

Portierung: mecrisp-stellaris auf PSoC-5LP von



...in drei Schritten

1. Der Controller – warum der?
2. Die Portierung – Treppauf und -ab
3. Das Evaluation-Board – ganz praktisch

Mecrisp: Who is Who

6 x ARM Cortex M3

LM4F120H5QR / TM4C123GH6PM
TM4C1294NCPDT
MSP432P401R
STM32L053C8T6
STM32L152RE
STM32F207ZG
STM32F303K8
STM32F401RE
STM32F411RET6
STM32F051R8
STM32F100RB
STM32F103C8T6
STM32F303VCT6
STM32F407VGT6
STM32F429ZIT6

KL25Z128VLK4
KL46Z256VLL4
MK64FN1M0
XMC1100Q024F0064
EFM32GG990F1024
EFM32HG322F64
LPC1114FN28
STM32F030F4
MK20DX256VLH7
LM4F232H5QC
nRF51822
STM32L152RBT6
STM32L476VG
STM32F746NG

Stand: mecrisp-stellaris-2.4.1

PSoC Programmable System on Chip

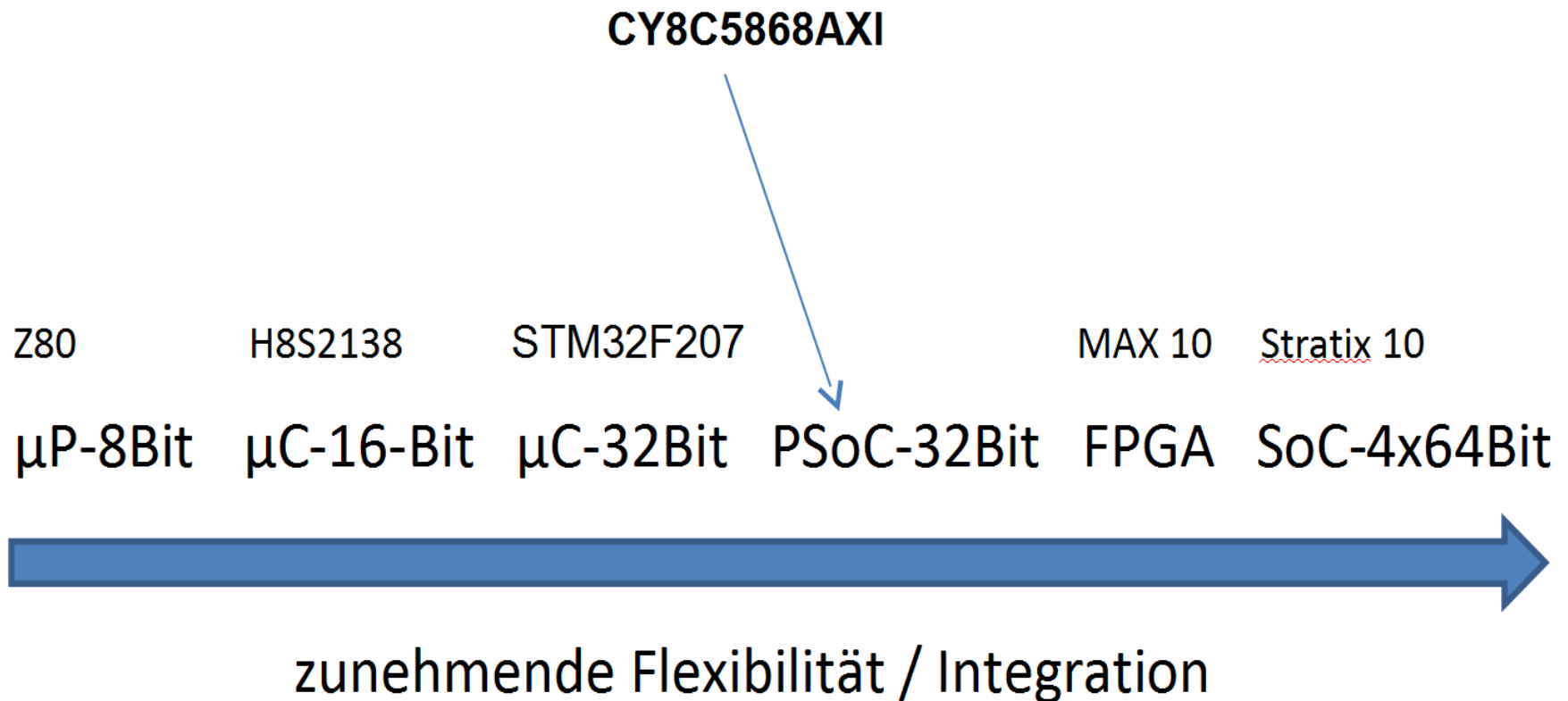


Figure 1-1. Simplified Block Diagram

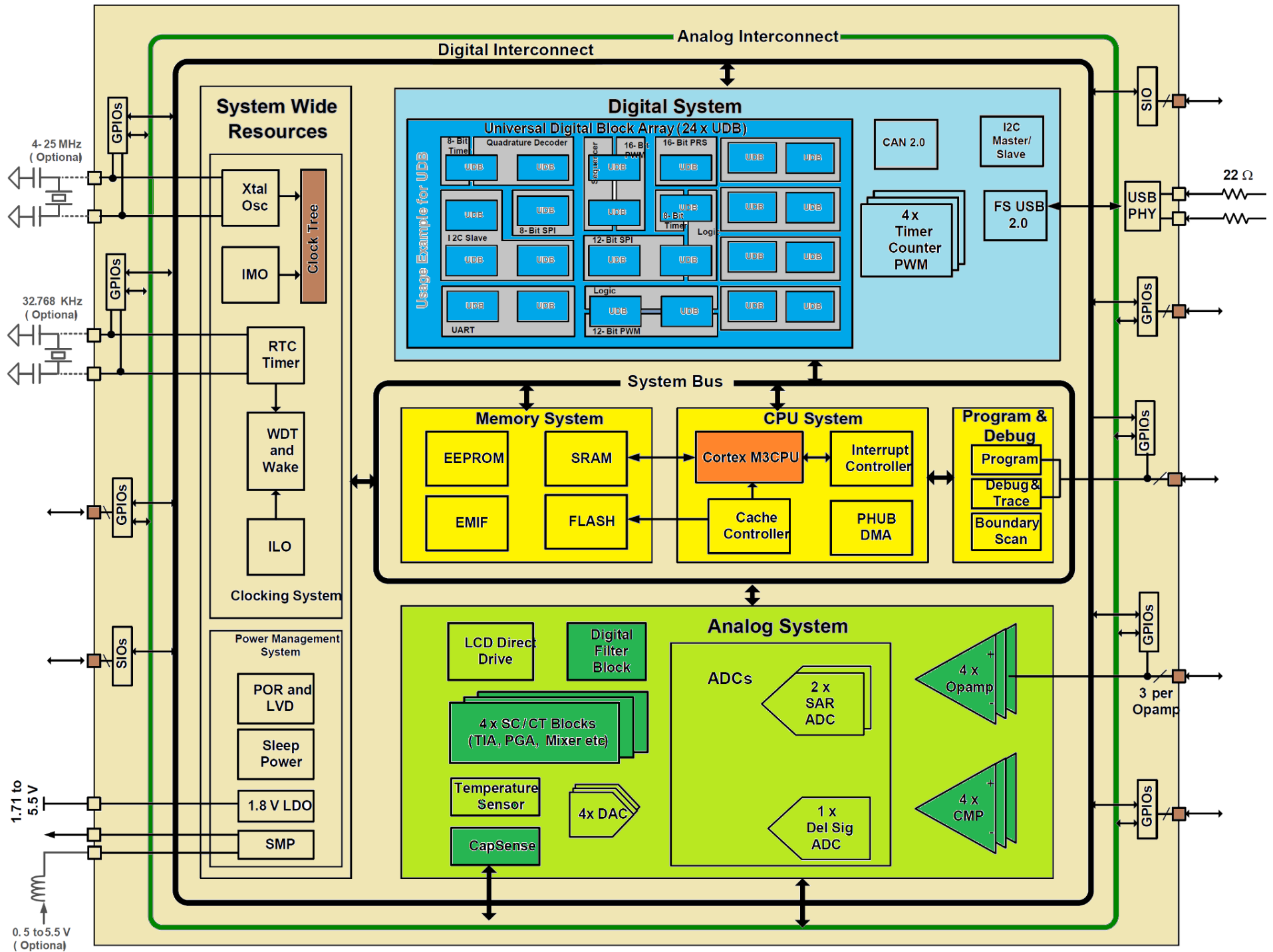


Figure 21-1. UDB Block Diagram

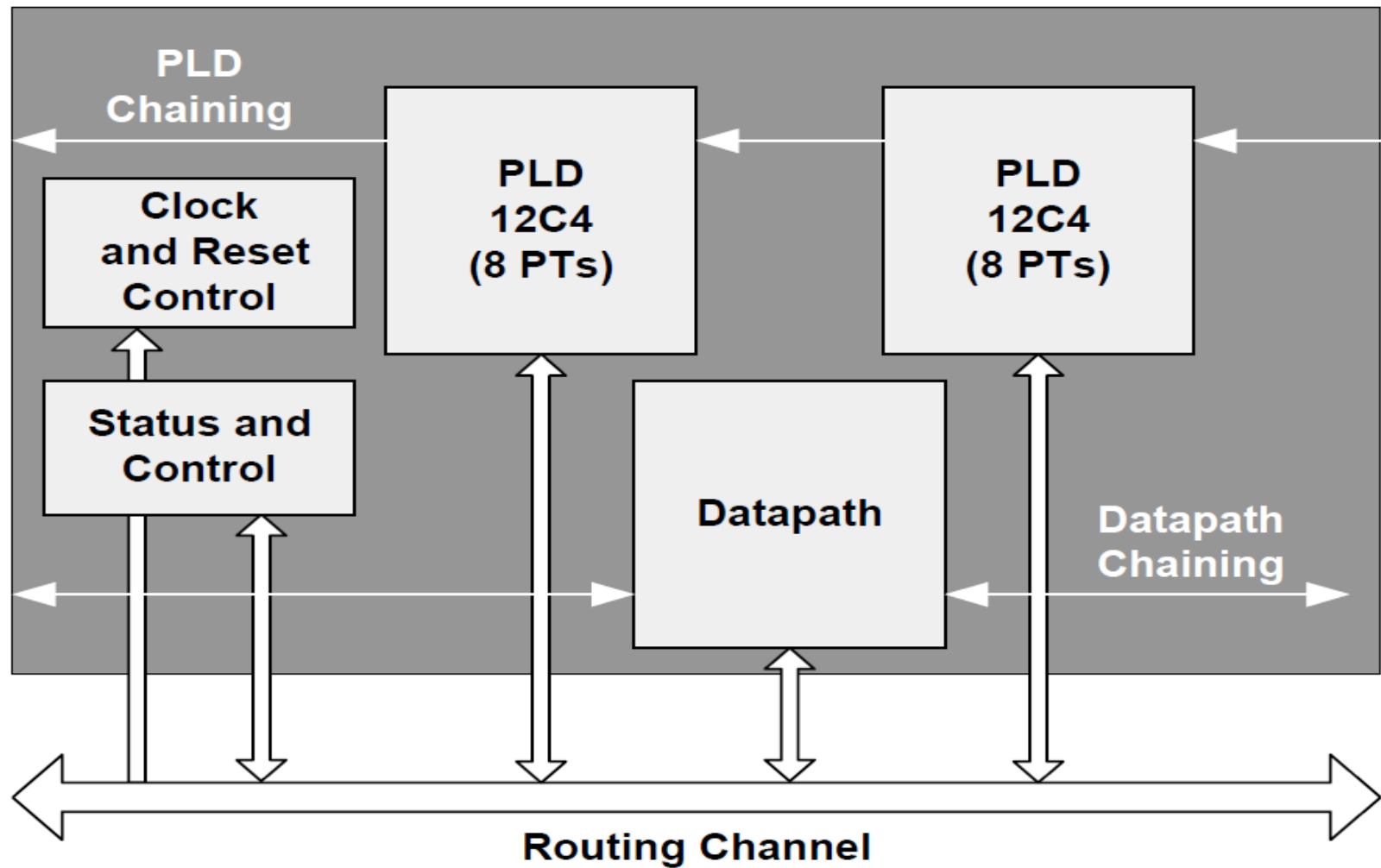


Figure 27-1. Digital Filter Block Diagram

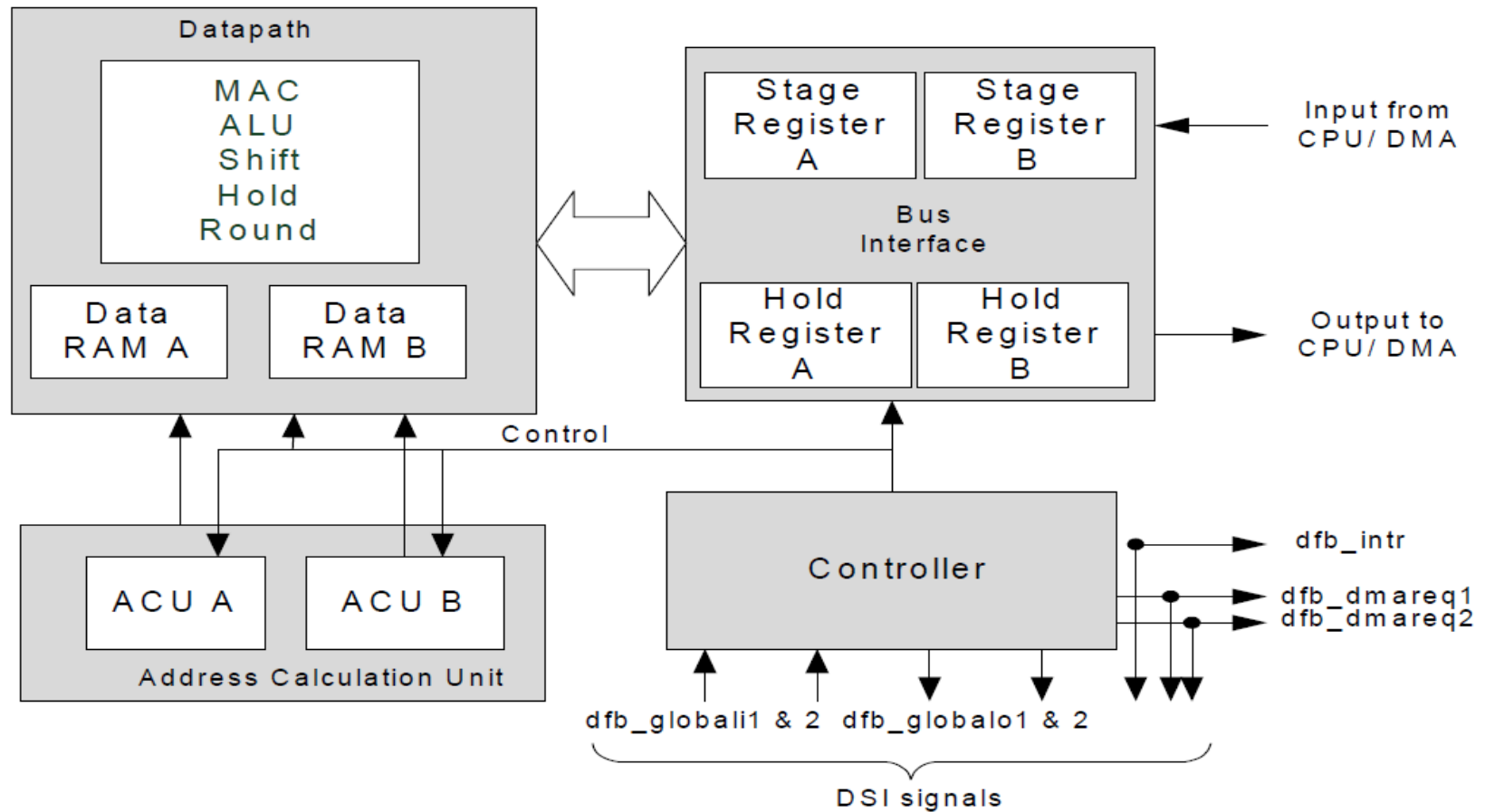


Figure 29-10. Switched Capacitor Routing, Interface

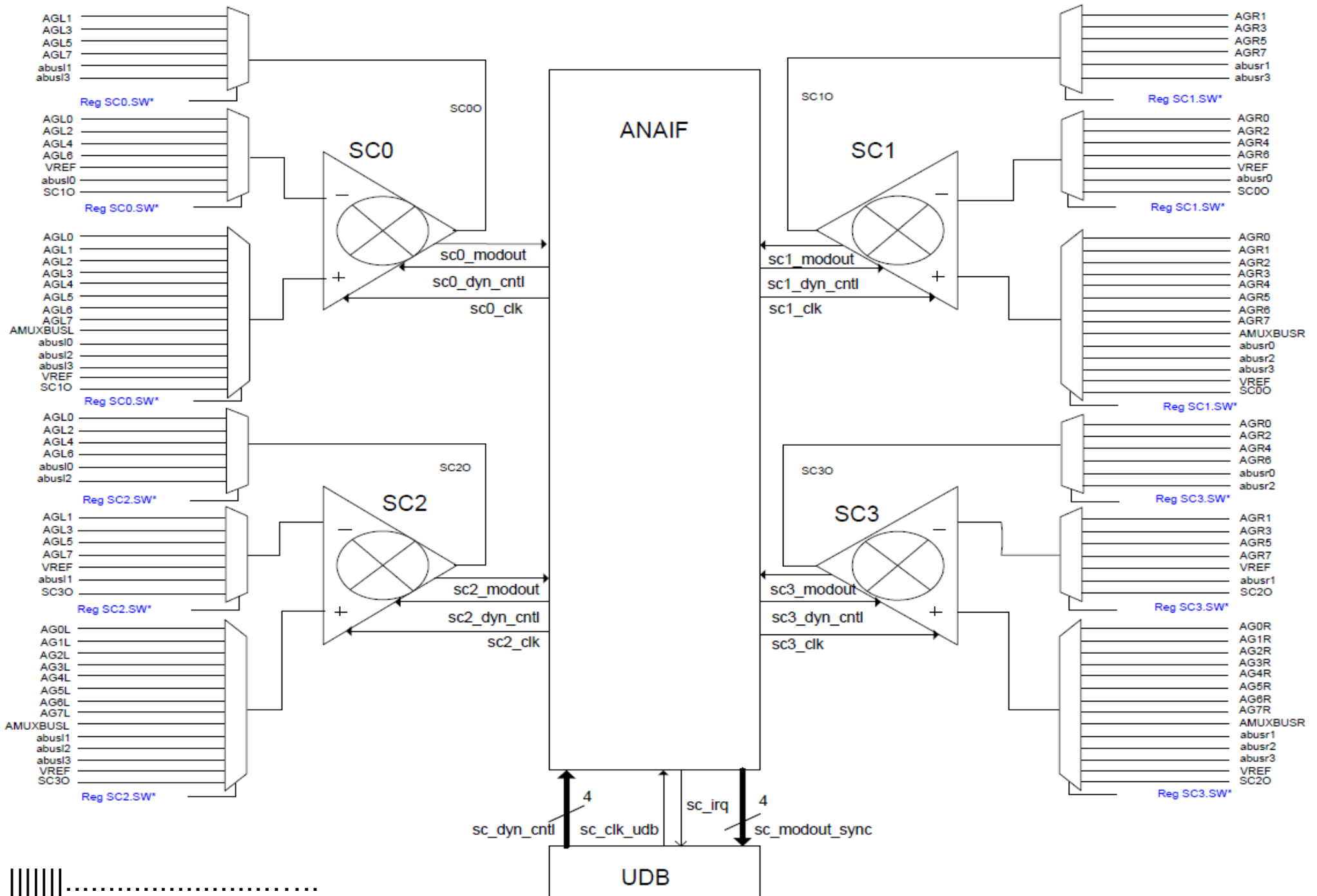


Figure 29-2. Analog Interconnect

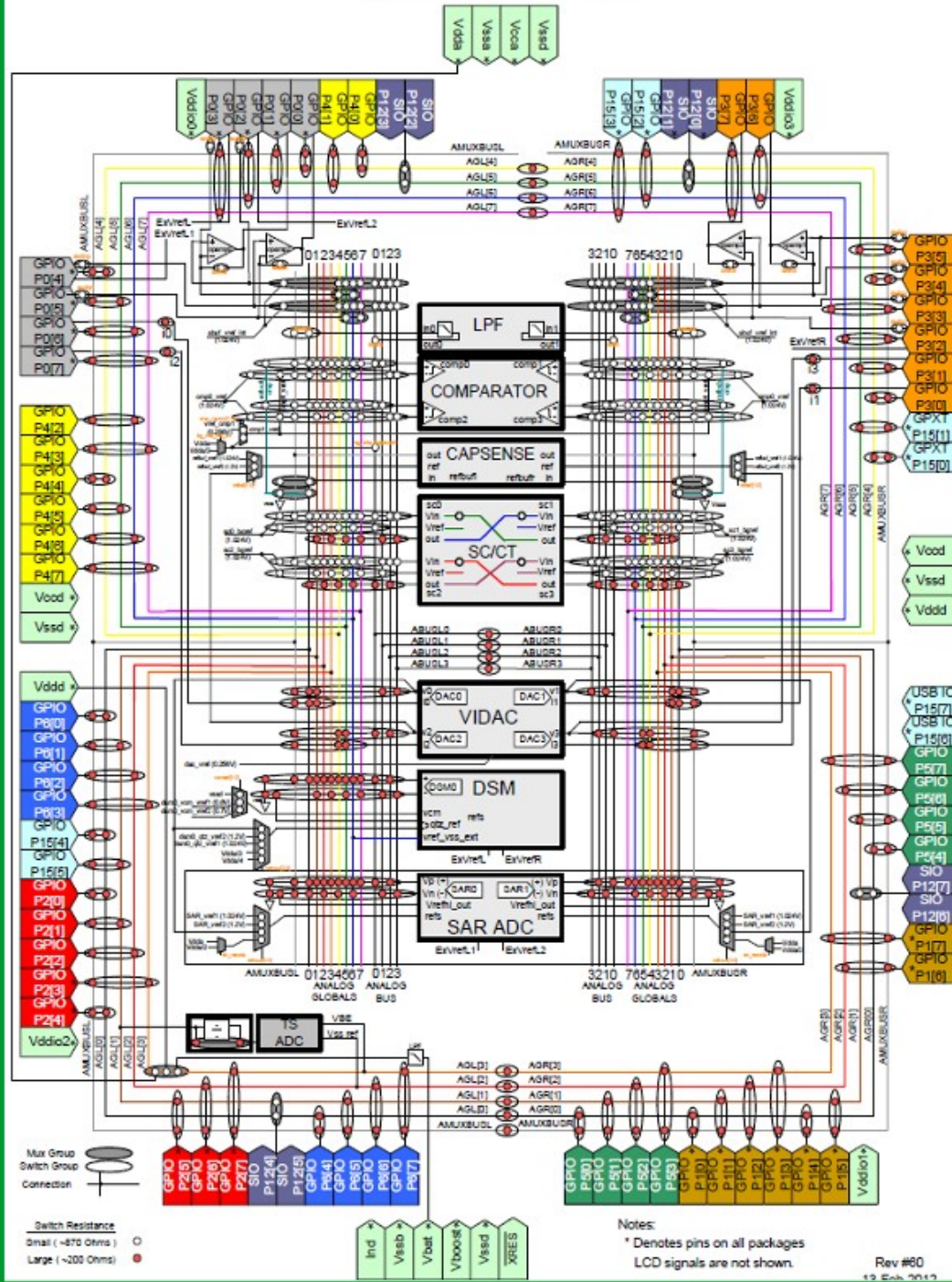
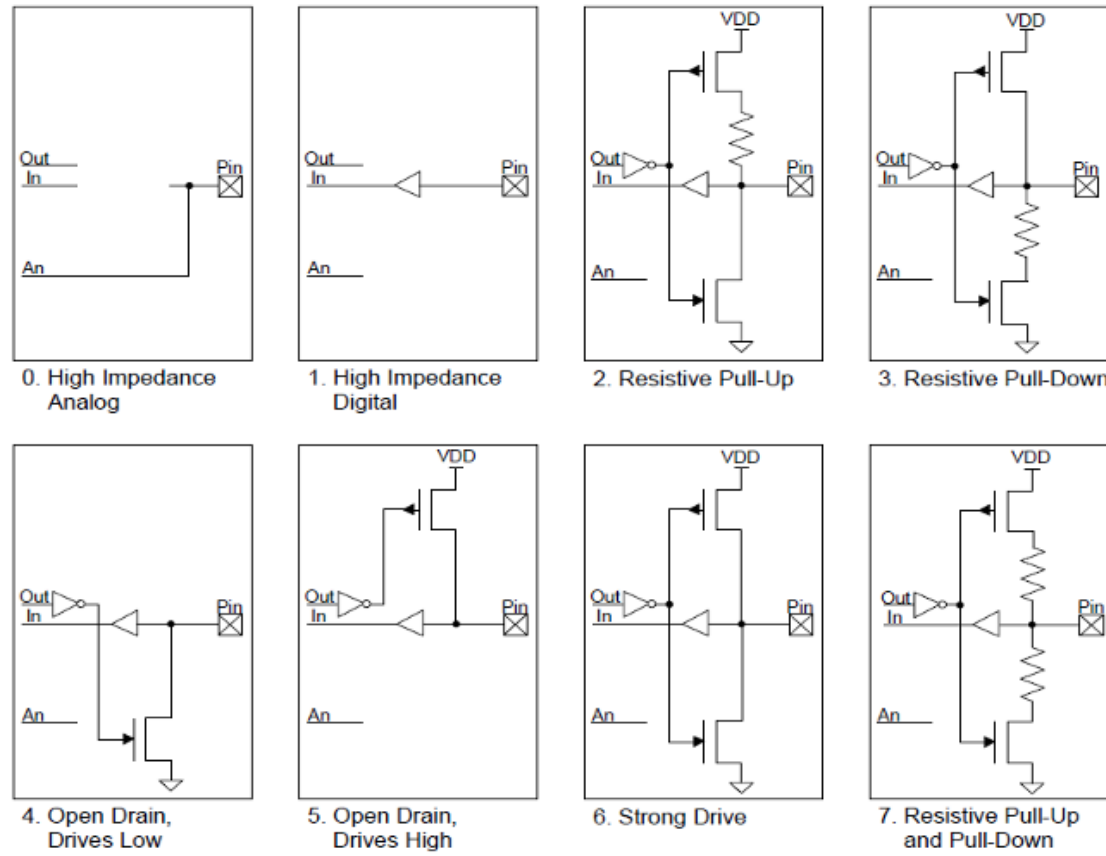


