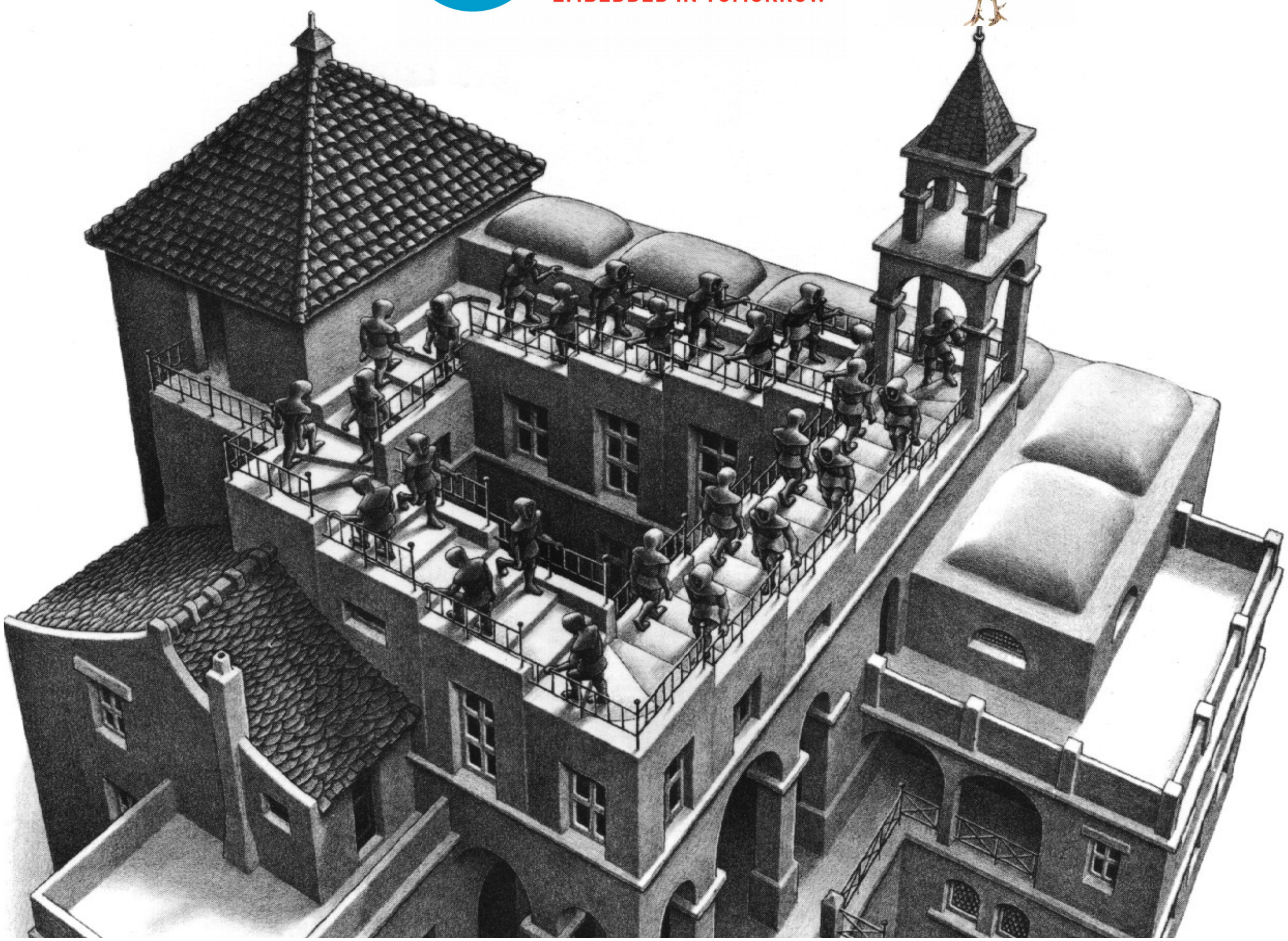
Portierung:
mecrisp-stellaris auf PSoC-5LP von

CYPRESS
EMBEDDED IN TOMORROW™



....

...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 **P**rogrammable **S**ystem **o**n **C**hip

