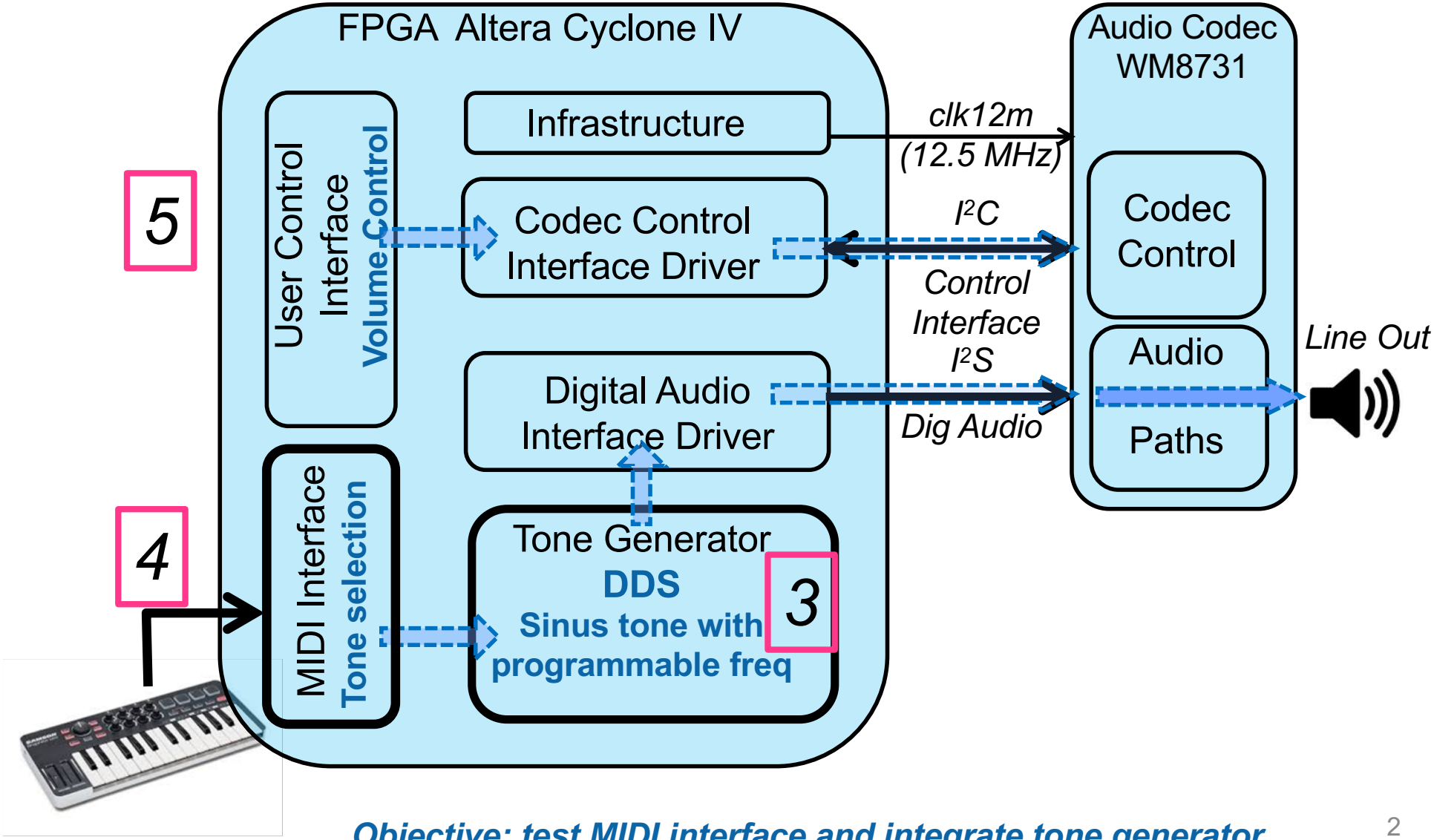
MS4: MIDI

School of Engineering

Musical Instrument Digital Interface

Milestones 3 bis 5
Tone Generator und MIDI-Interface

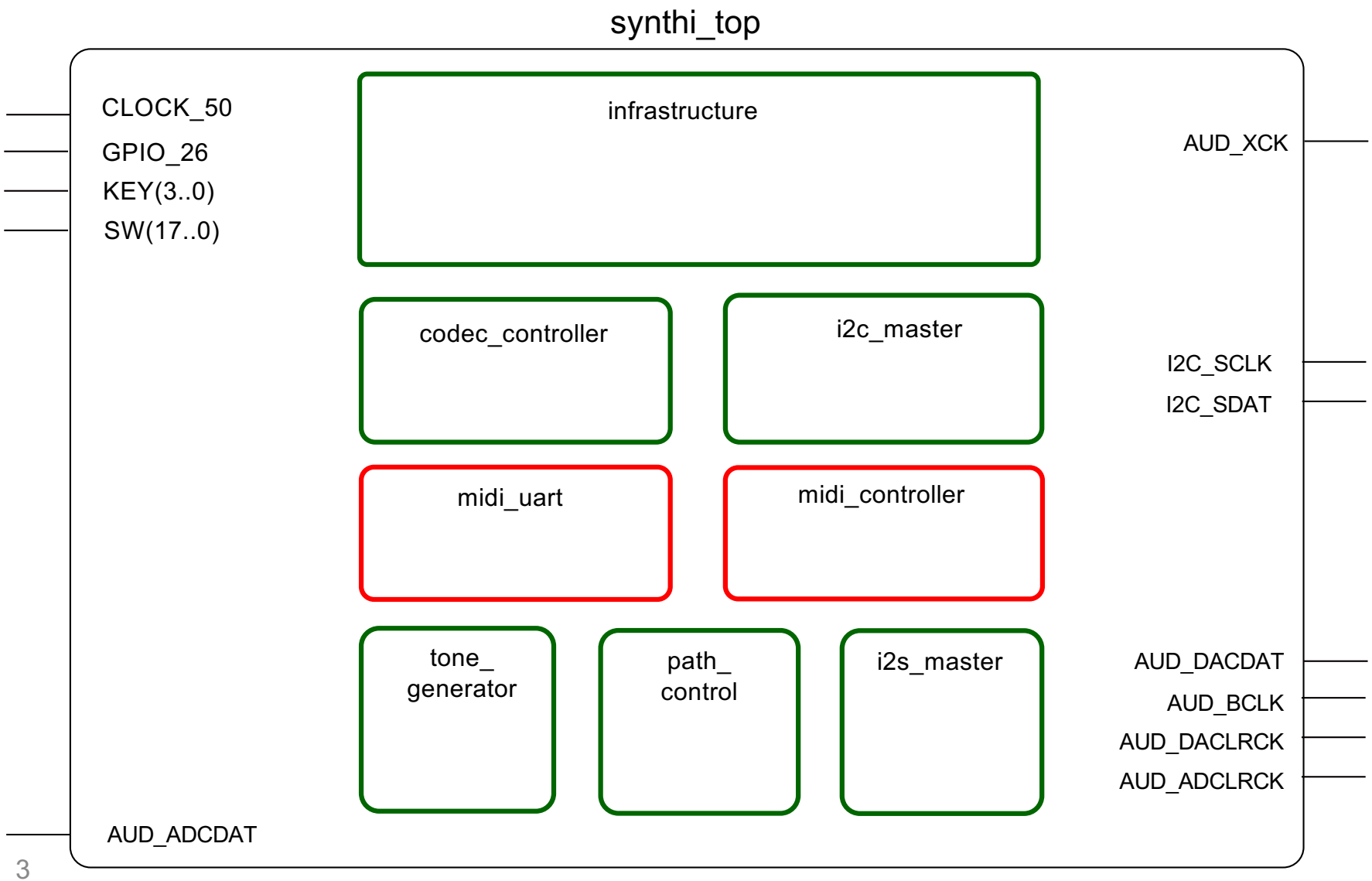
School of Engineering



Objective: test MIDI interface and integrate tone generator

MS4: MIDI

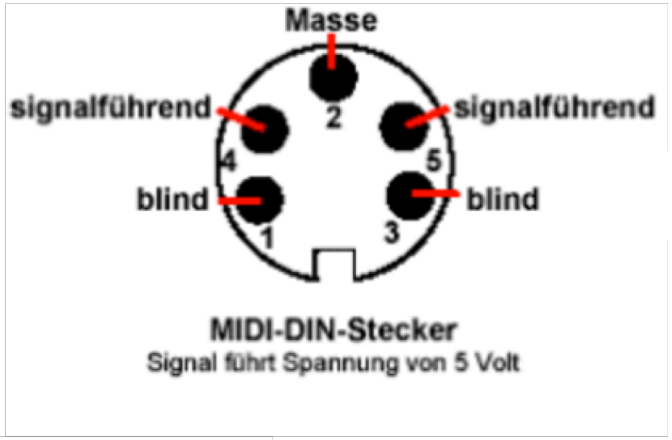
School of Engineering

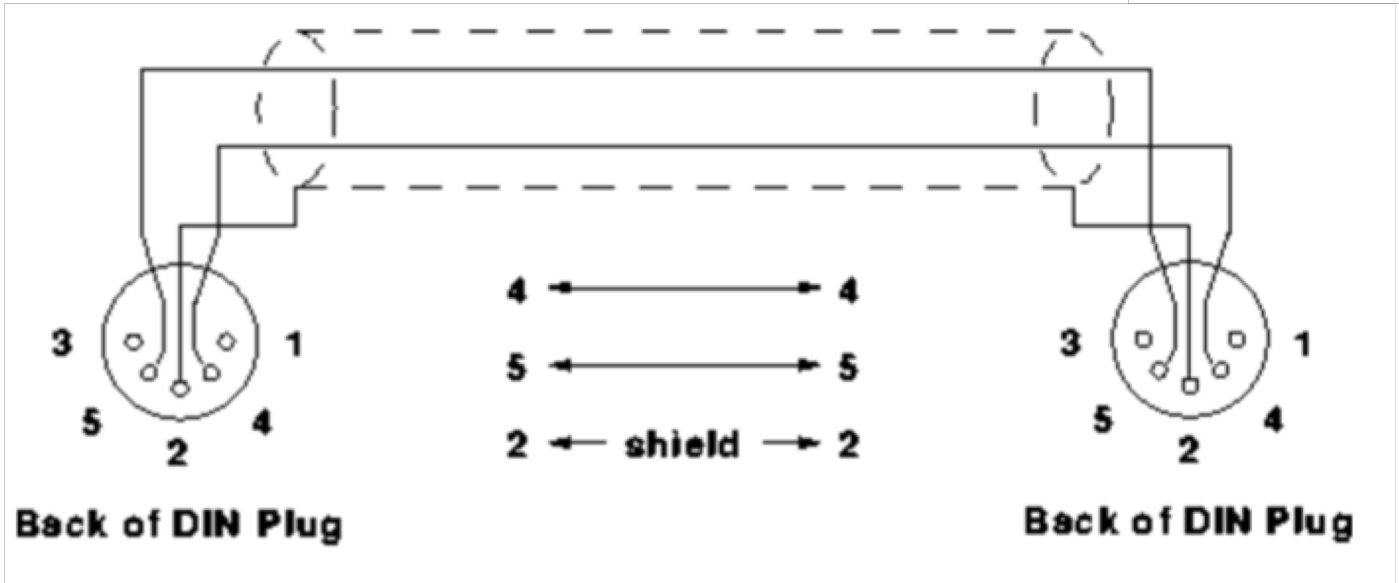


Physikalischer Layer

School of Engineering

MIDI has a single baud rate 31250 baud





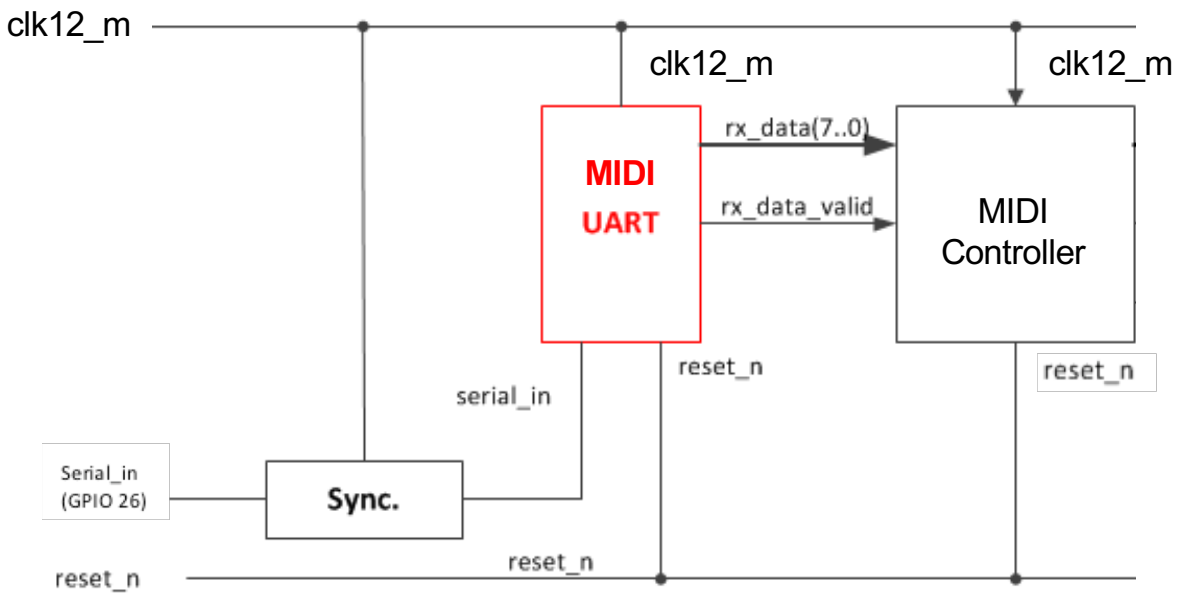
Vergleich UART vs MIDI

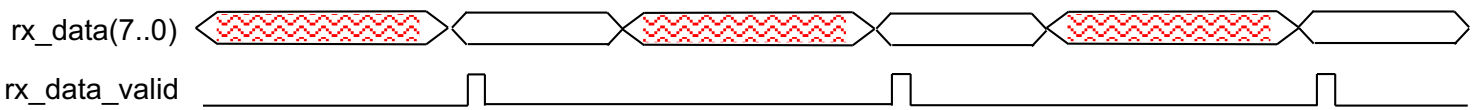
School of Engineering