Figure 6-11. Drive Mode



Register Name	Purpose	Address
FLASH_DATA[0..262143]	DATA	[0..262143 * 0x1]
SRAM_CODE64K[0..16383]	Code System Memory Bank	0x1ff8000 + [0..16383 * 0x1]
SRAM_CODE32K[0..8191]	Code System Memory Bank	0x1ffc000 + [0..8191 * 0x1]
SRAM_CODE16K[0..4095]	Code System Memory Bank	0x1ffe000 + [0..4095 * 0x1]
SRAM_CODE[0..4095]	Code System Memory Bank	0x1fff000 + [0..4095 * 0x1]
SRAM_DATA[0..4095]	Data System Memory Bank	0x20000000 + [0..4095 * 0x1]
SRAM_DATA16K[0..4095]	Data System Memory Bank	0x20001000 + [0..4095 * 0x1]
SRAM_DATA32K[0..8191]	Data System Memory Bank	0x20002000 + [0..8191 * 0x1]
SRAM_DATA64K[0..16383]	Data System Memory Bank	0x20004000 + [0..16383 * 0x1]
DMA_SRAM64K[0..16383]	Data System Memory Bank	0x20008000 + [0..16383 * 0x1]
DMA_SRAM32K[0..8191]	Data System Memory Bank	0x2000c000 + [0..8191 * 0x1]
DMA_SRAM16K[0..4095]	Data System Memory Bank	0x2000e000 + [0..4095 * 0x1]
DMA_SRAM[0..4095]	Data System Memory Bank	0x2000f000 + [0..4095 * 0x1]
CLKDIST_CR	Configuration Register CR	0x40004000
CLKDIST_LD	LOAD Register	0x40004001
CLKDIST_WRK0	LSB Shadow Divider Value Register	0x40004002
CLKDIST_WRK1	MSB Shadow Divider Value Register	0x40004003
CLKDIST_MSTR0	Master clock (clk_sync_d) Divider Value Register	0x40004004
CLKDIST_MSTR1	Master (clk_sync_d) Configuration Register/CPU Divider Value	0x40004005
CLKDIST_BCFG0	CLK_BUS LSB Divider Value Register	0x40004006
CLKDIST_BCFG1	CLK_BUS MSB Divider Value Register	0x40004007
CLKDIST_BCFG2	CLK_BUS Configuration Register	0x40004008
CLKDIST_UCFG	USB Configuration Register	0x40004009
CLKDIST_DLY0	Delay block Configuration Register	0x4000400a
CLKDIST_DLY1	Delay block Configuration Register	0x4000400b
CLKDIST_DMASK	Digital Clock Mask Register	0x40004010
CLKDIST_AMASK	Analog Clock Mask Register	0x40004014
CLKDIST_DCFG[0..7]_CFG0	LSB Divider Value Register	0x40004080 + [0..7 * 0x4]
CLKDIST_DCFG[0..7]_CFG1	MSB Divider Value Register	0x40004080 + [0..7 * 0x4] + 0x1
CLKDIST_DCFG[0..7]_CFG2	Configuration Register	0x40004080 + [0..7 * 0x4] + 0x2
CLKDIST_ACFG[0..3]_CFG0	LSB Divider Value Register	0x40004100 + [0..3 * 0x4]
CLKDIST_ACFG[0..3]_CFG1	MSB Divider Value Register	0x40004100 + [0..3 * 0x4] + 0x1
CLKDIST_ACFG[0..3]_CFG2	Configuration Register	0x40004100 + [0..3 * 0x4] + 0x2
CLKDIST_ACFG[0..3]_CFG3	Analog clocks Configuration Register	0x40004100 + [0..3 * 0x4] + 0x3
FASTCLK_IMO_CR	Internal Main Oscillator Control Register	0x40004200



1.3.731 B[0..3]_UDB00_F0

UDB00_F0

Reset: N/A

Register : Address

B0_UDB00_F0: 0x40006440

Bits	7	6	5	4	3	2	1	0
SW Access:Reset	RW:UUUUUUUU							
HW Access	RW							
Retention	NONRET							
Name	F0							

FIFO 0

Bits	Name	Description
7:0	F0[7:0]	Generic field for 8 bit working registers



Complexer Chip:

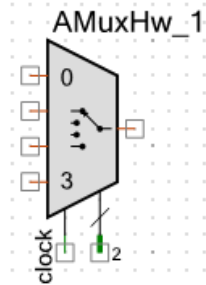
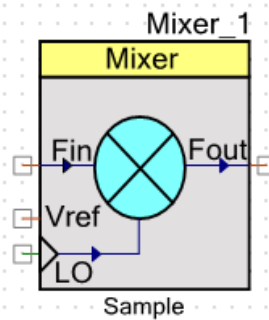
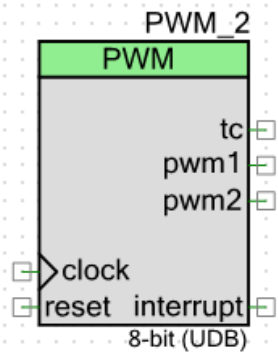
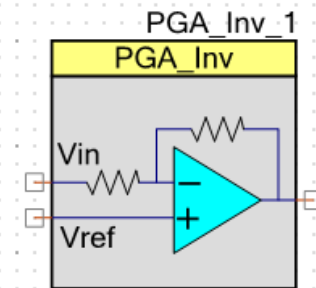
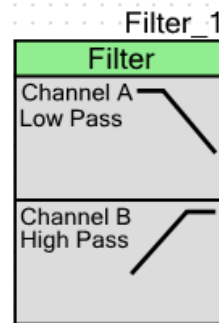
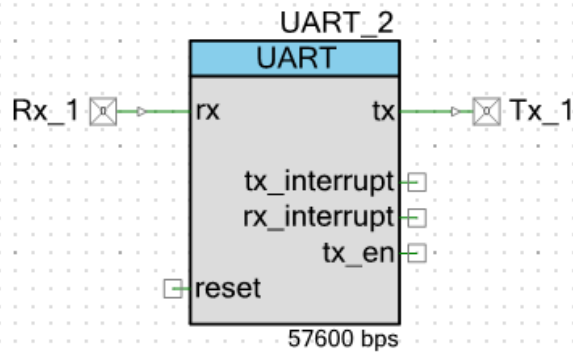
- UDB's
- Analoge Bauteile
- Digitaler Filter (DSP)
- Flexible IO
- Interconnect
- Pinzuweisung
- Hohe Anzahl an Registern



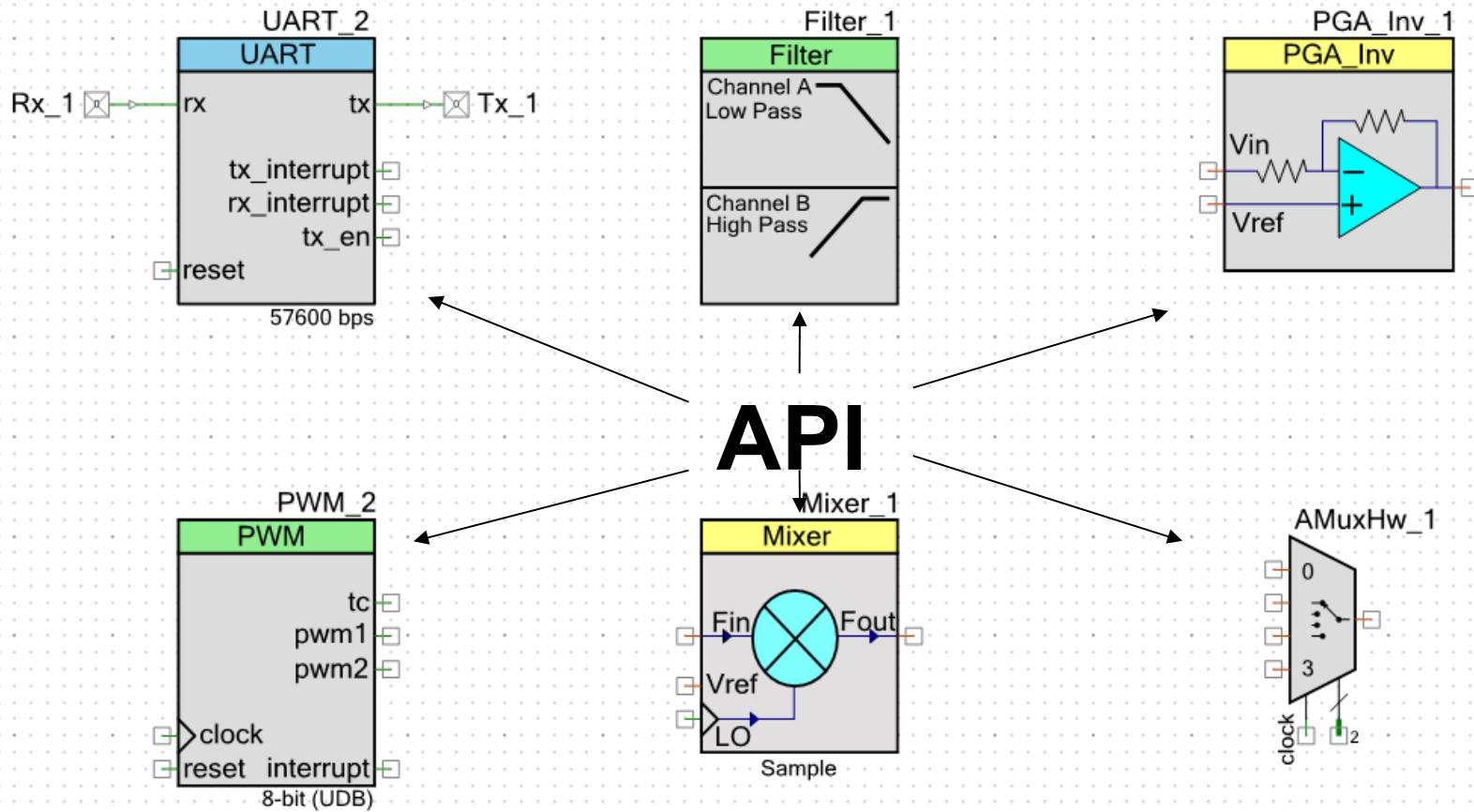
Application Programming Interface

API

PSoC Component



PSoC Componenten



PSoC Creator

The screenshot displays the PSoC Creator 3.3 interface for a project named 'UART_RxTx01'. The main workspace is divided into two sections: 'UART datasheet example project' and 'Development kit configuration'.