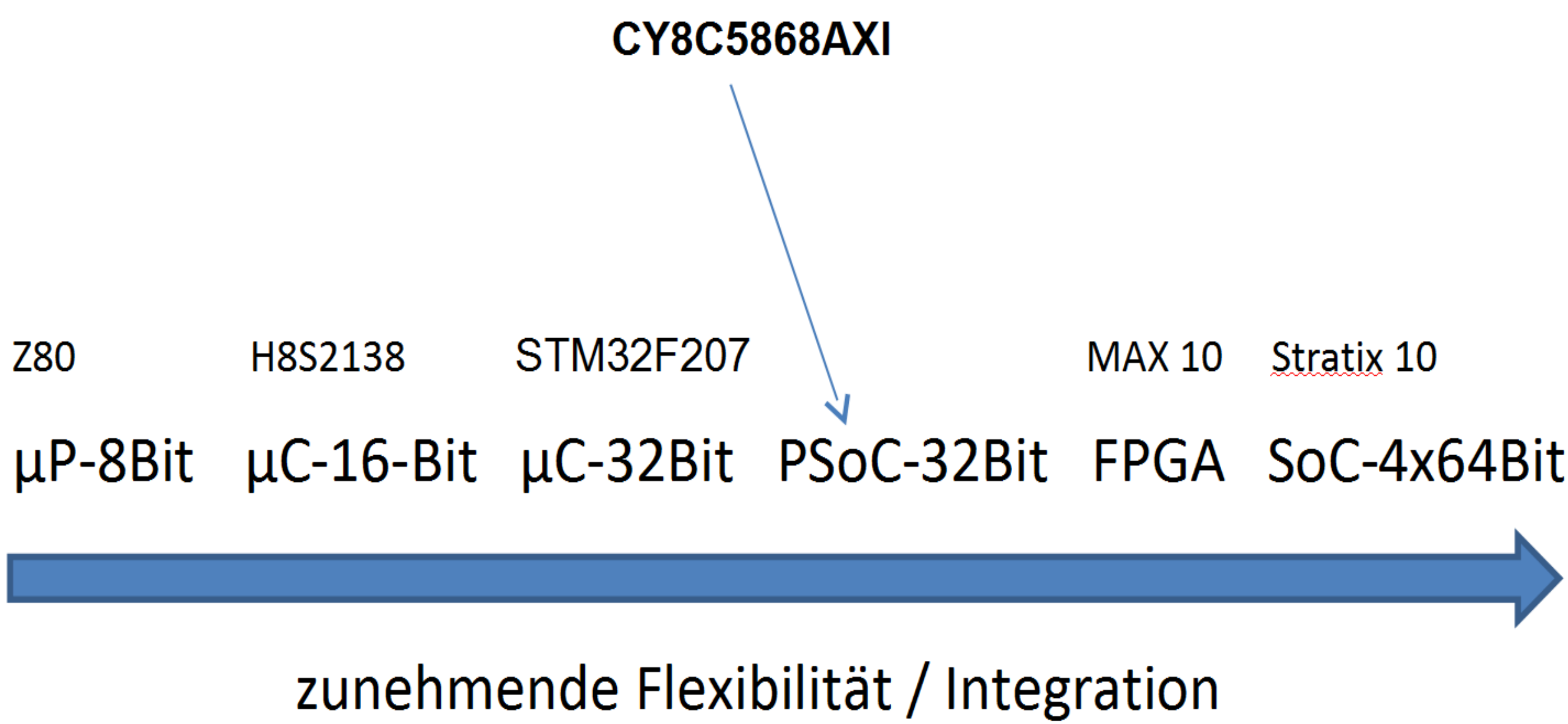


Figure 1-1. Simplified Block Diagram

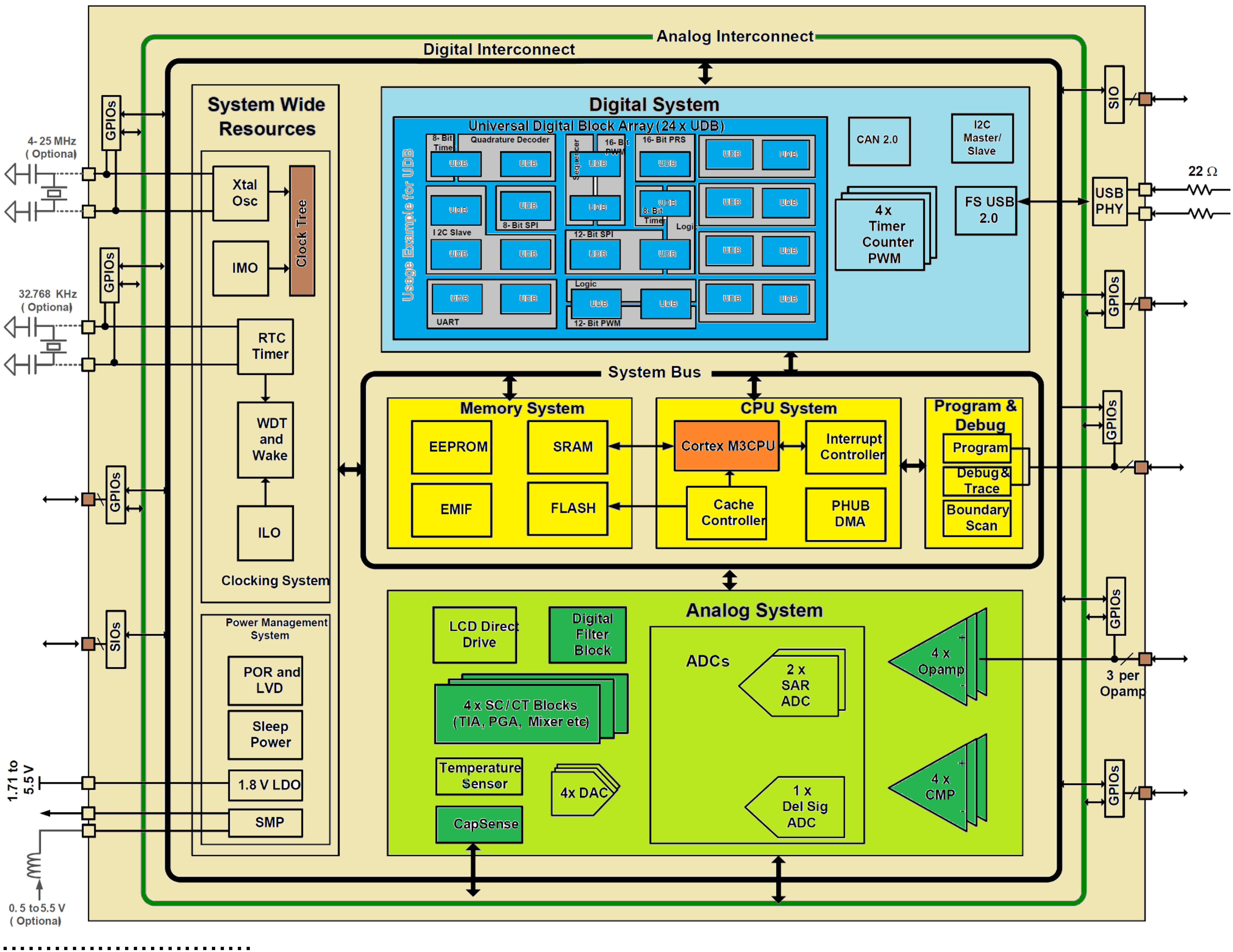
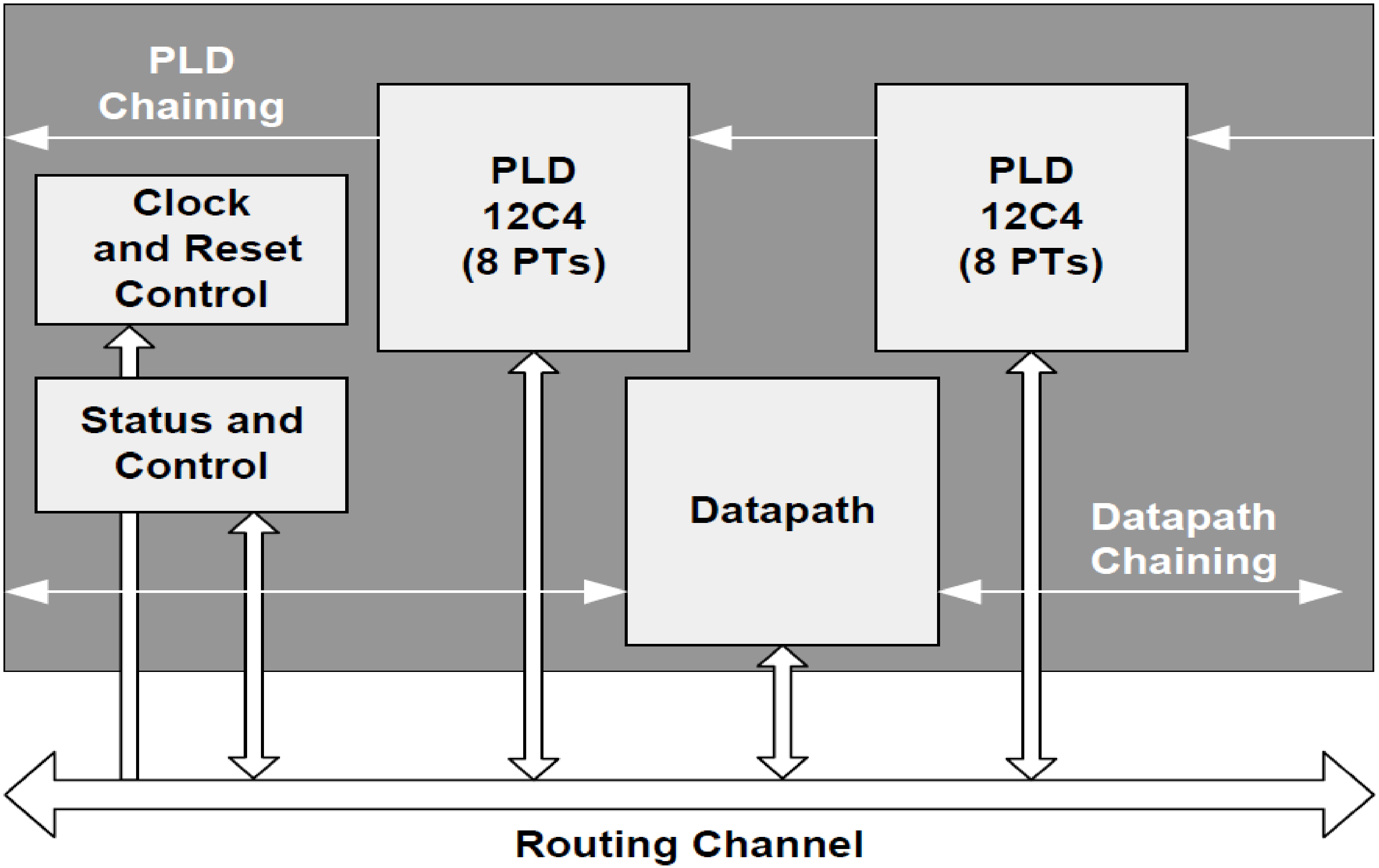
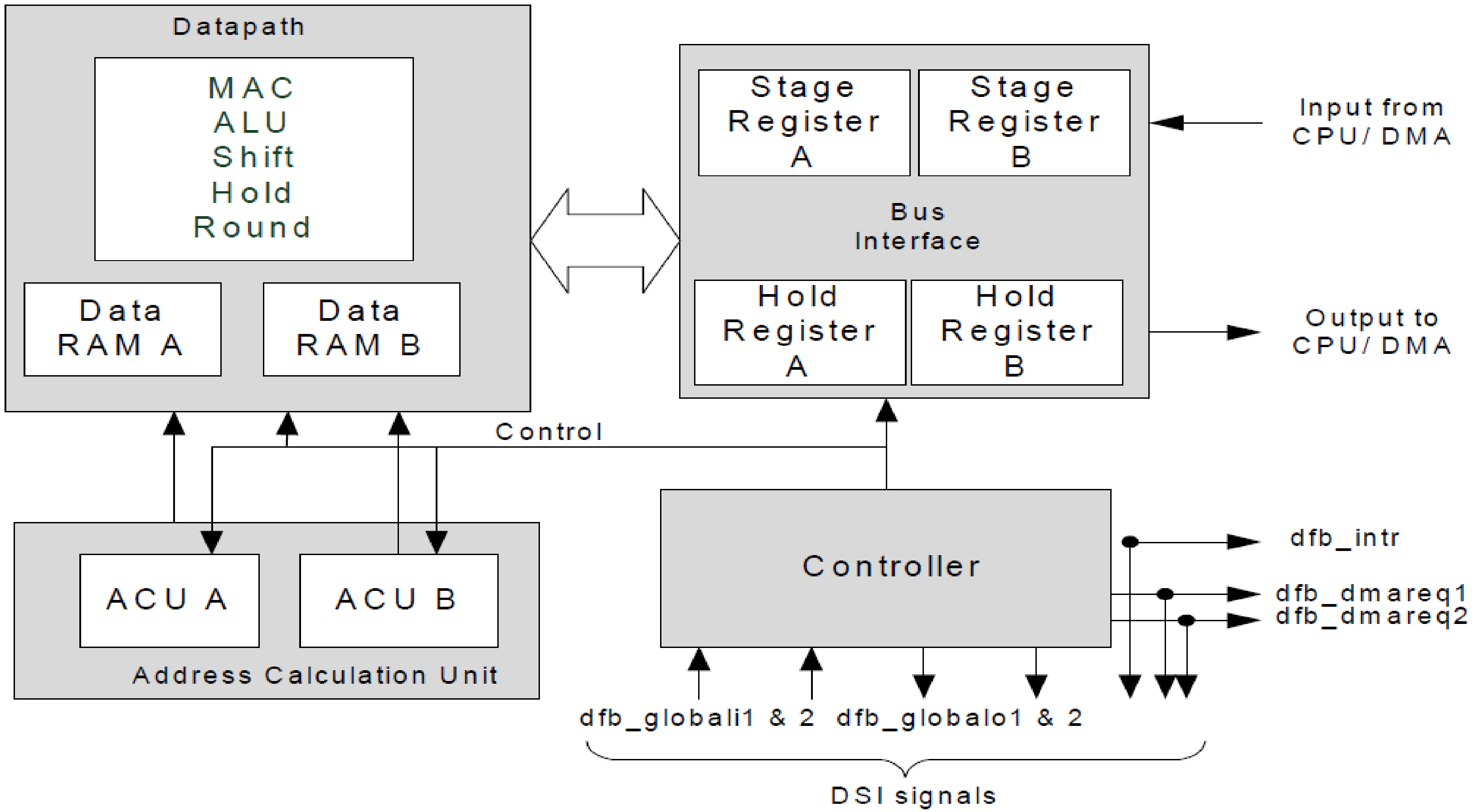


Figure 21-1. UDB Block Diagram



|||||..............................

Figure 27-1. Digital Filter Block Diagram



||||||..............................

Figure 29-10. Switched Capacitor Routing, Interface

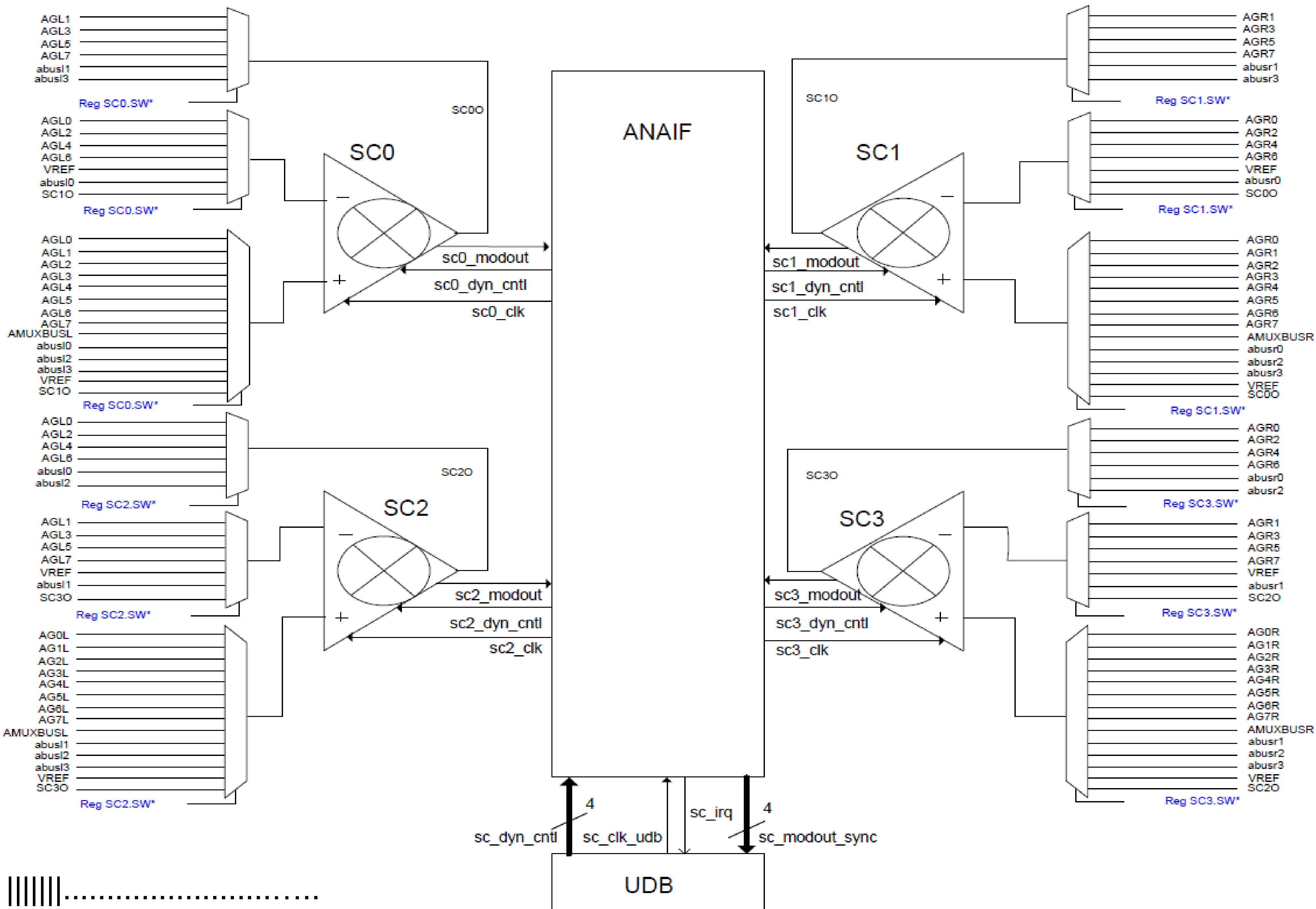
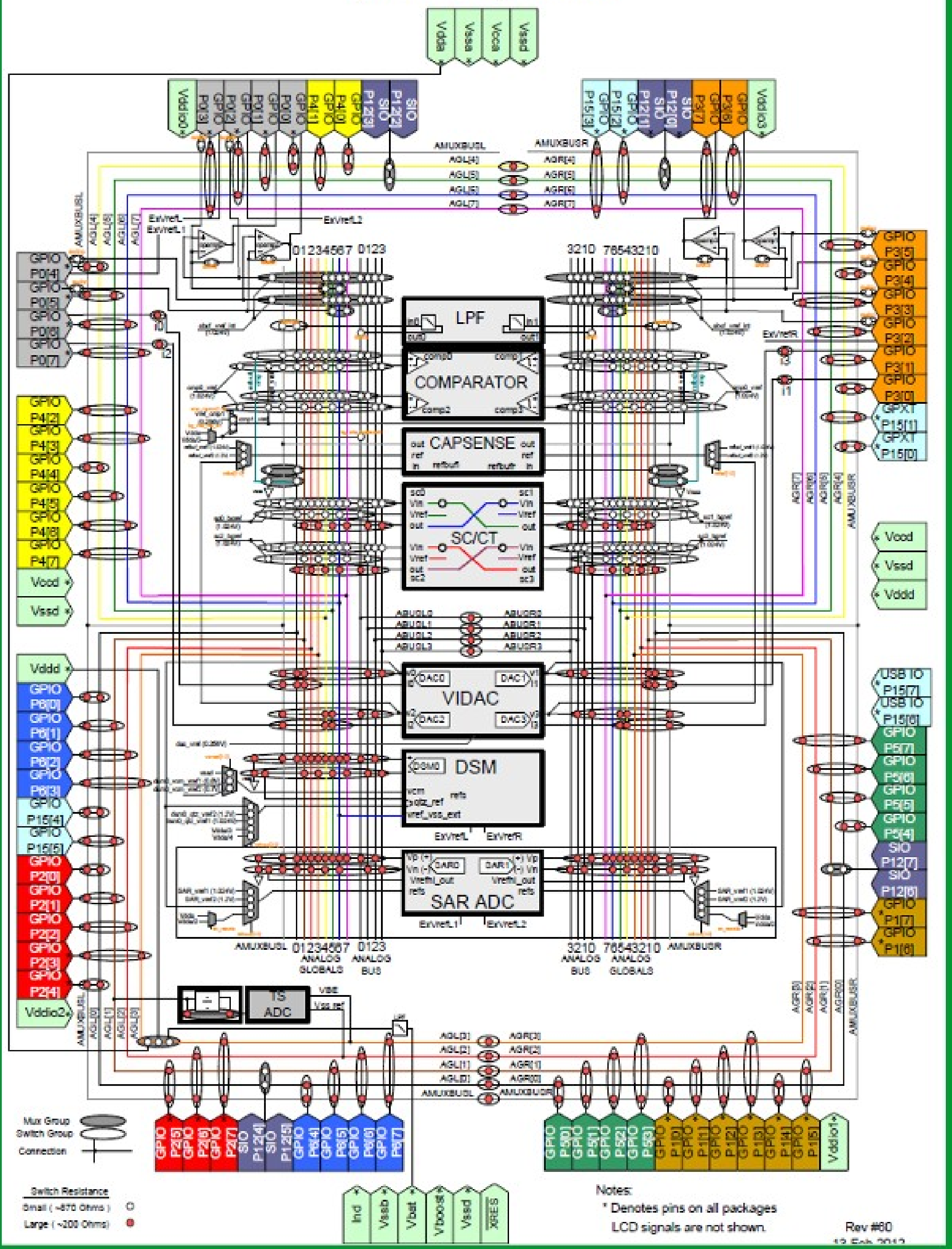
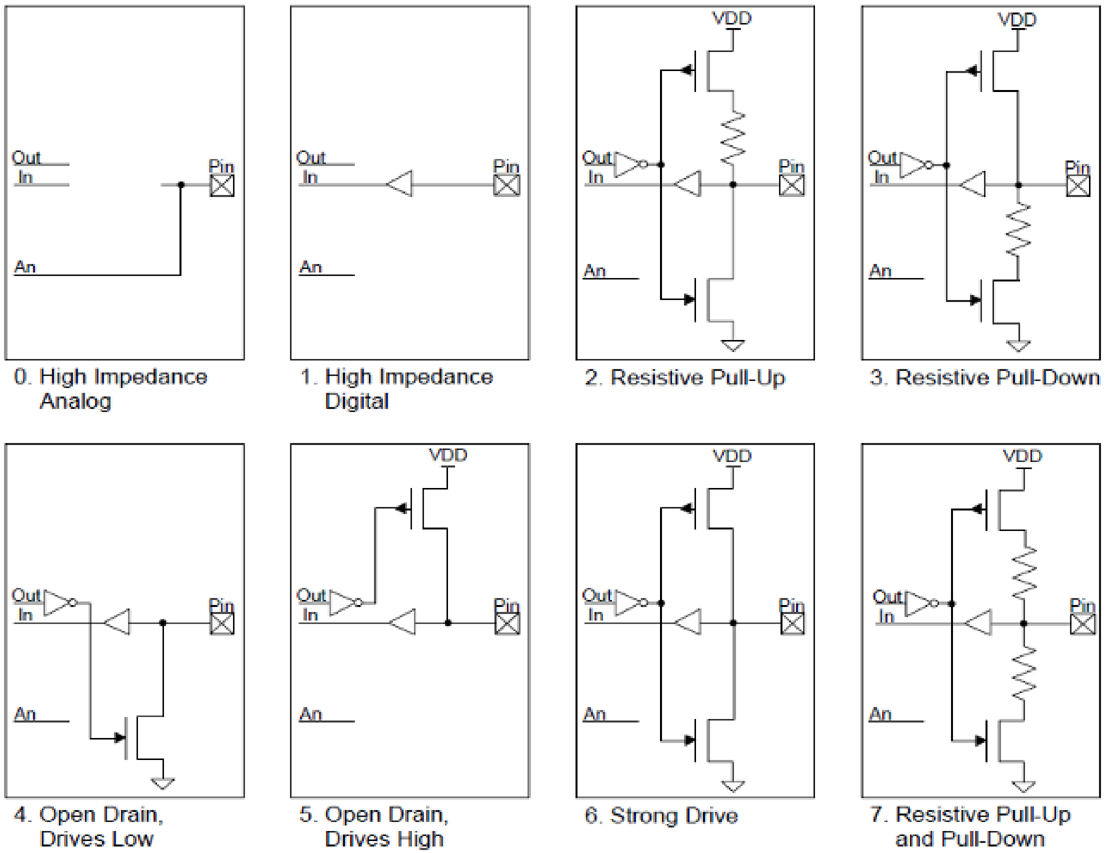


