in terms of 7 3-bytes packets, according to the format

```
1aaadddd 0ddddddd 0ddddddd
```

where 'd' denotes the sensor data and 'a' denotes an address bit, according to the following table. Rather than

address	sensor	address	sensor
000	LIS344AL accelerometer X	011	IDG500 gyro X
001	LIS344AL accelerometer Y	100	IDG500 gyro Y
010	LIS344AL accelerometer Z	101	IDG500 gyro Z
		110	LiPo battery voltage

Table 1: Sensor data format

receiving the individual bytes by the X32 software, a specific X32 peripheral has been implemented (in VHDL) to receive the entire 21 bytes frame, decode it, and present the sensor data to the C program in terms of 7 memory-mapped registers, thus avoiding the prohibitive overhead every 1,270 kHz frequent frame transfer (at 500,000 bps) would create for the X32 soft core. An interrupt is raised (`INTERRUPT_XUFO`) once the 7 new values are available to the X32. As an additional feature, there is also a counter register (`PERIPHERAL_XUFO_COUNT`) that holds the number of times a 7-value frame has been received (every 1/(1,270) $\approx$ 0.7 ms). In order to facilitate accurate sensor data logging, the counter register `PERIPHERAL_XUFO_TIMESTAMP` holds the 32 bit timestamp of the last received sensor data frame (in units of 20 ns, based on the X32's 50 MHz clock).

3.3.2 Actuators

Data transmitted from the FPGA (X32) to the AtMega through the same 500,000 baud 8n1 UART (X32 peripheral address `PERIPHERAL_XUFO_A`) also uses a 3 bytes format according to (MSB - LSB):

```
1aaaaaaa 00000ddd 0ddddddd
```

where 'a' denotes an address bit, and 'd' denotes actual data according to the following table. Note that no

address	actuator	data format
0000000	QR engine `ae1`	`0x0000` = off, `0x03ff` = full thrust
0000001	QR engine `ae2`	`0x0000` = off, `0x03ff` = full thrust
0000010	QR engine `ae3`	`0x0000` = off, `0x03ff` = full thrust
0000011	QR engine `ae4`	`0x0000` = off, `0x03ff` = full thrust
0000100	QR leds (4, currently not used) + camera	ddddd: leds 0-3 + camera

Table 2: Actuator data format

values above 0x03ff may be sent to the QR as this may result in erratic engine behavior. Given the lower bound mentioned earlier, this implies that RPM setpoints must be clipped at both sides. NOTE: a 10 microseconds time delay needs to be inserted between each consecutive byte that is sent to the uC in order for the AtMega firmware to keep up with the incoming data stream.

Similar to the sensor receiption case, a specific X32 peripheral has been implemented (in VHDL) to send an entire 12 bytes frame (4 engines, including the above-mentioned delays inbetween) to off-load the X32 (which otherwise would have to send all individual bytes through `X32_PERIPHERAL_A`). The engine setpoints appear to the C program in terms of 4 memory-mapped registers `PERIPHERAL_XUFO_A0` .. `PERIPHERAL_XUFO_A3`, respectively. Rather than reading the registers during QR (sensor) interrupt, the engine updates are communicated to the QR after writing to `PERIPHERAL_XUFO_A3`. Thus the X32 program (controller) is in charge, rather than the QR, which ensures that all engines receive consistent and timely information.

NOTE: the AtMega will automatically enter safe mode (all engines shut down) if no commands have been received within a *timeout window* of 250 ms. On the other extreme, don't send commands to the AtMega too often since repeating commands with the same RPM arguments doesn't make any difference, but *does* create severe (interrupt) processing overhead on the AtMega, which creates a potential stability hazzard.

3.4 Video Link

As mentioned earlier, the camera operates on either one of 6 channels (one per QR). this implies that each QR comes with a matching receiver. The receiver is connected to the Linux PC using a USB port. The camera is turned on/off by switch 0 of the QR board DIP switch. In addition, the camera can be turned on/off by bit 5 of the LED command sent to the AtMega (through the above interface protocol via Pc2Uc or Fpga2Uc, depending on QR FPGA mode). The video signal can be rendered on screen using `mplayer` which is involked by

```
$ mplayer tv:// -tv driver=v4l2:with=320:height=240:norm=PAL:input=1:device=/dev/video0:noaudio -vo x11
```

4 TE0300 Mode Operation

In TE0300 mode (the mode used at the in4073 lab), the X32 resides on the on-QR TE0300 FPGA. When initial, incoming data activity on the Fpga2Uc link is sensed by the AtMega it will automatically switch focus to this link while turning the green LED on, until a 0.5 s timeout has occurred.

As mentioned earlier, there are two communication modes between TE0300 and the Linux PC, i.e., wired and wireless, corresponding to whether the QR is operated in tethered or wireless mode. In tethered mode (the predominant experimentation mode), the TE0300 connects to the Linux PC through the 115,200 bps 8n1 RS232 PC link (Pc2Fpga/Fpga2Pc) in exactly the same way as the NEXYS board did (at either a native RS232 port at the PC, e.g., `/dev/ttyS0`, or via a USB2RS232 converter, e.g., `/dev/ttyUSB0`). In wireless mode, the TE0300 connects to the Linux PC through the Xbee transceiver link (Xb2Fpga/Fpga2Xb) which operates at 9,600 bps 8n1.

