



# SYSTECH J.Schnyder GmbH

Schliefweg 30  
CH-4106 Therwil  
**Telefon 091 827 15 87**  
www.systech-gmbh.ch

## HCS08-QD4TUT

**Beschrieb V 0.6 (Draft)**

Tutorial-PCB for the HCS08QD4 MPU

Inhalt / Content:

Shortform .....	<u>2</u>
Beschrieb / Description .....	<u>3</u>
Hardware .....	<u>3</u>
The power control section: .....	<u>3</u>
The LED's .....	<u>3</u>
The Software .....	<u>4</u>
First step: (The Basics) .....	<u>4</u>
Second Step: (The Improvement) .....	<u>5</u>
Third step: (The Enhancement) .....	<u>5</u>
Steckerbelegung / Pinout .....	<u>7</u>
Speisug / Power .....	<u>7</u>
externer Schalter / external Switch .....	<u>7</u>
externe LED0 / external LED0 .....	<u>7</u>
externe LED2 / external LED2 .....	<u>7</u>
externer LDR oder Termistor / external LDR or termistor .....	<u>7</u>
Programmier-Stecker / BDM Connector .....	<u>8</u>
Bestückungsplan / Component Placement .....	<u>9</u>
Stückliste / BOM .....	<u>10</u>
Schema / Schematics .....	<u>12</u>
Print / PCB .....	<u>13</u>
Links .....	<u>14</u>

## Inhalt

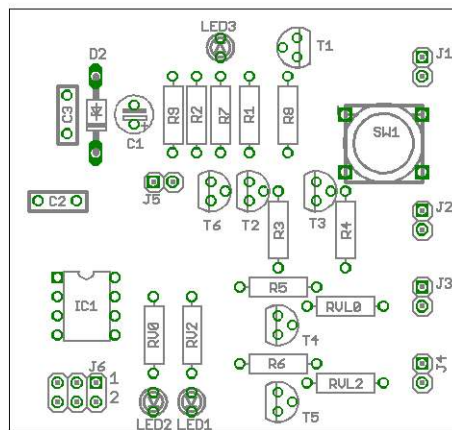
## Shortform



SYSTECH J.Schnyder GmbH

www.systemech-gmbh.ch

### HCS08-QD4PTUT HCS08QD4 tutorial PCB



- HC9S08QD4 MPU
- BDM-Conector
- Power indicator LED

Dimension: 60x56mm

- |    |                        |
|----|------------------------|
| J1 | Power                  |
| J2 | external switch        |
| J3 | external LED0          |
| J4 | external LED2          |
| J5 | external LDR/termistor |
| J6 | BDM interface          |

Version 1.0

## Beschrieb / Description

This is the tutorial PCB for the HCS08QD4 MPU

### Hardware

This PCB will show how to use a MPU in a circuit which is switched on by a pushbutton. The MPU then hold the power itself until the pushbutton is pressed a second time. This function was asked by a member of the Freescale forums and the PCB was designed to help on this topic. The PCB has some more functions so there are more possibilities to do some experiments.

For the basic function we need the switch on circuit and the LED3 to show if the MPU is working or not. Additionally LED2 can be switched on if the MPU is running. Since LED2 is under control of the MPU we could force LED2 to blink. The next step could be to control LED0 with the PWM output. It is possible to change the brightness of LED0 controlled by pressing the pushbutton for different periods (IE short pushes are switching the unit on and OFF; long pushes can change the brightness).

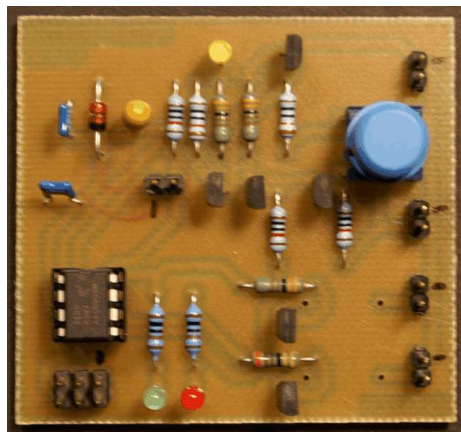
#### The power control section:

If a pushbutton (onboard or external) is pressed, a small current will flow through the base to the emitter of the transistor T3. This causes a higher current flowing through the collector of T3. This current, limited by R1, flows also through the base of the PNP transistor T1 which will conduct and therefore the VCC\_CPU is switched ON. The LED3 is ON in this case. If there would be no program in the MPU or the MPU would be missing releasing the pushbutton switch would also switch off VCC\_CPU. Assuming the MPU is programmed the program has to put the output PA3 (PTAD3) to HI (about VCC\_CPU). so the transistor T2 will conduct and VCC\_CPU will be switched on until the MPU puts the PA3 output to LO. The transistor T6 has the scope to signal if the pushbutton switch is pressed or not. If pressed the level on PA5 (PTAD5; input only) is LO otherwise HI. This input will be used in the program to decide if the MPU has to be switched off. This can be done by polling this input.

