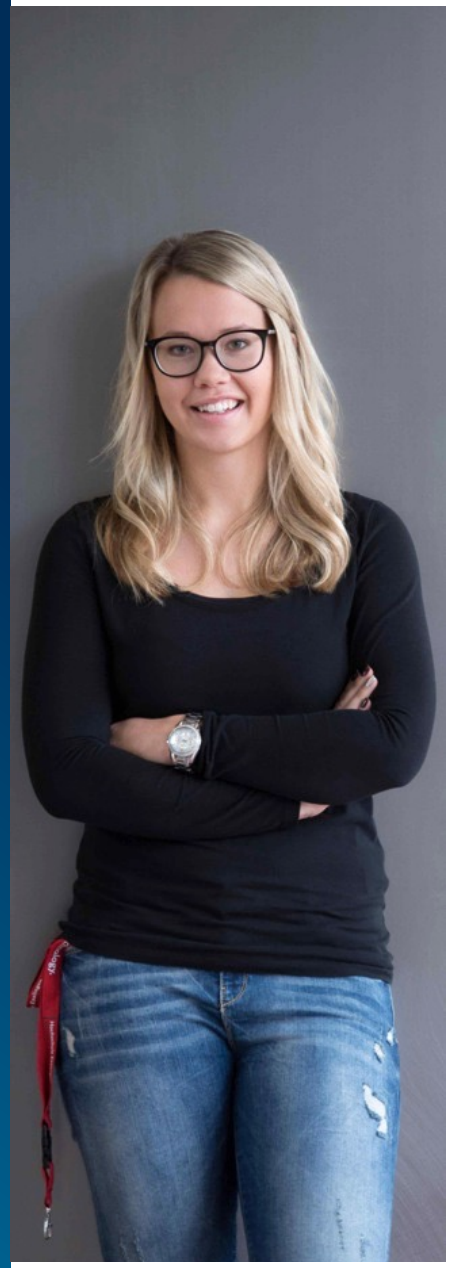
HOCHSCHULE
ESSLINGEN



LECTURE COMPUTER ARCHITECTURE

PERIPHERALS, DIGITAL/ ANALOG CONVERSION AND TIMED PERIPHERALS

RAINER KELLER

HOCHSCHULE
ESSLINGEN

CONTENT

HOCHSCHULE ESSLINGEN

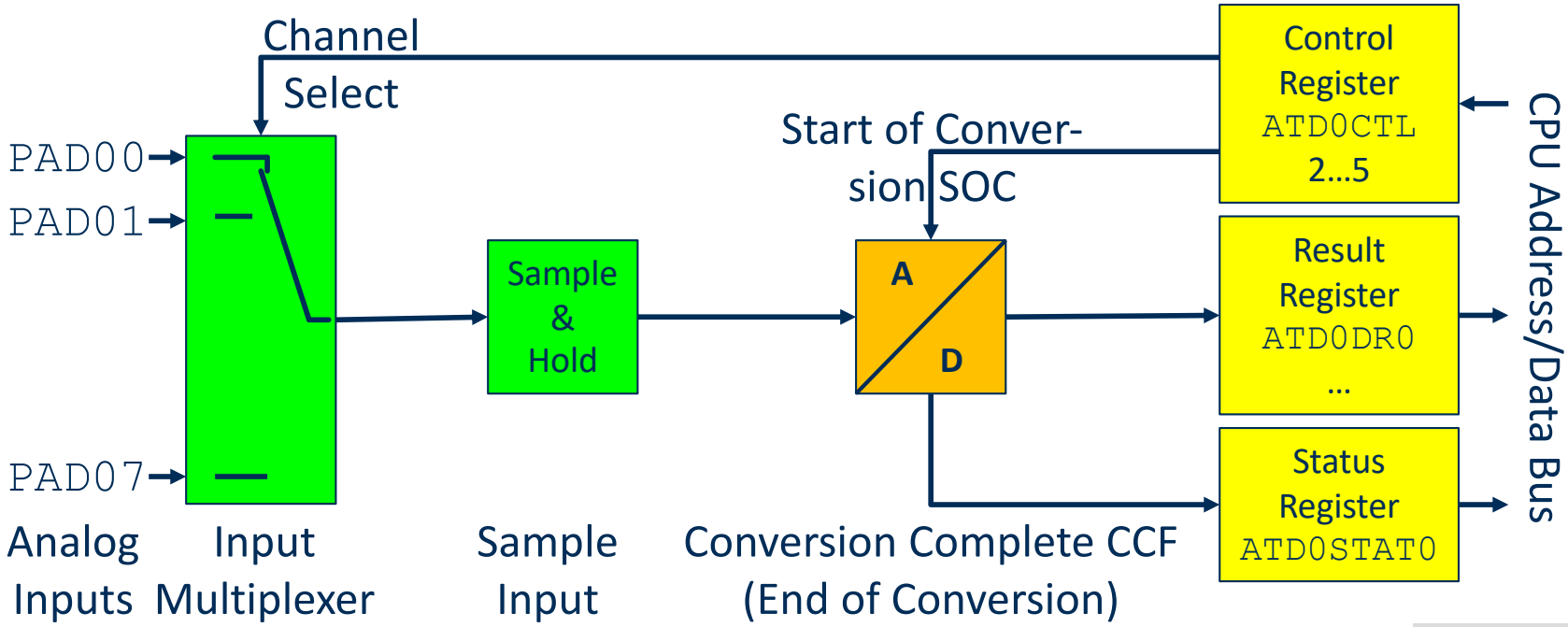
Analog Digital Conversion

ANALOG TO DIGITAL CONVERSION 1/4

HOCHSCHULE ESSLINGEN

Analog Digital Conversion (see `S12ATD10B8CV2.pdf`)