UART datasheet example project: This section describes the UART reception mechanism. It states: "This project demonstrates the UART reception mechanism. Data typed on the hyperterminal is sent through the serial port."

Development kit configuration: This section provides a list of steps for hardware setup:

1. The CY8CKIT-050 DVK board should be configured to the default switch and jumper settings.
2. Connect the RS232 cable from the computer to the DVK.
3. Setup Termini program in the computer with the following settings:
 - Baudrate : 57600 bits
 - Data bits : 8
 - Parity : None
 - Stop bits : 1
 - Flow control : Hardware
4. Connect P0_1 to RX pin on the CY8CKIT-050 kit.
5. Connect P0_0 to TX pin on the CY8CKIT-050 kit.
6. Connect P6_0 to CTS pin on the CY8CKIT-050 kit.
7. Connect P6_6 to RTS pin on the CY8CKIT-050 kit.

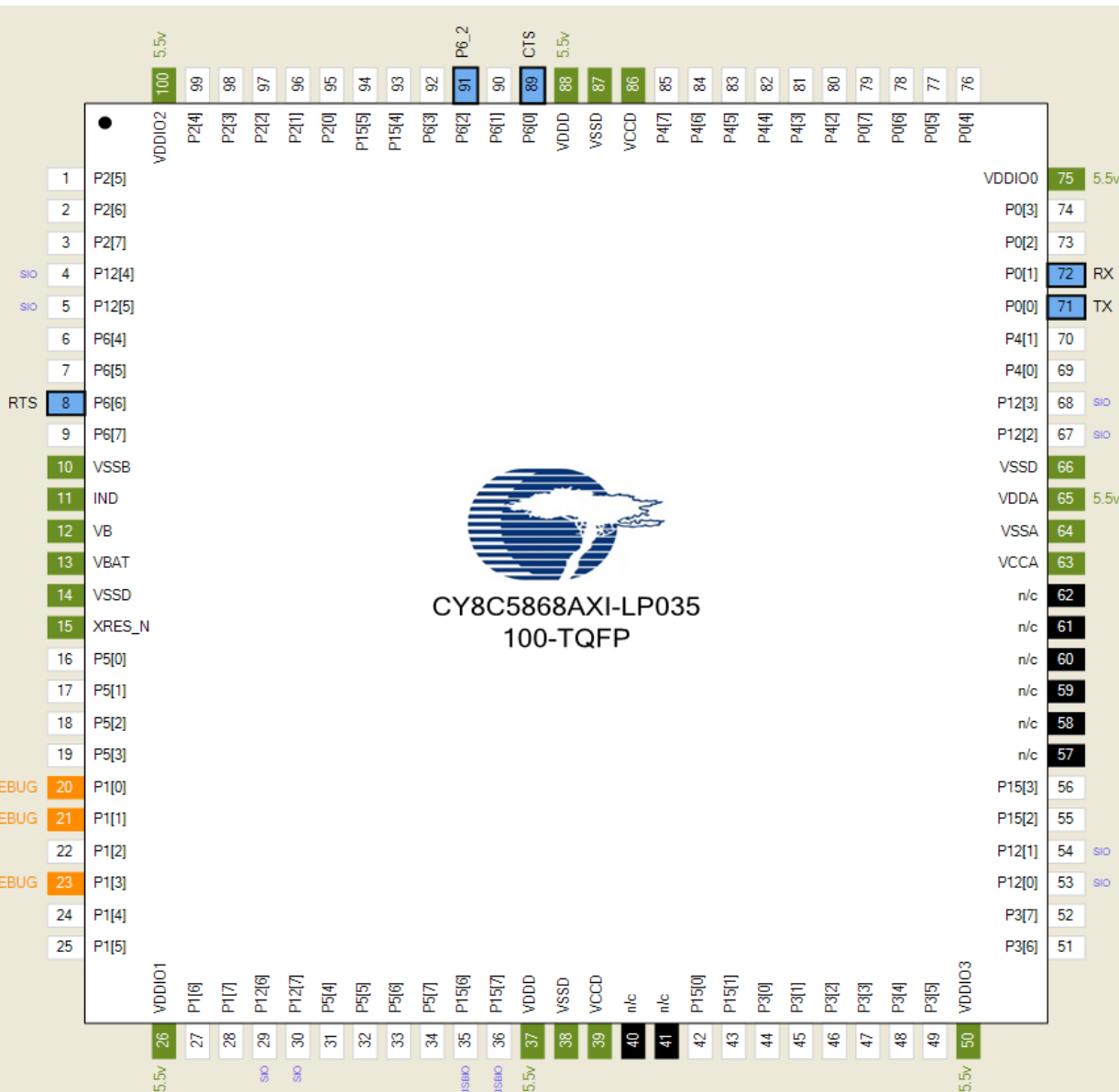
The schematic diagram shows the following components and connections:

- UART 1:** A component with pins for RX, CTS, TX, RTS, tx_interrupt, rx_interrupt, and a clock input. It is connected to a 460 kHz clock source.
- PWM 1:** A component with pins for tc, pwm1, pwm2, clock, and reset. It is connected to a 460 kHz clock source.
- LED3:** A blue LED connected to pin P6_2 through a 330Ω resistor (R60).
- MDIO Interface 1:** A component with pins for enable, disable, clock, read, address, force_clear, and en_page. It is connected to MDIO_1.

The interface also includes a Component Catalog on the right, an Output window at the bottom, and a workspace explorer on the left showing the project's file structure.



Port/Pin-Zuordnung der Signale



Name	Port	Pin	Lock
CTS	P6[0]	89	<input checked="" type="checkbox"/>
P6_2	P6[2]	91	<input checked="" type="checkbox"/>
RTS	P6[6]	8	<input checked="" type="checkbox"/>
RX	P0[1]	72	<input checked="" type="checkbox"/>
TX	P0[0]	71	<input checked="" type="checkbox"/>



2. Die Portierung

Wahl zwischen
Erstbegehung des 5868 er
und

Weg / Umweg mit dem geringsten
Widerstand

...

Ich hab beides gewählt - zunächst

Die Wahl:

Erstbegehung: FORTH only

- 1540 Register je nach Verschaltung initialisieren
- Primitives für Hardwarezugriff entwickeln
- Bugs, Änderungen etc. eigenständig pflegen

Um- / Weg R_{\min} : FORTH & C-API

- kein FORTH only
- C-Funktionen als Primitives für Hardwarezugriff

Tools, Rules & Moore

Cortex M3 Kern

GNU-Assembler und -Linker

Cortex M3 Assembler

PSoC Creator

Schaltplan Editor

API's

Main.c

GNU C-Compiler

Projekt Build

Mapfile

Linker

Debugger: Bin oder Asm Ebene

Mecrisp

Struktur, Aufbau, Fkt.

