Ziel

- „lesbarer“ Zugriff auf die Register der Peripherie
- einfaches Abbilden der Register
- bessere Lesbarkeit durch weniger Schreibarbeit
- Wunsch-Syntax: “Peripherie.Registername“
  z.B. `UART3.DATA`

Peripherie erzeugen

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0401_0003h
4h	Parameter Register (PARAM)	32	RO	0000_0202h
8h	LPUART Global Register (GLOBAL)	32	RW	0000_0000h
Ch	LPUART Pin Configuration Register (PINCFG)	32	RW	0000_0000h
10h	LPUART Baud Rate Register (BAUD)	32	RW	0F00_0004h
14h	LPUART Status Register (STAT)	32	RW	00C0_0000h
18h	LPUART Control Register (CTRL)	32	RW	0000_0000h
1Ch	LPUART Data Register (DATA)	32	RW	0000_1000h
20h	LPUART Match Address Register (MATCH)	32	RW	0000_0000h
24h	LPUART Modem IrDA Register (MODIR)	32	RW	0000_0000h
28h	LPUART FIFO Register (FIFO)	32	RW	00C0_0011h
2Ch	LPUART Watermark Register (WATER)	32	RW	0000_0000h

```
PERIPH UART
  REG 0x00 VERID
  REG 0x04 PARAM
  REG 0x08 GLOBAL
  REG 0x0C PINCFG
  REG 0x10 BAUD
  REG 0x14 STAT
  REG 0x18 CTRL
  REG 0x1C DATA
  REG 0x20 MATCH
  REG 0x24 MODIR
  REG 0x28 FIFO
  REG 0x2C WATER
END-PERIPH

0x4018C000 UART UART3
```

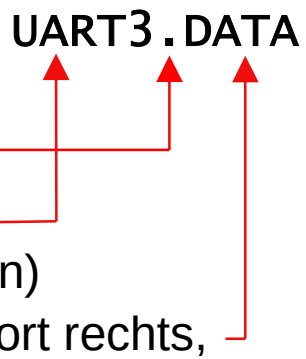
die Liste der Register

```
PERIPH UART
  REG 0x00 VERID
  REG 0x04 PARAM
  REG 0x08 GLOBAL
  REG 0x0C PINCFG
  REG 0x10 BAUD
  REG 0x14 STAT
  REG 0x18 CTRL
  REG 0x1C DATA
  REG 0x20 MATCH
  REG 0x24 MODIR
  REG 0x28 FIFO
  REG 0x2C WATER
END-PERIPH

0x4018C000 UART UART3
```

```
' uart3 >body ? 4018C000  ok
' uart3 >body cell+ ? 2001280C  ok
2001280c 80 dump
            0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
 20012800                                          20 56 45 52                VER
 20012810   49 44 20 30 78 30 20 50 41 52 41 4D 20 30 78 34  ID 0x0 PARAM 0x4
 20012820   20 47 4C 4F 42 41 4C 20 30 78 38 20 50 49 4E 43   GLOBAL 0x8 PINC
 20012830   46 47 20 30 78 43 20 42 41 55 44 20 30 78 31 30  FG 0xC BAUD 0x10
 20012840   20 53 54 41 54 20 30 78 31 34 20 43 54 52 4C 20   STAT 0x14 CTRL
 20012850   30 78 31 38 20 44 41 54 41 20 30 78 31 43 20 4D  0x18 DATA 0x1C M
 20012860   41 54 43 48 20 30 78 32 30 20 4D 4F 44 49 52 20  ATCH 0x20 MODIR
 20012870   30 78 32 34 20 46 49 46 4F 20 30 78 32 38 20 57  0x24 FIFO 0x28 W
 20012880   41 54 45 52 20 30 78 32 43 20 00 00              ATER 0x2C ..
 ok
```

STRUCT?