Figure 29-2. Analog Interconnect



||||||||............................

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	0x1fff8000 + [0..16383 * 0x1]
SRAM_CODE32K[0..8191]	Code System Memory Bank	0x1fffc000 + [0..8191 * 0x1]
SRAM_CODE16K[0..4095]	Code System Memory Bank	0x1fffe000 + [0..4095 * 0x1]
SRAM_CODE[0..4095]	Code System Memory Bank	0x1ffff000 + [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

Register Name	Purpose	Address
FLASH_DATA[0..262143]	DATA	[0..262143 * 0x1]
SRAM_CODE64K[0..16383]	Code System Memory Bank	0x1fff8000 + [0..16383 * 0x1]
SRAM_CODE32K[0..8191]	Code System Memory Bank	0x1fffc000 + [0..8191 * 0x1]
SRAM_CODE16K[0..4095]	Code System Memory Bank	0x1fffe000 + [0..4095 * 0x1]
SRAM_CODE[0..4095]	Code System Memory Bank	0x1ffff000 + [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

ca. 1540 Register

von denen 483 durch die Init. der anderen Register ihre Funktion erhalten

|||||||||||.......................

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	R/W:UUUUUUUU							
HW Access	R/W							
Retention	NONRET							
Name	F0							

FIFO 0

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

IIIIIIIIIII.........................

Complexer Chip:

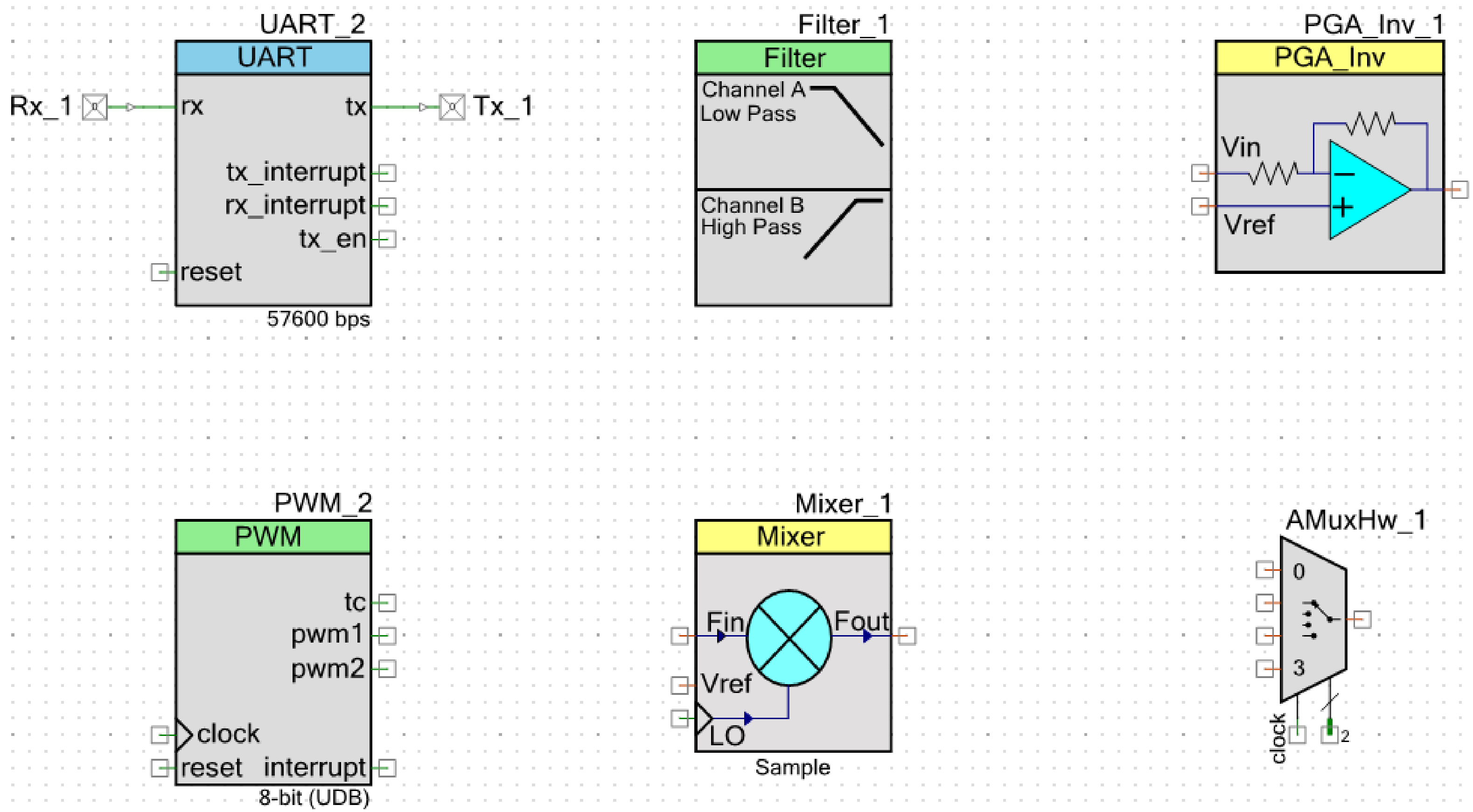
- UDB's
- Analoge Bauteile
- Digitaler Filter (DSP)
- Flexible IO
- Interconnect
- Pinzuweisung
- Hohe Anzahl an Registern