- The HCS12 on the Dragon12 provides **two** independent **10 bit** Analog-to-Digital Converters `ATD0` and `ATD1`.
- The converters use the successive approximation principle and have a **Sample & Hold** unit at their input.
- The **conversion time** is 14 clocks (2 for switching the input multiplexer + 2 for sampling + 10 clocks A/D-conversion) @ 2 MHz clock, i.e. 7µs
- Via the **input multiplexer** each converter can select one of 8 analog input channels: `ATD0`: Port PAD.07 … 00, `ATD1`: Port PAD.15 … 08.



ANALOG TO DIGITAL CONVERSION 2/4

Both converters ATD0 and ATD1 have the same set of registers and several operating modes. Only the major options are discussed, see `S12ATD10B8CV2.pdf`
At program start a one-time configuration is done via the following control registers:

- Control Registers

`ATD0CTL2` (8 bit register)	**Enable the ADC and interrupts**	
	Bit 7 = 1	Enable the ADC module
	Bit 6 = 1	Automatic resetting of the CCF flag
	Bit 5...2 = 0000_B	Miscellaneous options, don't change
	Bit 1 = 1	Enable interrupt after conversion completes
	Bit 0 = 1	Interrupt Flag, indicates an interrupt event, must be reset by writing a 1 into this bit.
`ATD0CTL3` (8 bit register)	**Conversion sequence**	
	Bit 7 = 0	Default
	Bit 6...3	Sequence Count SC, see below
	Bit 2...0 = 000_B	Default
`ATD0CTL4` (8 bit register)	**Resolution and conversion speed**	
	Bit 7 = 0	10 bit resolution (Bit 7=1 ... 8 Bit)
	Bit 6,5 = 00_B	Sampling length 2 clocks (don't change)
	Bit 4...0 = 00101_B	Clock divider f_{ADC}=2MHz @ f_{BUSCLK} = 24 MHz (maximum clock frequency)
`ATD0CTL5` (8 bit register)	**Data format and start of conversion**	
	Bit 7...5 = 100_B	Result in result register right-adjusted and unsigned, e.g. 0V = 0_D, 5V = 1023_D
	Bit 4	Multichannel conversion MULT, see below
	Bit 3 = 0	Default
	Bit 2...0	Channel select code C, see below

ANALOG TO DIGITAL CONVERSION 3/4

If the conversion is complete and the result available, can be polled via

- Status Register `ATD0STAT0` (8 bit register)
  - Bit 7 = 1 — End of conversion EOC
  - Bit 6…0 — Other status info, not described here

The conversion result can be read from the 16 bit **result register** `ATD0DR0` (and in registers `ATD0DR1`, …, `ATD0DR7`, see below).

Operating modes and start of conversion:

A. Single measurement of a single channel

Required setting	in $\texttt{ATD0CTL3}_{6\ldots3}$:	$SC=0001_B$	single measurement
	in $\texttt{ATD0CTL5}_4$:	MULT=0	no multi-channel operation
	in $\texttt{ATD0CTL5}_{2\ldots0}$:	C=0, 1 ,…, 7	select channel to measure
Start of conversion	when writing `ATD0CTL5`		
End of conversion	$\texttt{ATD0STAT0}_7 = 1$	Poll status register of configure interrupt	
Conversion result	in `ATD0DR0` (always, no matter what channel was selected)		

B. Multiple measurements of a single channel (settings see above in A.)

Except setting	in $\texttt{ATD0CTL3}_{6\ldots3}$:	SC=1, 2, …, 8	number of measurements
Conversion result	in `ATD0DR0`, `ATD0DR1`, … (first result, second result, …)		