	UART	MIDI
Frame Format	1 start bit 5-8 data bits 1-2 stop bits optional parity bit	1 start bit 8 data bits 1 stop bits no parity
Baud Rate	up to 20k standard rates of 300, 600,...19k2 extended rates of 38k4, 57k6, 115k2	31250 baud +/- 1%
Receiver Load	3k to 7k ohm	Opto-isolator (otherwise unspecified)
Maximum cable	2500pF (approx. 15m)	15m
Isolation	Unspecified	Opto-isolator on receiver circuit
Handshaking signals	Transmitter: RTS, DTR Receiver: CTS, DSR, DCD, RI	None
Physical connector	D25	5-pin DIN (180 degree)

Aufteilung der Blöcke

School of Engineering

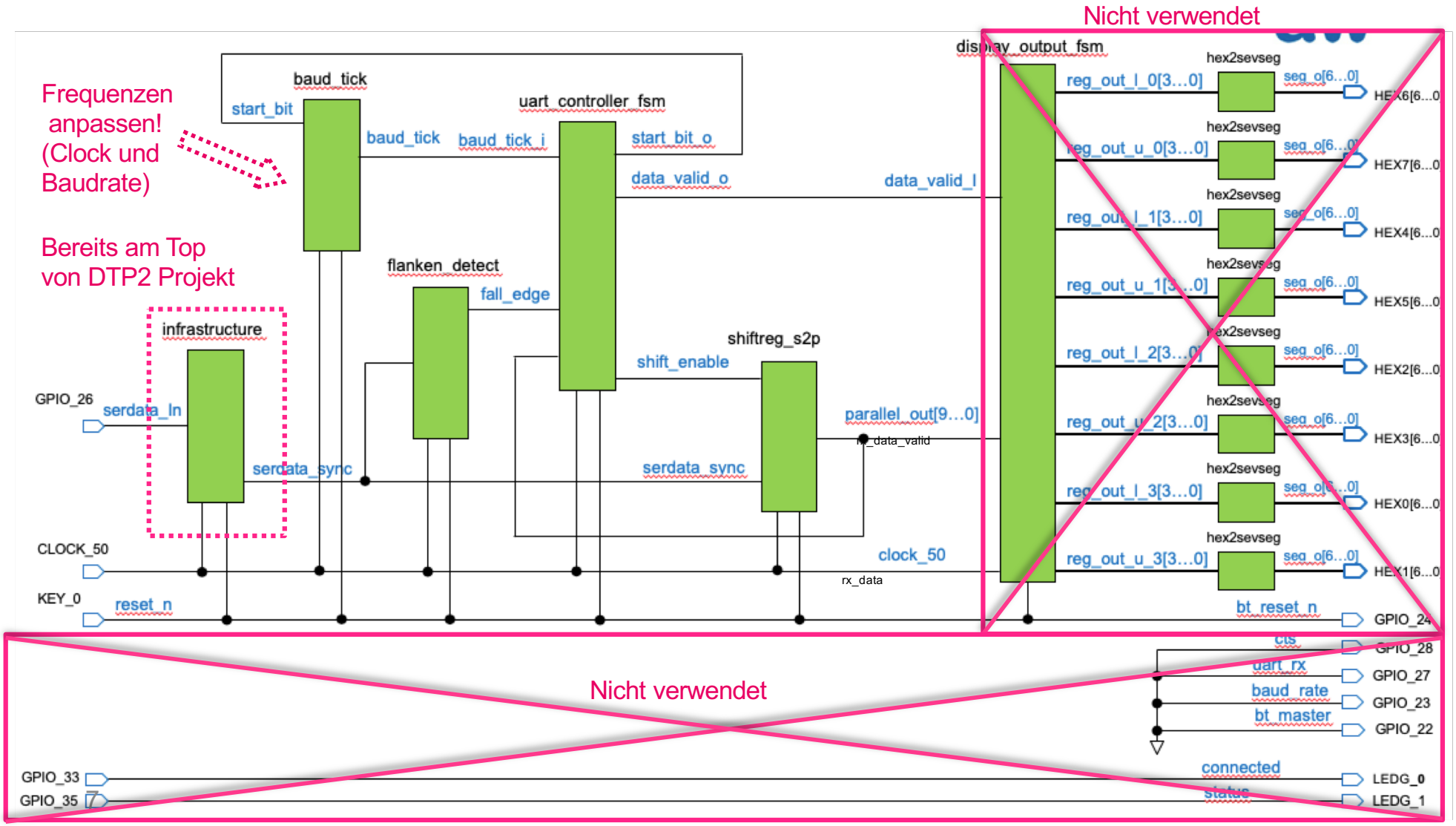




Block Diagram of UAR(T)