Application Programming Interface

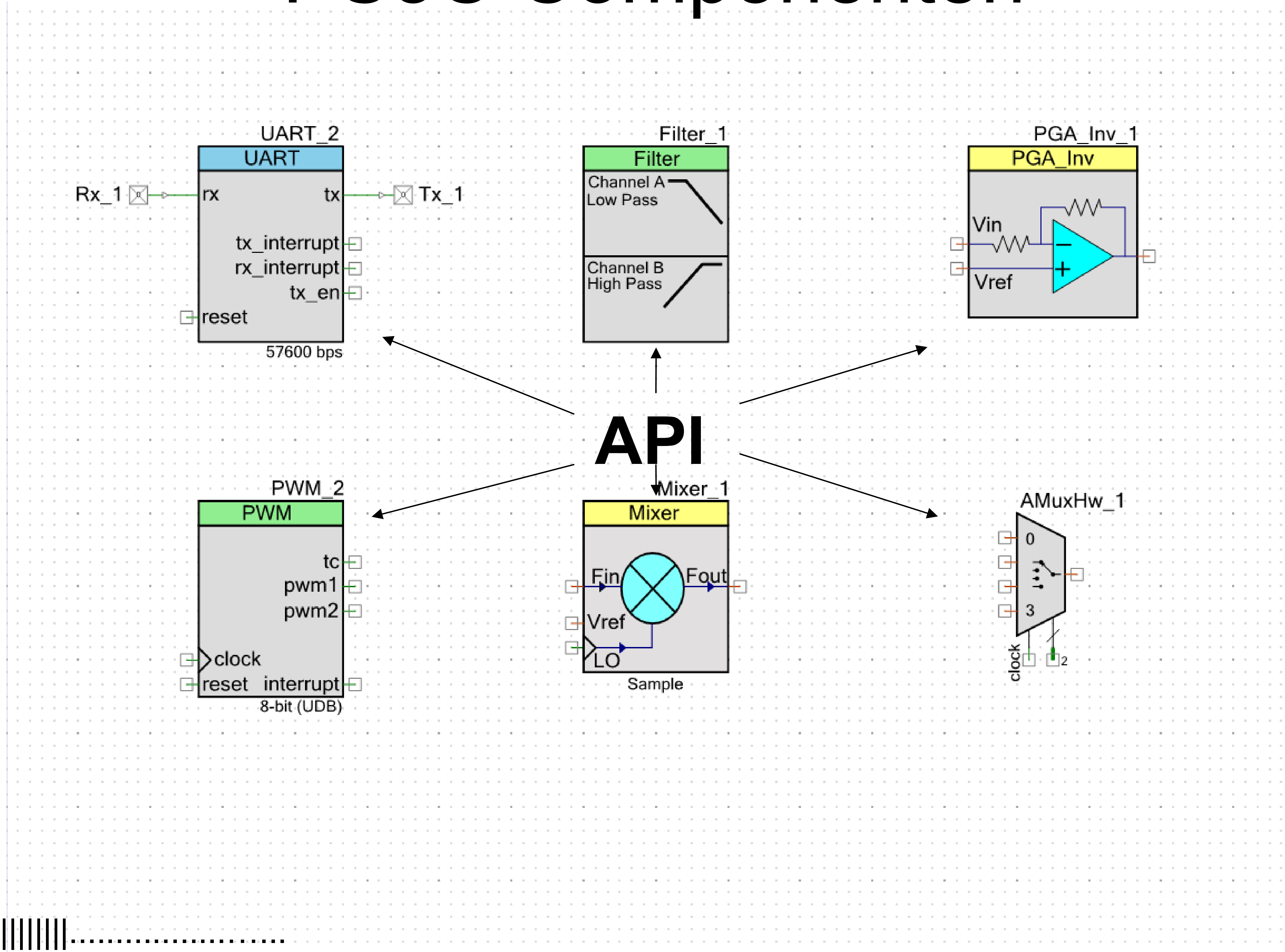
API

||||||||||||.........................

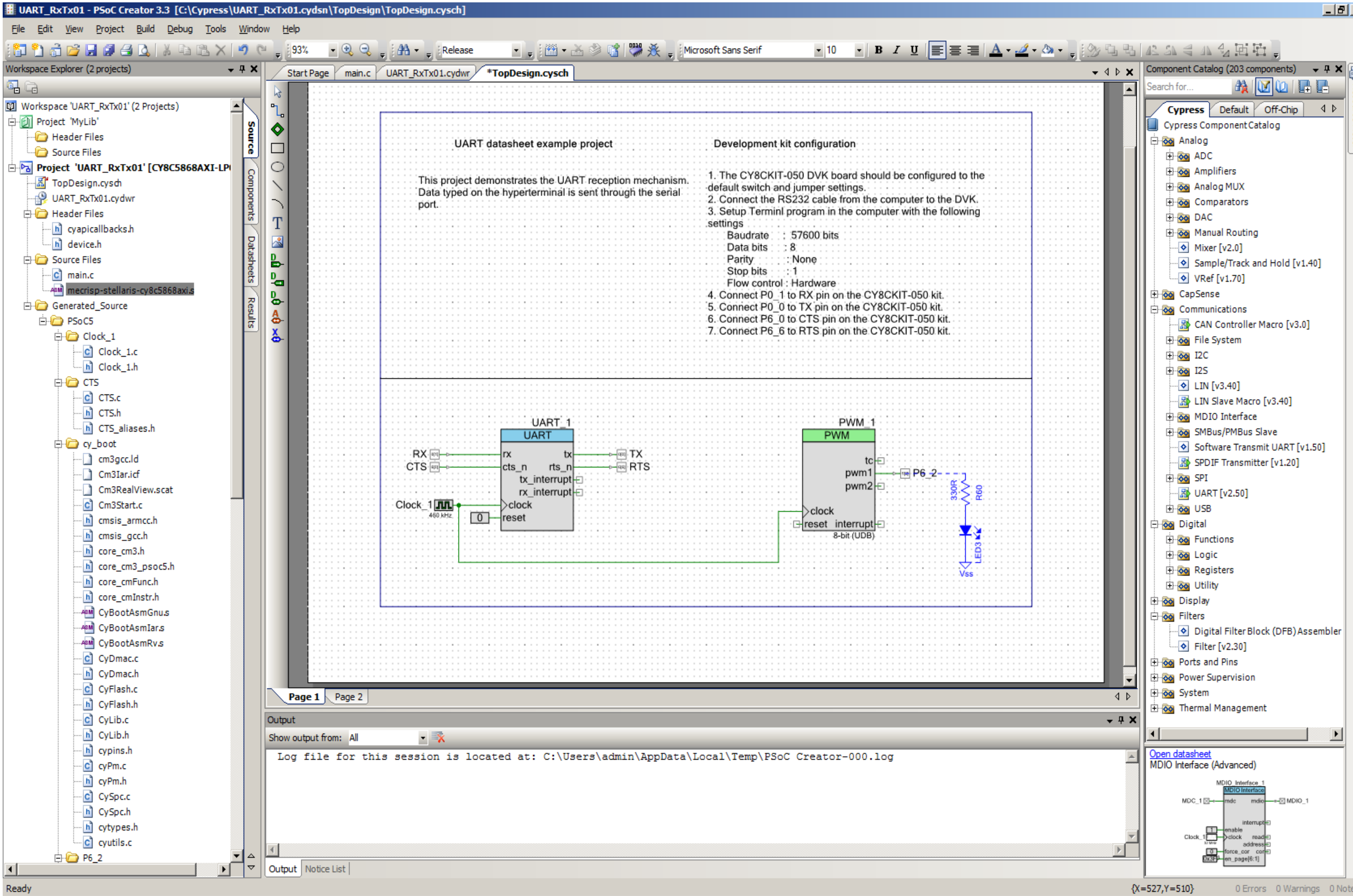
PSoC Componenten



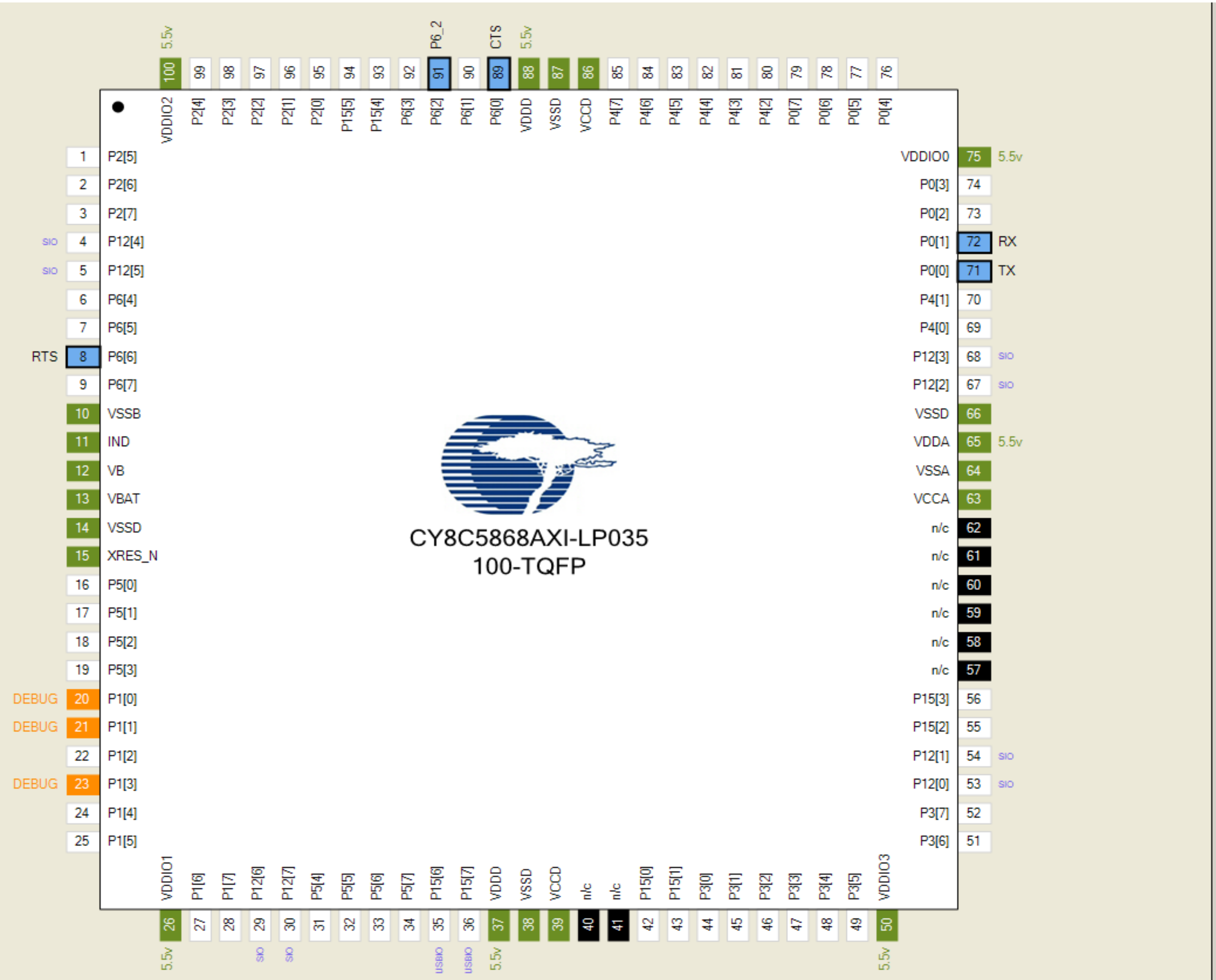
PSoC Componenten



PSoC Creator



Port/Pin-Zuordnung der Signale



Name	Port	Pin	Lock
CTS	P6[0]	89	☑
P6_2	P6[2]	91	☑
RTS	P6[6]	8	☑
RX	P0[1]	72	☑
TX	P0[0]	71	☑

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

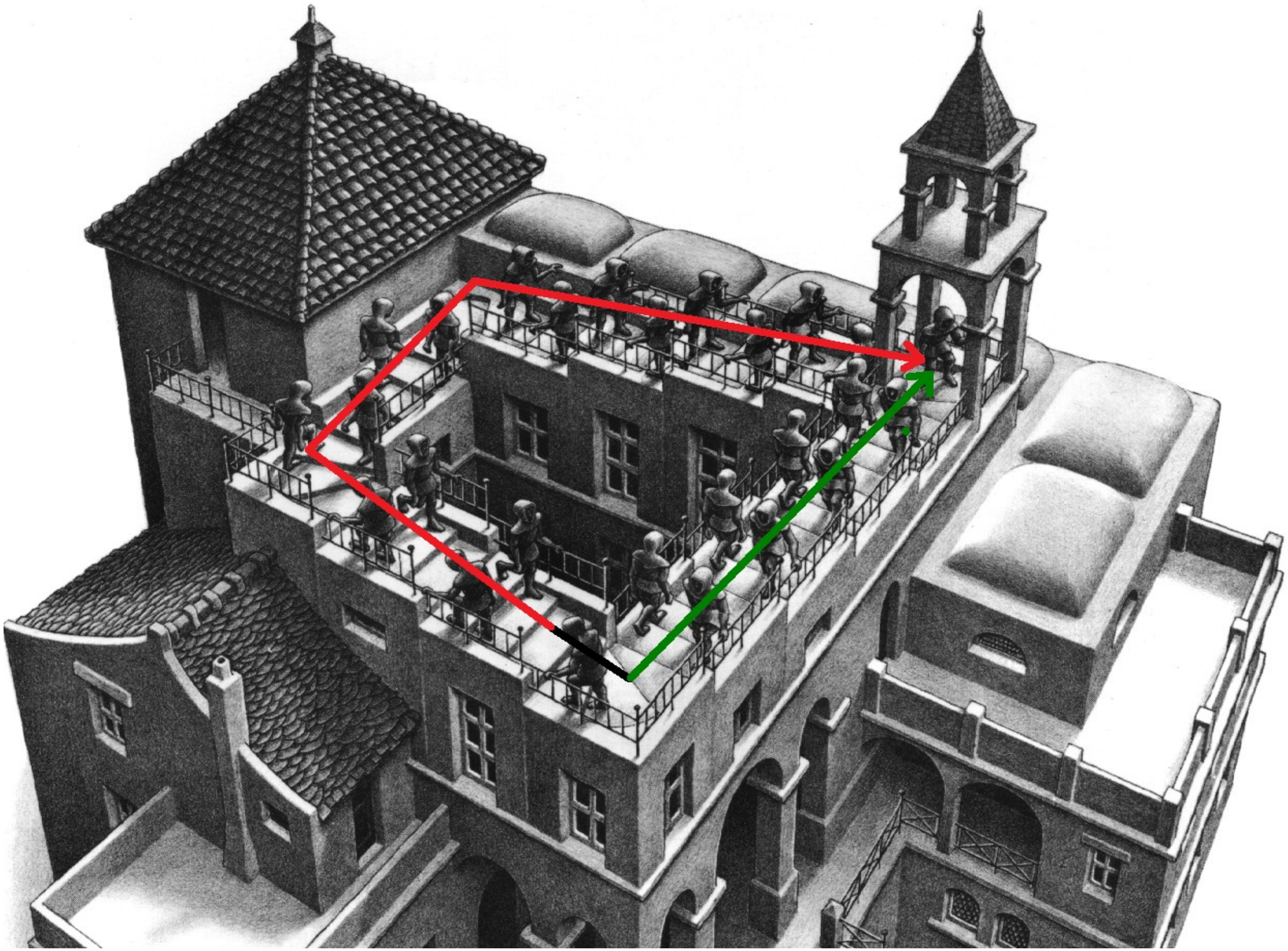
Schaltplan

Programmierung Flash

Datenblätter, Foren

||||||||||||||||||.................