#### The LED's

Care should be taken when choosing the resistors RV0 and RV2 as well as RVL0 and RVL2. LED0 and LED2 are direct driven by the MPU and the led current should not exceed 5mA. For a 3V supply voltage these resistors should be greater than 330 Ohms and for a 5V supply greater than 680 Ohms. You can connect external led's on J3 and J4. This gives the possibility to use led's with higher currents i.e. power led's. Make sure to choose the right resistor for limiting the led current. With a 3v supply the current should not exceed 100mA and 150mA with a 5V supply. If higher currents are needed you may use an transistor with a higher DC gain (about 300).

$$RVLx = (V_{\text{supply}} - V_{\text{led}} - 0.3V) / I_{\text{led}}$$



## The Software

The base for programming is the EBS08C (the C version of the EBS08).

The project is called EBS08C\_QD4\_51\_TUT\_x, where the x stands for the different steps we will do.

The following in and outputs are used:

### INPUTS:

**PA1:** PTAD1     input can be used as input for a second switch or analog input for ADC1P1

**PA5:** PTAD5     input for sensing the pushbutton switch SW1

### OUTPUTS:

**PA0:** PTAD0     output for LED0 can also be used as PWM output for TIM1 CHANNEL0

**PA2:** PTAD2     output for LED2

**PA3:** PTAD3     output for holding the power switch on

### SPECIAL:

**PA4:** PTAD4     input/output used for BDM communication

### NAMES:

PA0:     LED0\_OUT  
PA1:     ANA\_IN  
PA2:     LED2\_OUT  
PA3:     POWER\_ON  
PA4:     ----  
PA5:     SW1\_IN

### Remark!

Currently (February 2007) the OSBDM is not included in the CW interface toolbar of the project. So Start with the *Full Chip Simulation*, then in the command window type *set gdi*. By browsing choose *osbdm\_s08.dll*. Then in the pull down menu *Component* select *Set Connection* and choose *HCS08 Open Source BDM*. The last step is to load the file by opening *Load* in the now named *HCS08 Open Source BDM* pull down menu. The *Project.abs* file can be found in the bin folder.

### First step: (The Basics)

This is the simple part:

If somebody presses SW1 the power of the MPU will be switched on. Then the MPU could set Bit 3

of PTAD to HI, keeping the power switched on even if the switch will be released. This method has the disadvantage, that if the circuit is switched on by a short impulse, caused by an external event, the MPU is turned on out of control and will therefore not be switched off by the user. To overcome this, we can make sure the user presses the switch for at least 100ms for switching the unit on. For switching the unit off we can define a minimum switch time of 200ms.

Since the software is based on the simple OS EBS08 we only have to write our program for the task manager. The timing with the flags is done by the OS.

First some flags in the TASK\_F1 variable are defined. As can be seen in the main.c file of the *EBS08C\_QD4\_51\_TUT\_1* project there a flag named F\_PWRON is defined in the TASK\_F1 byte. To ease programming we define the TASK\_F1.F\_PWRON simply as F\_PWRON and this flag will be set if the power is held by the MPU. The Z\_SW1 variable is the counter for the time the switch is operated.

This part of the program does simply look if the 4ms flag (CORE\_F0.F\_TIC4) is set. If so there are two possibilities: The MPU was waked up and therefore the F\_PWRON flag is cleared or the power is switched already held on by the MPU so the flag is set.

If the flag is cleared we look if the switch is pressed, a logical LO on the SW1\_IN means the switch is pressed, a logical HI means the switch is released. If the switch is pressed, we are incrementing Z\_SW1 until this counter reaches a number of 25 ( $4\text{ms} \times 25 = 100\text{ms}$ ). By reaching 25 we are setting the variable POWER\_ON to 1 as well as the F\_PWRON flag. So the MPU is holding the power for operating.

