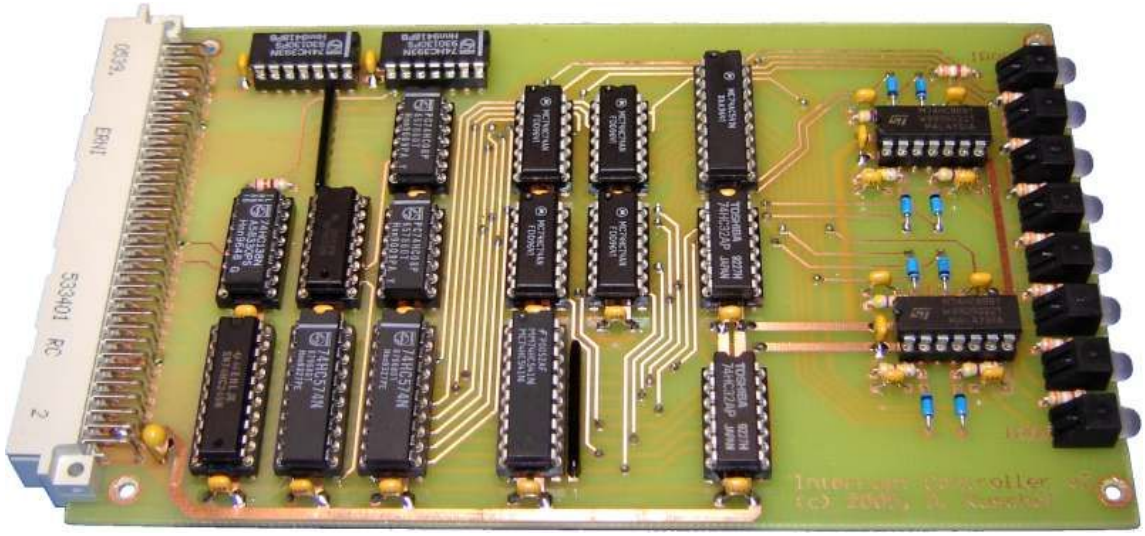
Interrupt - Controller

For the MyCPU

- Selfbuild Guide -



uses `/IOEN1:2100h-217Fh`

Dennis Kuschel · Corintostraße 21 · 28279 Bremen · Germany · www.mycpu.eu · dennis_k@freenet.de 2015-07-13

Content

Figures

Tables

Author:

