„Can you speak USB-ish?“

Low-speed USB host implemented in GA144

Daniel Kalny
on behalf of GreenArrays

Forth Day 2019

USB is…

…Universal Serial Bus

USB is...

... a means of connecting peripherals to computers

how does USB work?

USB in 10 minutes

specification

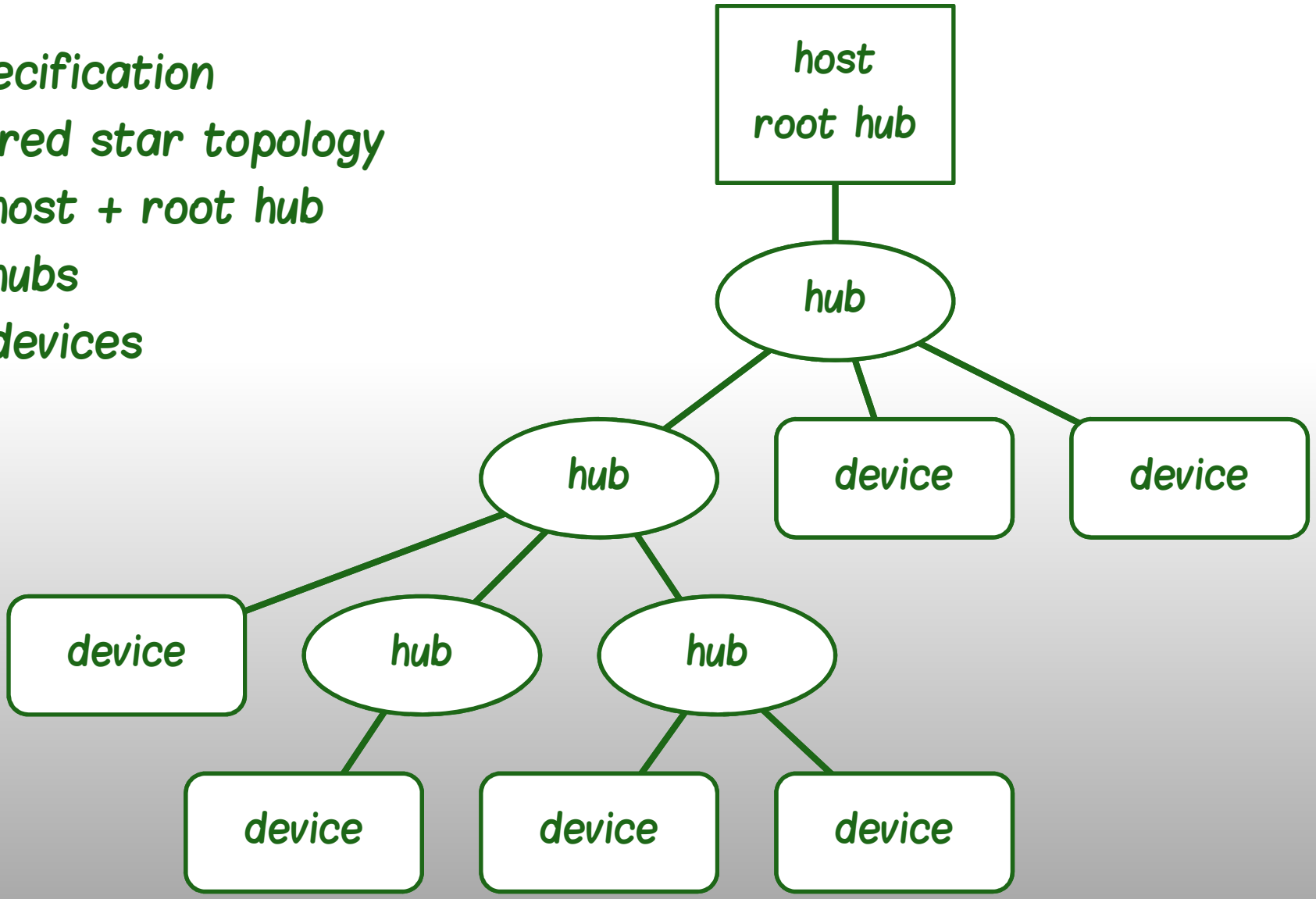
usbmadesimple.co.uk

USB in 10 minutes

specification

tiered star topology

- host + root hub
- hubs
- devices



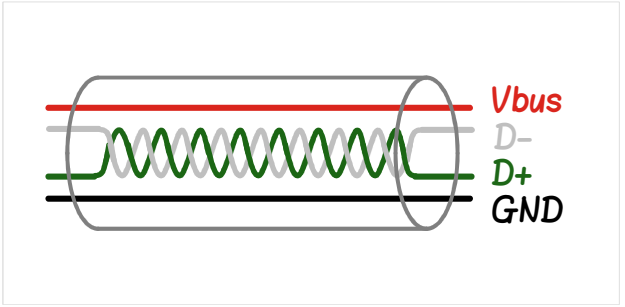
USB in 10 minutes

specification

tiered star topology

- host + root hub
- hubs
- devices

cables and connectors



type A plug

USB in 10 minutes

specification

tiered star topology

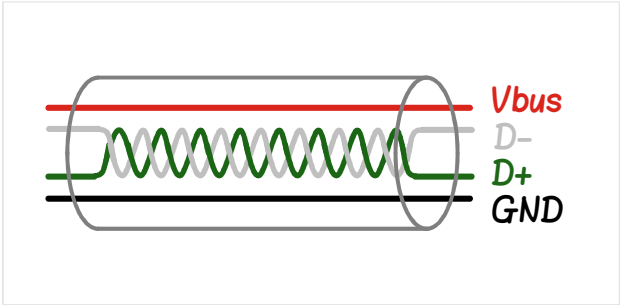
- host + root hub
- hubs
- devices

cables and connectors

half-duplex differential signaling

power distribution

host D+ and D- pulled down with
15k resistors



type A plug

USB in 10 minutes

data speeds

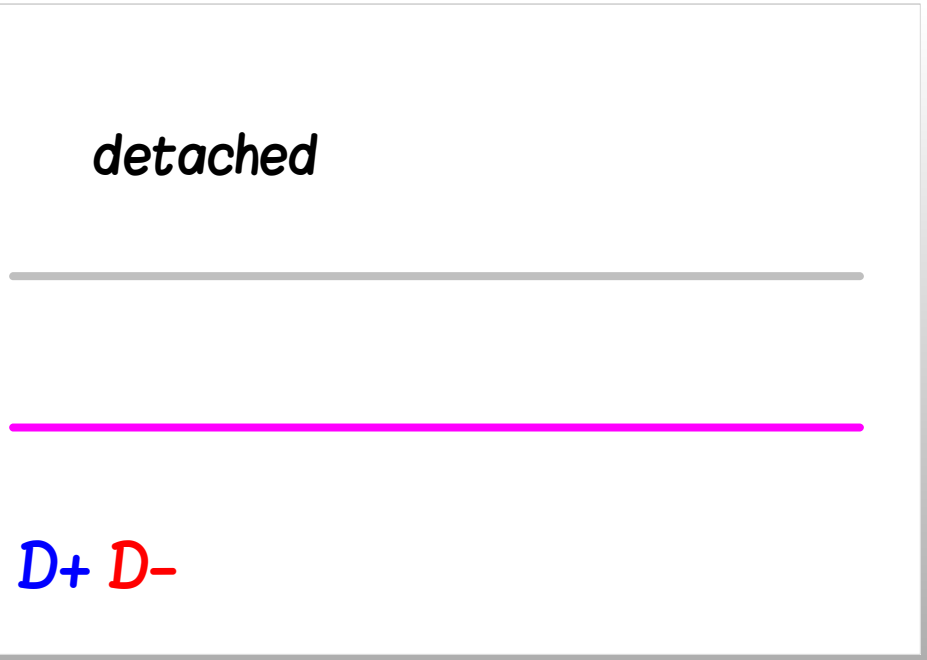
speed defined with

a pull up 1k5 resistor

line states

- detached

low speed	1.5 Mb/s	D-
full speed	12 Mb/s	D+
high speed	480 Mb/s	D+



USB in 10 minutes

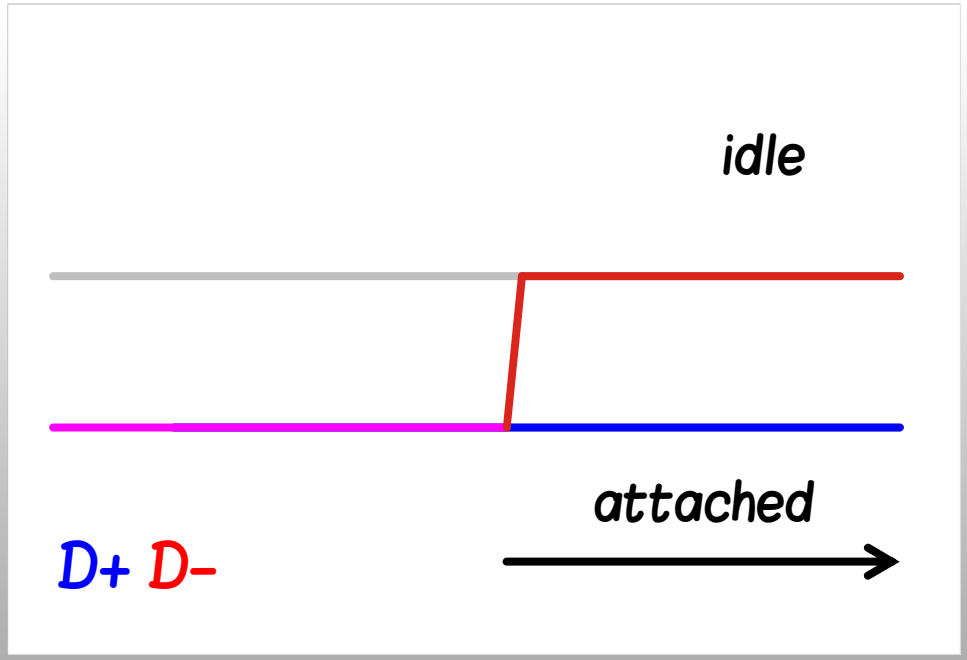
data speeds

speed defined with

a pull up 1k5 resistor

line states