ANALOG TO DIGITAL CONVERSION 4/4

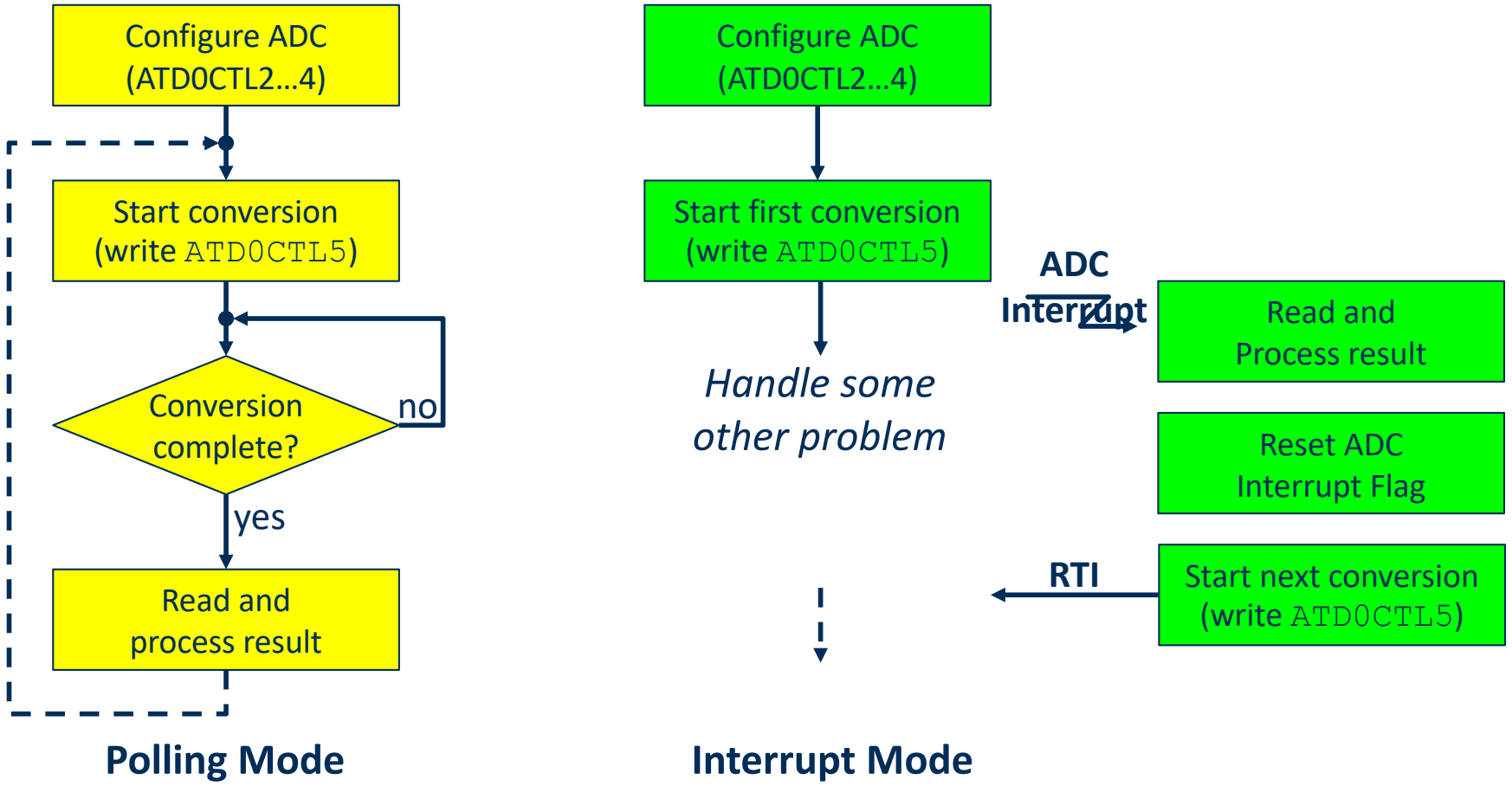
HOCHSCHULE ESSLINGEN

C. Single measurement of a multiple channels (channels in asc. order); settings see A.

Except setting
- in $\texttt{ATD0CTL3}_{6...3}$: SC=1, 2, …, 8 number of channels
- in $\texttt{ATD0CTL5}_{4}$: MULT=1 multi-channel mode
- in $\texttt{ATD0CTL5}_{2...0}$: C=0, 1 ,…, 7 first channel to measure

(E.g. when starting at C=6 with SC=4, channels 6, 7, 0 and 1 will be measured)

Conversion result in ATD0DR0, ATD0DR1, … (first channel, second channel, …)



ANALOG TO DIGITAL CONVERSION – EXAMPLE ASM

HOCHSCHULE ESSLINGEN

Example: Single Polling of Channel 7

```
main:
     …
    MOVB   #$C0, ATD0CTL2   ; Enable ATD, no interrupt
    MOVB   #$08, ATD0CTL3   ; Single conversion only
    MOVB   #$05, ATD0CTL4   ; 10 bit, 2 MHz ATD0 clock
    MOVB   #$87, ATD0CTL5   ; Start conversion on channel 7
wait1:
    ; Wait for End of Conversion (EOC), busy waiting:
    BRCLR  ATD0STAT0, #$80, wait1
    LDD ATD0DR0             ; Read conversion result → D
    …
```

HOCHSCHULE ESSLINGEN

Example: (see `ADCInterruptC.mcp`)