Treppauf / Treppab



||||||||||||||||||||||……………..

KEY & EMIT
via revers engineering

```
     .equ Terminal_USART_Base,  0x40006400
.equ Terminal_USART_RX_SR, Terminal_USART_Base + 0x63
.equ Terminal_USART_RX_DR, Terminal_USART_Base + 0x43
.equ Terminal_USART_TX_SR, Terminal_USART_Base + 0x60
.equ Terminal_USART_TX_DR, Terminal_USART_Base + 0x40
```

```
@ -----------------------------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit"
serial_emit: @ ( c -- ) Emit one character
@ -----------------------------------------------------------------------------
   push {lr}

1: bl serial_qemit                                    @ 06.02.16/4
   cmp tos, #0
   drop
   beq 1b

   ldr r2, =Terminal_USART_TX_DR   @ 06.02.16/1
   strb tos, [r2]                            @ Output the character
   drop

   pop {pc}

@ -----------------------------------------------------------------------------
  Wortbirne Flag_visible, "serial-key"
serial_key: @ ( -- c ) Receive one character
@ -----------------------------------------------------------------------------
   push {lr}

1: bl serial_qkey
   cmp tos, #0
   drop
   beq 1b

   pushdatos
   ldr r2, =Terminal_USART_RX_DR   @ 06.02.16/1
   ldrb tos, [r2]                            @ Fetch the character

   pop {pc}

@ -----------------------------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit?"
serial_qemit:  @ ( -- ? ) Ready to send a character ?
@ -----------------------------------------------------------------------------
   push {lr}
   bl pause

   pushdaconst 0
   ldr r0, =Terminal_USART_TX_SR   @ 06.02.16/1
   ldr r1, [r0]                                       @ Fetch status
   movs r0, #TXE
   ands r1, r0
   beq 1f
     mvns tos, tos
1: pop {pc}

@ -----------------------------------------------------------------------------
  Wortbirne Flag_visible, "serial-key?"
serial_qkey:  @ ( -- ? ) Is there a key press ?
@ -----------------------------------------------------------------------------
   push {lr}
   bl pause

   pushdaconst 0
   ldr r0, =Terminal_USART_RX_SR   @ 06.02.16/1
   ldr r1, [r0]                                       @ Fetch status
   movs r0, #RXNE
   ands r1, r0
   beq 1f
     mvns tos, tos
1: pop {pc}
```

KEY & EMIT via API-Adressen

```
@ -----------------------------------------------------------
    Wortbirne Flag_visible, "UART_1_PutChar"
UART_1_PutChar:
@ -----------------------------------------------------------
    popda r0

    ldr r1, =#0x00000434          @ execute
  mov pc, r1

@ -----------------------------------------------------------
    Wortbirne Flag_visible, "UART_1_GetChar"
UART_1_GetChar:
@ -----------------------------------------------------------
    ldr r0, =#0x00000400          @ execute
  mov pc, r0

    pushda r0

@ -----------------------------------------------------------
    Wortbirne Flag_visible, "UART_1_ReadTxStatus"
UART_1_ReadTxStatus:
@ -----------------------------------------------------------
    ldr r0, =#0x00000428          @ execute
  mov pc, r0

    pushda r0


@ -----------------------------------------------------------
    Wortbirne Flag_visible, "UART_1_ReadRxStatus"
UART_1_ReadRxStatus:
@ -----------------------------------------------------------
    ldr r0, =#0x000003f0          @ execute
  mov pc, r0

    pushda r0

```

```
@ -----------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit"
serial_emit: @ ( c -- ) Emit one character
@ -----------------------------------------------------------
  push {lr}

     bl UART_1_PutChar

  pop {pc}

@ -----------------------------------------------------------
  Wortbirne Flag_visible, "serial-key"
serial_key: @ ( -- c ) Receive one character
@ -----------------------------------------------------------
  push {lr}

1:bl pause

     bl UART_1_GetChar

     cmp tos, #0
     beq 1b
     pushdatos
     mov tos, r0

  pop {pc}

@ -----------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit?"
serial_qemit:  @ ( -- ? ) Ready to send a character ?
@ -----------------------------------------------------------
  push {lr}
  bl pause

     bl UART_1_ReadRxStatus

     movs r0, #TXE
     ands tos, r0

  pop {pc}

@ -----------------------------------------------------------
  Wortbirne Flag_visible, "serial-key?"
serial_qkey:  @ ( -- ? ) Is there a key press ?
@ -----------------------------------------------------------
   push {lr}
   bl pause

     bl UART_1_ReadTxStatus

     movs r0, #RXNE
     ands tos, r0

1:pop {pc}
```

Mecrisp.bin als char Tabelle in C eingebunden

```c
/* C:\FORTH\Mecrisp\mecrisp-stellaris-2.2.0\cy8c5868axi\mecrisp-stellaris-cy8c5868axi.bin (05.08.2017 00:13:28)
   StartOffset: 00000000, EndeOffset: 00003CAB, L∑nge: 00003CAC */

const unsigned char  mecrisp[] __attribute__ ((section ("FORTH"))) = {
	0x30, 0x04, 0x00, 0x20, 0x0B, 0xBB, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00,
	0x4B, 0xB9, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00, 0x4B, 0xB9, 0x00, 0x00,
	0x4B, 0xB9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x23, 0xB9, 0x00, 0x00, 0x30, 0x04, 0x00, 0x20, 0x0B, 0xBB, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0xA1, 0xB9, 0x00, 0x00, 0xC9, 0xB9, 0x00, 0x00, 0xF1, 0xB9, 0x00, 0x00,
	0x19, 0xBA, 0x00, 0x00, 0x41, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x67, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x8F, 0xBA, 0x00, 0x00, 0xB7, 0xBA, 0x00, 0x00,
	0xDF, 0xBA, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00, 0x79, 0xB9, 0x00, 0x00,
	0x79, 0xB9, 0x00, 0x00, 0xB6, 0x81, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x2D,
	0x2D, 0x2D, 0x20, 0x4D, 0x65, 0x63, 0x72, 0x69, 0x73, 0x70, 0x2D, 0x53,
	0x74, 0x65, 0x6C, 0x6C, 0x61, 0x72, 0x69, 0x73, 0x20, 0x43, 0x6F, 0x72,
	0x65, 0x20, 0x2D, 0x2D, 0x2D, 0x00, 0xCE, 0x81, 0x00, 0x00, 0x42, 0x80,
	0x04, 0x32, 0x64, 0x75, 0x70, 0x00, 0x38, 0x68, 0x47, 0xF8, 0x04, 0x6D,
	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

```
@ -----------------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit"
serial_emit: @ ( c -- ) Emit one character
@ -----------------------------------------------------------------
  push {lr}

    popda r0                                                @
    bl UART_1_PutChar                                       @

  pop {pc}

@ -----------------------------------------------------------------
  Wortbirne Flag_visible, "serial-key"
serial_key: @ ( -- c ) Receive one character
@ -----------------------------------------------------------------
  push {lr}

1:  bl pause

    bl UART_1_GetChar                                       @

    @ popda r0                                              @
    cmp r0, #0                                              @
    beq 1b

    pushda r0                                               @

  pop {pc}

@ -----------------------------------------------------------------
  Wortbirne Flag_visible, "serial-emit?"
serial_qemit:  @ ( -- ? ) Ready to send a char
@ -----------------------------------------------------------------
  push {lr}
    bl pause

    bl UART_1_ReadRxStatus

    pushda r0                                               @
    movs r0, #TXE
    ands tos, r0

  pop {pc}

@ -----------------------------------------------------------------
  Wortbirne Flag_visible, "serial-key?"
serial_qkey:  @ ( -- ? ) Is there a key press
@ -----------------------------------------------------------------
   push {lr}
    bl pause

    bl UART_1_ReadTxStatus

    pushda r0                                               @
    movs r0, #RXNE
    ands tos, r0

1:pop {pc}
```

HardwareAbstractionLayer HAL.s

```
@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "LEDon()"                         @ ( --- )
_LEDon:
@ -----------------------------------------------------------------------------------------
     b LEDon


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "LEDoff()"                        @ ( --- )
_LEDoff:
@ -----------------------------------------------------------------------------------------
     b LEDoff


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WritePeriod()"             @ ( c --- )
_PWM_1_WritePeriod:
@ -----------------------------------------------------------------------------------------
     popda r0                                  @ pop character from stack to input-register r0
     b PWM_1_WritePeriod


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WriteCompare1()"      @ ( c --- )
_PWM_1_WriteCompare1:
@ -----------------------------------------------------------------------------------------
     popda r0                                  @ pop character from stack to input-register r0
     b PWM_1_WriteCompare1