- detached
- attached
- idle



USB in 10 minutes

data speeds

speed defined with

a pull up 1k5 resistor

line states

- detached
- attached
- idle

bus states

- J, K, SE0, ~~SE1~~

low speed bus

J	D+ low, D- high
K	D+ high, D- low
SE0	D+ low, D- low

USB in 10 minutes

data speeds

speed defined with

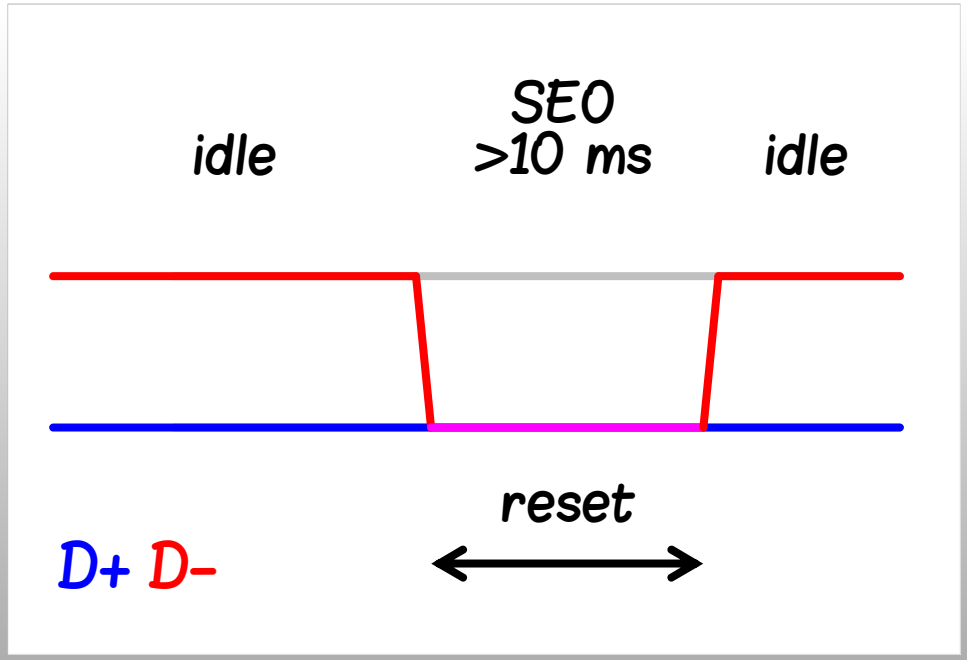
a pull up 1k5 resistor

line states

- detached
- attached
- idle

bus states

- J, K, SE0, ~~SE1~~
- reset



USB in 10 minutes

data speeds

speed defined with

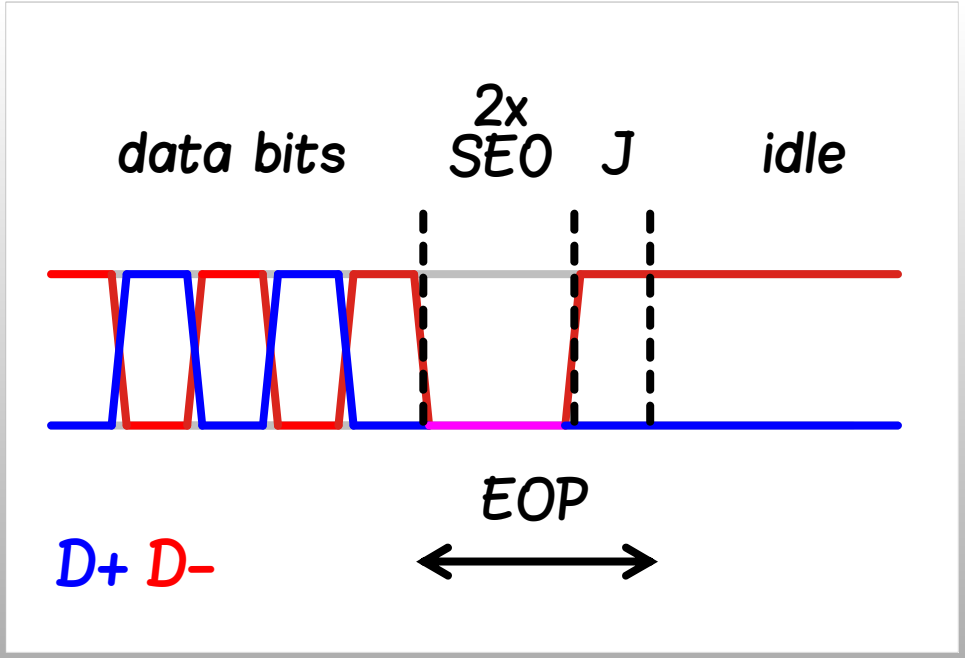
a pull up 1k5 resistor

line states

- detached
- attached
- idle

bus states

- J, K, SE0, ~~SE1~~
- reset
- end of packet



USB in 10 minutes

data speeds

speed defined with

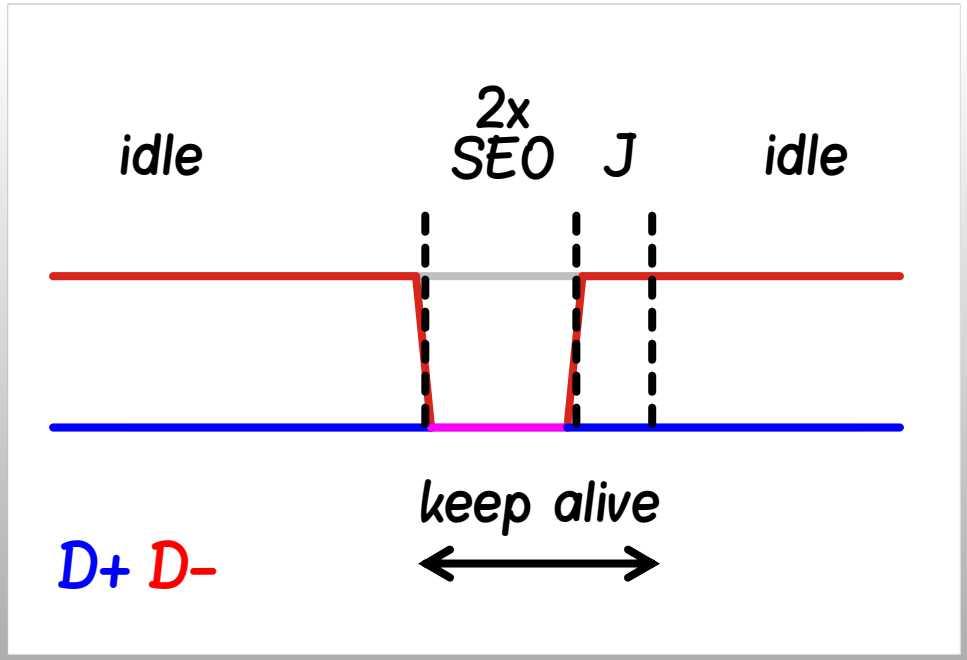
a pull up 1k5 resistor

line states

- detached
- attached
- idle

bus states

- J, K, SE0, ~~SE1~~
- reset
- end of packet
- keep alive



USB in 10 minutes

addresses

- range 0 – 127 (7 bits)
- uninitialized device addr = 0

endpoints

- range 0 – 15 (4 bits)
- direction either IN or OUT (endpoint 0 both directions)

endianness

- bytes transfered least significant bit first
- multi-byte data least significant byte first