Portierung

Adressen

Makefile

HAL.s

Terminal.s

Flash.s

Hex Editor

Terminal Programm

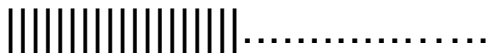
Evaluation Board

Inbetriebnahme

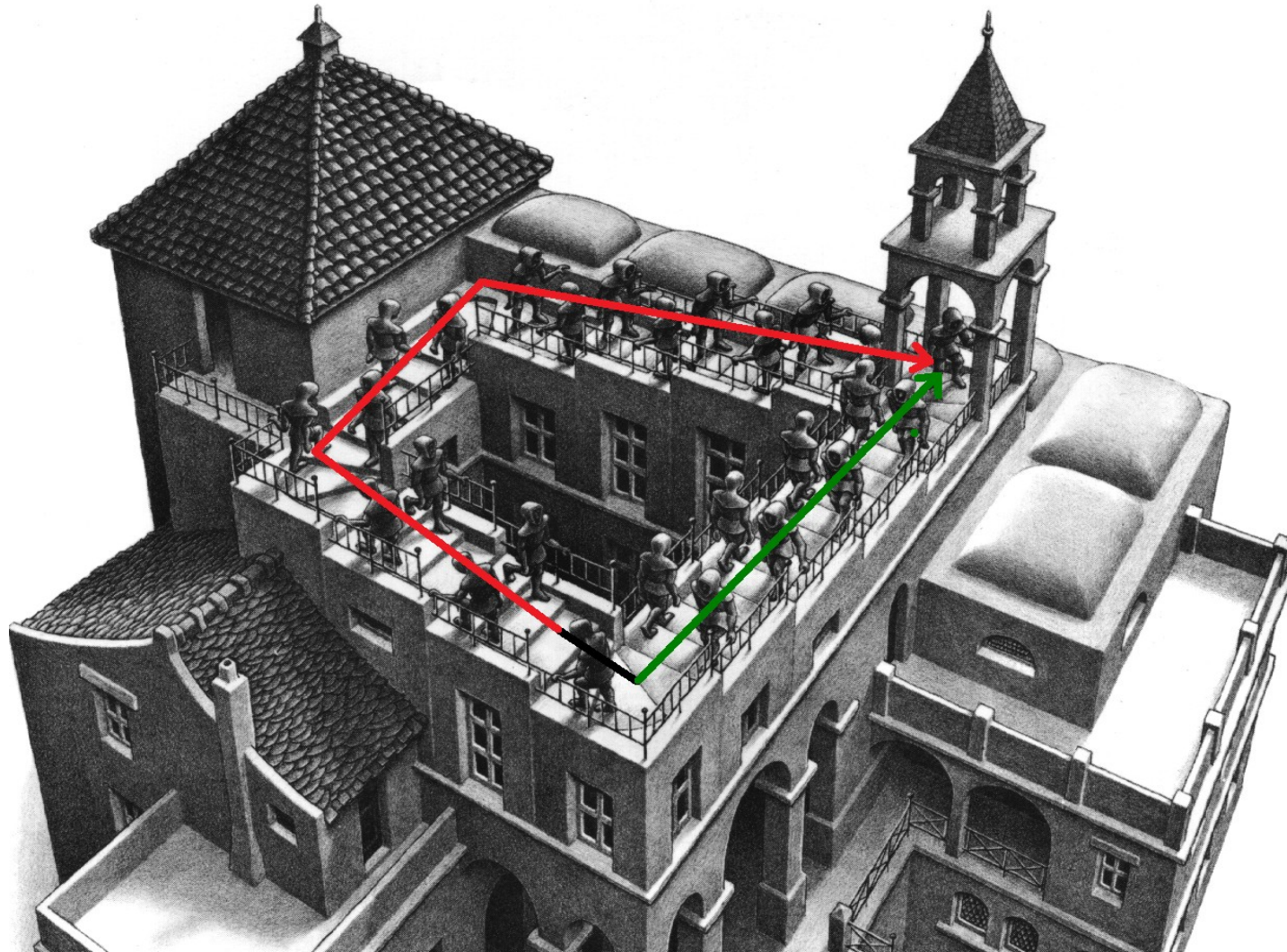
Schaltplan

Datenblätter, Foren

Programmierung Flash



Treppauf / Treppab



KEY & EMIT via revers engineering

```
27  
28  
29 .equ Terminal_USART_RX_SR, Terminal_USART_Base + 0x63  
30 .equ Terminal_USART_RX_DR, Terminal_USART_Base + 0x43  
31 .equ Terminal_USART_TX_SR, Terminal_USART_Base + 0x60  
32 .equ Terminal_USART_TX_DR, Terminal_USART_Base + 0x40
```

```
79 @ -----  
80 Wortbirne Flag_visible, "serial-emit"  
81 serial_emit: @ ( c -- ) Emit one character  
82 @ -----  
83 push {lr}  
84  
85 1: bl serial_qemit @ 06.02.16/4  
86 cmp tos, #0  
87 drop  
88 beq 1b  
89  
90 ldr r2, =Terminal_USART_TX_DR @ 06.02.16/1  
91 strb tos, [r2] @ Output the character  
92 drop  
93  
94 pop {pc}  
95  
96 @ -----  
97 Wortbirne Flag_visible, "serial-key"  
98 serial_key: @ ( -- c ) Receive one character  
99 @ -----  
100 push {lr}  
101  
102 1: bl serial_qkey  
103 cmp tos, #0  
104 drop  
105 beq 1b  
106  
107 pushdatas  
108 ldr r2, =Terminal_USART_RX_DR @ 06.02.16/1  
109 ldrb tos, [r2] @ Fetch the character  
110  
111 pop {pc}  
112  
113 @ -----  
114 Wortbirne Flag_visible, "serial-emit?"  
115 serial_qemit: @ ( -- ? ) Ready to send a character ?  
116 @ -----  
117 push {lr}  
118 bl pause  
119  
120 pushdconst 0  
121 ldr r0, =Terminal_USART_TX_SR @ 06.02.16/1  
122 ldr r1, [r0] @ Fetch status  
123 movs r0, #TXE  
124 ands r1, r0  
125 beq 1f  
126 mvns tos, tos  
127 1: pop {pc}  
128  
129 @ -----  
130 Wortbirne Flag_visible, "serial-key?"  
131 serial_qkey: @ ( -- ? ) Is there a key press ?  
132 @ -----  
133 push {lr}  
134 bl pause  
135  
136 pushdconst 0  
137 ldr r0, =Terminal_USART_RX_SR @ 06.02.16/1  
138 ldr r1, [r0] @ Fetch status  
139 movs r0, #RXNE  
140 ands r1, r0  
141 beq 1f  
142 mvns tos, tos  
143 1: pop {pc}  
144
```



KEY & EMIT via API-Adressen

```
43 @ -----
44      Wortbirne Flag_visible, "UART_1_PutChar"
45 UART_1_PutChar:
46 @ -----
47      popda r0
48
49      ldr r1,  =#0x00000434      @ execute
50      mov pc, r1
51
52 @ -----
53      Wortbirne Flag_visible, "UART_1_GetChar"
54 UART_1_GetChar:
55 @ -----
56      ldr r0,  =#0x00000400      @ execute
57      mov pc, r0
58
59      pushda r0
60
61 @ -----
62      Wortbirne Flag_visible, "UART_1_ReadTxStatus"
63 UART_1_ReadTxStatus:
64 @ -----
65      ldr r0,  =#0x00000428      @ execute
66      mov pc, r0
67
68      pushda r0
69
70
71 @ -----
72      Wortbirne Flag_visible, "UART_1_ReadRxStatus"
73 UART_1_ReadRxStatus:
74 @ -----
75      ldr r0,  =#0x000003f0      @ execute
76      mov pc, r0
77
78      pushda r0
79
```

```
81 @ -----
82      Wortbirne Flag_visible, "serial-emit"
83 serial_emit: @ ( c -- ) Emit one character
84 @ -----
85      push {lr}
86
87      bl UART_1_PutChar
88
89      pop {pc}
90
91 @ -----
92      Wortbirne Flag_visible, "serial-key"
93 serial_key: @ ( -- c ) Receive one character
94 @ -----
95      push {lr}
96
97      1:bl pause
98
99      bl UART_1_GetChar
100
101      cmp tos, #0
102      beq 1b
103      pushdatas
104      mov tos, r0
105
106      pop {pc}
107
108 @ -----
109      Wortbirne Flag_visible, "serial-emit?"
110 serial_qemit: @ ( -- ? ) Ready to send a character ?
111 @ -----
112      push {lr}
113      bl pause
114
115      bl UART_1_ReadRxStatus
116
117      movs r0, #TXE
118      ands tos, r0
119
120      pop {pc}
121
122 @ -----
123      Wortbirne Flag_visible, "serial-key?"
124 serial_qkey: @ ( -- ? ) Is there a key press ?
125 @ -----
126      push {lr}
127      bl pause
128
129      bl UART_1_ReadTxStatus
130
131      movs r0, #RXNE
132      ands tos, r0
133
134      1:pop {pc}
135
```



Mecrisp.bin als char Tabelle in C eingebunden

```
1 /* C:\FORTH\Mecrisp\mecrisp-stellaris-2.2.0\cy8c5868axi\mecrisp-stellaris-cy8c5868axi.bin (05.08.2017 00:13:28)
2 StartOffset: 00000000, EndeOffset: 00003CAB, L[änge: 00003CAC */
3
4 const unsigned char mecrisp[] __attribute__((section("FORTH"))) = {
5     0x30, 0x04, 0x00, 0x20, 0x0B, 0xBB, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00,
6     0x4B, 0xB9, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00,
7     0x4B, 0xB9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
8     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
9     0x79, 0xB9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
10    0x23, 0xB9, 0x00, 0x00, 0x30, 0x04, 0x00, 0x20, 0x0B, 0xBB, 0x00, 0x00,
11    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
12    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
13    0xA1, 0xB9, 0x00, 0x00, 0xC9, 0xB9, 0x00, 0x00, 0xF1, 0xB9, 0x00, 0x00,
14    0x19, 0xBA, 0x00, 0x00, 0x41, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
15    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
16    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
17    0x67, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
18    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
19    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
20    0x79, 0xB9, 0x00, 0x00, 0x8F, 0xBA, 0x00, 0x00, 0xB7, 0xBA, 0x00, 0x00,
21    0xDF, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
22    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
23    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
24    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
25    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
26    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
27    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
28    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
29    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
30    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
31    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
32    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
33    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
34    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
35    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
36    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
37    0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
38    0x79, 0xB9, 0x00, 0x00, 0xB6, 0x81, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x2D,
39    0x2D, 0x2D, 0x20, 0x4D, 0x65, 0x63, 0x72, 0x69, 0x73, 0x70, 0x2D, 0x53,
40    0x74, 0x65, 0x6C, 0x6C, 0x61, 0x72, 0x69, 0x73, 0x20, 0x43, 0x6F, 0x72,
41    0x65, 0x20, 0x2D, 0x2D, 0x2D, 0x00, 0xCE, 0x81, 0x00, 0x00, 0x42, 0x80,
42    0x04, 0x32, 0x64, 0x75, 0x70, 0x00, 0x38, 0x68, 0x47, 0xF8, 0x04, 0x6D,
43    0x04, 0x3F, 0x38, 0x60, 0x70, 0x47, 0xE0, 0x81, 0x00, 0x00, 0x62, 0x80,
```

Main.c → extern const unsigned char mecrisp[];



KEY & EMIT via API ohne Adressen und C-Tabelle

```
28 @ -----
29 Wortbirne Flag_visible, "serial-emit"
30 serial_emit: @ ( c -- ) Emit one character
31 @ -----
32 push {lr}
33
34     popda r0                                @
35     bl UART_1_PutChar                        @
36
37     pop {pc}
38
39 @ -----
40 Wortbirne Flag_visible, "serial-key"
41 serial_key: @ ( -- c ) Receive one character
42 @ -----
43     push {lr}
44
45 1:    bl pause
46
47     bl UART_1_GetChar                        @
48
49     @ popda r0                                @
50     cmp r0, #0                               @
51     beq 1b
52
53     pushda r0                                @
54     .....
55     pop {pc}
56
57 @ -----
58 Wortbirne Flag_visible, "serial-emit?"
59 serial_qemit: @ ( -- ? ) Ready to send a char
60 @ -----
61     push {lr}
62     bl pause
63
64     bl UART_1_ReadRxStatus
65
66     pushda r0                                @
67     moves r0, #TXE
68     ands tos, r0
69
70     pop {pc}
71
72 @ -----
73 Wortbirne Flag_visible, "serial-key?"
74 serial_qkey: @ ( -- ? ) Is there a key press
75 @ -----
76     push {lr}
77     bl pause
78
79     bl UART_1_ReadTxStatus
80
81     pushda r0                                @
82     moves r0, #RXNE
83     ands tos, r0
84
85 1:pop {pc}
86
```



HardwareAbstractionLayer HAL.s

```
28 @ -----
29     Wortbirne Flag_visible, "LEDOn()"           @ ( --- )
30 _LEDOn:
31 @ -----
32     b LEDOn
33
34
35 @ -----
36     Wortbirne Flag_visible, "LEDOff()"         @ ( --- )
37 _LEDOff:
38 @ -----
39     b LEDOff
40
41
42 @ -----
43     Wortbirne Flag_visible, "PWM_1_WritePeriod()" @ ( c --- )
44 _PWM_1_WritePeriod:
45 @ -----
46     popda r0                                   @ pop character from stack to input-register r0
47     b PWM_1_WritePeriod
48
49
50 @ -----
51     Wortbirne Flag_visible, "PWM_1_WriteCompare1()" @ ( c --- )
52 _PWM_1_WriteCompare1:
53 @ -----
54     popda r0                                   @ pop character from stack to input-register r0
55     b PWM_1_WriteCompare1
56
57
58 @ -----
59     Wortbirne Flag_visible, "PWM_1_WriteCompare2()" @ ( c --- )
60 _PWM_1_WriteCompare2:
61 @ -----
62     popda r0                                   @ pop character from stack to input-register r0
63     b PWM_1_WriteCompare2
64
```