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WriteCompare2()"      @ ( c --- )
_PWM_1_WriteCompare2:
@ -----------------------------------------------------------------------------------------
     popda r0                                  @ pop character from stack to input-register r0
     b PWM_1_WriteCompare2

```

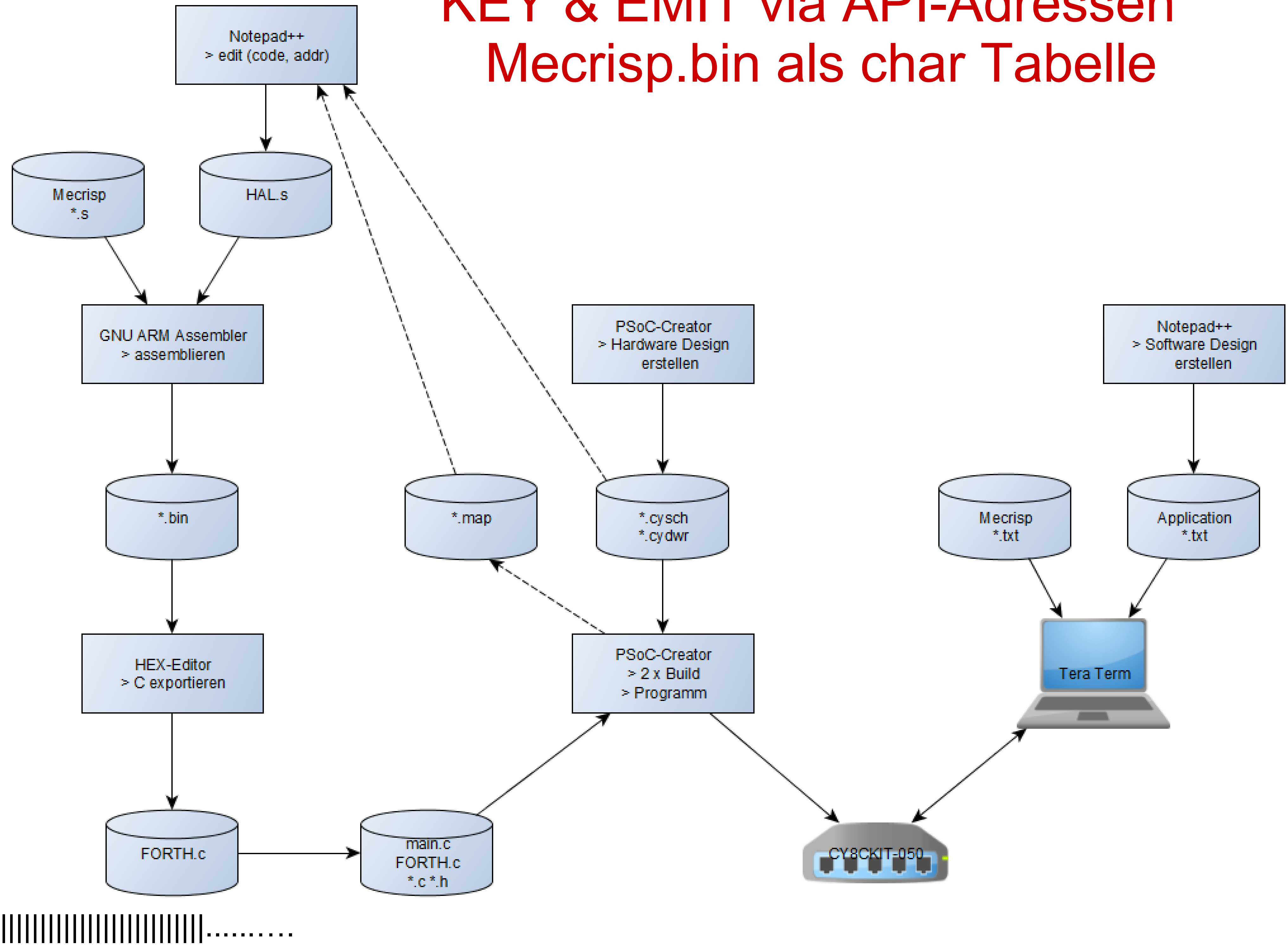
HardwareAbstractionLayer HAL.s

```
@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "LEDon()"                       @ ( --- )
_LEDon:
@ -----------------------------------------------------------------------------------------
     b LEDon


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "LEDoff()"                      @ ( --- )
_LEDoff:
@ -----------------------------------------------------------------------------------------
     b LEDoff


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WritePeriod()"           @ ( c --- )
_PWM_1_WritePeriod:
@ -----------------------------------------------------------------------------------------
     popda r0                           @ pop character from stack to input-register r0
     b PWM_1_WritePeriod


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WriteCompare1()"    @ ( c --- )
_PWM_1_WriteCompare1:
@ -----------------------------------------------------------------------------------------
     popda r0                           @ pop character from stack to input-register r0
     b PWM_1_WriteCompare1