Dennis Kuschel
Corintostraße 21
28279 Bremen
Germany

web : http://www.mycpu.eu
email: dennis_k@freenet.de

1 Boards

1.1 Interrupt Controller Board

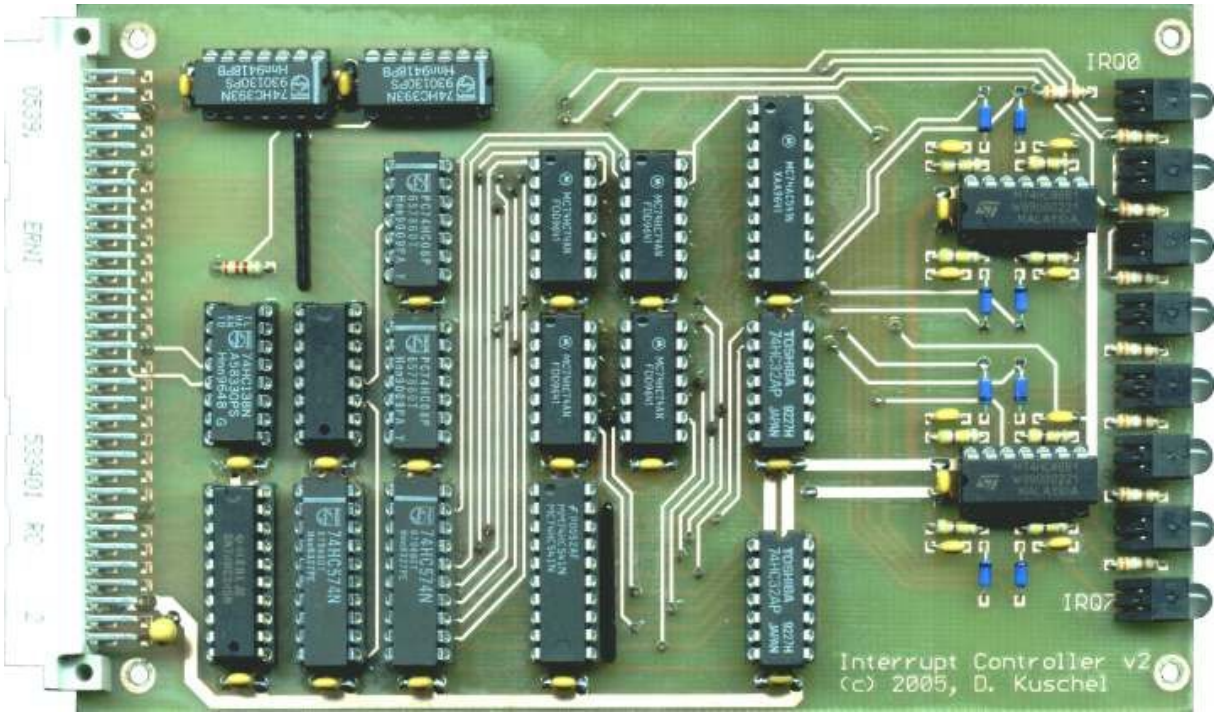


Fig. 1: Interrupt Controller Board

1.1.1 Description

The Interrupt Controller Board extends the interrupt capability of the MyCPU21. The Interrupt Controller provides 8 edge-triggered interrupt channels. The interrupts can be enabled on a per-channel basis. The first interrupt (IRQ0) is defined as timer interrupt. The board provides a timer interrupt with a timer rate of 61.035156 HZ (= 4MHz / 2^{16}).
The Operating System is responsible for the interrupt priorities. The provided MyCPU Operating System does not priotize the interrupts, thus all interrupts have the same priority (the interrupts are scheduled on a round-robin basis).

1.1.2 Selection of Components

No special parts are required. You should use the 74HCxxx types (74ACxxx types are not required). For IC18 and IC19 you MUST use 74**HC**86.

1.1.3 Placement of Components

After you have soldered all Via's, you can continue with the integrated circuits. I suggest you not to use sockets for the IC's because it is easier to solder the integrated circuits directly to the PCB. If you wish to use sockets anyway, you must use the high precision sockets. Only the high quality sockets allow you to solder the pads on the top side of the board. I strongly recommend you to follow the placement order I have noted in the placeplan below (see blue numbers, and start with the IC that has the blue number 1). When all IC's are placed and soldered, you can continue to place the capacitors. In the last round the resistors, diodes and board connectors are placed and soldered.

ATTENTION!
Please be careful, and don't forget to solder a pad on the top side of the board. I have marked all critical pads with red colour in the placeplan below. Please check if you have really soldered these pads!