If the F\_PWRON flag is set we do waiting if the switch is pressed for at least  $50 \times 4\text{ms}$ . If we reach 200ms then the F\_PWRON flag is cleared and the variable POWER\_ON is set to 0. so the power is switched of. We then are entering an endless loop feeding only the watchdog to make sure not causing a reset.

By downloading to the board you can test the program.

It works, but has a handicap! you have to press the switch for a moment, but this moment has to be really short. If you do press to long the LED3 will not stay on after releasing the switch; why?

It is simple: when the program has seen the switch on for 100ms it holds the power as we have been expecting, but after that the program checks the switch again and releases the power after 200ms! Not really user friendly this way. We have to improve that.

### **Second Step: (The Improvement)**

If we where waiting until the switch is released, we can overcome the problem of *EBS08C\_QD4\_51\_TUT\_1*. So have a look at *EBS08C\_QD4\_51\_TUT\_2*, wher you can see how it is realized.

The original problem from a member of the Freescale forum is solved this way, but the MPU is only used to hold its power until it is switched off. Adding some LED's and changing the program a bit, we could do a lot more!

### **Third step: (The Enhancement)**

As told above: With a LED (LED2) and some changes of the software we can enhance the unit with multifunction switch!

In the *EBS08C\_QD4\_51\_TUT\_3* project, the following functions are realized:

By pressing SW1, the MPU gets started and after having initialized all the components needed, checks if the switch is pressed for at least 100ms ( $25 \times 4\text{TIC's}$ ). If so the LED2 is turned ON. Every additional long push (500ms and more) on SW1 toggle's the LED2 between blinking and continuous mode. Any short push on SW1 ( $>100\text{ms}$ , but  $< 500\text{ms}$ ) will turn off the MPU.

To realize this some more flags have been introduced:

F\_SW1LG        (set if the switch is pressed at least 500ms)  
F\_SW1NW        (set if SW1 is newly pressed)  
F\_LEDBL        (set if LED2 should blink)

You may have a look at the code to get an idea how it works.

The next step will be to include a dimming function with the PWM modulator of TIM1!

(Coming soon)

## Steckerbelegung / Pinout

### Speisug / Power

J1 (max. 5V!)