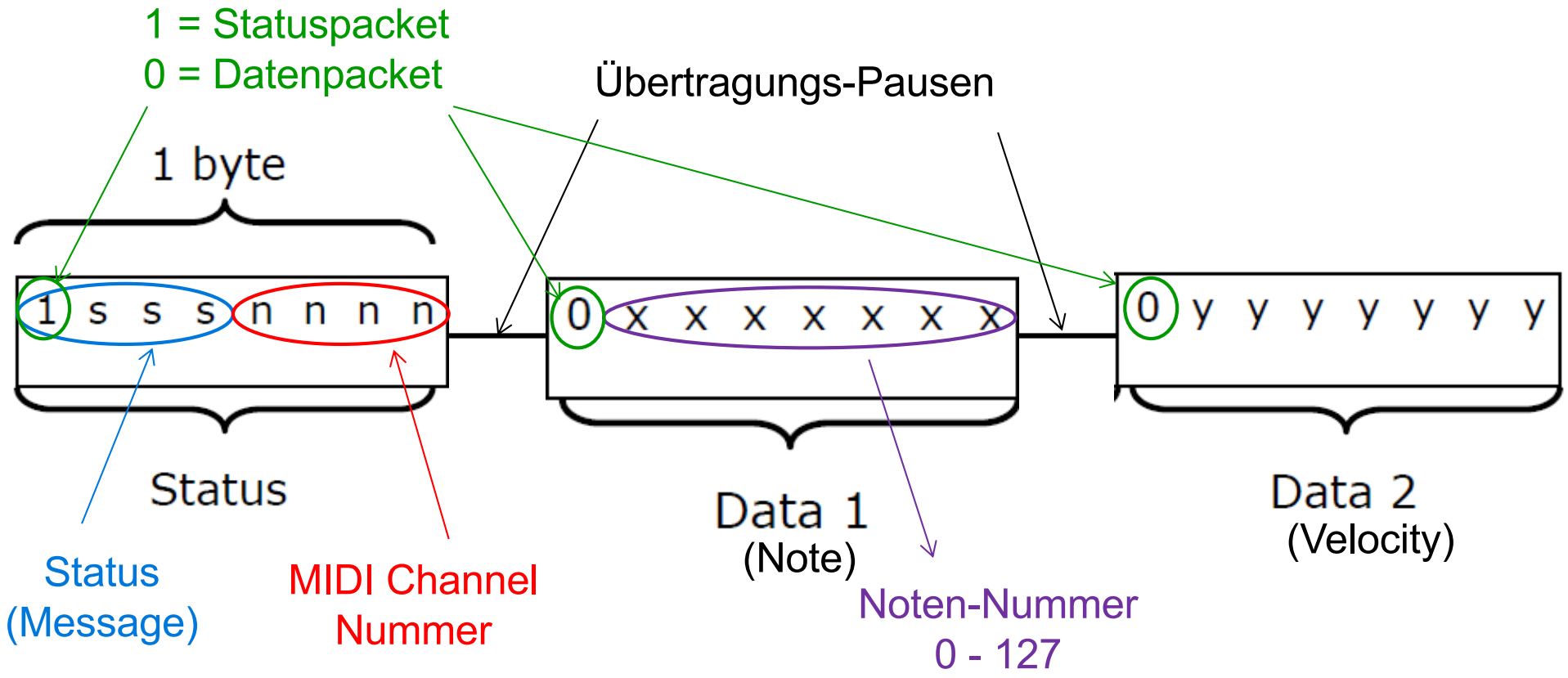
School of Engineering

UART vom Mini Projekt DTP1 : anpassen und als MIDI_UART verwenden!



MIDI Channel Messages

School of Engineering



Achtung:
1.) Status und Daten unterscheiden sich in dem Bit-7 = 1 bei Status und 0 bei Daten
2.) Aus diesem Grund ist der Inhalt eines Datepackets nur 7 bits, d.h. max 0x7f = 127

MIDI Channel Messages

School of Engineering

Bedeutung	Status Byte	Data Byte 1	Data Byte 2	
Note Off	8n	kk	vv	Beendet das Spielen der Note kk. Zusätzlich wird eine „Release velocity“ vv gesendet.
Note On	9n	kk	vv	Beginnt das Spielen einer Note kk. Zusätzlich wird die Anschlagsdynamik v (engl. Velocity) angegeben, die der Druckstärke auf die Taste in 127 Schritten von 1 (sehr schwach) bis 127 (sehr stark) entspricht. Der Wert 0 ist als Note-Off-Befehl definiert.
Polyphonic Aftertouch	An	kk	vv	Beschreibt das Ändern des Tastendrucks vv während die Taste kk bereits gedrückt ist, für jede Taste einzeln.
Control Change	Bn	cc	vv	Ändert den Zustand eines Controllers cc mit dem Wert vv.(Nicht verwendet!)
Program Change	Cn	pp		Legt das für den angegebenen Kanal zu spielende Instrument pp fest. (Nicht verwendet!)
Monophonic bzw. Channel Aftertouch	Dn	vv		Beschreibt das Ändern des Tastendrucks vv während die Tasten bereits gedrückt sind, für alle Tasten gemeinsam.
Pitch Wheel	En	vv	ww	Einstellung des Pitchbend-Rades. Die zwei Datenbytes vv (LSB) und ww (MSB) ergeben zusammen den 14 Bit breiten Wert zwischen 0 und 16383. Zu beachten ist, dass sie vertauscht kombiniert werden, also (Byte 2 · 128) + Byte 1 = 14-Bit-Wert. Die Mittelstellung des Rades wird durch den Wert 8192 signalisiert.
System (exclusive) Message	F0…7	xx		Steuermeldungen, häufig gerätespezifisch, Länge ebenfalls gerätespezifisch (xx = Datenbytes) (Nicht verwendet!)

MIDI Channel Messages

School of Engineering

Message	Status	Data 1	Data 2
Note off	8n	Note number unterstützt	Velocity Nicht unterstützt
Note on	9n	Note number unterstützt	Velocity unterstützt
Polyphonic aftertouch	An	Note number Nicht	Pressure Nicht unterstützt
Control change	Bn	Controller number	Data Nicht getestet
Program change	Cn	Program number	- Nicht getestet
Channel aftertouch	Dn	Pressure Unterstützt nur mit Slider	
Pitch wheel	En	LSbyte unterstützt	MSbyte unterstützt

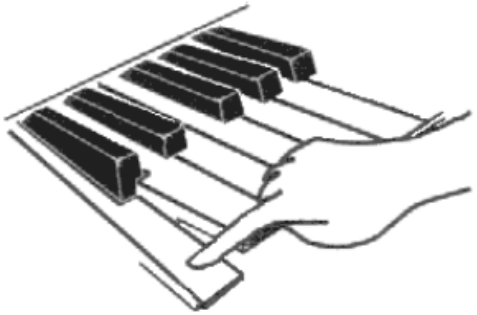
Hinweise beschreiben was Alesis 25 unterstützt

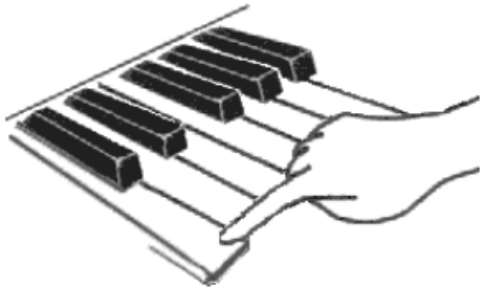
Note On / Note Off

School of Engineering

Message	Status	Data 1	Data 2
Note off	8n	Note number	Velocity
Note on	9n	Note number	Velocity

- "Note on" triggers a musical note, "note off" turns it off



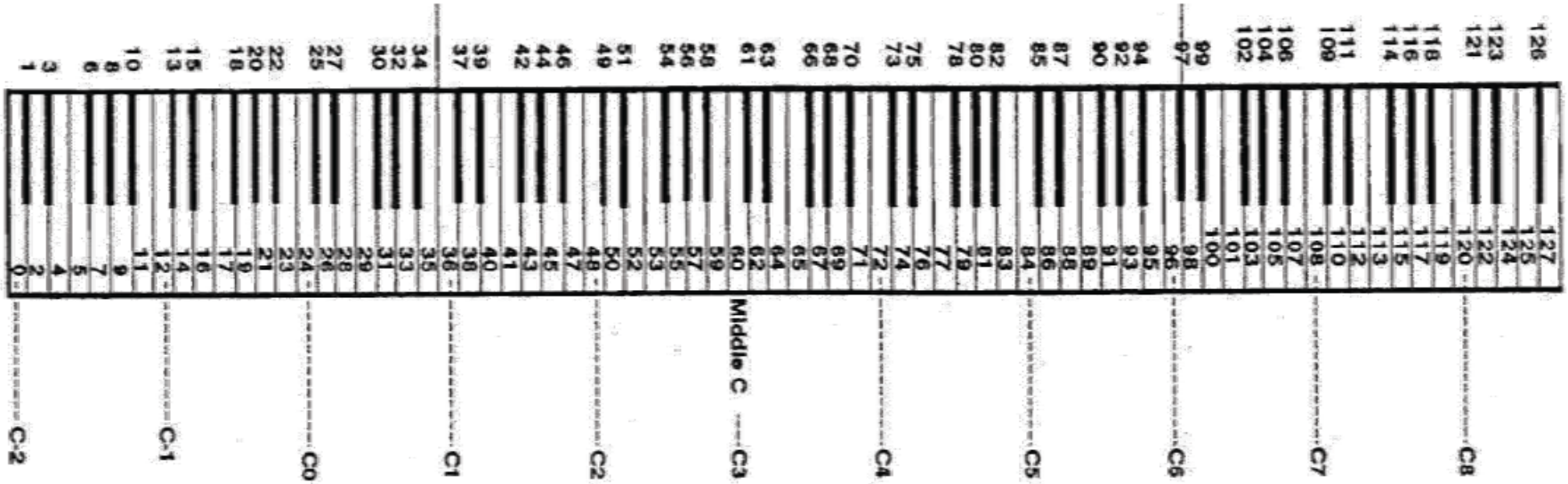


- All notes MUST be turned off (otherwise they'll sound indefinitely)

Note on / Note Off (2)

School of Engineering

- There are 128 (0-127) possible note values (~10 octaves) mapped to the chromatic western music scale.



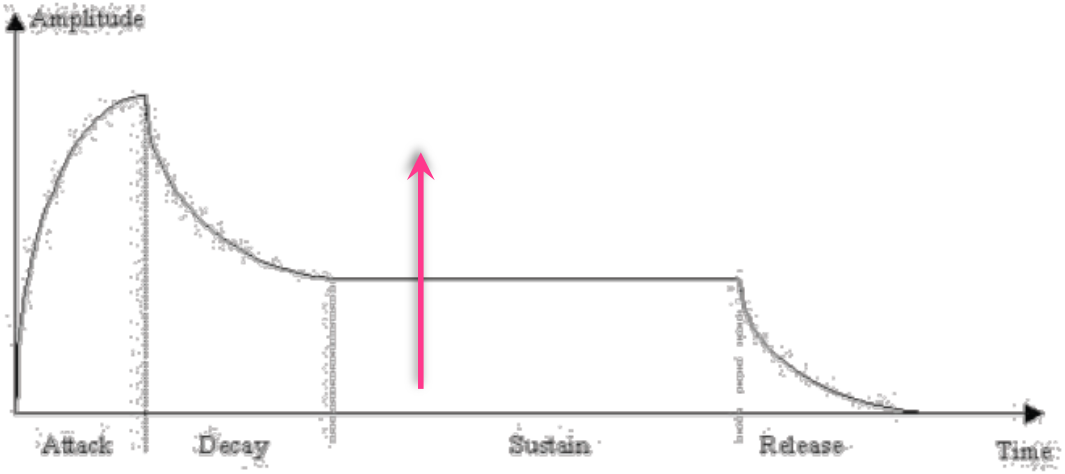
- Commonly, middle C is mapped to MIDI's C-3 (note number 60, 6th octave).