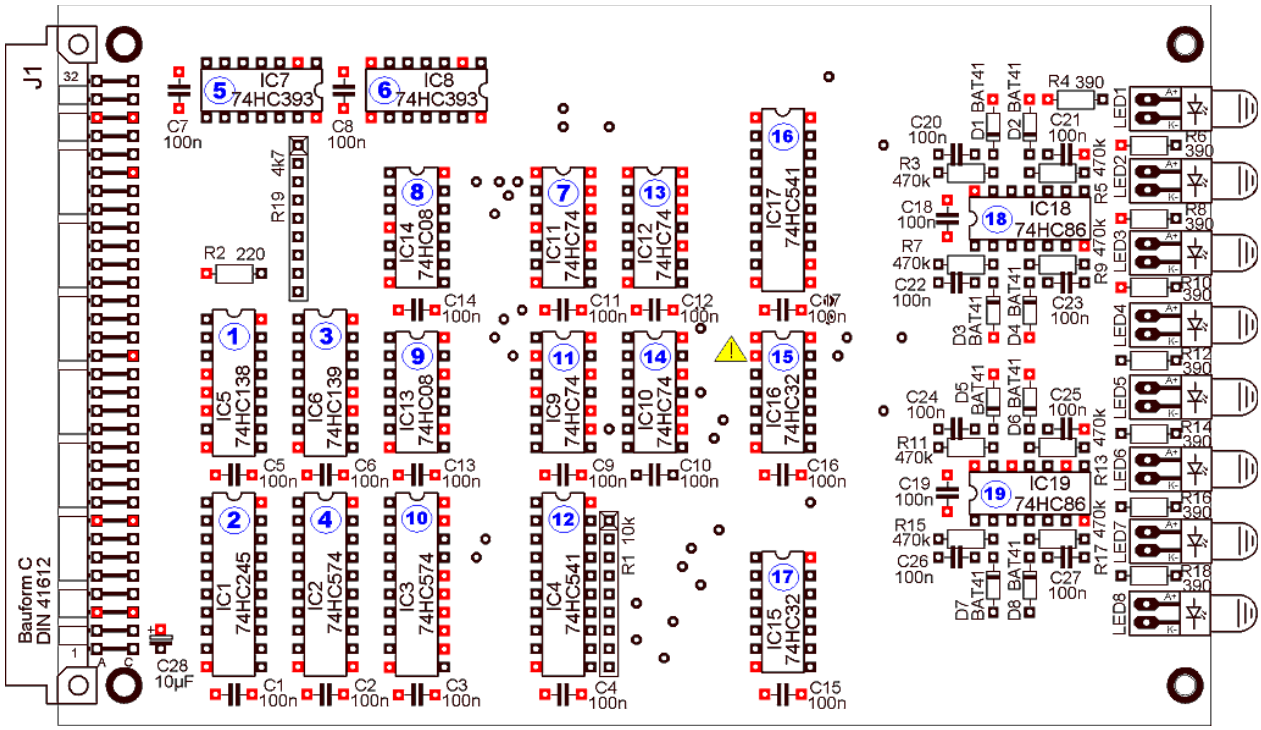


Fig. 2: Interrupt Controller Board - Placeplan

1.1.4 Partlist

74HC08	IC13, IC14	
74HC32	IC15, IC16	
74HC74	IC9, IC10, IC11, IC12	
74HC86	IC18, IC19	***MUST***
74HC138	IC5	
74HC139	IC6	
74HC245	IC1	
74HC393	IC7, IC8	
74HC541	IC4, IC17	
74HC574	IC2, IC3	
BAT41	D1 – D8	
Duo-LED, 2 wires red/green	LED1 – LED8	
220 Ohm	R2	
390 Ohm	R4, R6, R8, R10, R12, R14, R16, R18	
470 kOhm	R3, R5, R7, R9, R11, R13, R15, R17	
SIL 8 x 4.7 kOhm (8x 4k7)	R19	
SIL 8 x 10 kOhm	R1	
100nF ceramic capacitor	C1 – C27	
10µF / 16V, tantal	C28	
DIN 41612 Connector	J1	

2 First Test

2.1 Test Setup

For this test you need the CPU, the Memory Base Board, the Multi-/IO Base Board and the Interrupt Controller Board. No further software is required, please disconnect the MyCPU from a PC fileserver.

IMPORTANT: You need at least Kernel Version 2.4 for the Interrupt Controller Board. You can verify your kernel version by entering the "`ver`"-command at the MyCPU shell prompt. Please visit http://mycpu.mikrocontroller.net to get the latest software.

2.2 Running the Test

Switch on the power and enter the following commands:

```
8:/>basic

10 goto 10
run
```

Now observe the four LEDs on the Multi-I/O Base Board. They must still show a moving light that walks from right to left one step per second when a basic program is executed. You can repeat this test without having the Interrupt Controller installed. Without the Interrupt Controller the light will stop moving when the basic program is executed.

Explanation:
The Interrupt Controller provides a timer interrupt that is triggered approximately 61 times per second. The Operating System uses this timebase to change the state of the LEDs. When no Interrupt Controller is present, the OS executes a "pseudo timer interrupt" when it is idle, e.g. when it waits for user input. But when the CPU executes a basic program the operating system is no more in its idle loop and the LEDs do not change their state.

2.3 Interrupt Status LEDs

Each LED represents the state of one interrupt channel. A LED is turned on every time an interrupt request is pending (that means the appropriated interrupt wire on the connector is forced to high state / +5V). When the interrupt is served by the Operating System, the LED lights green. But when there is no device driver waiting for that interrupt, the LED lights red and the interrupt is ignored. You can test this by connecting pin A26 at the backplane connector with +5V (e.g. pin A30). The LED for interrupt 7 should light red now, whereas the LED for interrupt 0 should always light green, because interrupt 0 is the timer interrupt that is generally served by the Operating System.