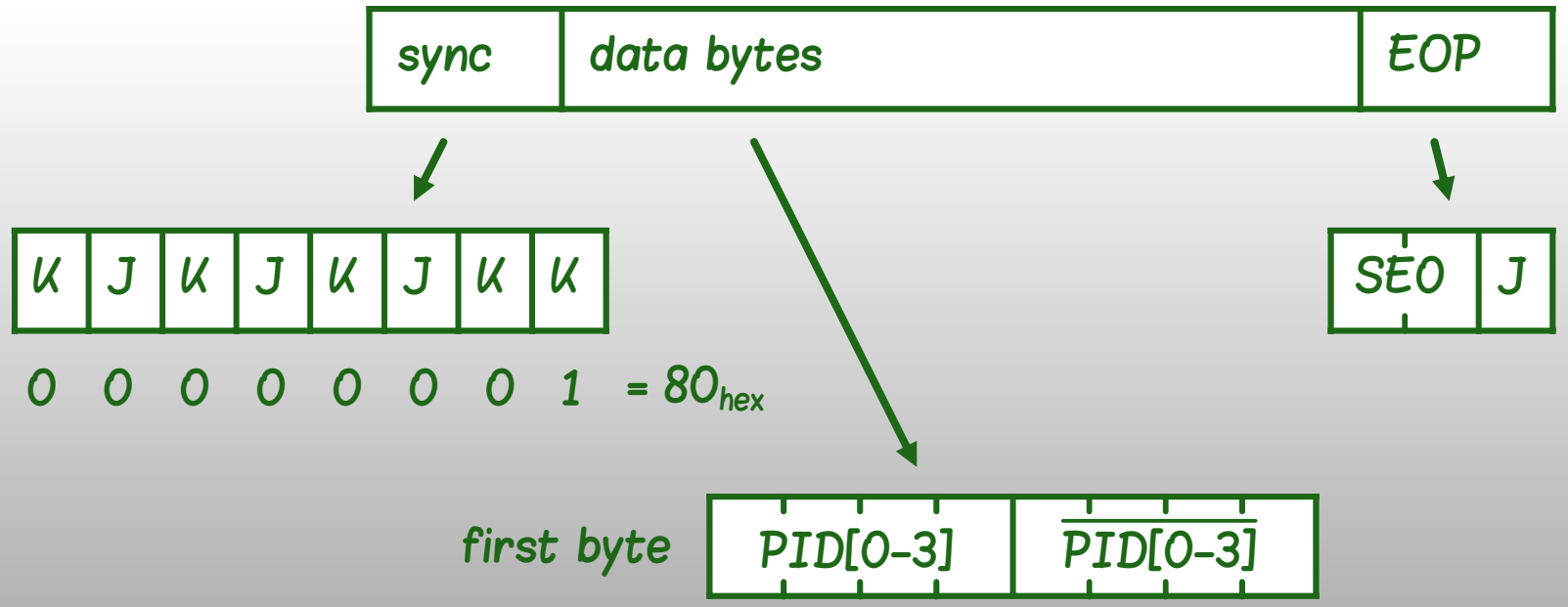
encoding

- NRZI: 0 bit – state change, 1 bit – no change
- up to 6 one bits, then a zero bit inserted (bit stuffing)

USB in 10 minutes

packets

- smallest transmission element
- integer number of bytes
- bus in idle state before and after a packet



USB in 10 minutes

token packet

- first packet in a transaction
- indicates address, endpoint, purpose of a transaction

IN
OUT
SETUP

USB in 10 minutes

data packet

- used for transfering data
- payload 0 to 8 bytes

DATA0
DATA1

USB in 10 minutes

handshake packet

- used to confirm status of a transaction
- provide handshake between host and device

ACK
NAK
STALL

USB in 10 minutes

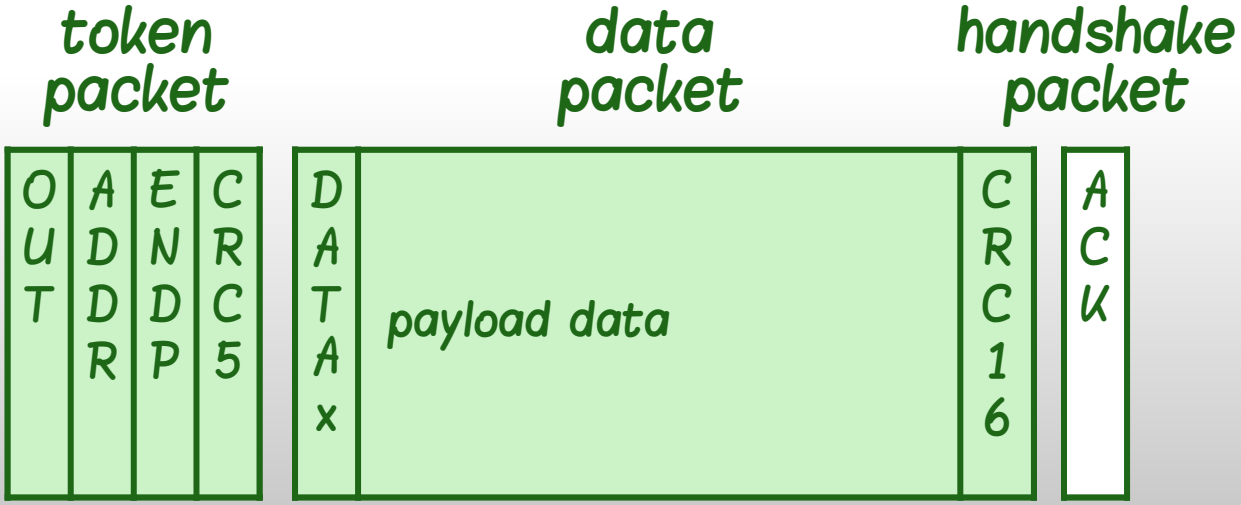
transactions

- sequence of three packets
- secure transfer of data

USB in 10 minutes

OUT transaction

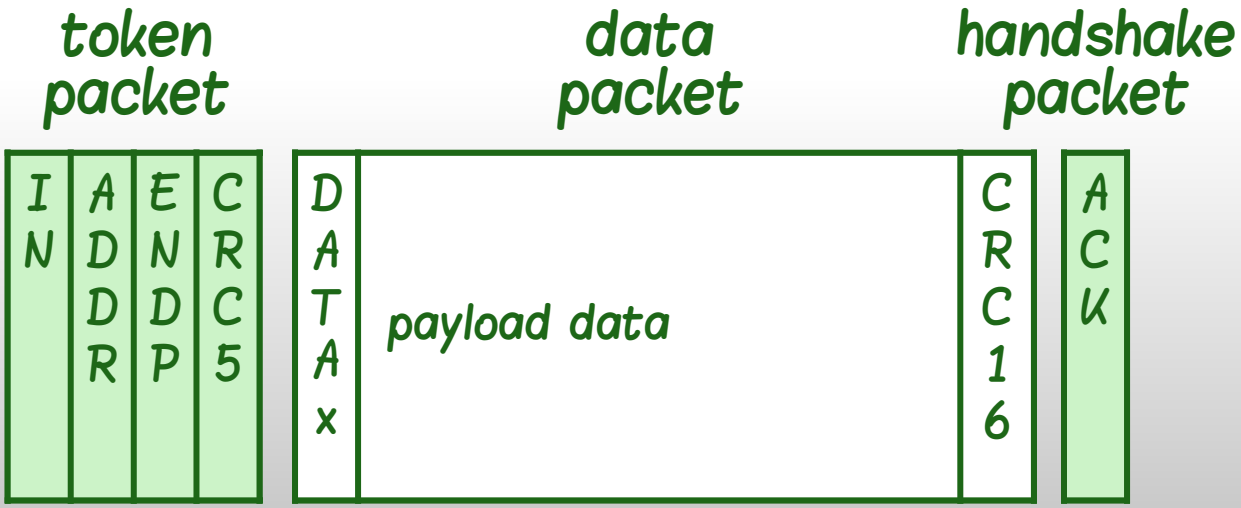
- transfer data from host to device
- DATAx is alternating between DATA0 and DATA1



USB in 10 minutes

IN transaction

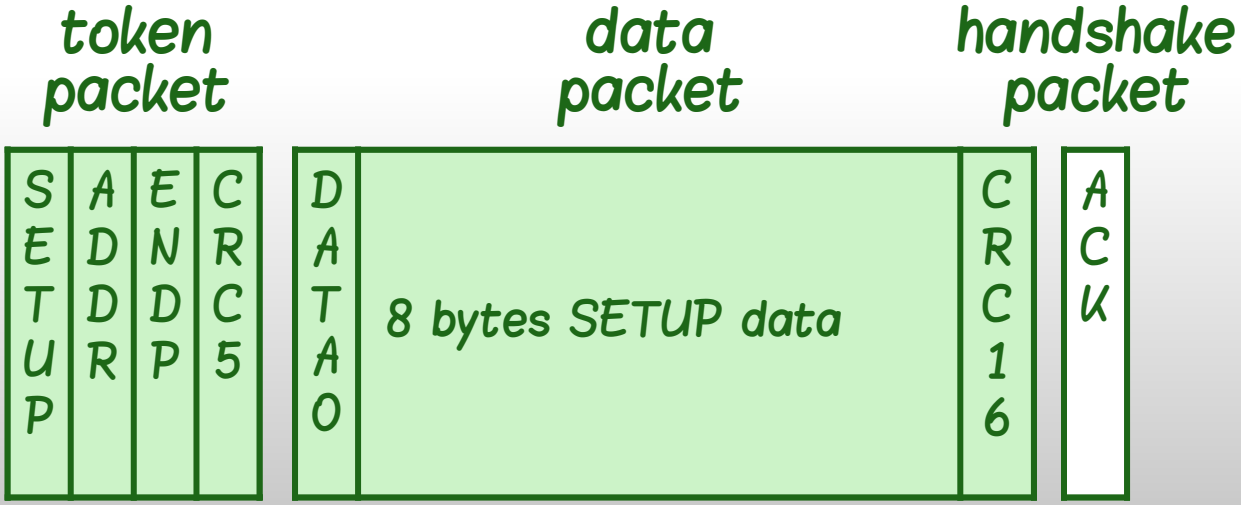
- transfer data from device to host
- DATAx is alternating between DATA0 and DATA1



USB in 10 minutes

SETUP transaction

- similar to OUT transaction, payload always 8 bytes
- only DATA0 used



USB in 10 minutes

transfers

- control
  - used for initial configuration of a device by the host
  - bidirectional on endpoint 0 IN and 0 OUT only
- interrupt
  - IN or OUT transactions
  - low throughput
  - regular data transfer (keyboard, mouse)

USB in 10 minutes

enumeration

- get descriptors (device, configuration, interface…)
- set address, select configuration, interface, etc.