Slide Courtesy of Juan P Bello

Note on / Note Off (3)

School of Engineering

- Note on messages are also associated to a "velocity" value, characterizing how hard the key was hit.
- Note on velocity can be used to control volume and timbre of a sound (e.g. by controlling the scaling of an envelope generator)



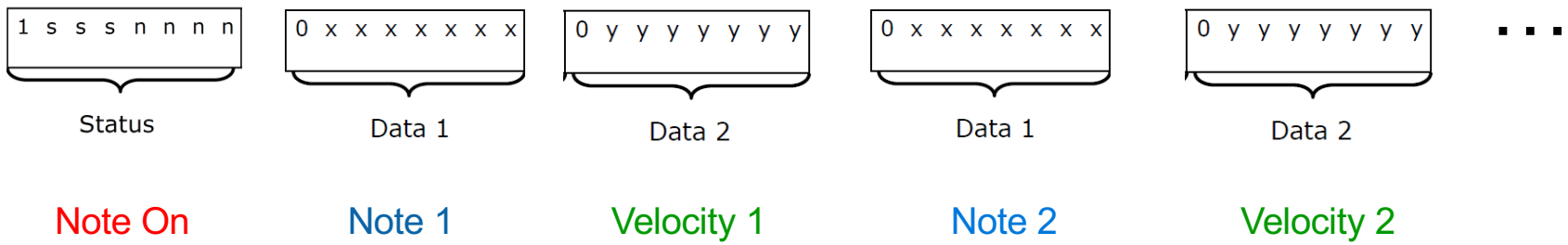
- The mapping between velocity and the parameter it controls is often logarithmic.
- Note off velocity relates to the speed at which a note was released
- It could be used to affect a sound, but it is not normally used.

Easy Method:
Use Velocity to Control Volume

Slide Courtesy of Juan P Bello

MIDI «Running Status» Übertragungsart

School of Engineering



MIDI «Running Status» Übertragungsart

School of Engineering

- It is convenient when large amounts of data are sent to the MIDI bus (e.g. a high-polyphony chord)
- Normally we will need 6 bytes for each note of a chord: [9n][pitch][velocity] and [8n][pitch][velocity]
- Instead we can clutter note on and off messages together: [9n][pitch][vel][pitch][vel]…[pitch][vel]
- This is known as running status
- For a 4-note chord it means 17 bytes are transmitted rather than 24 bytes (assuming running status remains unchanged).

Slide Courtesy of Juan P Bello

Aftertouch

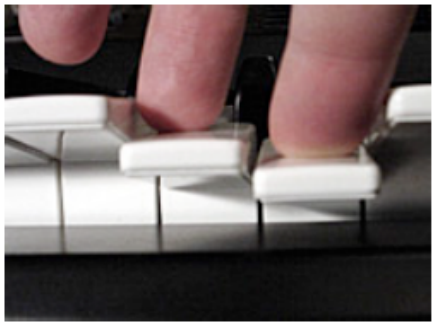
School of Engineering

Unterstützt nur mit Slider, **Optional!**

- Key pressure messages are called aftertouch
- It refers to the amount of pressure placed on a key at the bottom of its travel (triggering performance parameters, e.g. vibrato)

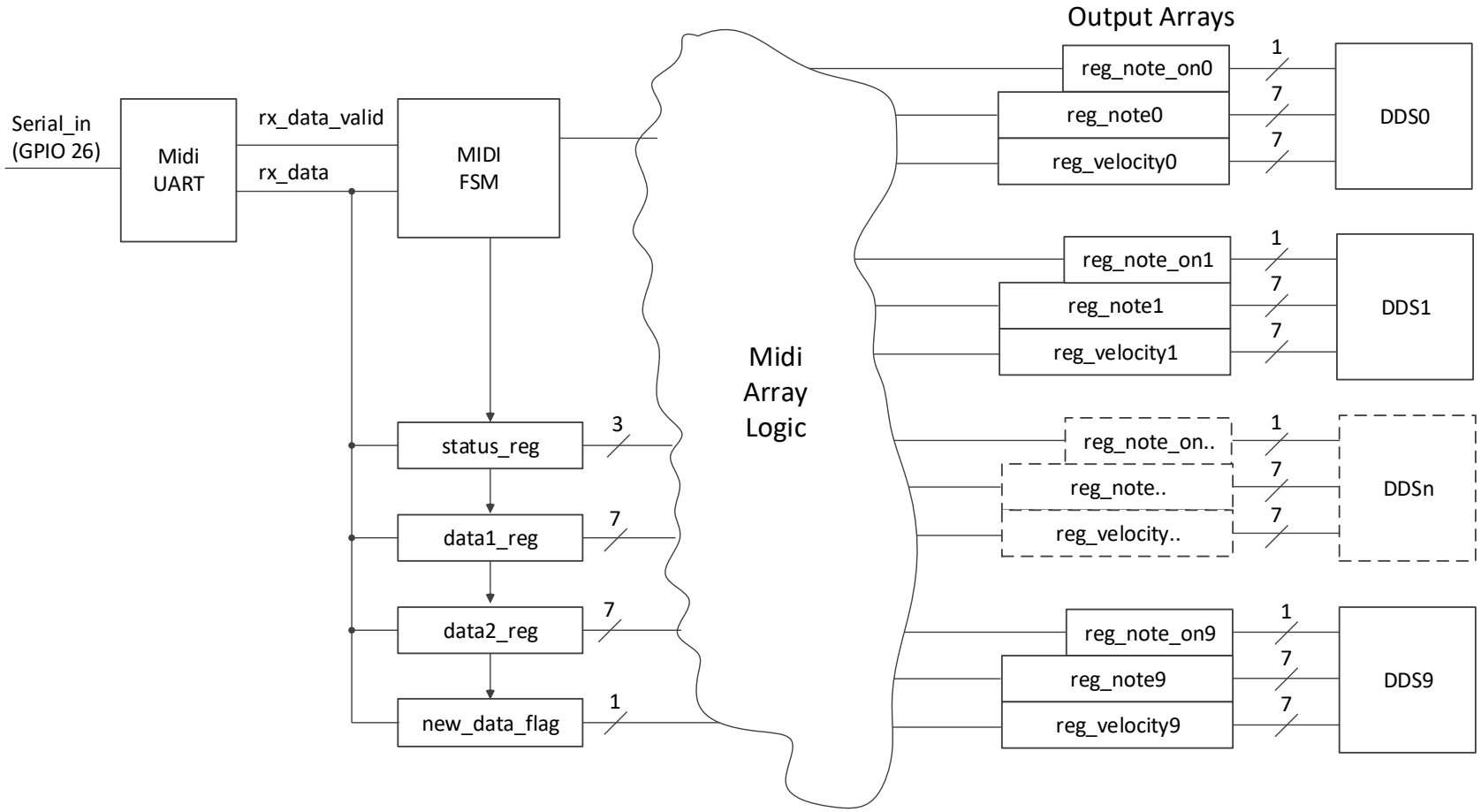
Message	Status	Data 1	Data 2
Polyphonic aftertouch	An	Note number	Pressure
Channel aftertouch	Dn	Pressure	-

- Polyphonic key pressure transmits a separate value per key (thus requiring separate sensors)
- This is expensive as most players do not maintain constant pressure on the bottom of the key
- Most instruments use a single sensor, thus one message is sent with the approx. total pressure sensed (channel aftertouch)



Slide Courtesy of Juan P Bello

MIDI_Controller



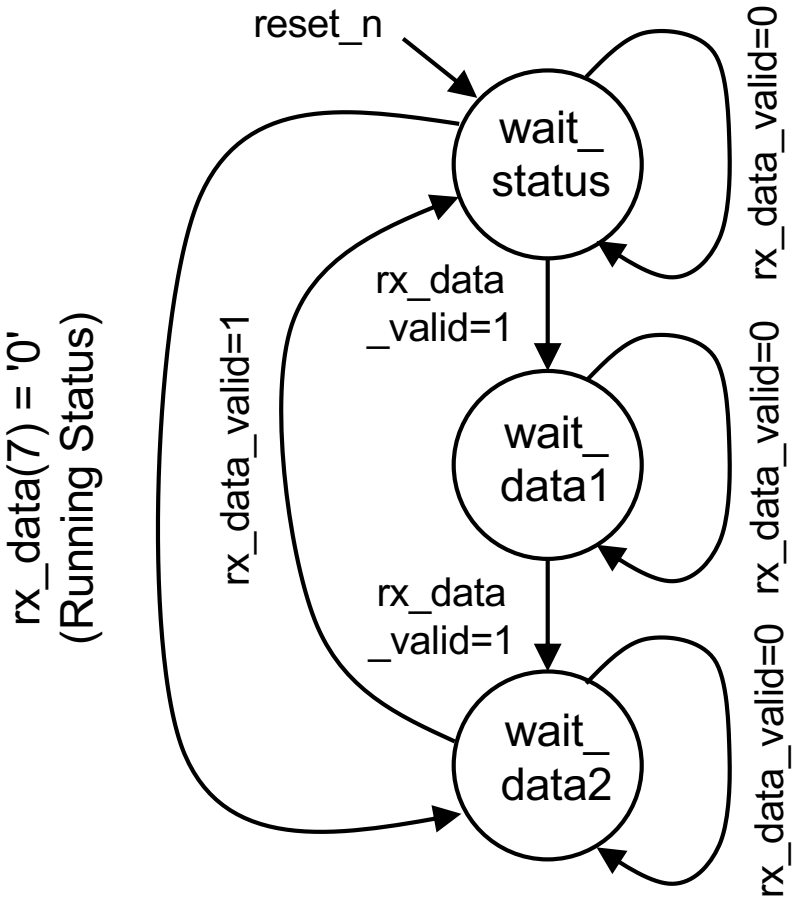
note_0 to 9 und velocity_0 to 9 sind in einem Array vom Typ t_tone_array gespeichert. Die definition Von t_tone_array muss ins tone_gen_pkg geschrieben werden

```
signal reg_note, next_reg_note          : t_tone_array;
signal reg_velocity, next_reg_velocity  : t_tone_array;

type t_tone_array is array (0 to 9) of std_logic_vector(6 downto 0) ;
```