@ -----------------------------------------------------------------------------------------
     Wortbirne Flag_visible, "PWM_1_WriteCompare2()"    @ ( c --- )
_PWM_1_WriteCompare2:
@ -----------------------------------------------------------------------------------------
     popda r0                           @ pop character from stack to input-register r0
     b PWM_1_WriteCompare2

```

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

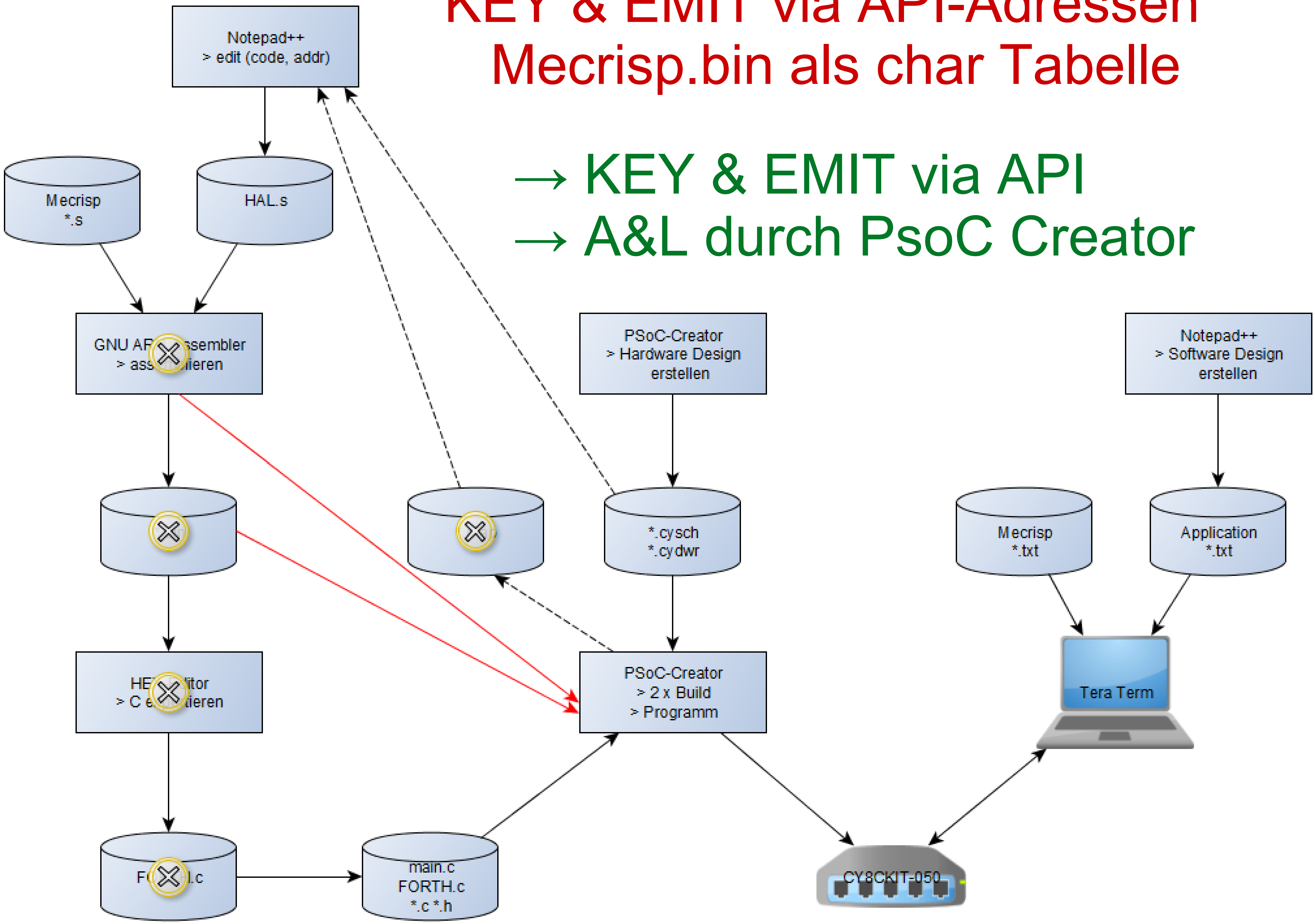
||||||||||||||||||||||||||||............

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



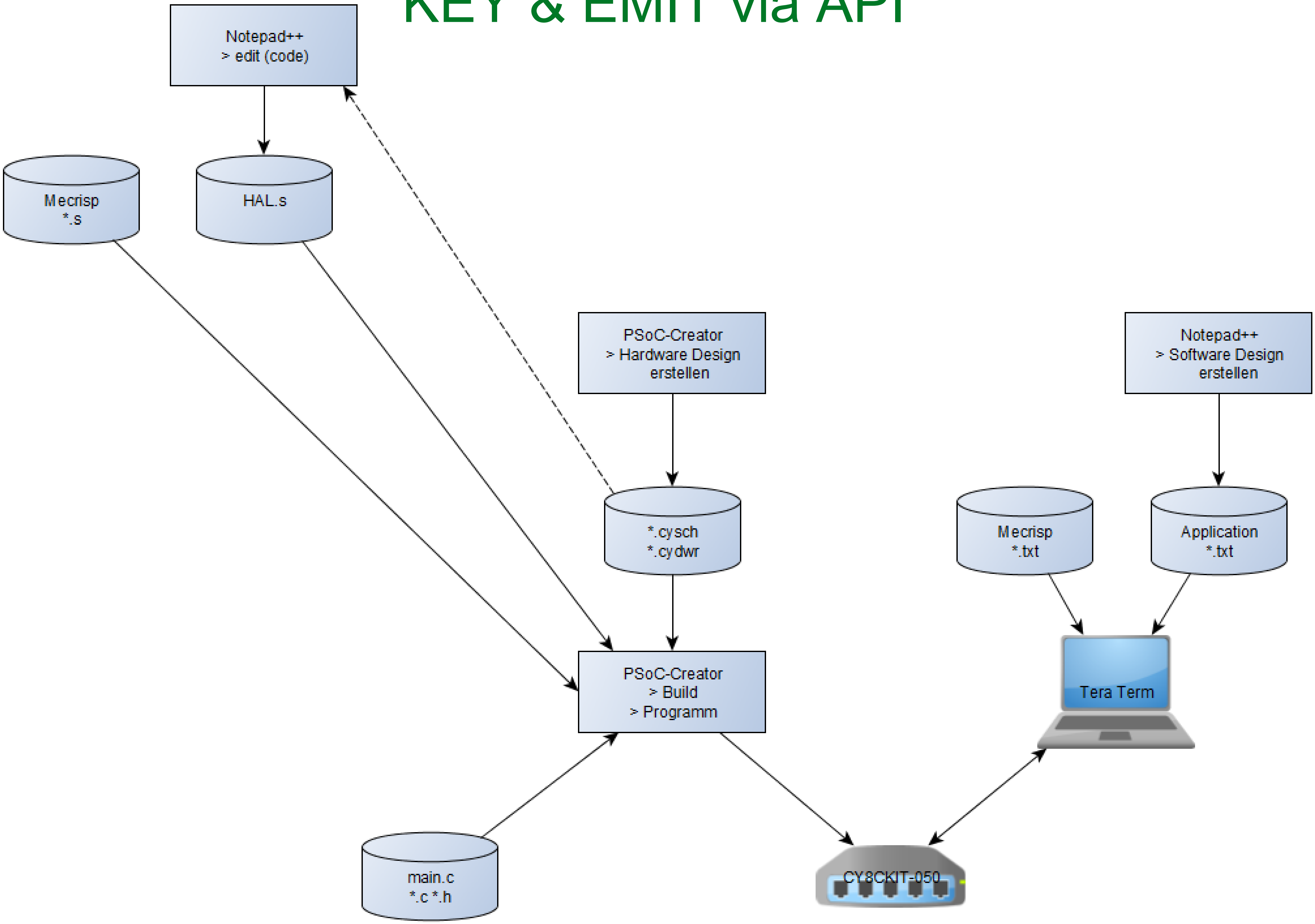
KEY & EMIT via API-Adressen
Mecrisp.bin als char Tabelle

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



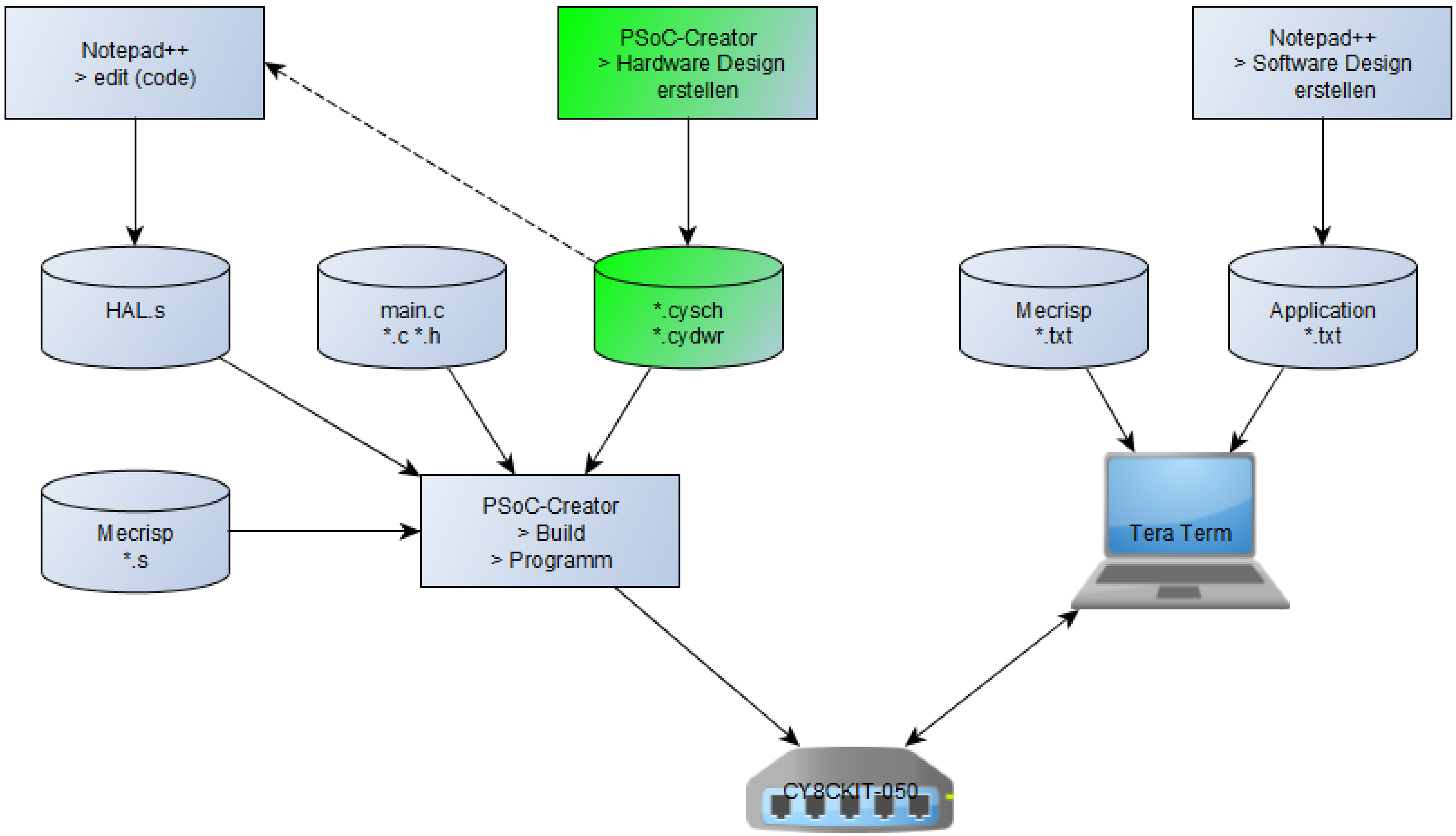
||||||||||||||||||||||||||||||||……..

KEY & EMIT via API



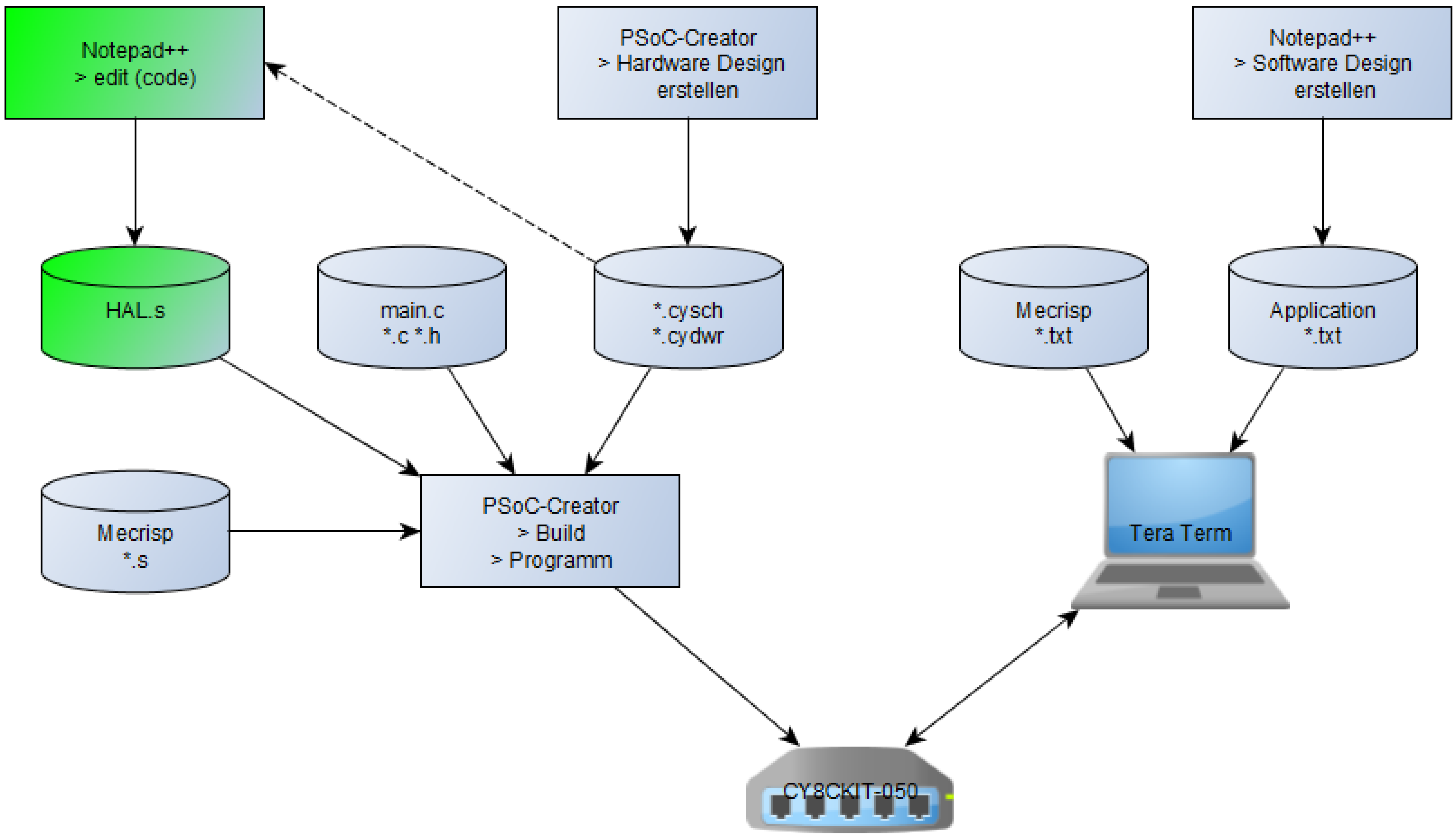
||||||||||||||||||||||||||||||........

Schaltplan erstellen

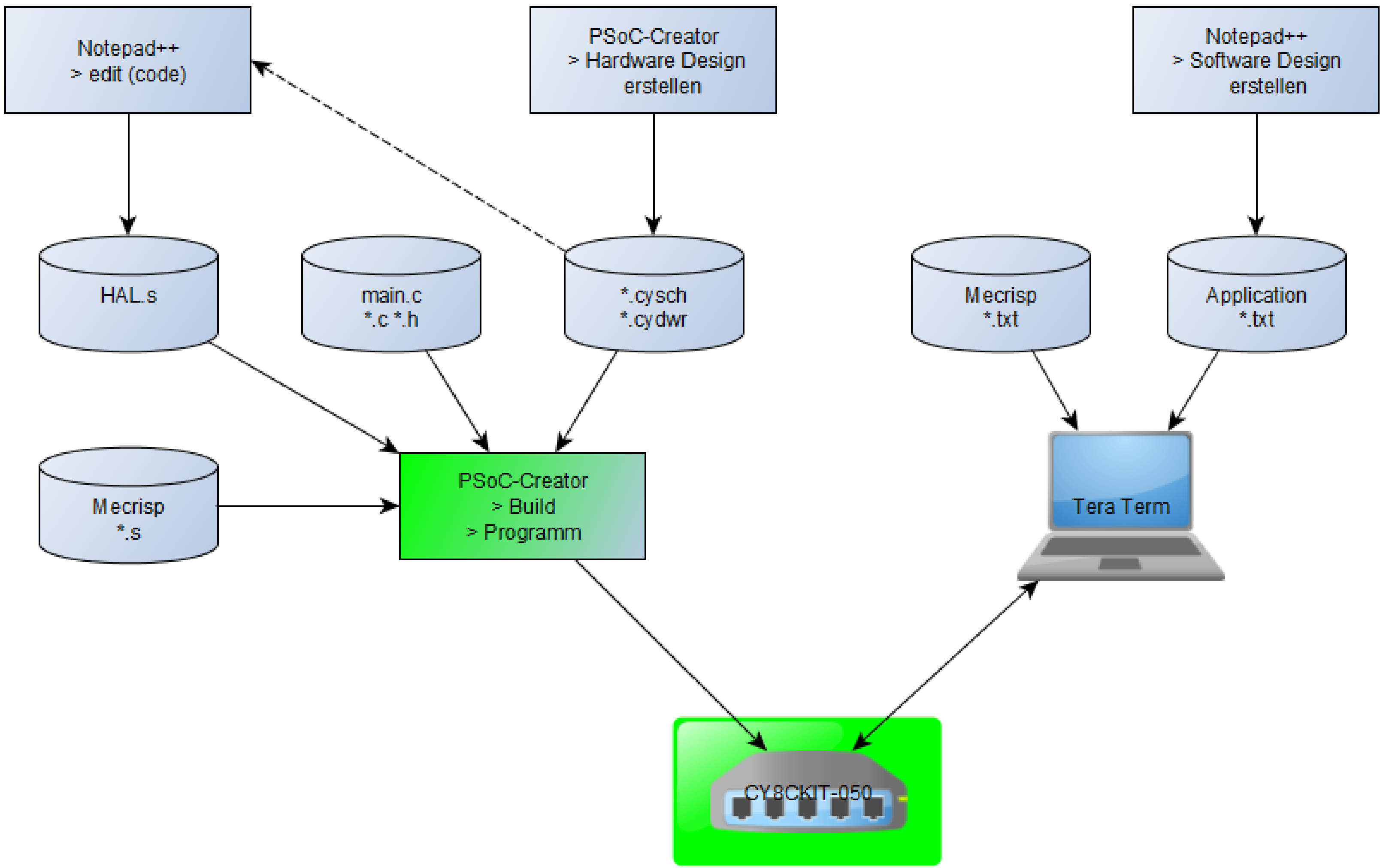


||||||||||||||||||||||||||||||||.......

Componenten API's in HAL.s eintragen

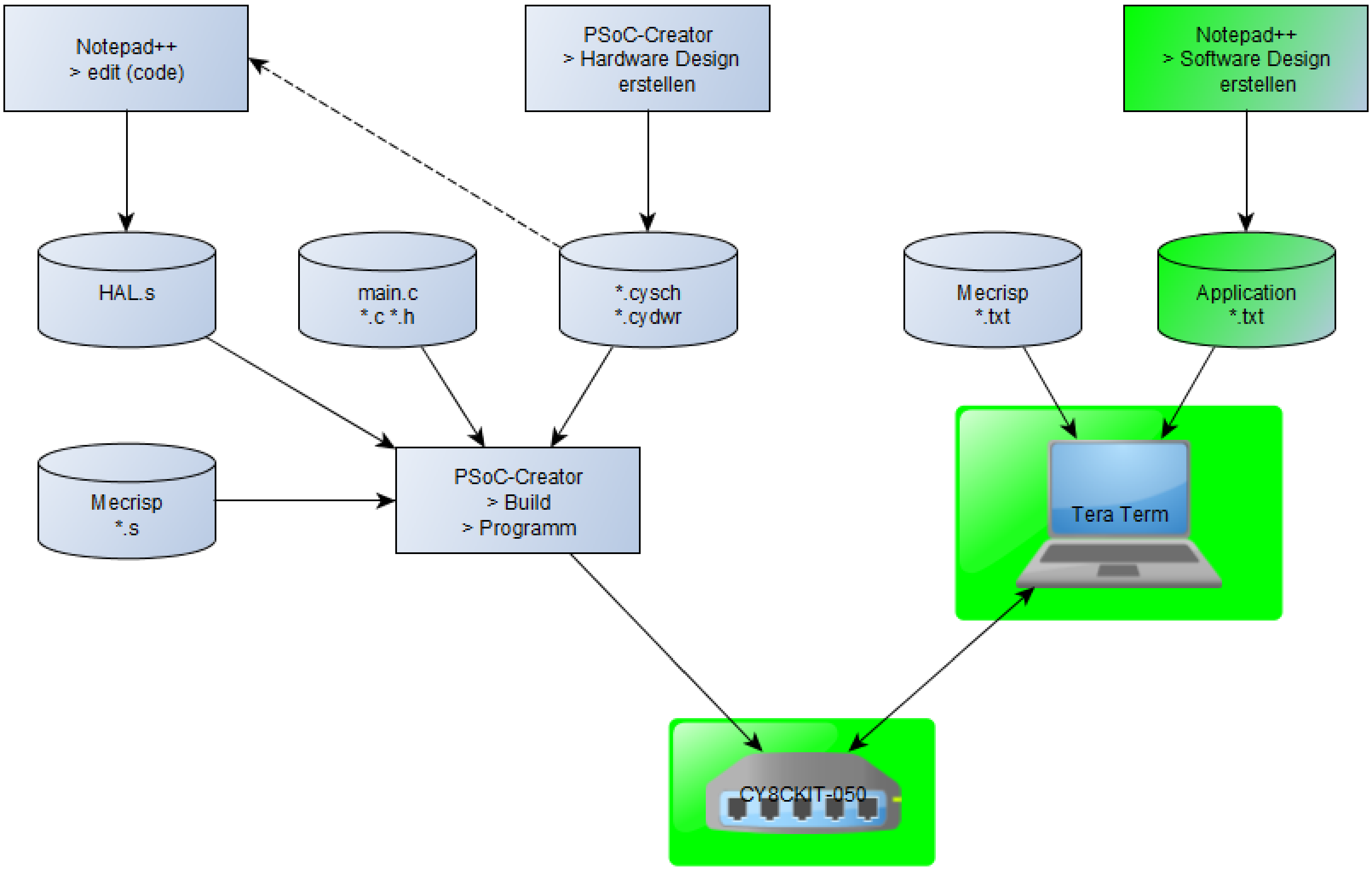