Human Interface Device (HID)

- HID class descriptors, reports
- Report protocol vs Boot protocol

PROJECT

aim of the project

1) low-speed USB host in GA144
2) keyboard controller
3) demo – USB keyboard & etherForth editor

simplified specs for GA144

hardware

- no control of rise and fall times
- no root hub, no support for hubs

protocol

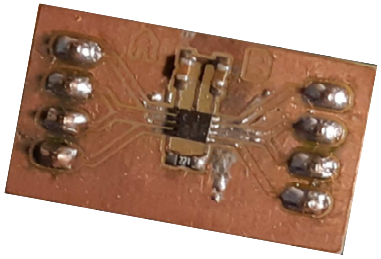
- no address assigned to a device (only address zero used)
- no check of DATA0 and DATA1 toggling
- disregard STALL handshake

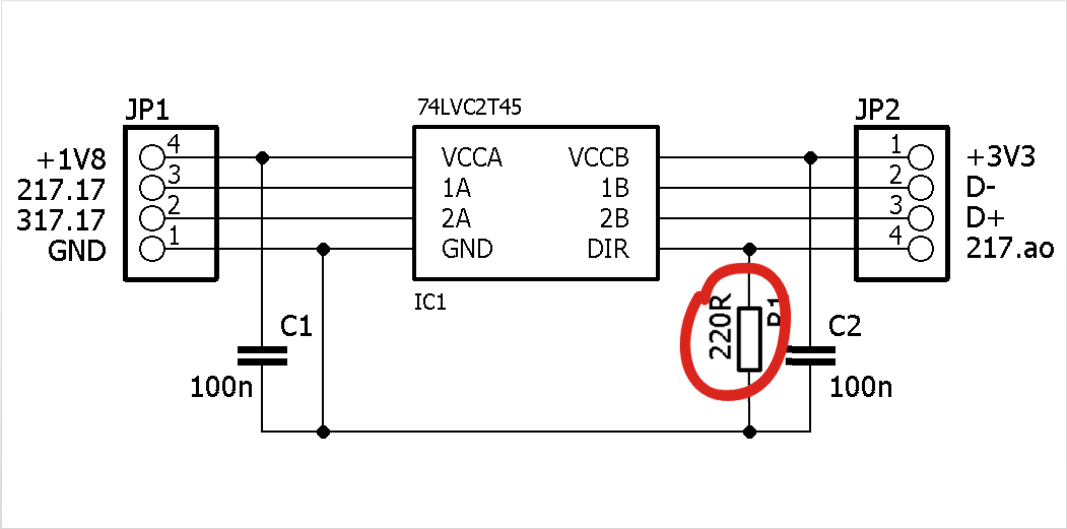
keyboard

- no auto repeat
- ignore simultaneously pressed keys

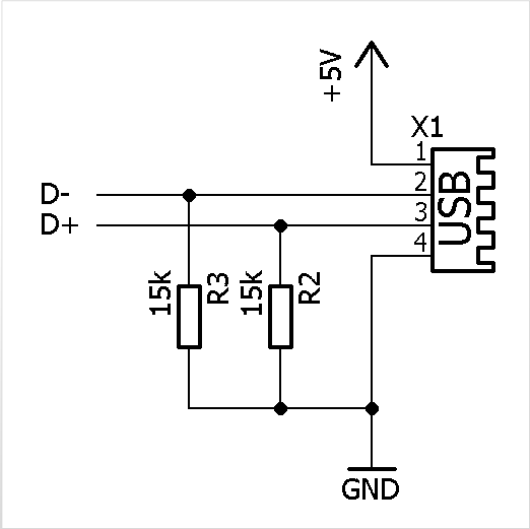
HARDWARE

hardware





voltage level shifter 74LVC2T45



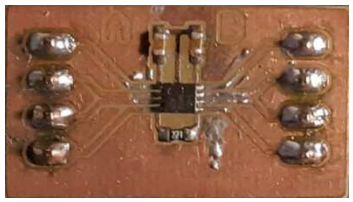
USB connector

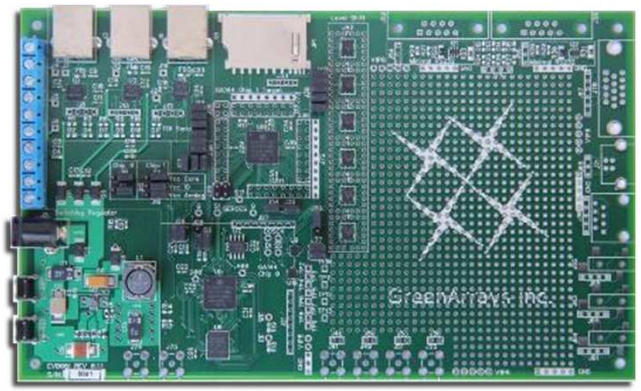
datasheet: assets.nexperia.com/documents/data-sheet/74LVC_LVCH2T45.pdf