MIDI_Controller : FSM

School of Engineering



FSM steuert folgende Register:

- Setzen von Status Register mit den Konstanten: SET_NOTE, DEL_NOTE etc.
- Laden der Register data1_reg, data2_reg
- Setzen von new_data_flag

VHDL For Loop

School of Engineering

Syntax For Loop:

```
"optional_label": for "parameter" in "range" loop
  sequential statements
end loop "optional_label";
```

Beispiel For Loop:

```
for i in 0 to 7 loop
   w(i) <= a(i) and b; -- 8bit bus "a" mit Einzelsignal "b" verbunden
end loop;
```

Loop Example : Delete Note

School of Engineering

Check ob Array Eintrag für Note bereits besetzt ist

```
…
IF (new_data_flag) THEN
note_available := ’0';
…
        -----------------------------------------------------
        -- CHECK IF NOTE IS ALREADY ENTERED IN MIDI ARRAY
        -----------------------------------------------------
        FOR i IN 0 TO 9 LOOP
                IF   reg_note(i) = data1_reg  AND  reg_note_on(i) = '1‘  THEN    -- Found a matching note
                                note_available := '1';
                                IF  status_reg = DEL_NOTE  THEN
                                                next_reg_note_on(i) <= '0'; -- turn off note
                                ELSIF status_reg = SET_NOTE  AND  data2_reg = "00000000"  THEN
                                                next_reg_note_on(i) <= '0'; -- turn off note if velocity is 0
                                END IF;
                END IF;
        END LOOP;
…
```

Loop Example: Set Note

School of Engineering

Check an welcher Stelle die neue Note gesetzt werden kann

```
-----------------------------------------
-- ENTER A NEW NOTE IF STILL EMPTY REGISTERS
-----------------------------------------
-- If the new note is not in the midi storage array yet, find a free space
-- if the valid flag is cleared, the note can be overwritten, at the same time a flag is set to mark that
-- the new note has found a place.

IF  note_available = '0'  THEN -- If there is not yet an entry for the note, look for an empty space and write it

        FOR i IN 0 TO 9 LOOP
                IF note_written = '0' THEN -- if the note already written, ignore the remaining loop runs

                        -- If a free space is found, enter the note number and velocity
                        -- or if until the end of the loop no space is found overwrite last entry

                        IF (reg_note_on(i) = '0' OR i = 9)  AND status_reg = SET_NOTE THEN
                                next_reg_note(i) <= data1_reg;
                                next_reg_velocity(i) <= data2_reg;
                                next_reg_note_on(i) <= '1'; -- And set the note_1_register to valid.
                                note_written := '1'; -- flag that note is written to supress remaining loop runs
                        END IF;
                END IF;
        END LOOP;
END IF;
```

DDS Instanziierung: Umständliche Lösung

School of Engineering

```
inst_dds_1 : dds
      port map(
        clk          => clk,
        reset_n      => rst_n,
        phi_incr_i   => LUT_midi2dds(to_integer(unsigned(note_1))),
        load_i     => load_i,
        attenu_i     => "001",
        dds_o => dds_1_o
       );

inst_dds_2 : dds
      port map(
        clk          => clk,
        reset_n      => rst_n,
        phi_incr_i   => LUT_midi2dds(to_integer(unsigned(note_2))),
        load_i     => load_i,
        attenu_i     => "001",
        dds_o => dds_2_o
       );

inst_dds_3 : dds
      port map(
        clk          => clk,
        reset_n      => rst_n,
        phi_incr_i   => LUT_midi2dds(to_integer(unsigned(note_3))),
        load_i     => load_i,
        attenu_i     => "001",
        dds_o => dds_3_o
       );
```

DDS Instanziierung: Übersichtliche Lösung

School of Engineering

```
dds_inst_gen : for i in 0 to 9 generate
    inst_dds : dds
      port map(
        clk          => clk,
        reset_n      => rst_n,
        phi_incr_i   => LUT_midi2dds(to_integer(unsigned(note_array(i)))),
        load_i     => load_i,
        attenu_i     => "001",
        dds_o => dds_o_array(i)
       );
end generate dds_inst_gen;
```

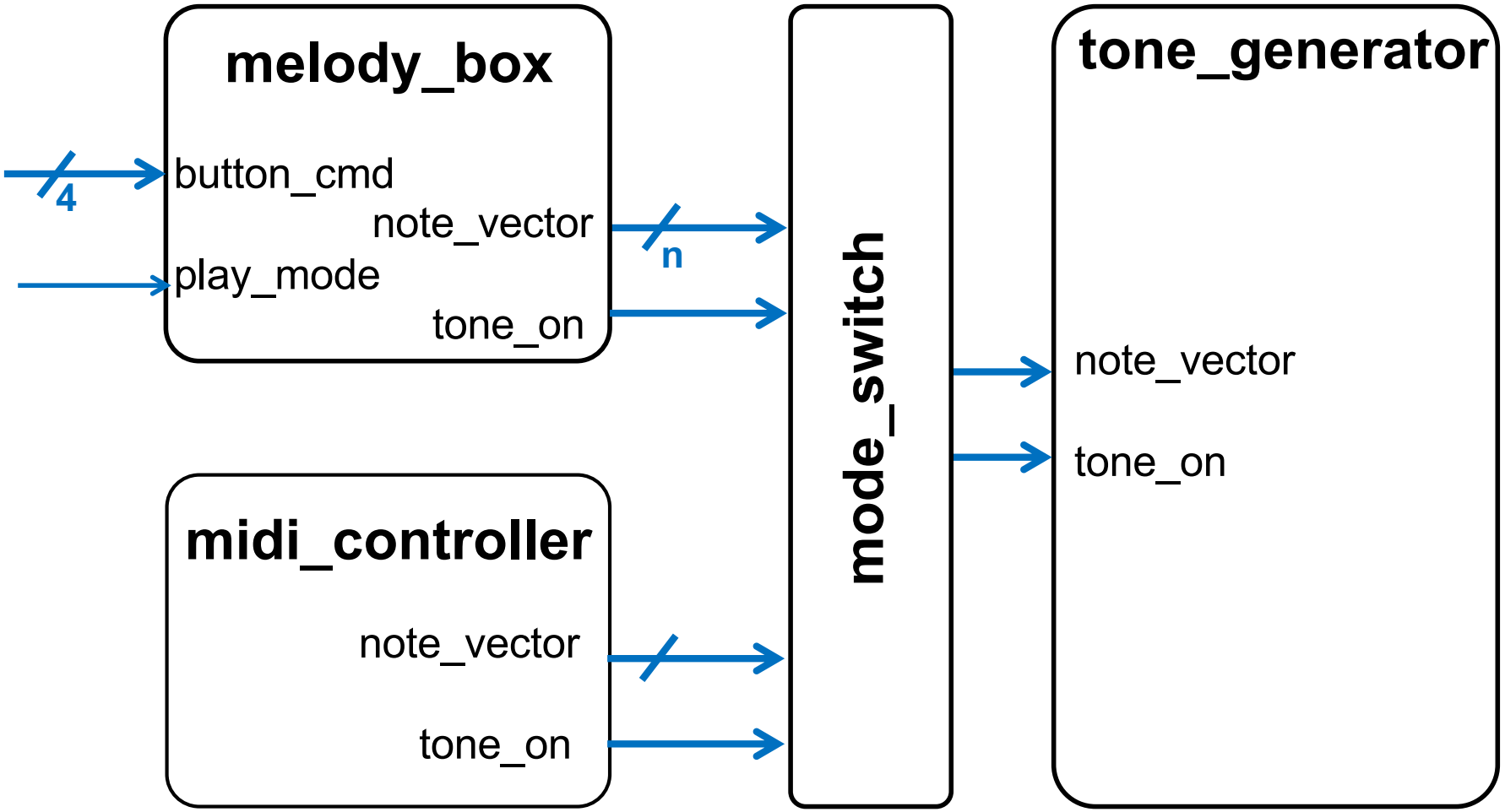
```
    note_array : in  t_tone_array;
    velocity_array : in  t_tone_array;
```

```
type t_tone_array is array (0 to 9) of std_logic_vector(6 downto 0) ;
```

```
type t_dds_o_array is array (0 to 9) of std_logic_vector(N_AUDIO-1 downto 0);
signal dds_o_array : t_dds_o_array;
```