1	+VBAT
2	GND

### externer Schalter / external Switch

J2 Druckschalter / Pushbutton

1	NO
2	COM

### externe LED0 / external LED0

J3

1	+ Anode
2	- Kathode

### externe LED2 / external LED2

J4

1	+ Anode
2	- Kathode

### externer LDR oder Termistor / external LDR or termistor

J5

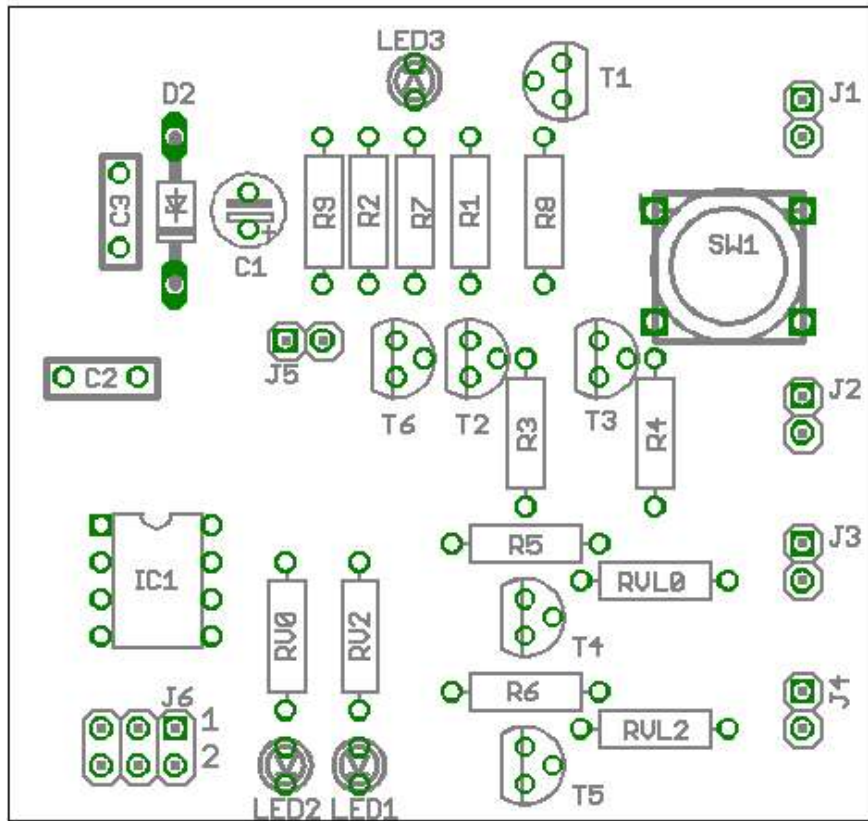
1	A
2	B

## Programmier-Stecker / BDM Connector

J6 (BDM)