setup

GA EVB001

voltage level shifter





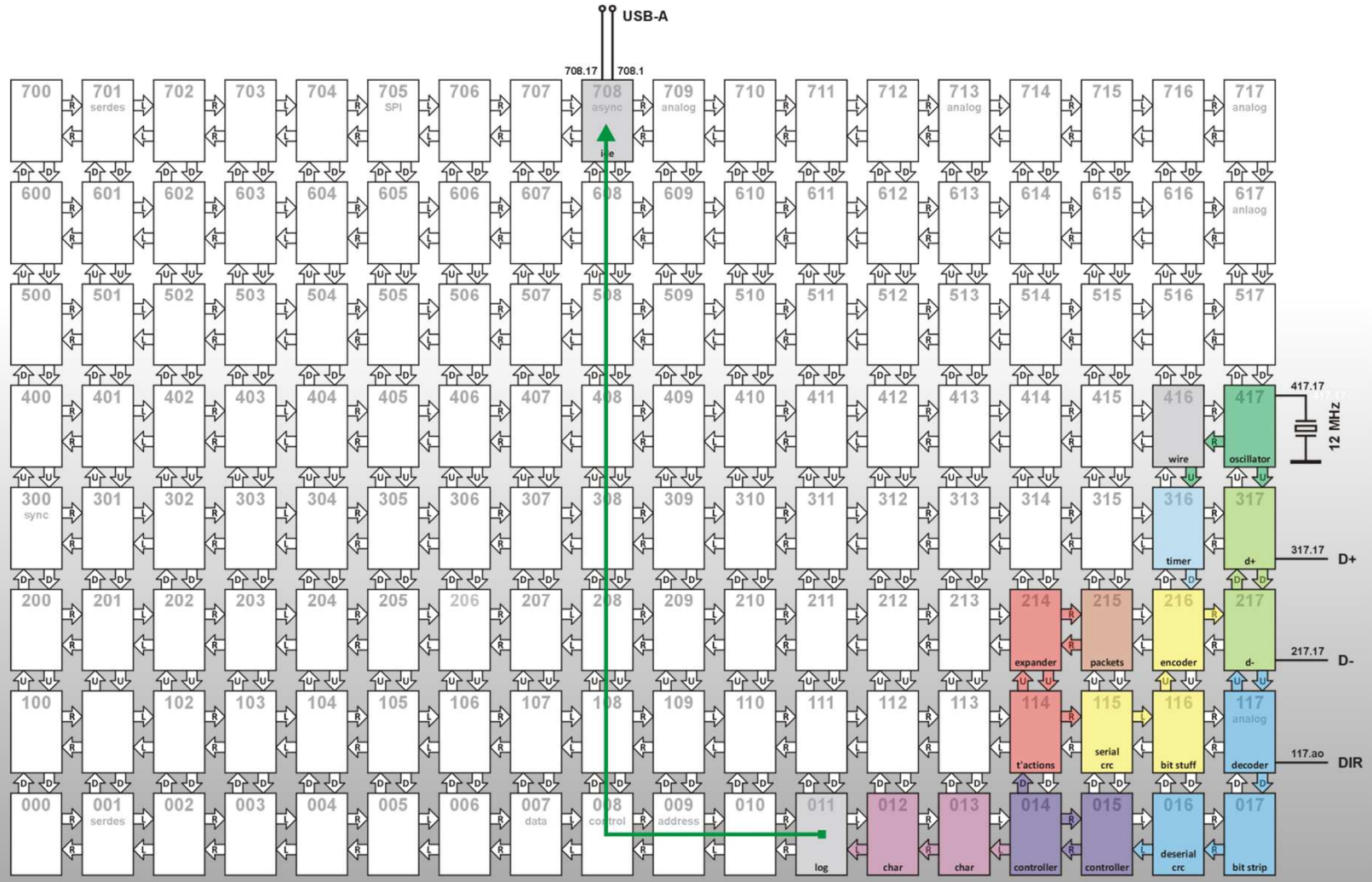
USB keyboard

arrayForth IDE

FLOORPLAN

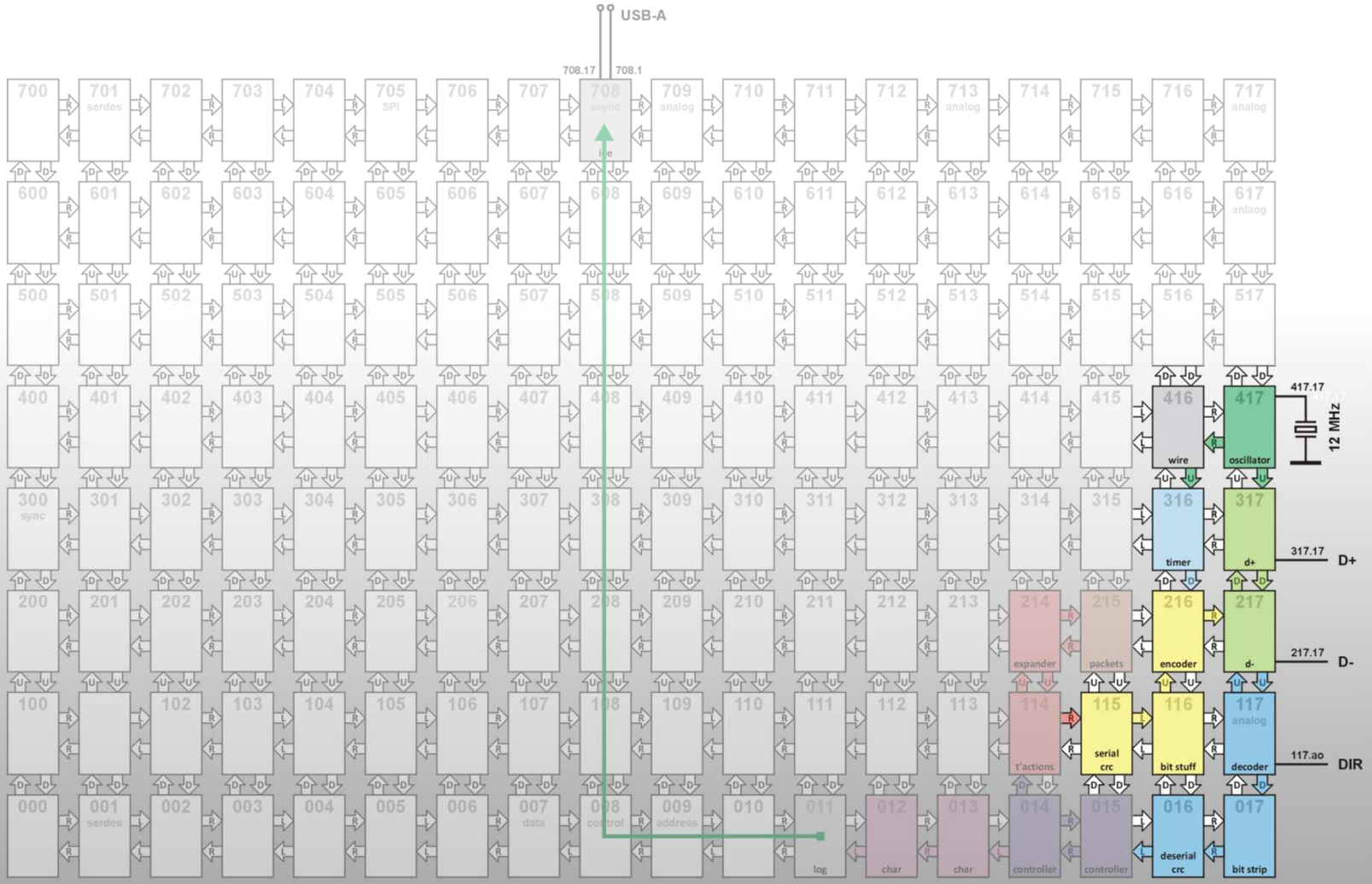
floorplan

project - host chip - 18 nodes



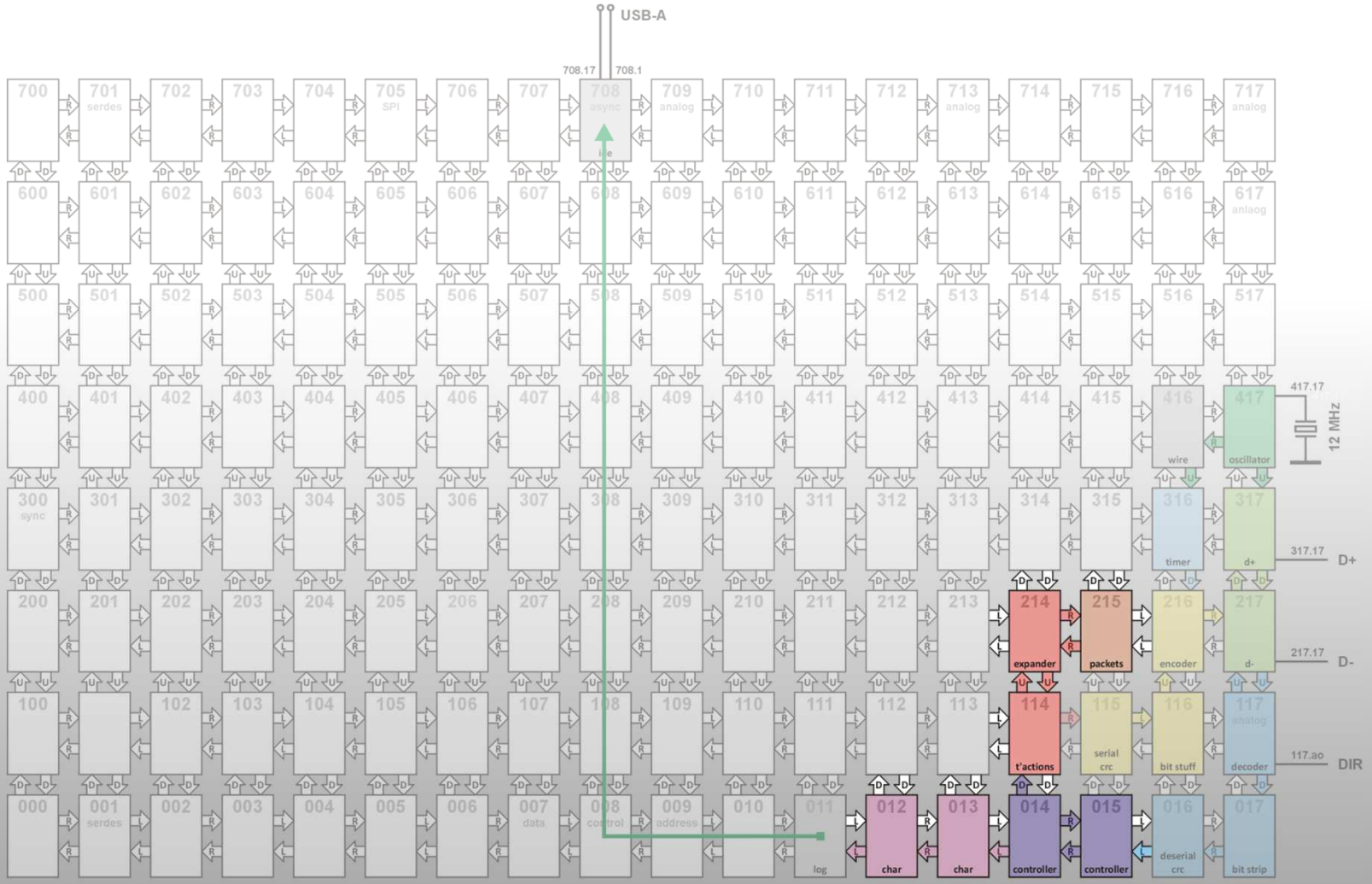