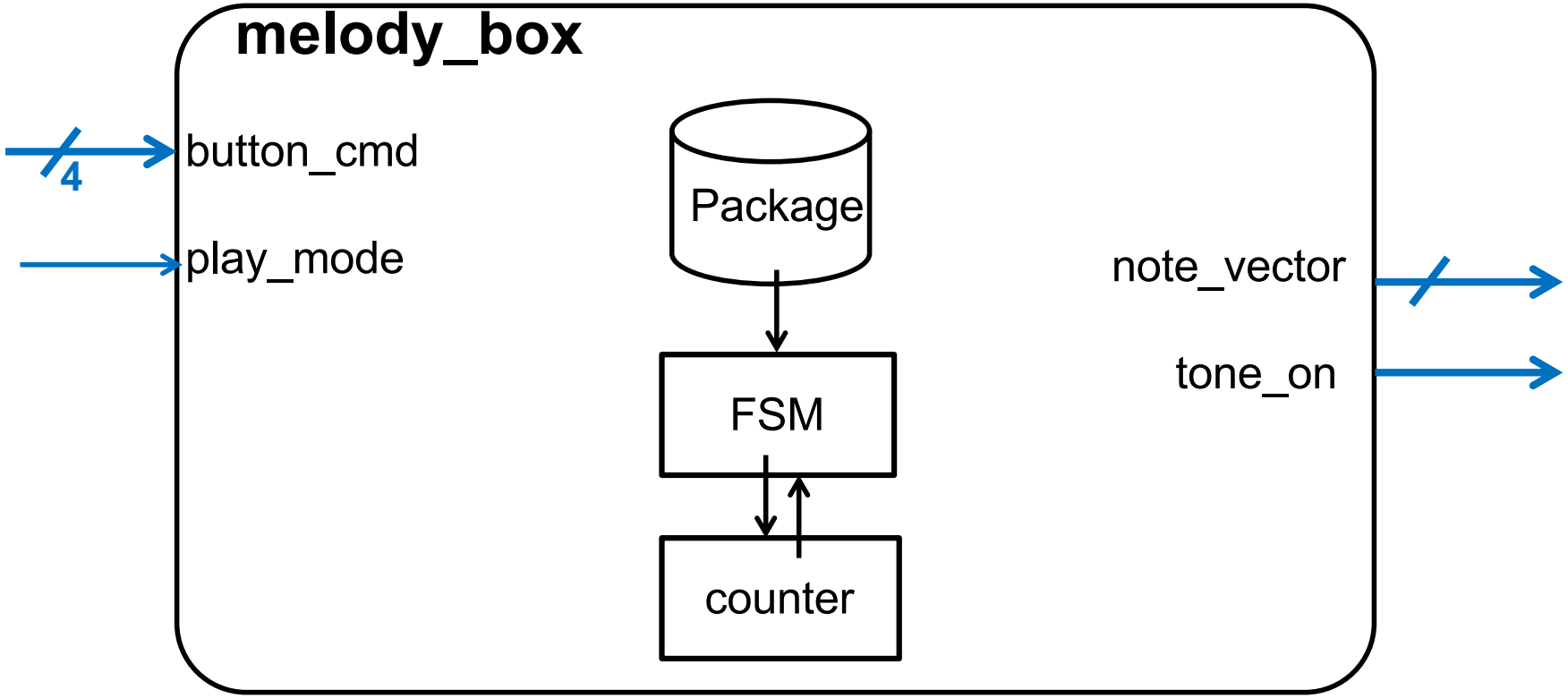
Extra Feature: Melody Box

Extra Feature: Melody Box



Melody-Box

School of Engineering



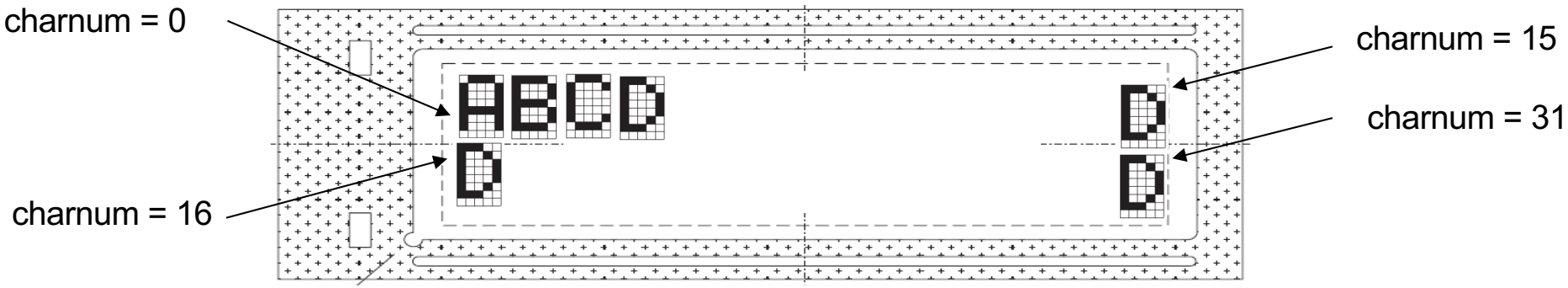
Comments

- *The package contains arrays describing the songs*
- *The songs can be described as a sequence of tones, and eventually have extra parameters like beat-speed (how long each tone lasts)*
- *The input play-mode selects either piano-mode (button_cmd are then the piano-keys) or song-mode (button_cmd are then the song-selector)*

LCD-Display and User Interface

2 Zeilen 16 Character LCD Display

School of Engineering



ASCII Zeichensatz

School of Engineering

ASCII-Codetabelle, Nummerierung in Hex

Code	…0	…1	…2	…3	…4	…5	…6	…7	…8	…9	…A	…B	… C	… D
0…	*NUL*	*SOH*	*STX*	*ETX*	*EOT*	*ENQ*	*ACK*	*BEL*	*BS*	*HT*	*LF*	*VT*	*FF*	*CR*
1…	*DLE*	*DC1*	*DC2*	*DC3*	*DC4*	*NAK*	*SYN*	*ETB*	*CAN*	*EM*	*SUB*	*ESC*	*FS*	*GS*
2…	*SP*	!	"	#	$	%	&	'	(	)	*	+	,	-
3…	0	1	2	3	4	5	6	7	8	9	:	;	<	=
4…	@	A	B	C	D	E	F	G	H	I	J	K	L	M
5…	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]
6…	`	a	b	c	d	e	f	g	h	i	j	k	l	m
7…	p	q	r	s	t	u	v	w	x	y	z	{	|	}

For more details check in LCD datasheet page 13:
Character Generator ROM Pattern

ASCII Zeichensatz

School of Engineering

Practical to define values in a package, for example:

```
-- NUMBERS ---------------------------------------------------------------------
constant Num_0   : std_logic_vector (7 downto 0) :=    x"30";
constant Num_1   : std_logic_vector (7 downto 0) :=    x"31";
constant Num_2   : std_logic_vector (7 downto 0) :=    x"32";
….

-- LETTERS ---------------------------------------------------------------------
constant Let_A     : std_logic_vector (7 downto 0) :=    x"41";
constant Let_B     : std_logic_vector (7 downto 0) :=    x"42";
constant Let_C     : std_logic_vector (7 downto 0) :=    x"43";
….
```

LCD Anschlüsse am DE2-115 und DE2-classic

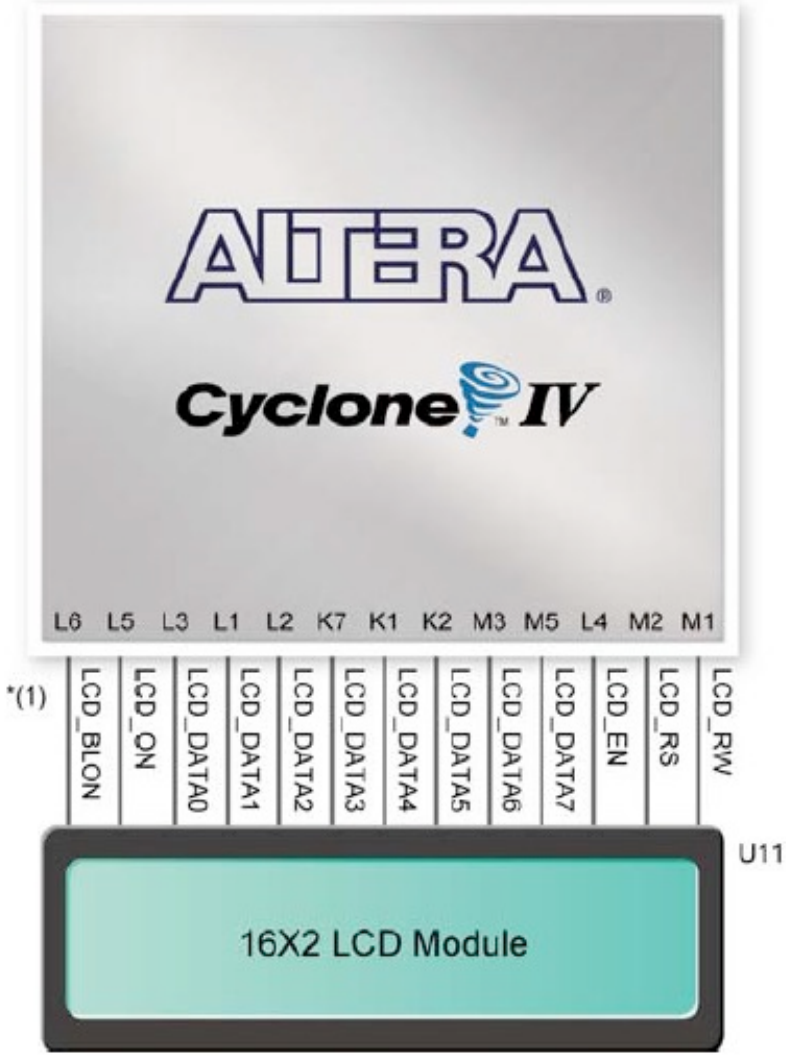
School of Engineering

```
LCD_RW = '0'
LCD_ON = '1'
```

LCD_BLON = Backlight nicht unterstützt (gemäss DE2_115 User_Manual, aber in schematics verbunden. Try out!!)