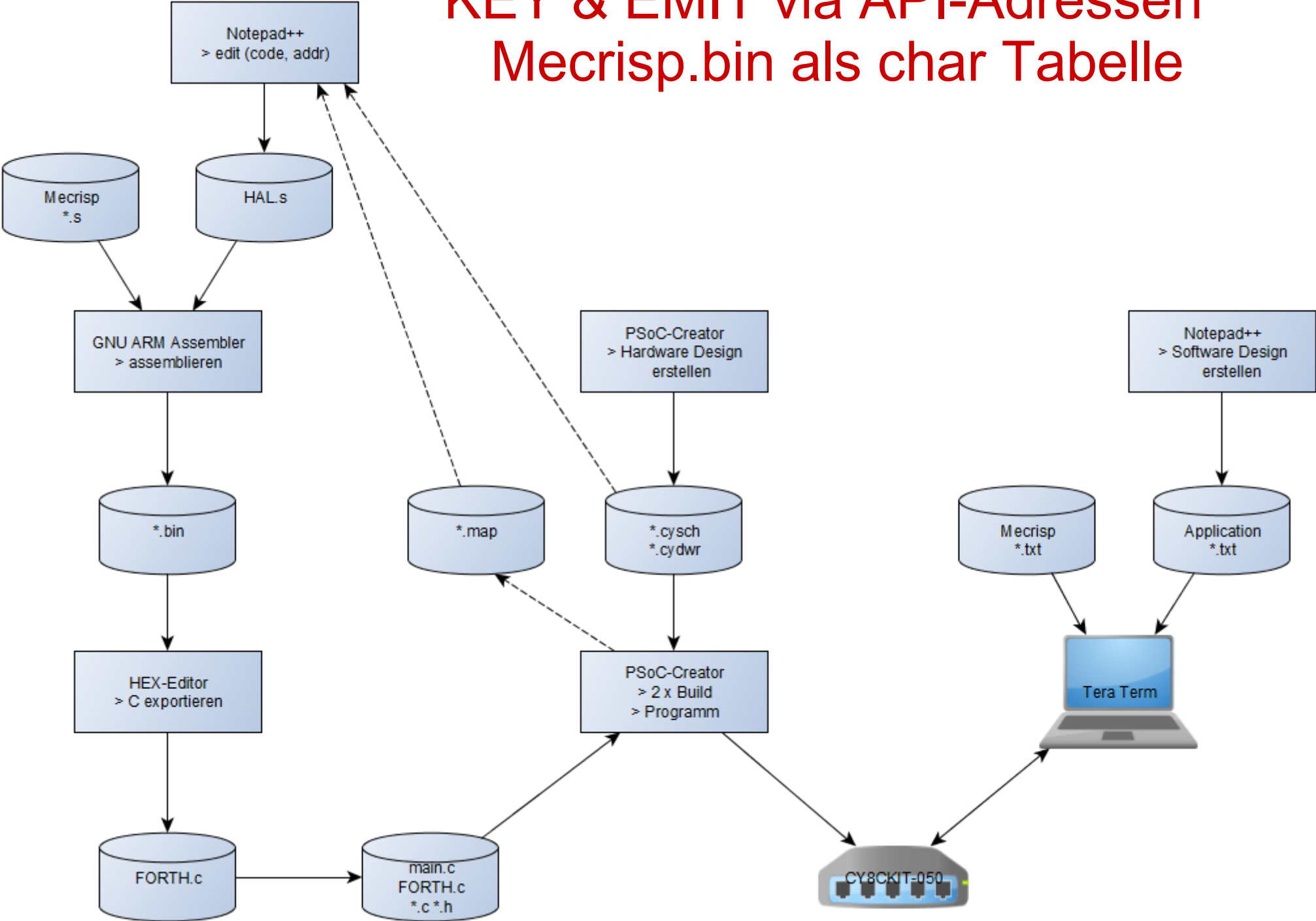
HardwareAbstractionLayer HAL.s

```
28 @ -----
29     Wortbirne Flag_visible, "LEDOn()"           @ ( --- )
30 _LEDOn:
31 @ -----
32     b LEDOn
33
34
35 @ -----
36     Wortbirne Flag_visible, "LEDOff()"          @ ( --- )
37 _LEDOff:
38 @ -----
39     b LEDOff
40
41
42 @ -----
43     Wortbirne Flag_visible, "PWM_1_WritePeriod()" @ ( c --- )
44 _PWM_1_WritePeriod:
45 @ -----
46     popda r0                                     @ pop character from stack to input-register r0
47     b PWM_1_WritePeriod
48
49
50 @ -----
51     Wortbirne Flag_visible, "PWM_1_WriteCompare1()" @ ( c --- )
52 _PWM_1_WriteCompare1:
53 @ -----
54     popda r0                                     @ pop character from stack to input-register r0
55     b PWM_1_WriteCompare1
56
57
58 @ -----
59     Wortbirne Flag_visible, "PWM_1_WriteCompare2()" @ ( c --- )
60 _PWM_1_WriteCompare2:
61 @ -----
62     popda r0                                     @ pop character from stack to input-register r0
63     b PWM_1_WriteCompare2
64
```

- eigene C-Library aus FORTH nutzbar
- ermöglicht Interaktives C
- ist Voraussetzung für den Umstieg von C auf FORTH