floorplan

serial interface engine



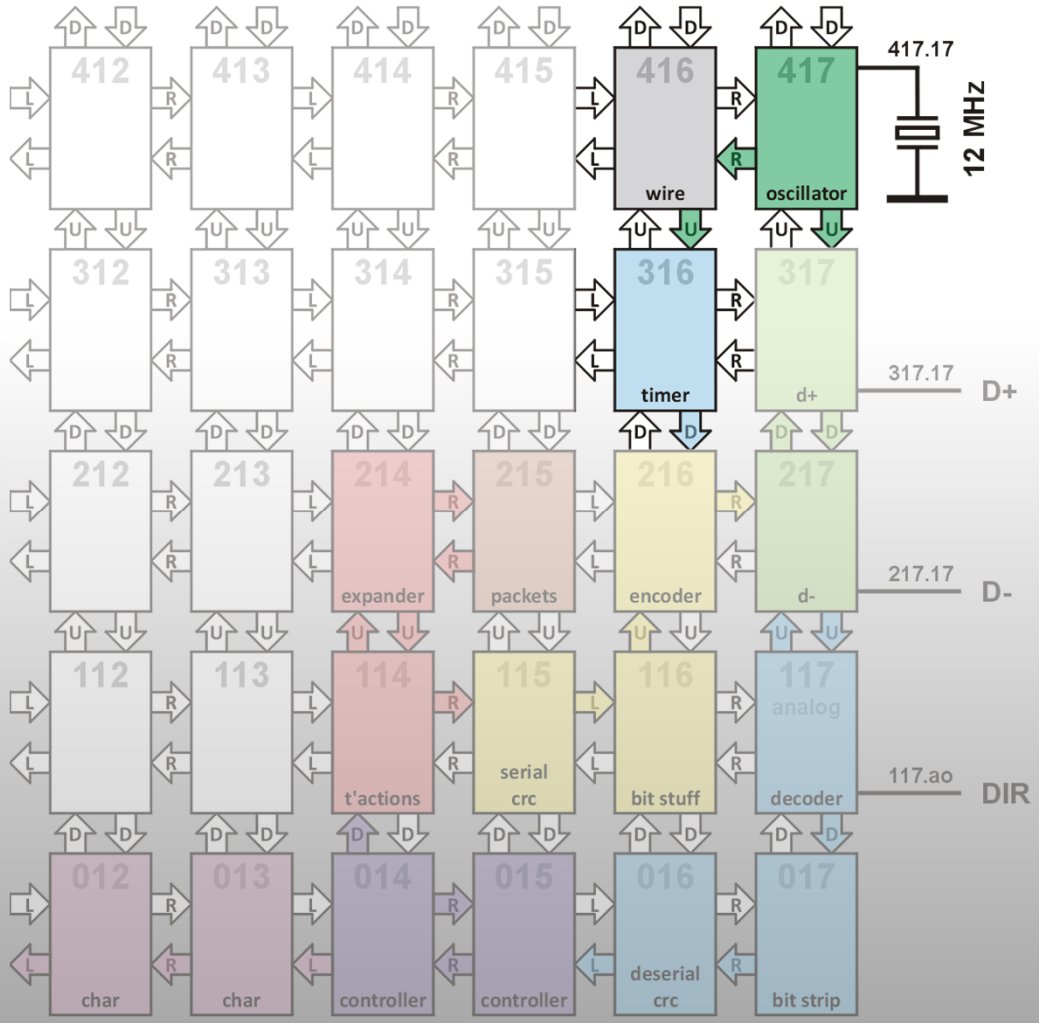
floorplan

keyboard controller



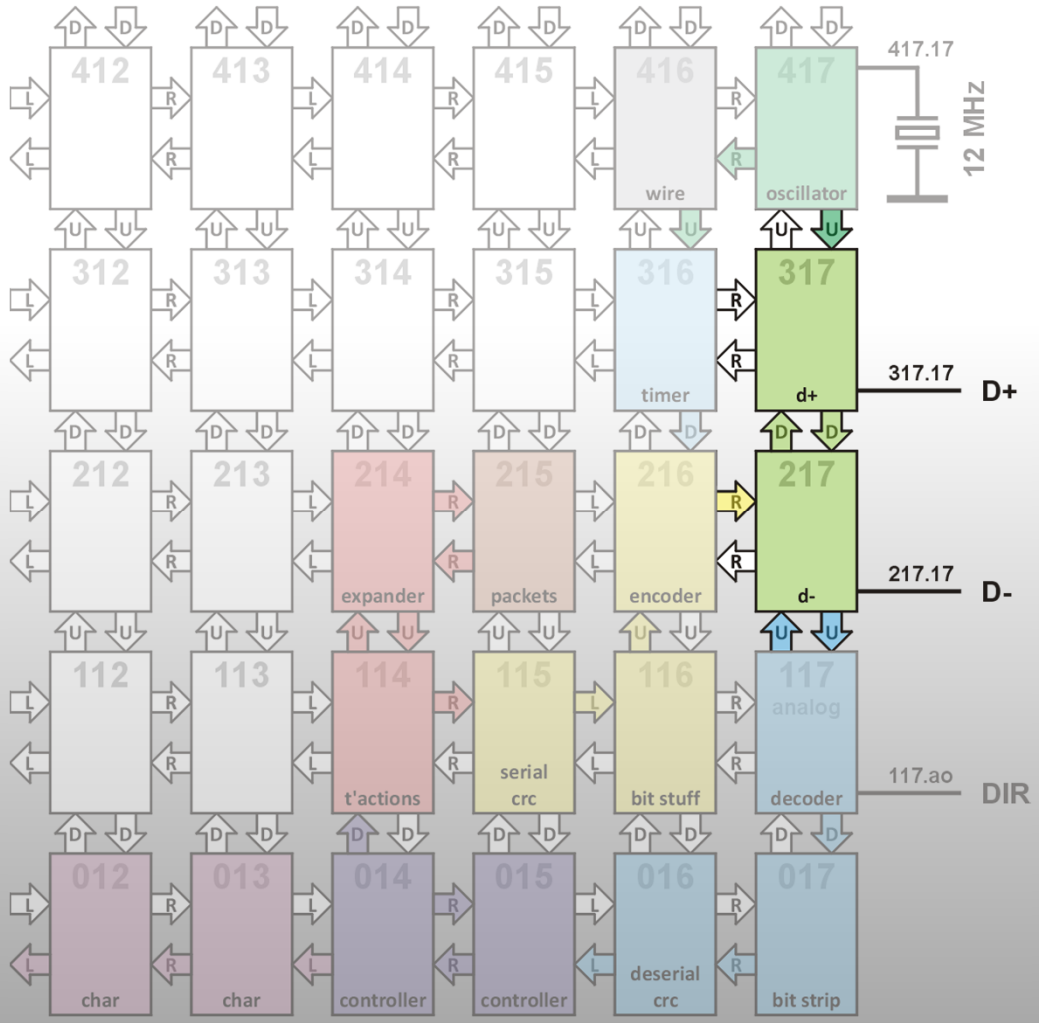
floorplan

serial interface engine - clock



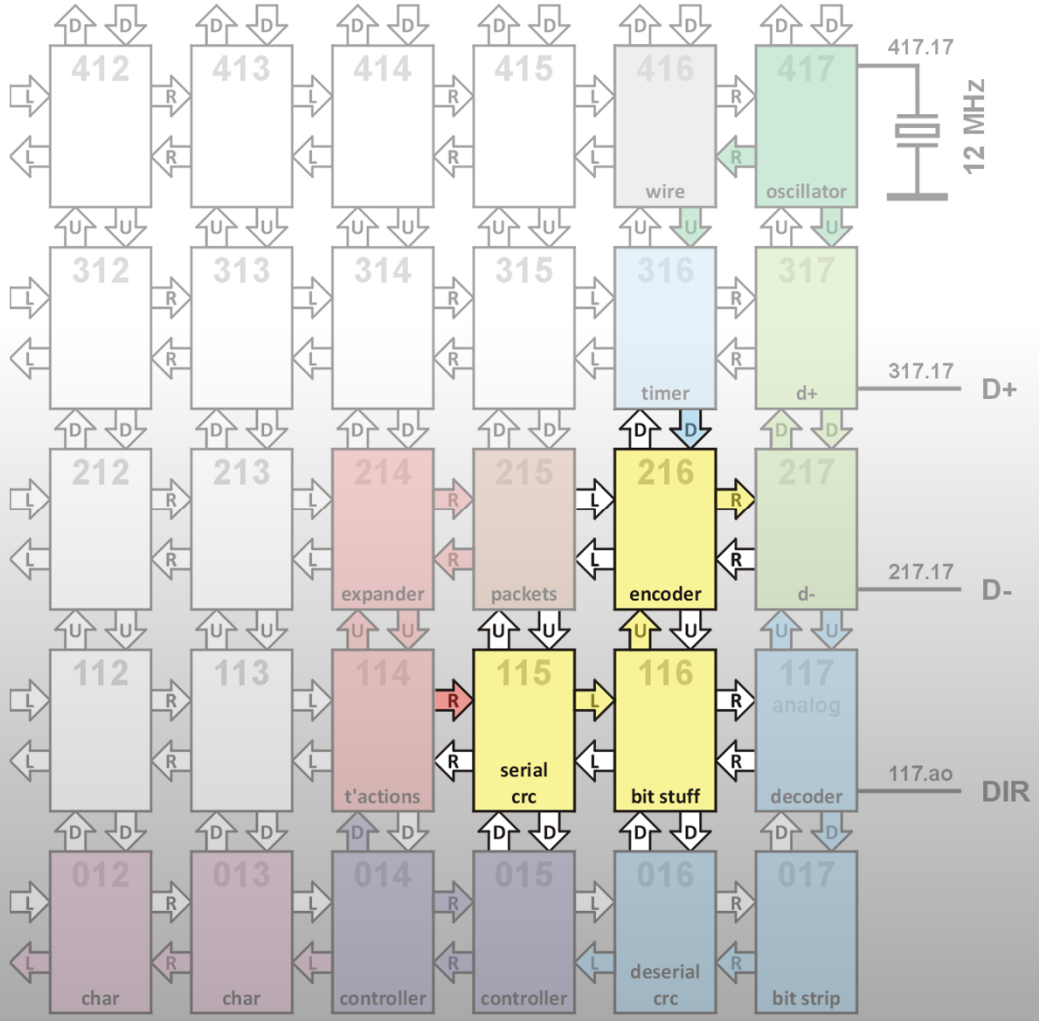
floorplan