- Measure the analog signal on channel 7 (connected to a poti on Dragon12)
- Output the arithmetic mean of 4 measurement values (in binary) to the LEDs on port B

```
…
// --- Global variables ------------------------------------
unsigned int value;     // Measurement value

void main(void) {
  EnableInterrupts;
  // --- Initialize ATD0 -------------------------------------
  ATD0CTL2 = 0b11000010; // Enable ATD0, enable interrupt
  ATD0CTL3 = 0b00100000; // Sequence: 4 measurements
  ATD0CTL4 = 0b00000101; // 10bit, 2MHz ATD0 clock
  ATD0CTL5 = 0b10000111; // Start 1st measurement on single ch.7

  for (;;) {              // Infinite loop
    // Show upper 8 of the 10bits measured value on LEDs 7…0
    PORTB = value >> 2;
  }
}
```

ANALOG TO DIGITAL CONVERSION – EXAMPLE C 2/2

HOCHSCHULE ESSLINGEN

```
// --- ADC interrupt service routine ----------------------
void interrupt 22 adcISR(void) {

  // Read the result registers and compute average of 4 measurements
  value=((unsigned long)ATD0DR0 + ATD0DR1 + ATD0DR2 + ATD0DR3)>> 2;
  ATD0CTL2 |= 0x1;   // Reset interrupt flag (bit0 in ATD0CTL2)

  ATD0CTL5 = 0b10000111; // Start next measurement on Channel 7



}
```

Note: Result registers `ATD0DR0`, … must be **read** (to automatically reset the CCF flag), **before** starting the **next conversion**.

Instead of starting the A/D conversion via SW (by writing to `ATD0CTL5`), the HCS12 allows other modes to trigger a conversion:

- Automatic Triggering: This mode (scan mode for free running mode, activated by bit 5=1 in register `ATD0CTL5`) software only triggers the first conversion. All following conversions will start automatically after the end of the previous conversion. Result registers can be read any time and always provide the latest result. However, in this mode sampling is asynchronous, i.e. the user program has no control of the signal sampling period.
- Hardware triggering: Rather than by software, a conversion will be started by an external hardware signal (a rising or falling edge of a pulse signal at channel 7 of the ADC).

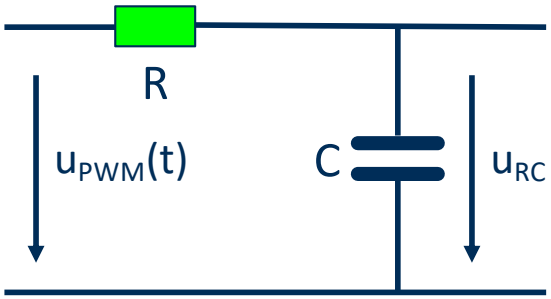
HOCHSCHULE
ESSLINGEN

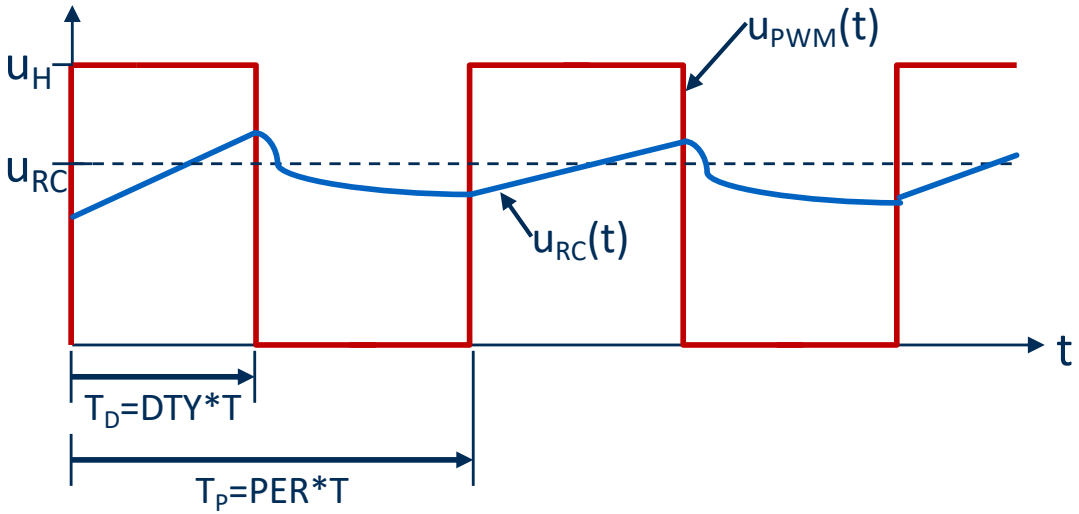
PWM Outputs

PWM OUTPUTS 1/4

Purpose: (see `S12PWM_8B8CV1.pdf`)

- Pulse-Width Modulation (PWM) allows generating signals with timers, using period T_P or pulse length (width) T_D (where D stands for Duty Cycle)





- Pseudo digital-to-analog conversion:
  Arithmetic mean $U_{Rc} \approx U_{PWM} = U_H \frac{T_D}{T_P}$ for RC >> T_P (valid only if `PWMPOL.x`=1)

- After filtering the output via a RC or RL low pass filter (e.g. the coil of a DC motor or solenoid) → control of servo motors, solenoids, lamps, …
- By changing T_D and/or T_P during run-time, the "analog" signal can be modulated (Pulse Width or Pulse Frequency Modulation)