Hint: For verification you can simulate this test setup with the MyCPU Emulator v4. The appropriated command line for the emulator is " `cpuemu4 -v3` ". But note that the CPU Emulator v4 does not show the LEDs. The switch `-v3` simulates a hardware configuration including the MyCPU, Memory Unit, Interrupt Controller and Multi-I/O-Unit.

3 Interrupt Assignment

The table shows the current interrupt assignment. Note that the interrupt 0 is a bit special: The interrupt 0 is hard wired with the timer-signal on the interrupt controller board. But you may supersede interrupt 0 with your own timer interrupt signal. For this purpose you need a driver that is capable of driving at least a current of 20mA. Please see R2 in the schematics for explanation. Note that also the wiring of IC13B is different from the other channels. Because of this interrupt 0 can only be triggered through a continuously rectangle signal.

Interrupt Number	Backplane Wire	Default Assignment
0	C23	Timer Interrupt (61.03515625 Hz)
1	A23	Keyboard Controller
2	C24	first UART / COM1:
3	A24	second UART / COM2:
4	C25	not yet assigned, but reserved for Operating System
5	A25	not yet assigned, but reserved for Operating System
6	C26	not yet assigned, free for your own use
7	A26	not yet assigned, free for your own use

Tab. 1: Interrupt Assignment

4 Interrupt Signal Timing

Because of the design of the Interrupt Controller a minimum pulse width is required on the interrupt inputs. The formula for calculating the minimum pulse width is

$p = \frac{1}{f}$ where p is the minimum pulse width in seconds and f is the CPU core frequency.

Thus a pulse width of 500ns is required if the CPU core is clocked with 2 MHz. Note that the interrupt inputs are edge-sensitive and an interrupt is triggered on the rising edge. See the figure below for explanation:

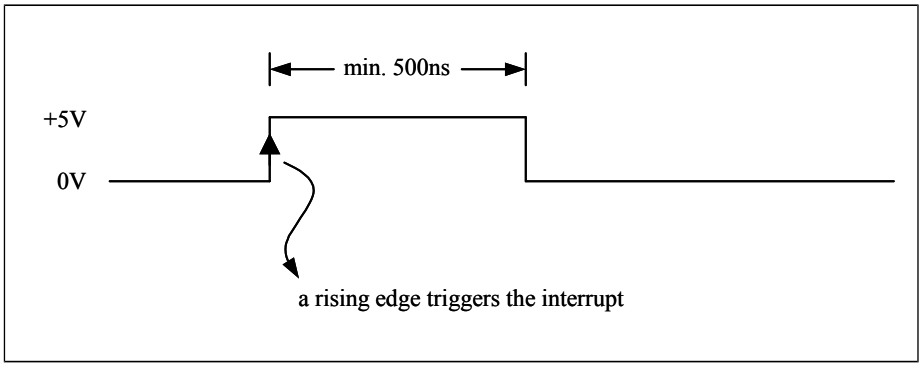


Fig. 3: A valid interrupt pulse

5 Registers

5.1 Overview

Address	Register Name	Read/Write	Function
2100h	REG_IC_INT	R	Interrupt Status (pending interrupts)
2100h	REG_IC_TRIGGER	W	Trigger all pending interrupts
2101h	REG_IC_IEMASK	R / W	Interrupt Enable Mask

5.2 REG_IC_INT

This register represents the status of the 8 interrupt inputs. If one bit is set in this status word, the appropriated interrupt is pending and waits for service by the CPU.

Address: 2100h (read only)

MSB(D7)							LSB(D0)
IS7	IS6	IS5	IS4	SI3	IS2	IS1	IS0

IS7 – IS0 Interrupt status. If one of this bits is set to 1, the appropriated interrupt is pending and waits for service. The status bit can be cleared by disabling the interrupt in the register REG_IC_IEMASK.

5.3 REG_IC_TRIGGER

It can happen that the first interrupt of a device gets lost after enabling the interrupt in the register REG_IC_IEMASK for that device. When the CPU writes to this register, all pending interrupts are re-fetched into the interrupt status register. The operating system writes to this register every time a driver requests a interrupt line for use. Note that interrupt 0 is not re-fetched when writing to this register.

Address: 2100h (write only)

MSB(D7)							LSB(D0)

5.4 REG_IC_IEMASK

Interrupt Enable Register. This register is used to enable/disable an interrupt and to clear an pending interrupt.