The standard X32 boot monitor I/O uses the Pc2Fpga port for uploading user programs. It is up to the user program to distinguish between tethered mode (continue using Pc2Fpga/Fpga2Pc) or wireless mode (use Xb2Fpga/Fpga2Xb). To this end X32 offers an additional Xb2Fpga UART to allow users to conveniently communicate via wireless in the same way as in the wired case, albeit at significantly reduced speed.

The Xbee modem is preset at one of the 6 numbered channels (one per QR), and thus needs a matching receiver at the PC side. At the PC side the matching Xbee transceiver module needs to be plugged in at a USB port (e.g., `/dev/ttyUSB1`). Despite the fact that the modems interface to the TE0300 on a duplex serial link of 9,600 pbs (8n1), the effective communication speed available is extremely low. This is due to the fact that the 868 MHz channel is legally constrained to 10% duty cycle in order to allow other (non-in4073) users to share the channel. While the modems have an air data rate of 24 kbps (approx. 2,000 bytes per second), due to the 10% duty cycle constraint, each modem is limited to sending only 200 bytes per second.

This implies that the wireless link can ONLY be used when the QR is propely controlled by the X32, such that it can be flown by limited joystick activity (requiring limited setpoint data transfer)! Also note that downloading the massive telemetry data off the QR can only be performed by the wired link. Consequently, the ideal operating sequence is: start QR in wired mode, upload X32 object file, test the controller, switch to wireless mode, test the controller, fly, record telemetry data on X32, touchdown, switch back to wired mode, download telemetry data.

5 NEXYS Mode Operation

In NEXYS mode the QR (AtMega, the TE0300 is not used in this mode) is connected to the off-QR NEXYS FPGA board where the X32 resides. As the FPGA board's PMOD connectors are 3.3V TTL level, an RS232 adapter has been implemented to interface to the QR's RS232 link Pc2Uc/Uc2Pc (which can sustain much higher data bandwidths than a simple, TTL link). The RS232 adapter board is plugged onto Pmod connector

JB (2nd from left on the NEXYS board when the Pmod connectors and the power jack are facing upwards), such that the RS232 D connector faces to the right.

As mentioned earlier, when the AtMega initially discovers incoming data from the NEXYS board, it will disable the Fpga2Uc link and switch focus to Pc2Uc link and will turn the orange LED from blinking to continuously burning until no data has been received for more than 0.5 s.

6 Safety Guidelines

Operating the QR implies that the user has read, understood, and accepts the following quad rotor safety guidelines.

The quad rotor "quad-pilot 2 F.3" is a complicated piece of equipment integrating engineering materials, electronics, mechanics, aerodynamics, high frequency radio and embedded software. Correct installation and operation are a must in order to prevent accidents from taking place. Operation of the quad rotor should be performed in a safe and responsible manner. Improper operation may result in serious bodily or property damage.

General safety guidelines:

- Keep the QR away from obstacles and people. The quad rotors are highly agile and can move at considerable speed. Therefore it is best to keep them as far away from other people as possible. Also, keep away from obstacles that are either expensive or accidentprone (such as high-tension lines).
- Keep the QR away from humidity. Humidity will affect the quad rotor's electronics and result in unpredictable behavior or a crash.
- If not an experienced model pilot, always fly under direct supervision of someone that is experienced: operating a quad rotor is difficult. Novices should first practice in a simulator and should only control the quad rotor under direct supervision of an experienced model pilot.
- Operate the QR with concentration. Do not operate a quad rotor if you are tired or in any other sense less acute than normally. Loss of attention can result in crashes and consequential damage.
- Keep the QR away from heat source. Heat may damage the quad rotor and impair subsequent use.
- Always check the quad rotor before flight. During flight, parts vibrate and may loosen, such as little screws, the battery connection, or the propeller-holding nuts. Verify before the flight that there are no loose parts, in order to prevent erratic behavior.

Also pay special attention to the following:

- The rotating blades: keep them away from other people and do not try to touch them as they can cause deep cuts. Wear safety goggles to prevent damage to your eyes.
- The landing rods: if the quad rotor undergoes a hard impact, the landing rods may break or come off. Again, it is important to keep the quad rotor away from people and wear safety clothes such as safety goggles if possible. The landing rods are made of fiber: do not touch them, since small splinters may come off.
- The LiPo batteries: if used wrongly they can ignite and even explode, for instance when overcharged. Do not use them outside the range of -20C to 60C, the voltage should never go below 3V per battery cell neither should it go above 4.2V per cell during charging. Avoid serious impact on the batteries and do not use sharp objects on it. Do not shortcircuit the battery's wires by, e.g., cutting them with scissors. Store the batteries at room temperature (19C-25C).

Finally, adhere to the following rules:

- Don't exceed the 12.6V external PS voltage.
- Don't exceed the 500 mA external PS current to the battery (i.e., when engines are at full stop).
- Don't exceed the 5 A external PS current to the battery when the QR is flying.
- Don't install the 3.3V jumper on the TE0300 micromodule as the QR already supplies 3.3V on the FPGA Bank 0 external 3.3V bus.

- The QR frame is not to be used as power GND since there is a 0.3mR MosFET protection in between, in order to protect against wrong battery polarity.
- Charge and discharge the LiPo battery with caution, adhering to the guidelines for 3-cell LiPo operation.
- The Xbee modems can be reprogrammed using Xbee's XCTU program that runs on the PC and connects to the modem via the Pc2Fpga link. However, do not reprogram the Xbee modems unless explicit consent has been obtained by the TAs.

References

[1] In4073 Course Web Site. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/index.html`.

[2] In4073 Assignment document. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/assignment.pdf`.

[3] In4073 Resources. `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/index.html`.