PWM OUTPUTS 2/4

The HCS12 PWM module has 8 PWM output channels (Port P.7…0). T_D and T_P can be set individually for each channel with a resolution of 8 bit:

- Register for Ch. X (x=0,…,7) (all registers 8 bit)

PWMPERx	Period T_P as multiple of the clock period T_x (for maximum resolution use PWMPER=255)
PWMDTYx	Length of phase T_D as multiple of the clock period T_x (Make sure PWMDTYx < PWMPERx)
PWMCNTx	Counter register of PWM channel x (Clear to 0 to restart the PWM signal. Automatically done by reset, so normally not required)

There are three 8 bit control registers with one bit per channel:

- Common Control Registers (1 bit/channel)

PWME (8 bit register)	Enable channel: Set a channel's bit to 1 to start its PWM signal. Otherwise, the port pin can be used as a normal digital input/output. **Enable only,** after all PWM channels including clock dividers have been configured.
PWMPOL (8 bit register)	PWM signal polarity: When a channel's bit is set to 1, the PWM signal period starts with the H phase, L otherwise.
PWMCLK (8 bit register)	Selection of one of the clock signals $T_{A/B}$ or $T_{SA/SB}$: If a channel's bit is set to 0, the channel uses the fast Clock $T_{A/B}$, if set to 1 it uses the slow clock $T_{SA/SB}$

PWM OUTPUTS 3/4

The PWM module has a total of 4 clock signals, paired in two groups. For each channel one out of the two signals in a group can be selected via the PWMCLK (see next page):

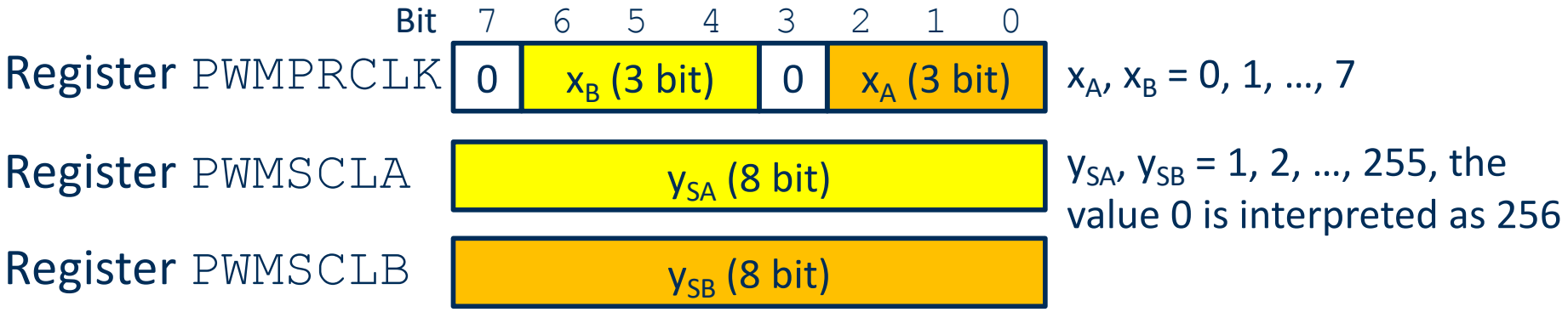
T_A or T_{SA} for PWM channels 0, 1, 4 ,5 T_B or T_{SB} for PWM channels 2, 3, 6, 7

These clock signals are generated via clock dividers from the CPU's clock BUSCLK. The clock signals are programmed via clock dividers x_A or x_B or y_{SA} or y_{SB}:

$T_A = 2^{XA} * T_{BUSCLK}$ $T_B = 2^{XB} * T_{BUSCLK}$

$T_{SA} = 2*y_{SA}*T_A$ $T_{SB} = 2*y_{SB} * T_B$

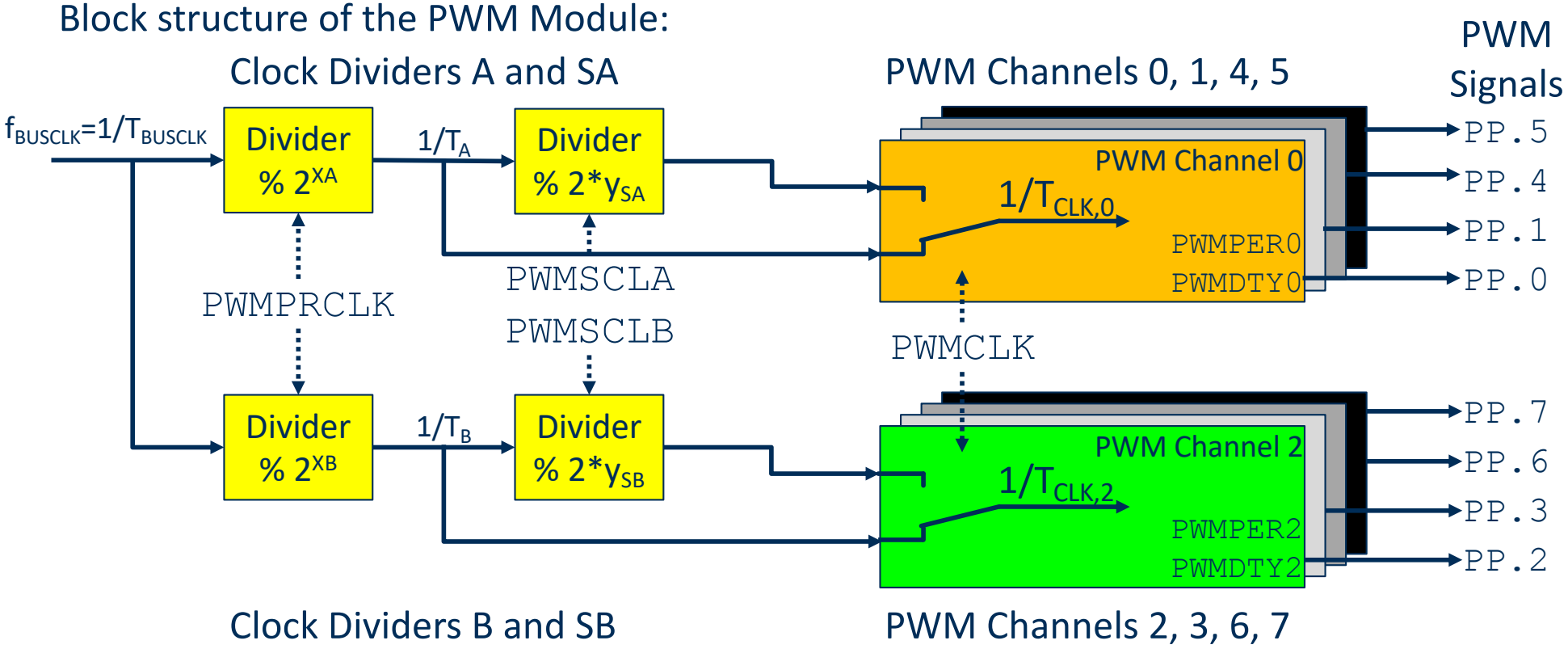
Dragon12 uses $T_{BUSCLK} = 1/f_{BUSCLK} = 1/24$ MHz. The clock dividers x_A, x_B, y_{SA} and y_{SB} are set via the following registers:



The PWM module has several options and operation mode, which are not discussed here. These options are turned off automatically after reset.

PWM OUTPUTS 4/4

Block structure of the PWM Module:



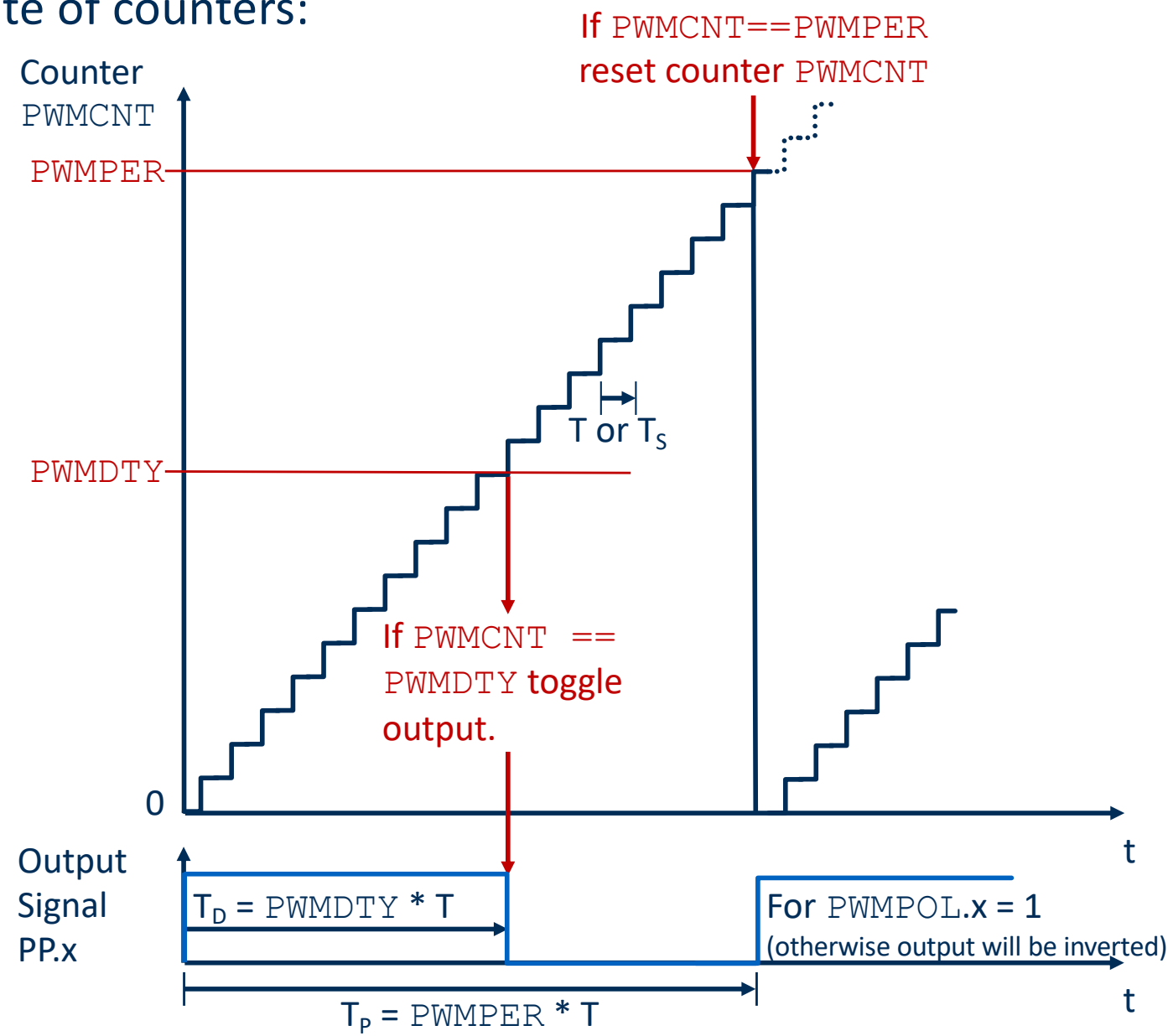
- 2 clock divider groups μ=A,B with 2 clocks each: $T\mu = 2^{x_\mu} * T_{BUSCLK}$ and $T_{S\mu}=2*y_{S\mu}*T_\mu$
- Channels v= 0, 1, 4, 5 use clock $T_{CLKv} = T_A$ or T_{SA} dependent on the setting of PWMCLK.v
- Channels v= 2, 3, 6, 7 use clock $T_{CLKv} = T_B$ or T_{SB} dependent on the setting of PWMCLK.v
- Period of the pulse signal of channel v is:

  $T_{Pv} = PWMPER_v * T_{CLKv}$
- Length of the H-phase (if PWMPOL.v=1) or the L-phase (if PWMPOL.v=0) for channel v is:

  $T_{Dv} = PWMDTY_v * T_{CLKv}$

PWM OUTPUTS – OPERATION OF A PWM CHANNEL

State of counters:



PWM OUTPUTS – SIGNAL FREQUENCY RANGE

HOCHSCHULE ESSLINGEN

What are the minimum and the maximum frequencies of a PWM signal on a Dragon12 board?
The duty cycle T_D/T_P shall be configure for 8 bit resolution.

PWM OUTPUTS – EXAMPLE 1/2

HOCHSCHULE ESSLINGEN

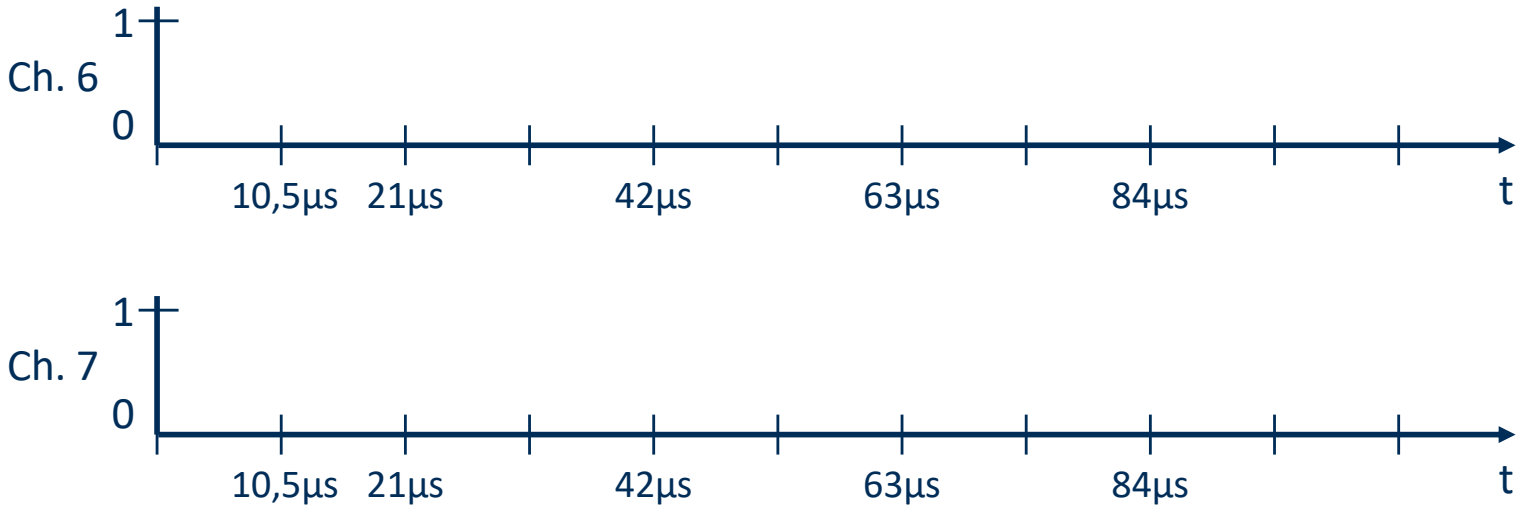
Which PWM signals are generated using the following?(see `PWM1.mcp`)

```
MOVB    #$80, PWMCLK

MOVB    #$10, PWMPRCLK

MOVB    #$02, PWMSCLB

MOVB    #$40, PWMPOL

MOVB    #255, PWMPER6

MOVB    #128, PWMDTY6

MOVB    #255, PWMPER7

MOVB    #32, PWMDTY7

BSET    PWME, #$C0
```