serial interface engine - transceivers



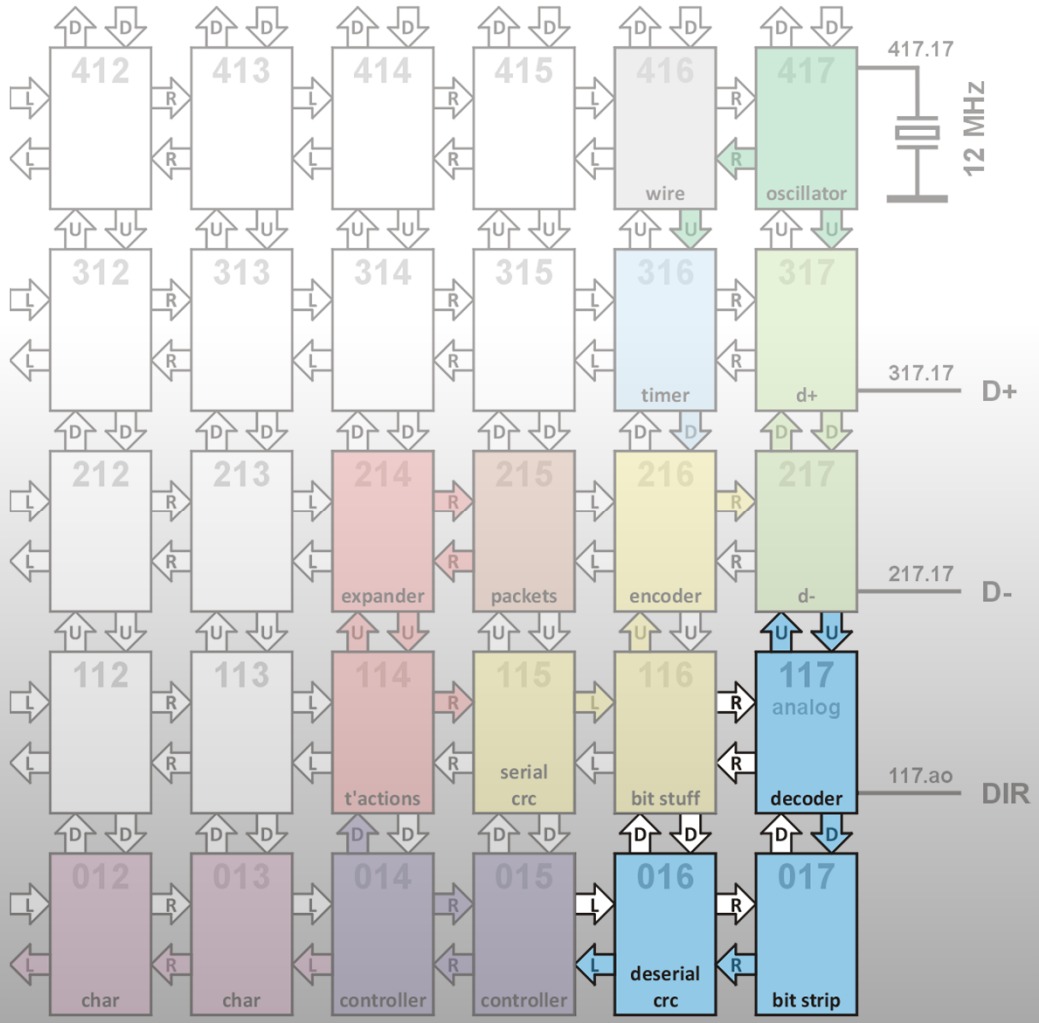
floorplan

serial interface engine - transmit paths



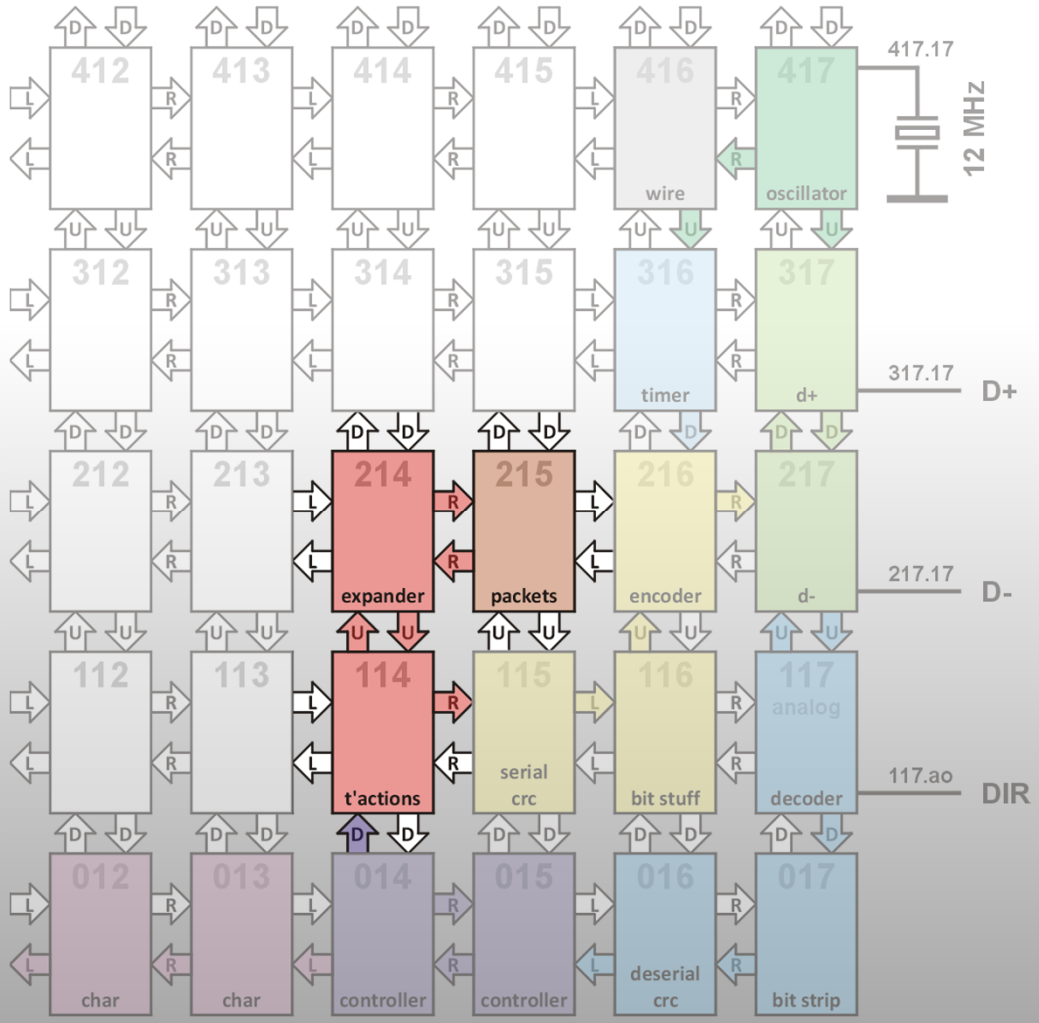
floorplan

serial interface engine - receive paths



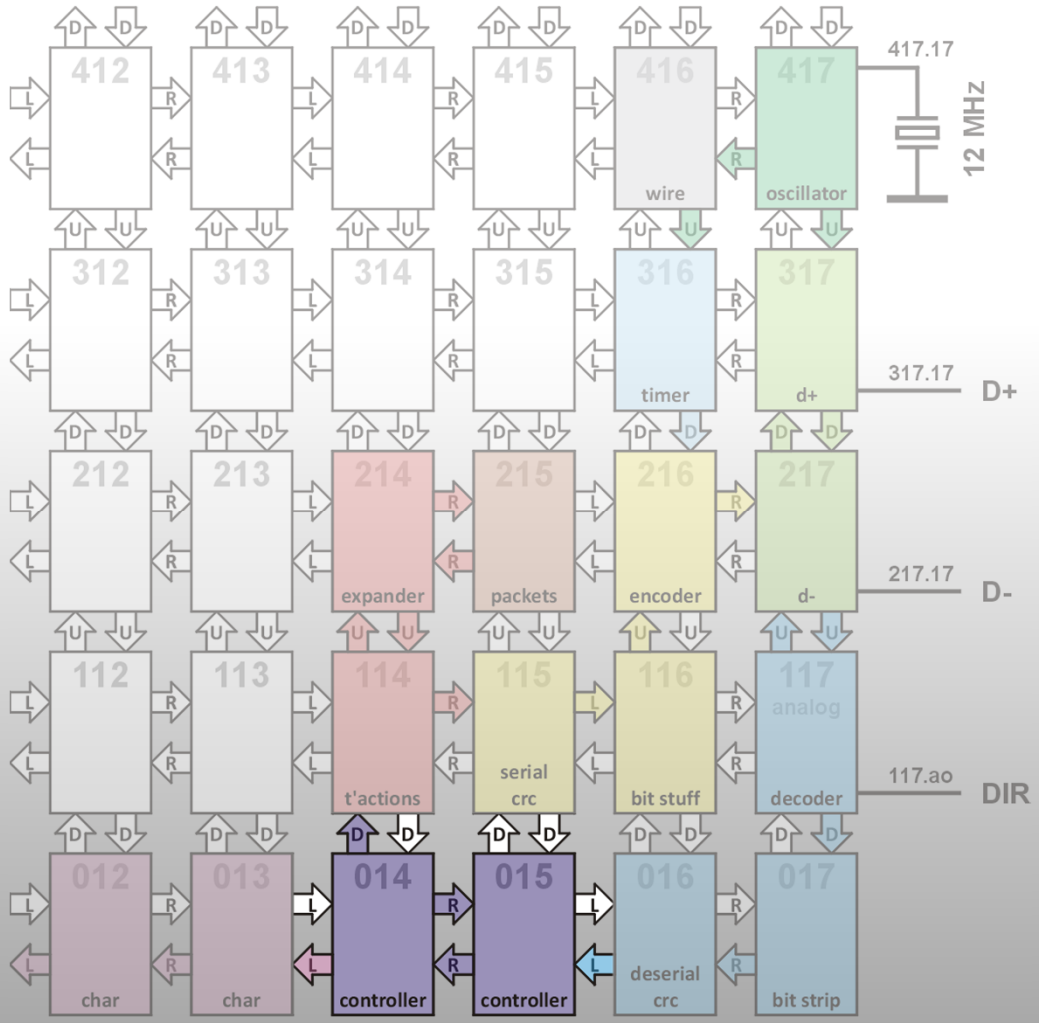