1	BKDG	2	GND
3	--	4	/RESET
5	--	16	VCC

# Bestückungsplan / Component Placement

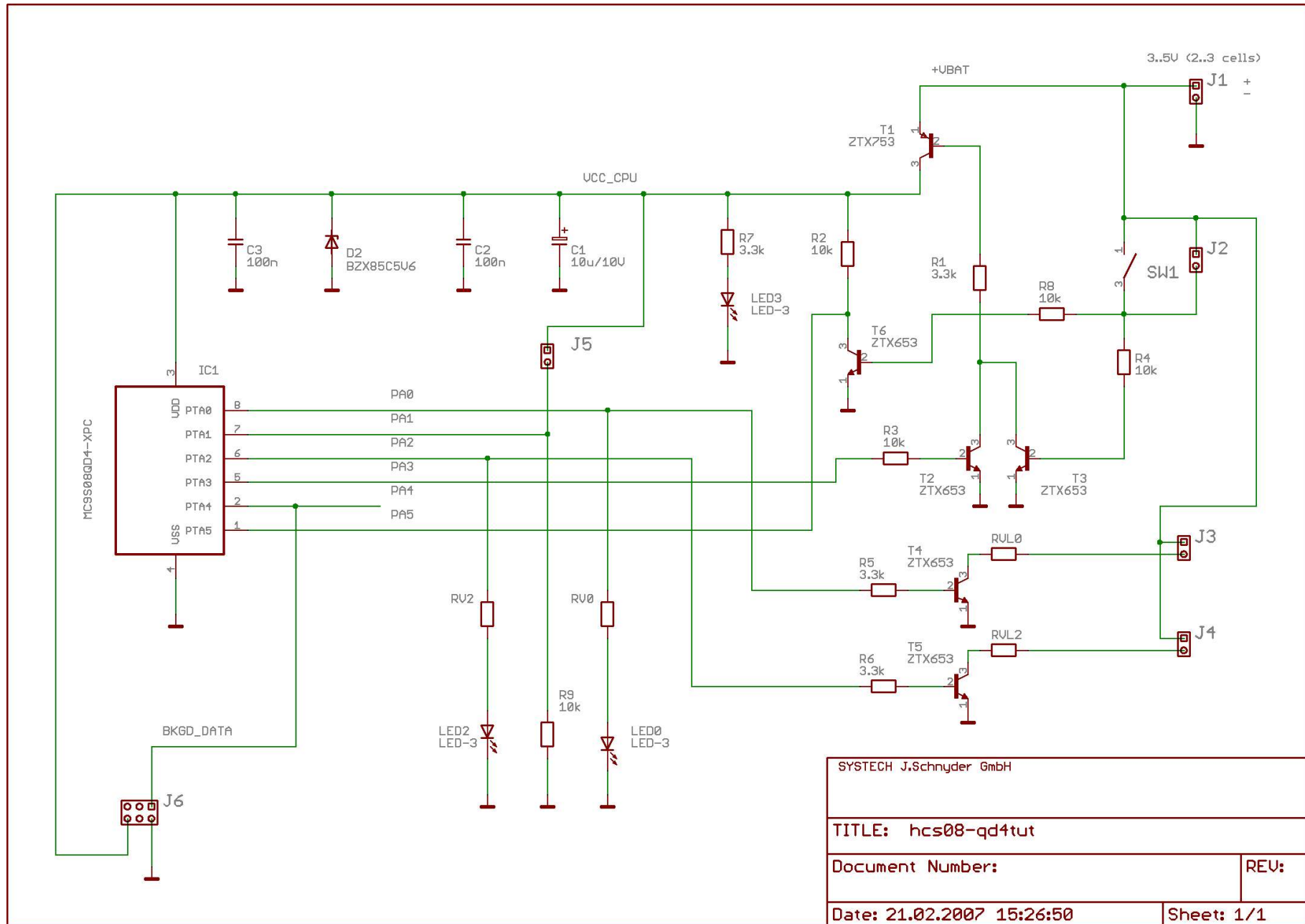


## Stückliste / BOM

HCS08-QD4TUT		BOM V 1.0			2007.02.21 SYSTECH			
Name	Type	Value	Case	Distrelec	Farnell	Digikey	Other	Remarks
C1	E-2.5-10U-10V	10U	E-2.5	802324				
C2	C-5-100N	100N	C-5	830284				
C3	C-5-100n	100N	C-5	830284				
D2	BZX85C5V6	5V6	DO41	602185				
IC1	MC9S08QD4CPC		DIL8	122096/18			Freescall	
J1	ST-1X2			6783.11111				
J2	ST-1X2			6783.11111				
J3	ST-1X2			6783.11111				
J4	ST-1X2			6783.11111				
J5	ST-1X2			6783.11111				
J6	ST-2X3			10175.5				
LED0	LED3-GN	green	LED3	254592				
LED2	LED3-RT	red	LED3	254588				
LED3	LED3-GE	yellow	LED3	254595				
R1	R-10-3K3	3K3	R-10	711706				
R2	R-10-10K	10K	R-10	711718				
R3	R-10-10K	10K	R-10	711718				
R4	R-10-10K	10K	R-10	711718				
R5	R-10-3K3	3K3	R-10	711706				
R6	R-10-3K3	3K3	R-10	711706				
R7	R-10-3K3	3K3	R-10	711706				
R8	R-10-10K	10K	R-10	711718				
R9	R-10-10K	10K	R-10	711718				

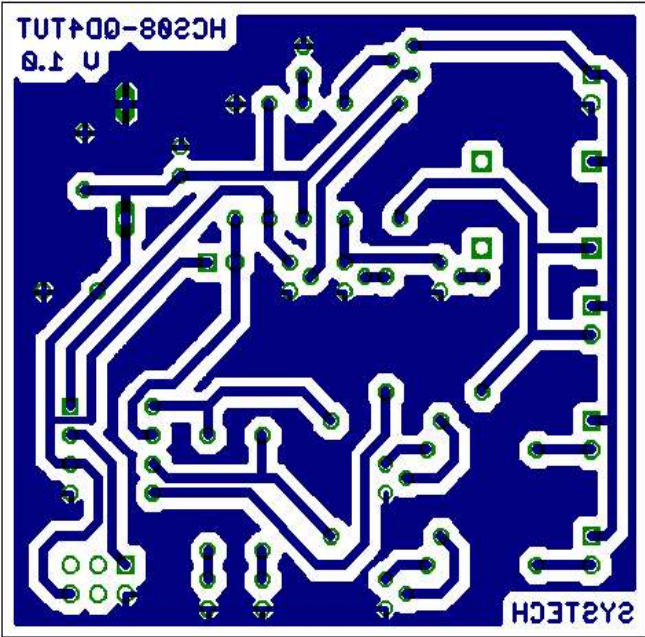
RV0	R-10-1K	1K	R-10	711692				
RV2	R-10-1K	1K	R-10	711692				
RVL0								to be calculated
RVL2								to be calculated
SW1	3FTL6		3FTL6	200330				
T1	ZTX753		ELINE	612934				
T2	ZTX653		ELINE	612933				
T3	ZTX653		ELINE	612933				
T4	ZTX653		ELINE	612933				
T5	ZTX653		ELINE	612933				
T6	ZTX653		ELINE	612933				
PCB	HCS08-QD4TUT						SYSTECH	

# Schema / Schematics



SYSTECH J.Schnyder GmbH	
TITLE: hcs08-qd4tut	
Document Number:	REV:
Date: 21.02.2007 15:26:50	Sheet: 1/1

Print / PCB



Print bottom

## **Links**

**Distrelec**

[www.distrelec.com](http://www.distrelec.com)

**Bauteile und mehr**

**Systech J.Schnyder GmbH**

[www.systech-gmbh.ch](http://www.systech-gmbh.ch)

**Entwicklung von Hard- und Software, Schulungs-Systeme  
Layout-Programme**

**Freescale forums**

[forums.freescale.com](http://forums.freescale.com)