The LCD Module supports two Data lengths (DL):
DL: 4bits (nibble transfers)
DL: 8 bits (byte transfers)

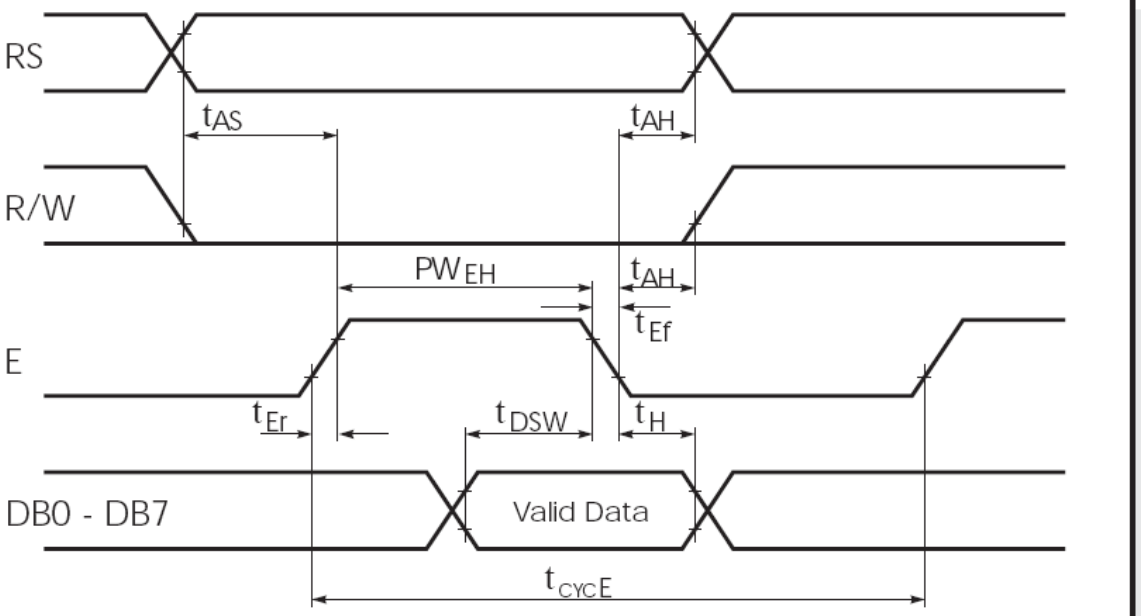
Achtung:
Der LCD VHDL Driver Block arbeitet im 4-bit Modus
LCD_DATA7 = MSB
LCD_DATA4 = LSB
LCD_DATA0-3 = nicht angeschlossen



Schnittstelle: LCD-Driver zu LCD-Display

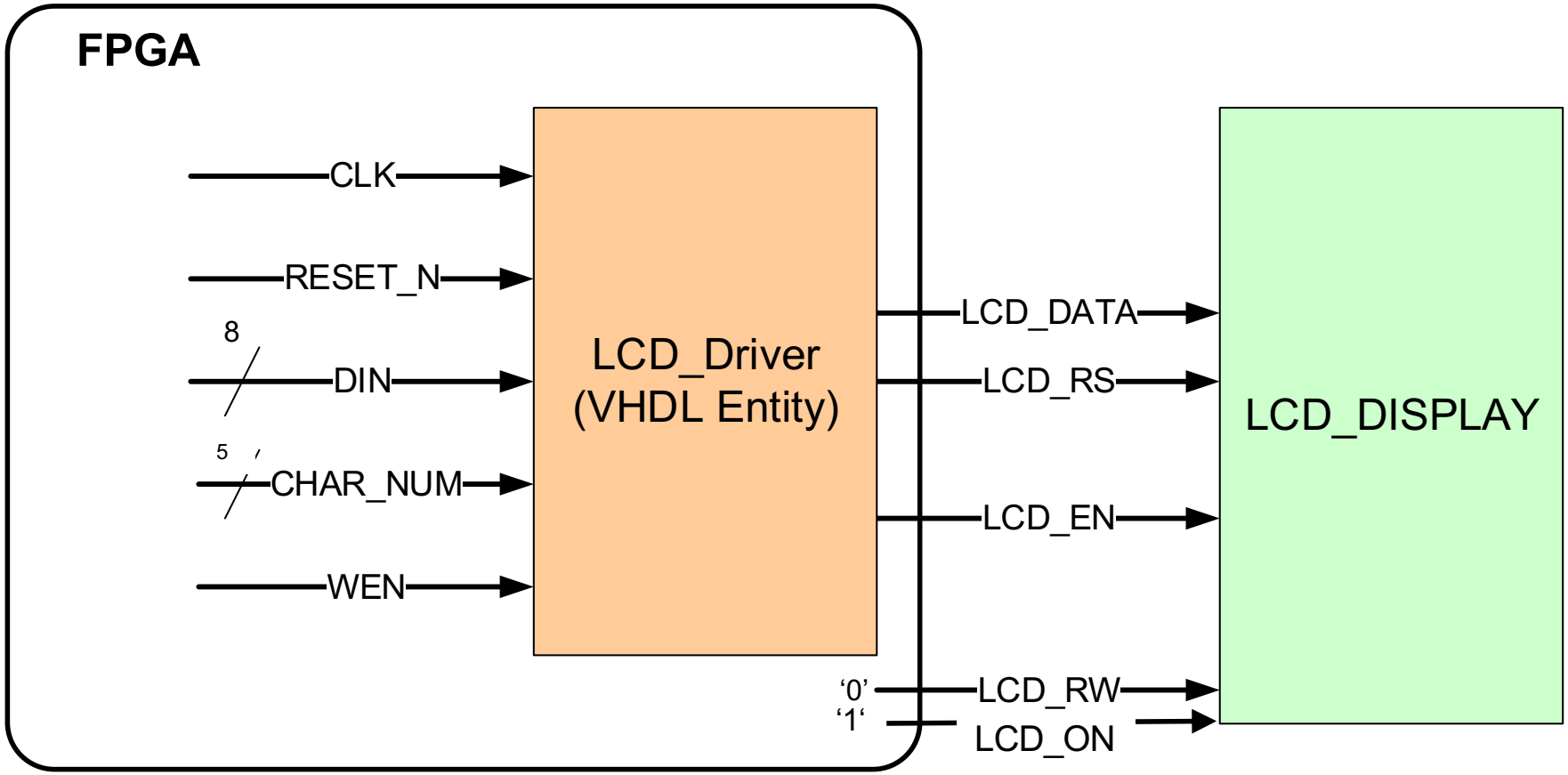
School of Engineering

Wird vom LCD Driver (gegebenes VHDL) erzeugt



Write Operation (from MPU to display)

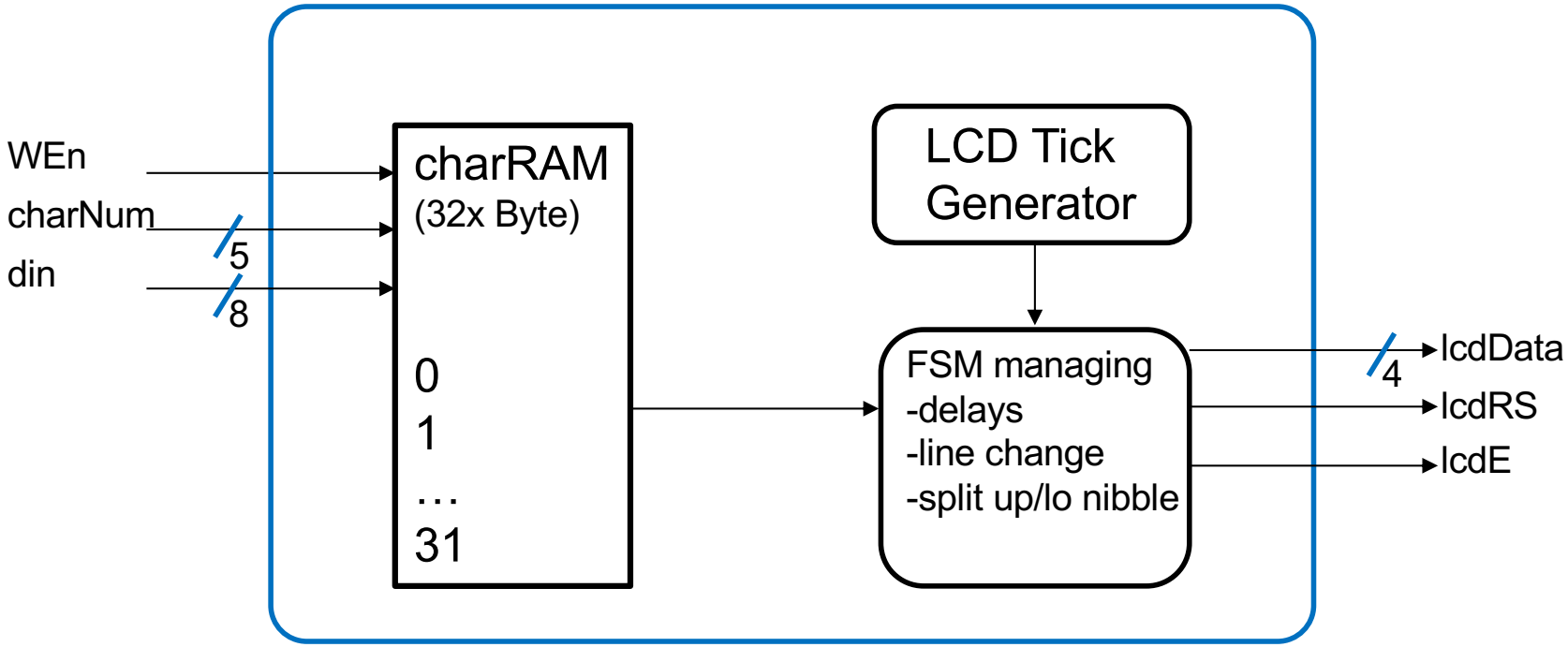
LCD Driver



Achtung:

- Nur die MSB Datenleitungen LCD_DATA(7..4) am Driver anschliessen
- Es werden nur Schreibvorgang (von MCU zum Display) unterstützt

LCDDRIVER 2x16



Achtung: Schlechter Code Style of LCD-Driver (gefunden im Internet)