PWM OUTPUTS – EXAMPLE 2/2

HOCHSCHULE ESSLINGEN

Timing Diagram:



Note:
The PWM can be configured in such a way, that channels 0+1, 2+3, 4+5 and 5+6 are combined into four PWM channels. Thus the duty cycle resolution can be doubled to 16 bit.
Rather than starting with the L- (or H-)phase, the PWM signal can be generated symmetrically within its period (Center Aligned).

APPENDIX B: CLOCK GENERATOR AND CLOCK DIVIDER

HOCHSCHULE ESSLINGEN

RTI Interrupt:

Interrupt period $T_{RTI} = 2^{9+X} * (Y + 1) / f_{OSCCLK}$. All values in ms @ f_{OSCCLK} = 4 MHz

	Y=0	Y=1	Y=2	Y=3	Y=4	Y=5	Y=6	Y=7	Y=8	Y=9	Y=10	Y=11	Y=12	Y=13	Y=14	Y=15
X=1:	0.256	0.512	0.768	1.024	1.280	1.536	1.792	2.048	2.304	2.560	2.816	3.072	3.324	3.584	3.840	4.096
X=2:	0.512	1.024	1.536	2.048	2.560	3.072	3.584	4.096	4.608	5.120	5.632	6.144	6.656	7.168	7.680	8.192
X=3:	1.024	2.048	3.072	4.096	5.120	6.144	7.168	8.192	9.216	10.240	11.264	12.288	13.312	14.336	15.360	16.384
X=4:	2.048	4.096	6.144	8.192	10.240	12.288	14.336	16.384	18.432	20.480	22.528	24.576	26.624	28.672	30.720	32.768
X=5:	4.096	8.192	12.288	16.384	20.480	24.576	28.672	32.768	36.864	40.960	45.056	49.152	53.248	57.344	51.440	65.536
X=6:	8.192	16.384	24.576	32.768	40.960	49.152	57.344	65.536	73.728	81.920	90.112	98.304	106.496	114.688	122.880	131.072
X=7:	16.384	32.768	49.152	65.356	81.920	98.304	114.68	131.072	147.456	163.840	180.224	196.608	212.992	229.376	245.760	262.144

ECT Timer: @ f_{BUSCLK} = 24 MHz

Clock period $T_{TCNT} = 1/f_{TCNT} = 2^X/f_{BUSCLK}$. In µs

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
0.042	0.083	0.167	0.333	0.667	1.333	2.667	5.333

Clock period $T_P = 2^{16} * TTC_{NT}$ Values in ms

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
2.731	5.461	10.293	21.845	43.691	87.381	174.763	349.525

PWM: @ f_{BUSCLK} = 24 MHz

Clock period of fast clock $T_{A/B} = 2^X/f_{BUSCLK}$. In µs

X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
0.042	0.083	0.167	0.333	0.667	1.333	2.667	5.333

Clock period of slow clock $T_{SA/B} = 2*y*T_{A/B}$. In µs

	X=0	X=1	X=2	X=3	X=4	X=5	X=6	X=7
Y=1	0.083	0.167	0.333	0.667	1.333	2.667	5.333	10.667
…	…	…	…					
Y=255	21.250	42.500	85.000	170000	340000	680000	1360000	2720000

Serial Interface: @ f_{BUSCLK} = 24 MHz

Clock Divider Register SCIxBD = $f_{BUSCLK}/(16*f_{BIT})$

f_{BIT}=300 bit/s	600 bit/s	1,2 kbit/s	2,4kbit/s	4,8kbit/s	9,6kbit/s	19,2kbits	38,4kbit/s	57,6kbit/s	115,2kbit/s
5000	2500	1250	625	313	156	78	39	26	13

LECTURE: LITERATURE

HOCHSCHULE ESSLINGEN

According to list of literature:

- Patterson, D., Hennessy, J.: *Computer Organization and Design*, Kaufmann, 2011
  (Deutsche Übersetzung: Rechnerorganisation und –entwurf, Spektrum)
- Hennessy, J., Patterson, D.: *Computer Architecture: A quantitative Approach, **5th edition***, Morgan Kaufmann, 2017
- Tanenbaum, A., Austin, T.: *Structured Computer Organization*, Pearson, 6th edition, 2013
- Tanenbaum, A., Austin, T.: *Rechnerarchitektur: von der digitalen Logik zum Parallelrechner*, Pearson, 6th ed., 2014
- Huang, H.W.: *The HCS12/9S12. An Introduction to the HW and SW interface*, Thomson Learning, 2009
- Beierlein, T.: *Taschenbuch Mikroprozessortechnik*, Carl-Hanser Verl., 2011