Address: 2101h (read/writeable)

MSB(D7)							LSB(D0)
EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0

EN7 – EN0 Interrupt enable bit. If the bit is set to 1, the appropriated interrupt is enabled and is signalled to the CPU when it is triggered. To clear a pending interrupt, set the appropriated bit in this register to 0 and then to 1 again.

6 LEDs

The LEDs are used to display the interrupt status on a per-channel basis. It is recommended to use duo-LEDs with two pins. The LEDs should be connected like shown in the figure below. The red LED will light when an interrupt request is pending but the interrupt is not enabled in the interrupt enable register. The green LED will light when the interrupt is enabled and will be served by the CPU. But it is also possible to use only a single LED for each channel. This is shown at the right side in the figure below. You may also use two separate LEDs and connect them like a duo-LED shown at the left side in the figure below.

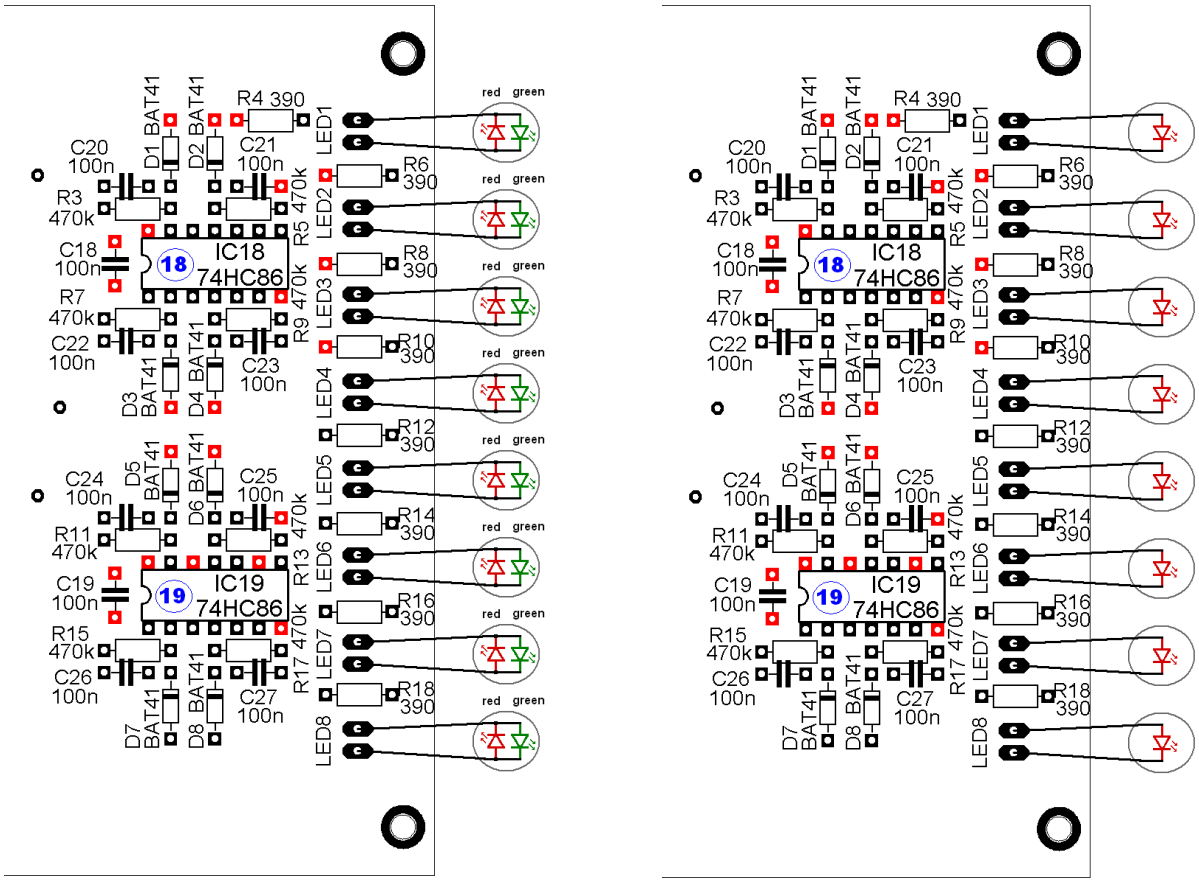


Fig. 4: Wiring of the Status LEDs

7 Schematics

On the following page you will find the schematics of the Interrupt Controller Board.
The PCB layouts can be found in a separate PDF file (see file IntCtrl_Layouts.pdf).