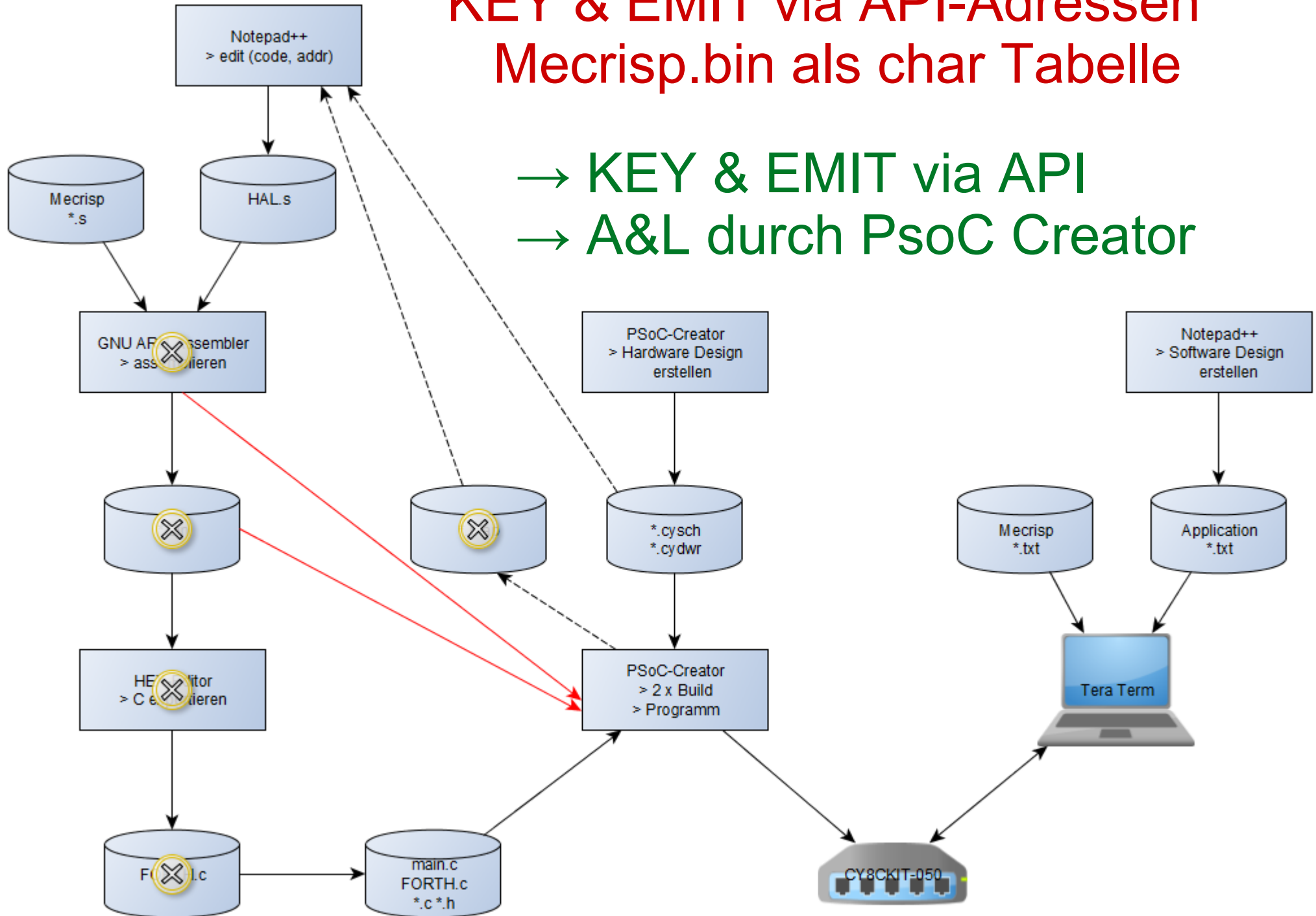


KEY & EMIT via API-Adressen Mecrisp.bin als char Tabelle

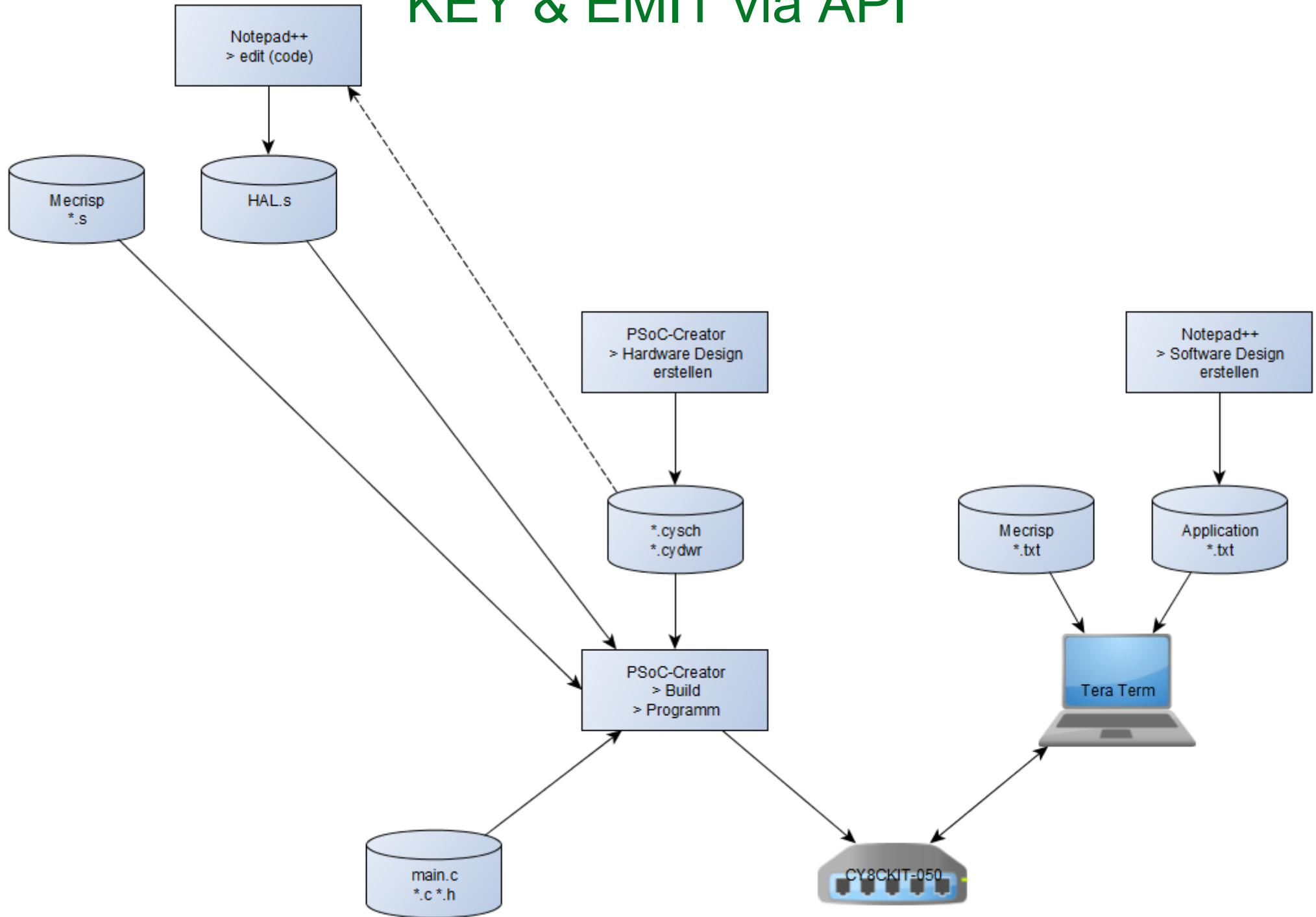


KEY & EMIT via API-Adressen Mecrisp.bin als char Tabelle

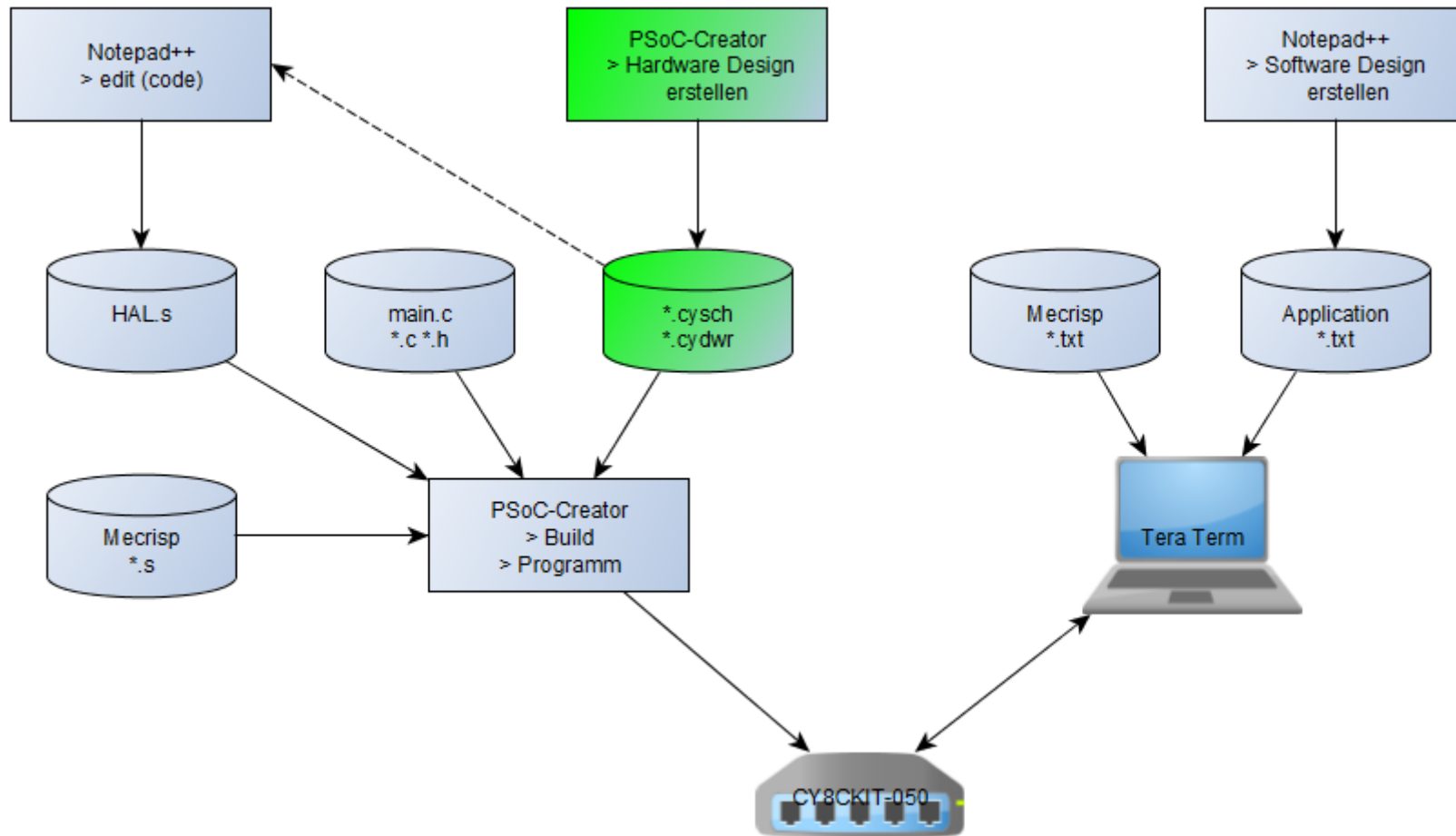
- KEY & EMIT via API
- A&L durch PsoC Creator