- gibt es einen Punkt im String?
- suche mit FIND das xt vom Wort links
  (jetzt ist auch die Registerliste gefunden)
- suche in der Registerliste nach dem Wort rechts,
  es wird der Offset auf den Stack gelegt
- jetzt gibt es das xt vom Wort links
  und den Offset vom Wort rechts
- das xt wird ausgeführt, es muss! eine Adresse zurückgeben
- die gesuchte Registeradresse ist Adresse plus Offset

Beispiel UART

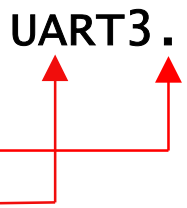
```
: INIT.UART3   (   --   )
          0 UART3.CTRL ! \ disable Sender u. Empfänger
0x0402008B UART3.BAUD ! \ setze Oversampling und Baudrate
0x000C0000 UART3.CTRL ! \ enable Sender u. Empfänger
;
```

Syntax am Beispiel UART

```
: INIT.UART3   (   --   )
          0 UART3.CTRL ! \ disable Sender u. Empfänger
0x0402008B UART3.BAUD ! \ setze Oversampling und Baudrate
0x000C0000 UART3.CTRL ! \ enable Sender u. Empfänger
;

: INIT.UART3  (   --   )
       UART3. (   )
          0 .CTRL !
0x0402008B .BAUD !
0x000C0000 .CTRL !
;
```

STRUCT?



- gibt es einen Punkt im String?
- suche mit FIND das xt vom Wort links
  (jetzt ist auch die Registerliste gefunden)
- es gibt aber keinen String rechts vom Punkt

  Für diesen Fall gilt:
- Führe xt aus und “merke“ die zurückgegebene Adresse sowie die gefundene Registerliste
- kein Stackeffekt!

STRUCT?

- gibt es einen Punkt im String?
- der String beginnt mit einen Punkt ,
  d.h. es gibt keinen String links vom Punkt

  Für diesen Fall gilt:
- verwende die “gemerkte“ Adresse und Registerliste
- suche in der Registerliste nach dem Wort rechts,
  es wird der Offset auf den Stack gelegt
- die gesuchte Registeradresse ist Adresse plus Offset

Syntax am Beispiel UART

```
: INIT.UART3   (   --   )
          0 UART3.CTRL ! \ disable Sender u. Empfänger
0x0402008B UART3.BAUD ! \ setze Oversampling und Baudrate
0x000C0000 UART3.CTRL ! \ enable Sender u. Empfänger
;

: INIT.UART3  (   --   )
       UART3. (   )
          0 .CTRL !
0x0402008B .BAUD !
0x000C0000 .CTRL !
;

: INIT.UART  ( addr --   ) \ addr ist Basisadresse von UART
UART. ( addr -- addr )
( addr )           0 OVER .CTRL !
( addr ) 0x0402008B OVER .BAUD !
( addr ) 0x000C0000 OVER .CTRL !
( addr ) DROP
;
```

```
: PREPARE.LPSPI5-DMA  ( count --   )
\ Tx Channel
0x20000000 DMAMUX0.CHCFG5 !  \ Mux channel ist Always On
0xA0000000 DMAMUX0.CHCFG5 !  \ Mux channel ist Enabled
5 DMA0.ERQ RESET.BIT         \ disable Requests für Channel 5
5 DMA0.INT SET.BIT           \ lösche evtl. noch anstehendes Interrupt Flag
5 DMA0.ERR SET.BIT           \ lösche evtl. noch anstehendes Error Flag
0x0490 DMA0.CR !             \ setze EMLM (Enable Minor Loop Mapping) u. HOE (Halt on Error)
TxTCD-LPSPI5 8 CELL-ERASE    \ lösche den Transfer Control Descriptor
TxTCD-LPSPI5.
TXBUFFER-SPI5 .SADDR !       \ setze Source Address
1             .SOFF W!       \ es wird Byteweise aus dem TxBuffer gelesen
0x0000        .ATTR W!       \ source data transfer size u. destination data transfer size ist 8-bit
1             .NBYTES !      \ probehalber Anzahl Bytes
LPSPI5.TDR    .DADDR !       \ Destination address ist das Transmit Data Register
0             .DOFF W!       \ Destination address soll nicht weiterzählen!
( count ) DUP
DUP .CITER W! .BITER W!      \ Major loop ist 1, die beiden müssen gleich sein
0             .DLASTSGA !    \ Last destination address adjustment wird nicht benötigt
8             .CSR W!        \ DREQ
\ Rx Channel
0x20000000 DMAMUX0.CHCFG6 !
0xA0000000 DMAMUX0.CHCFG6 !
6 DMA0.ERQ RESET.BIT
6 DMA0.INT SET.BIT
6 DMA0.ERR SET.BIT
RxTCD-LPSPI5 8 CELL-ERASE
RxTCD-LPSPI5.
( count )
LPSPI5.RDR    .SADDR !
0             .SOFF W!       \ Source address soll nicht weiterzählen
0x0000        .ATTR W!       \ source data transfer size u. destination data transfer size ist 8-bit
```