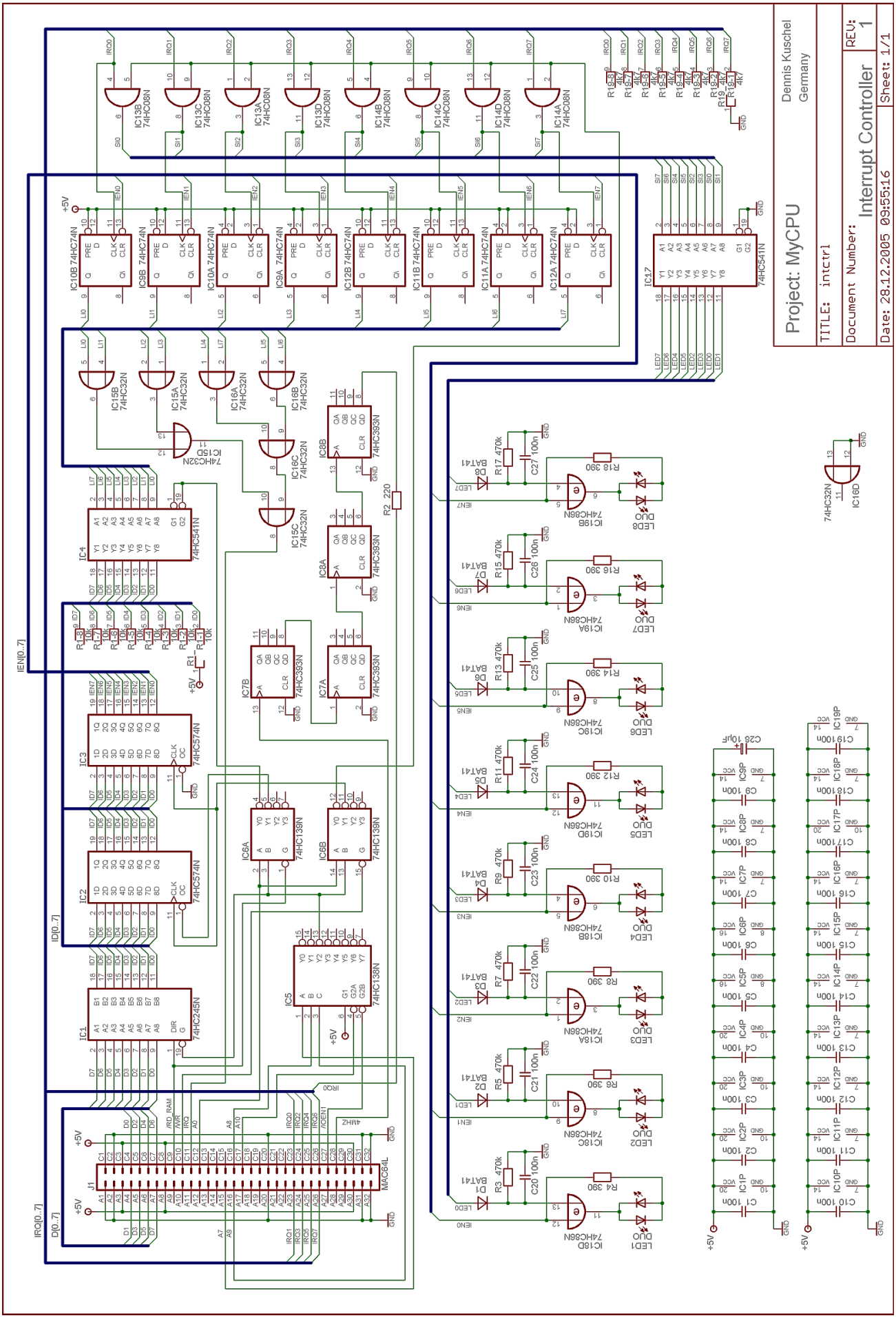


Fig. 5: Schematic of the Interrupt Controller Board

8 Change Log

8.1 Changes in the Interrupt-Controller design

Date	Name	Chapter	Description
2008-12-22	D.Kuschel	all	Document converted to OpenOffice format
2015-07-13	D.Kuschel		Document revised