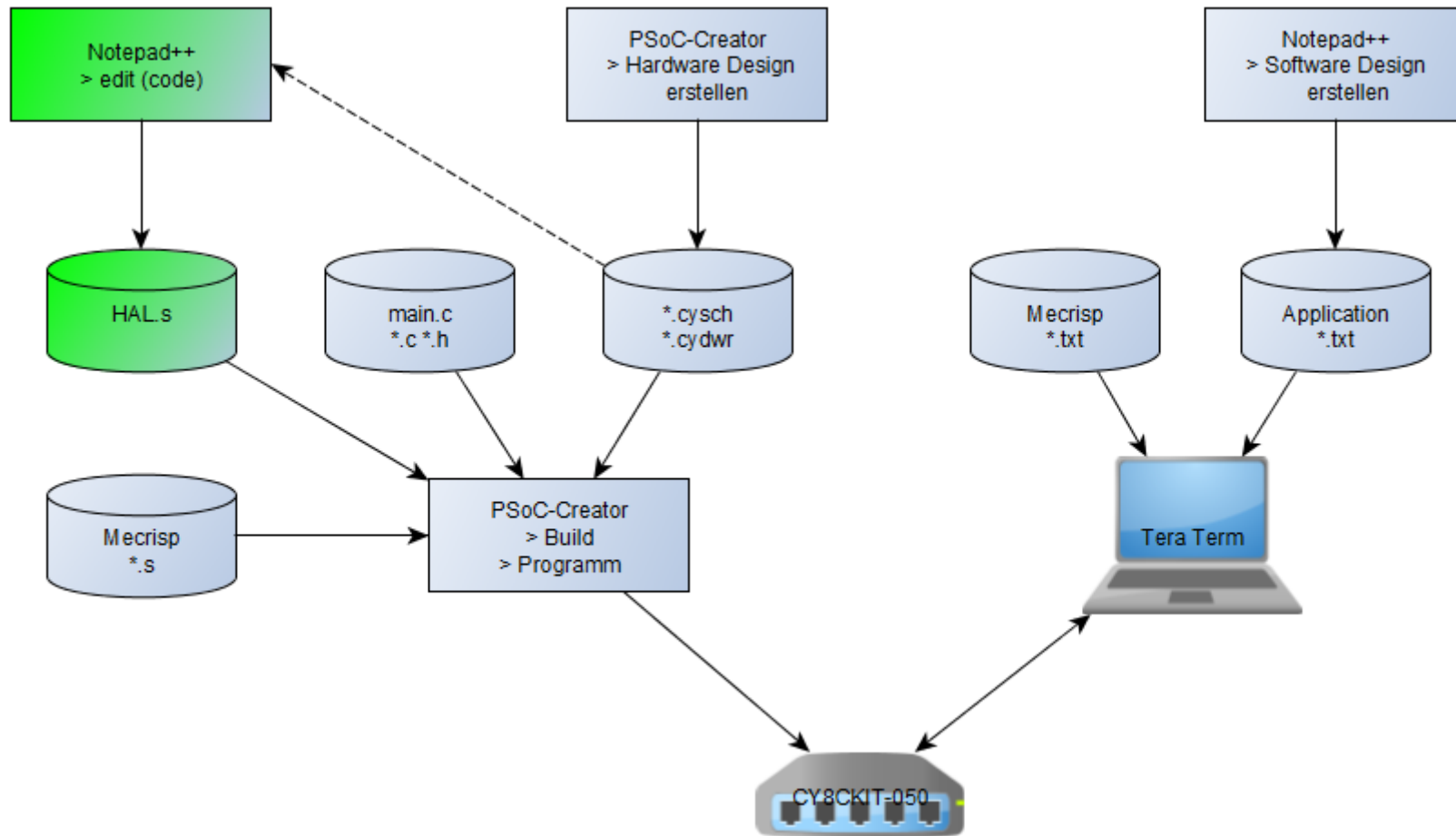
KEY & EMIT via API



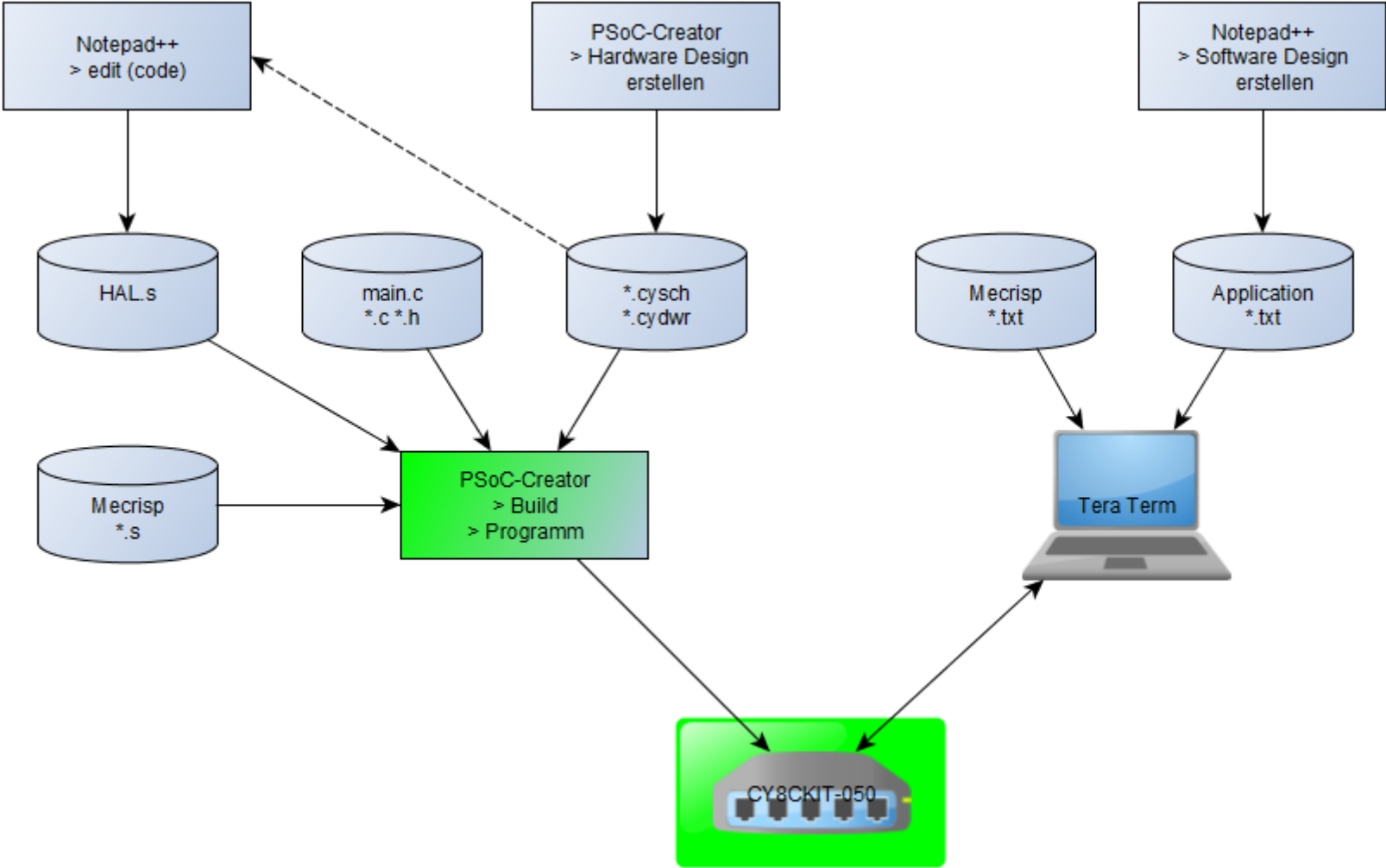
Schaltplan erstellen



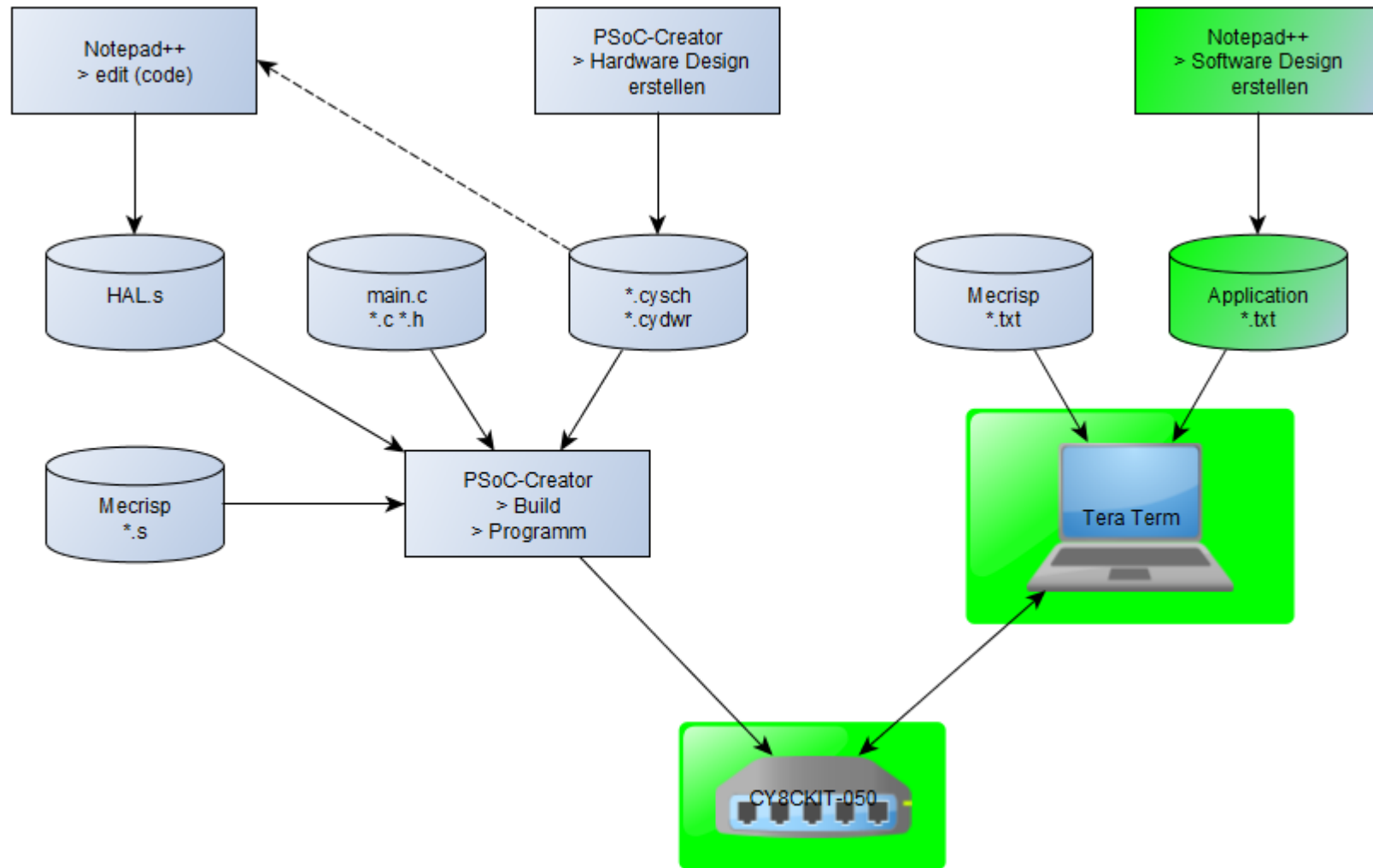
Componenten API's in HAL.s eintragen



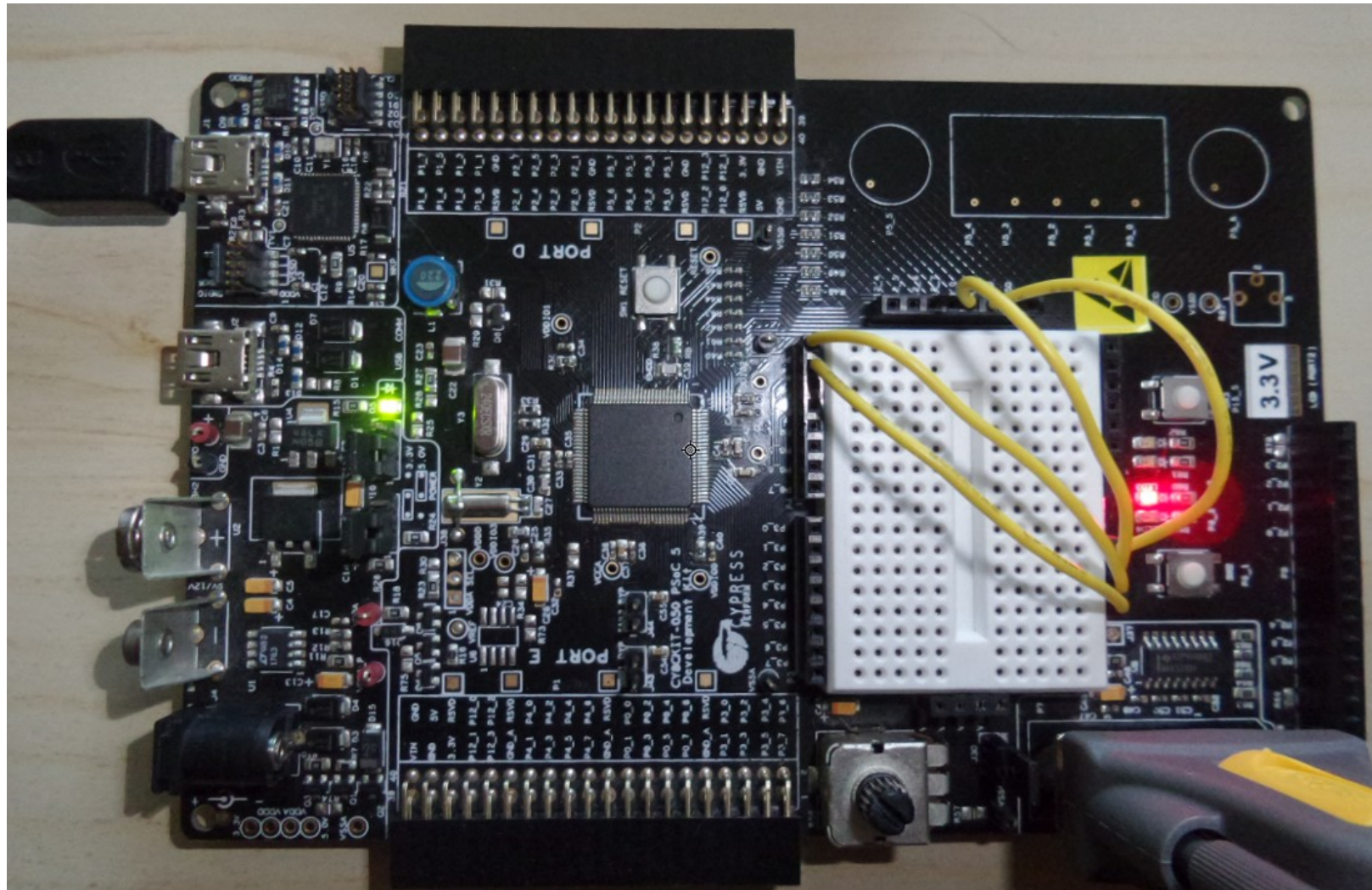
Hardware / Software generieren und laden



FORTH

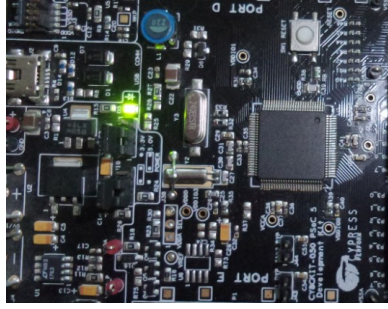
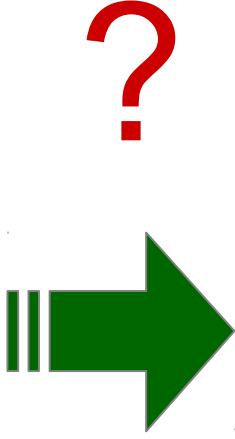


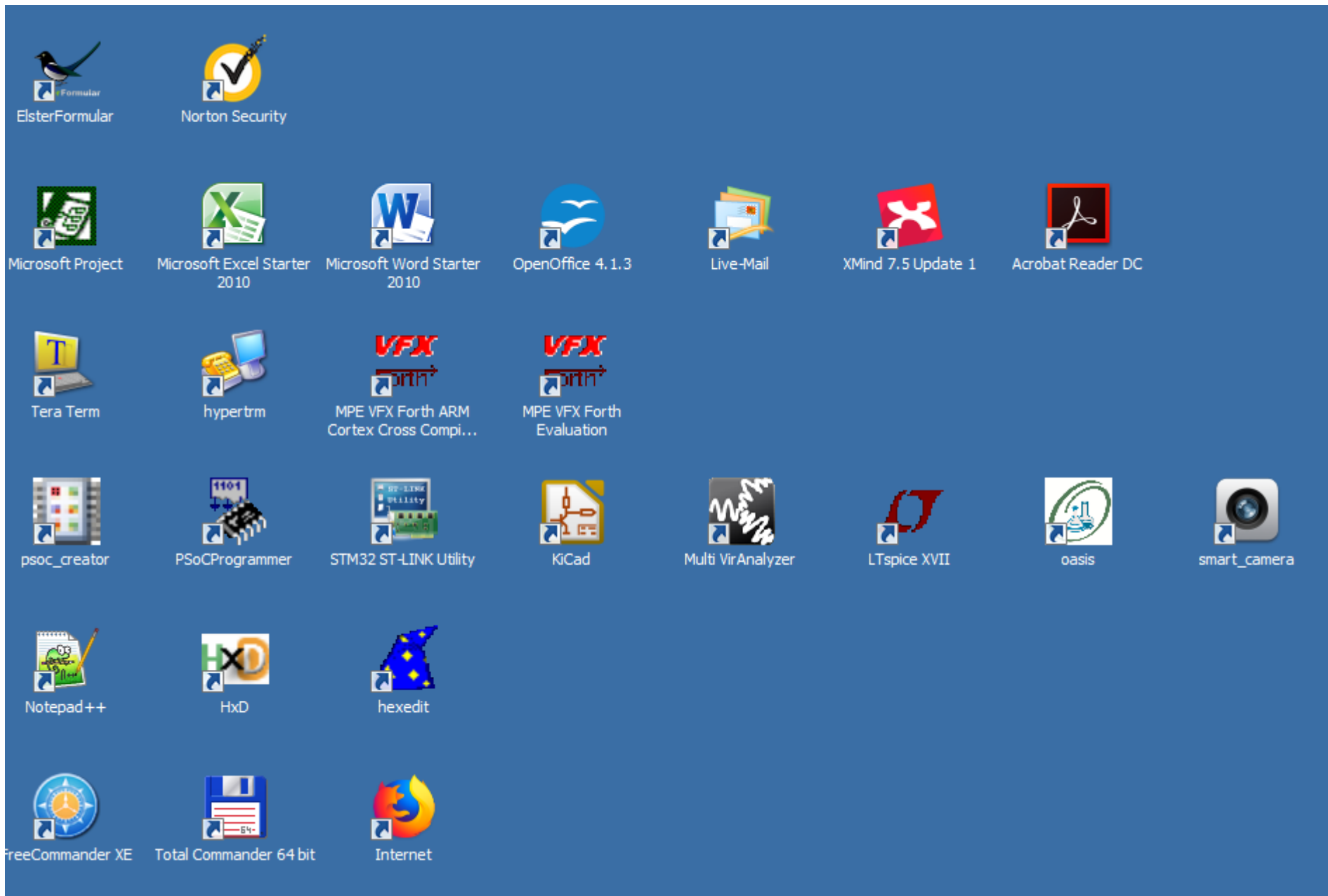
3. Das Evaluation-Board

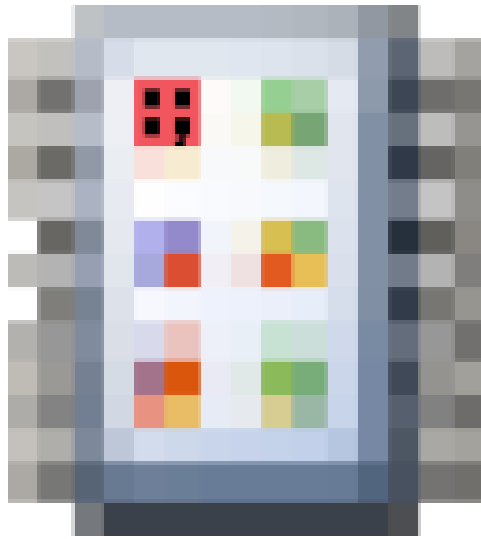


CY8C5868AXI: 100 Pin 10,- 1 cm²









C°y°SWAP