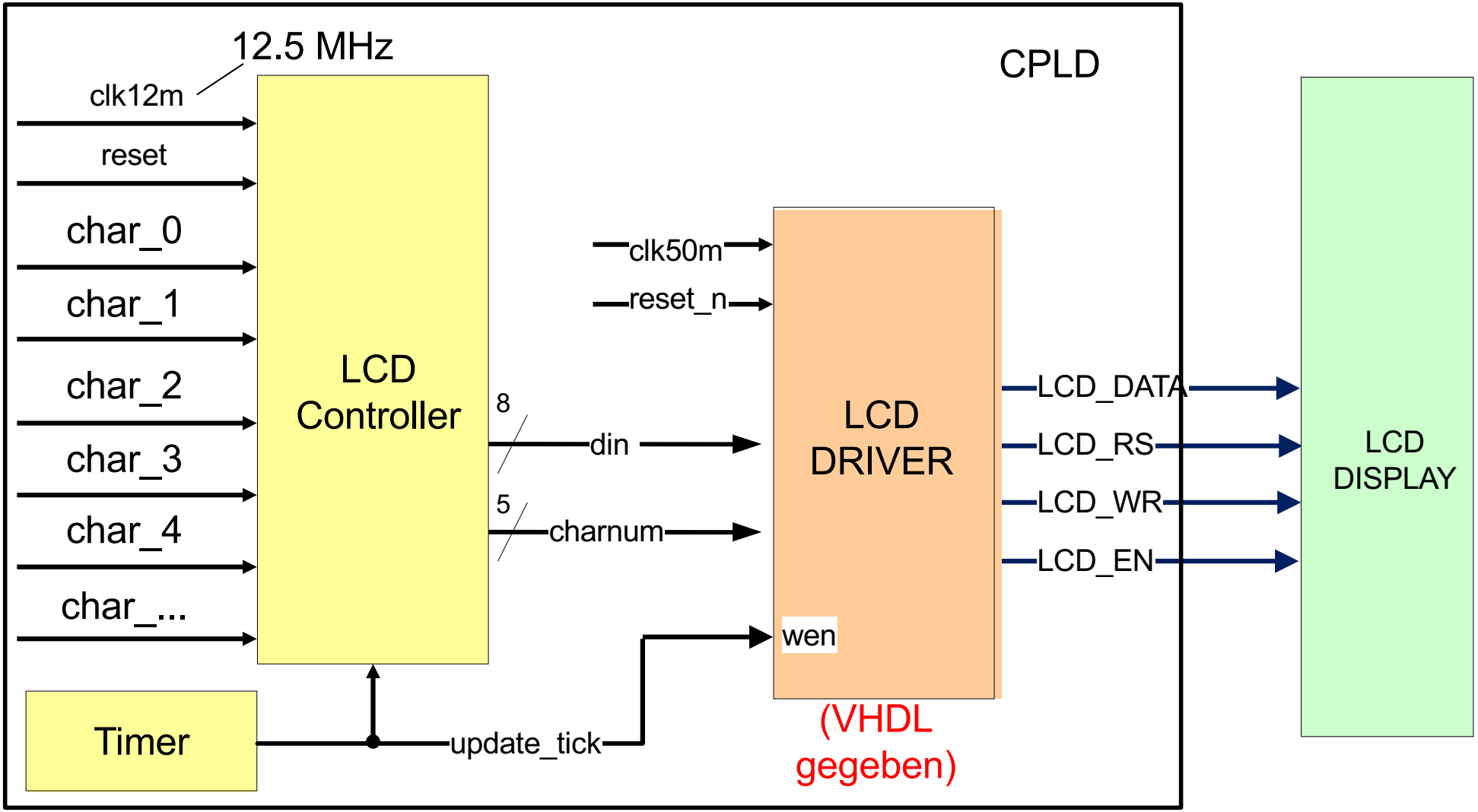
Register and Comb-logic processes are not separated.

Recommend to use such code style only after further experience acquired in VHDL.

Because harder to recognize which hardware parts are generated with code.

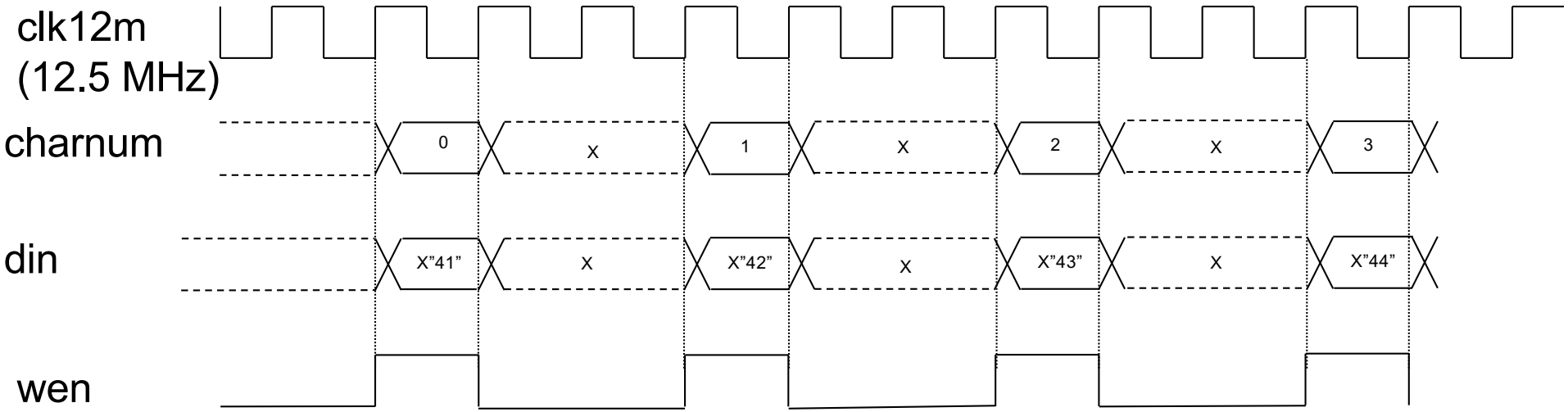
LCD Controller

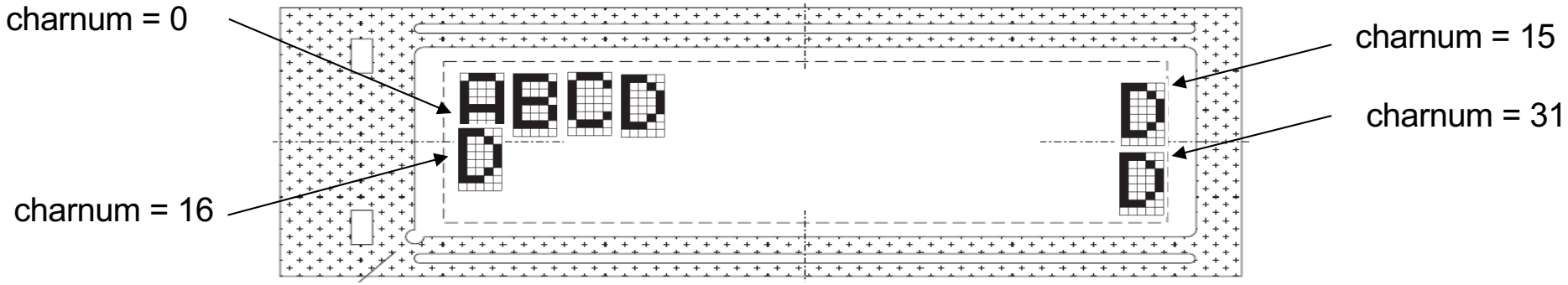
School of Engineering

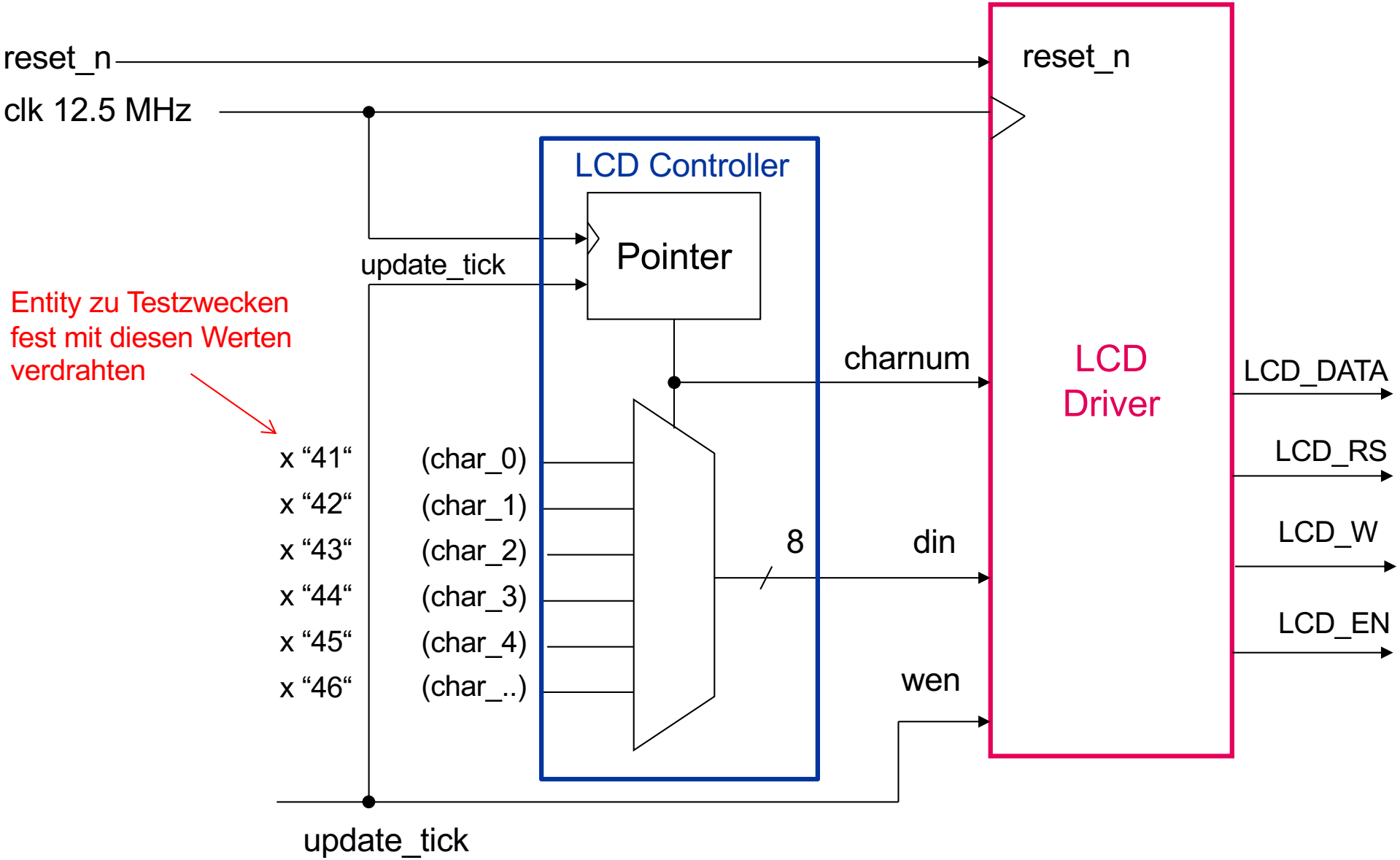


Schnittstelle: LCD-Controller zu LCD-Driver

School of Engineering





Testen des LCD–Controllers auf Board

School of Engineering