Hardware / Software generieren und laden



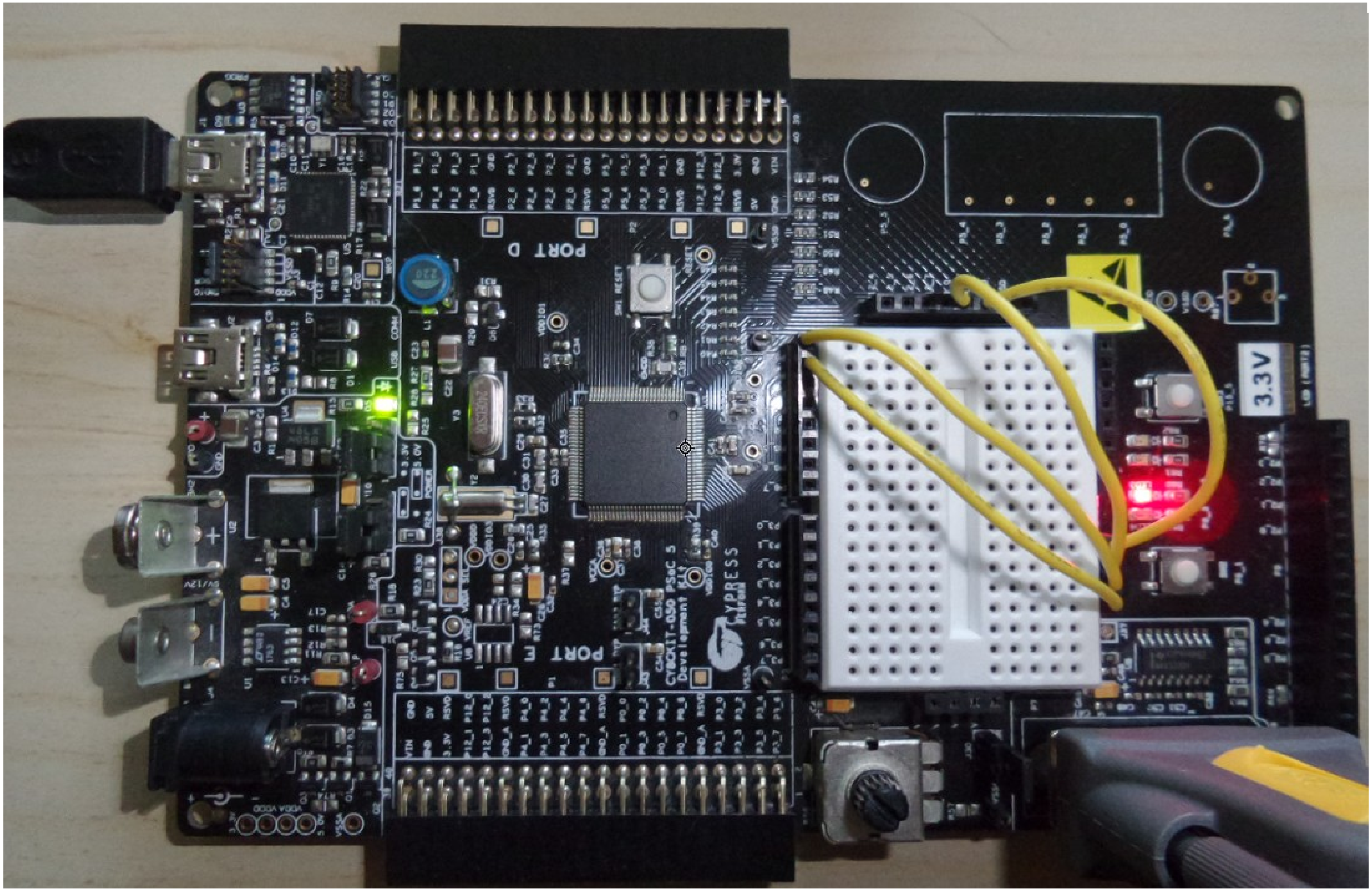
|||||||||||||||||||||||||||||||||||.....

FORTH


Notepad++
> edit (code)
PSoC-Creator
> Hardware Design
erstellen
Notepad++
> Software Design
erstellen
HAL.s
main.c
*.c *.h
*.cysch
*.cydwr
Mecrisp
*.txt
Application
*.txt
Mecrisp
*.s
PSoC-Creator
> Build
> Programm
Tera Term
CY8CKIT-050

||||||||||||||||||||||||||||||||||||||||....

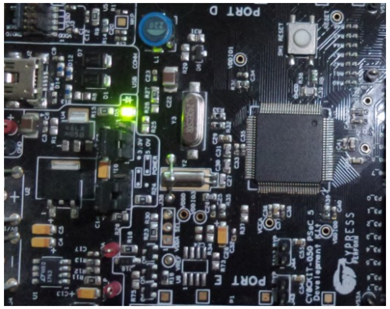
3. Das Evaluation-Board

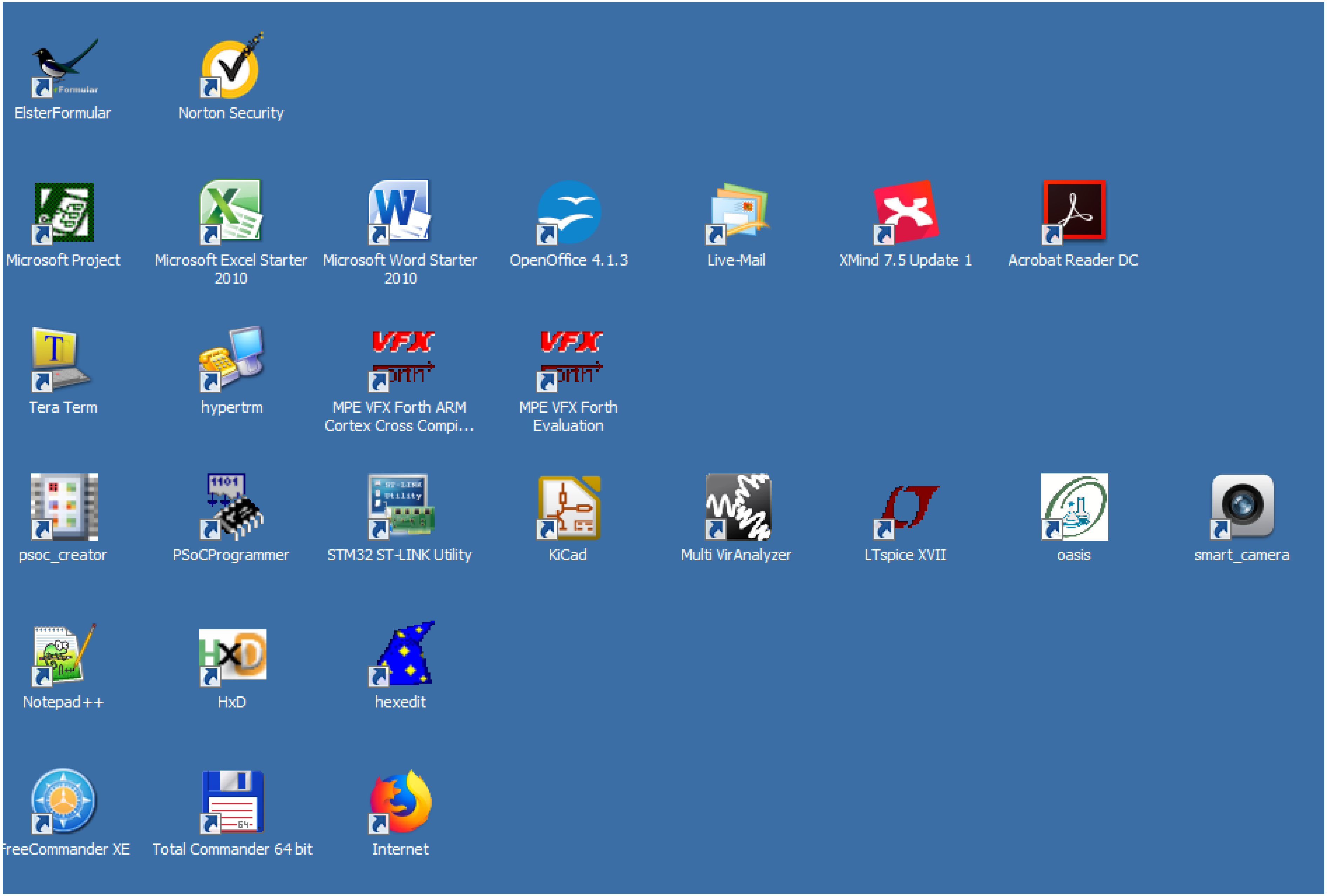


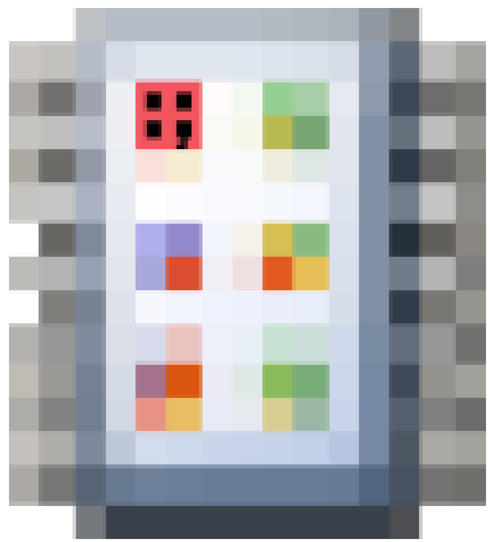
CY8C5868AXI: 100 Pin   10,-   1 cm²

||||||||||||||||||||||||||||||||||||...

?




ElsterFormular
Norton Security
Microsoft Project
Microsoft Excel Starter 2010
Microsoft Word Starter 2010
OpenOffice 4.1.3
Live-Mail
XMind 7.5 Update 1
Acrobat Reader DC
Tera Term
hypertrm
MPE VFX Forth ARM Cortex Cross Compi...
MPE VFX Forth Evaluation
psoc_creator
PSoCProgrammer
STM32 ST-LINK Utility
KiCad
Multi VirAnalyzer
LTspice XVII
oasis
smart_camera
Notepad++
HxD
hexedit
FreeCommander XE
Total Commander 64 bit
Internet



C°y°SWAP

||||||||||||||||||||||||||||||||||||||||