floorplan

keyboard controller – packets, transactions



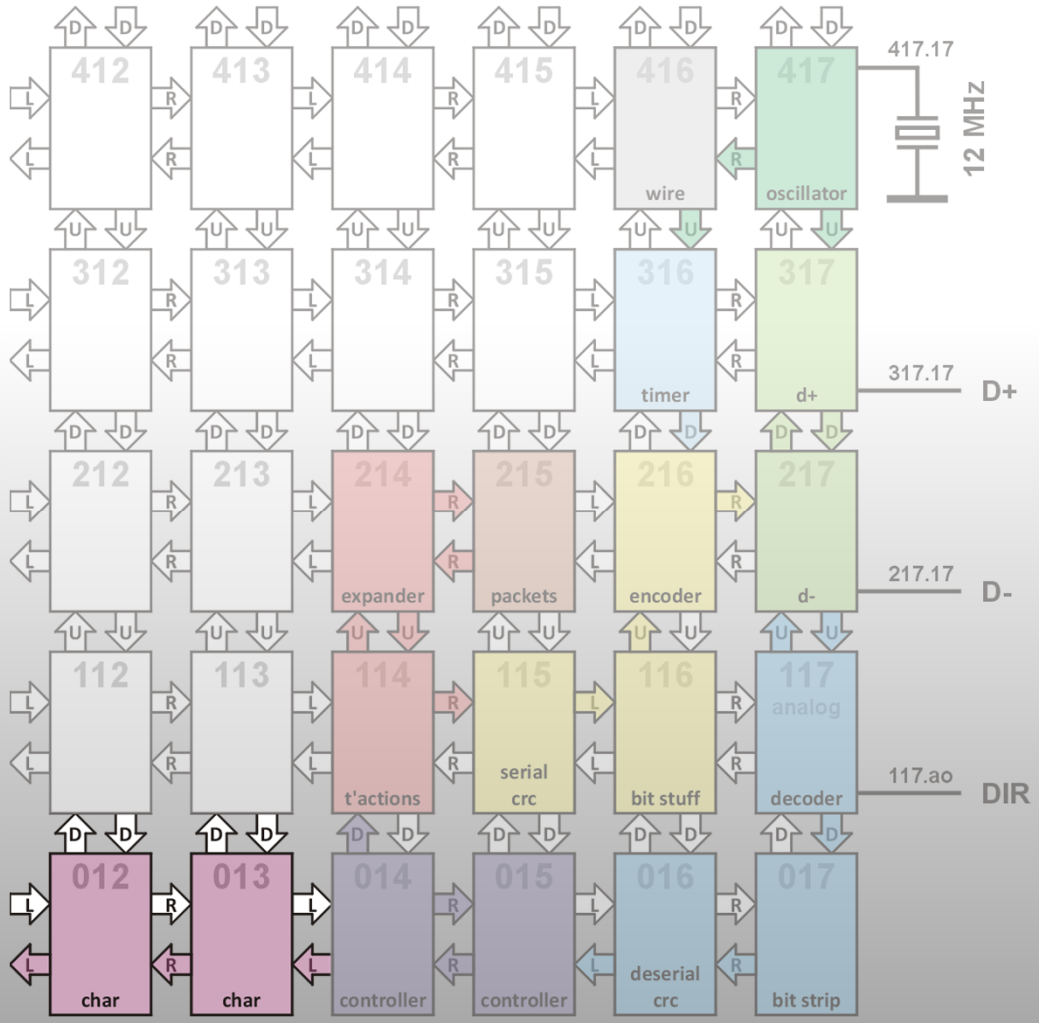
floorplan

keyboard controller - controller



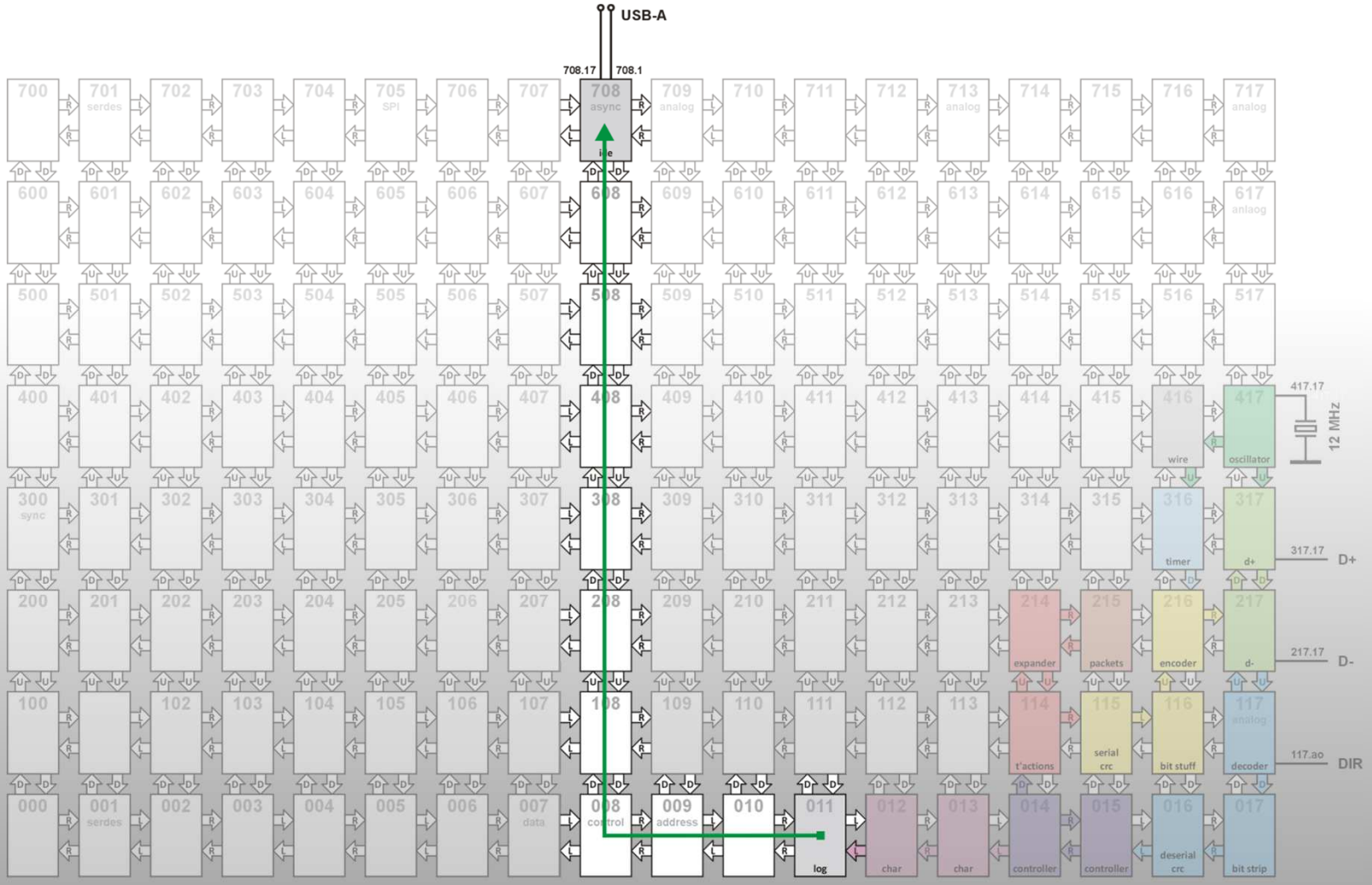
floorplan

keyboard controller – character decoder



floorplan

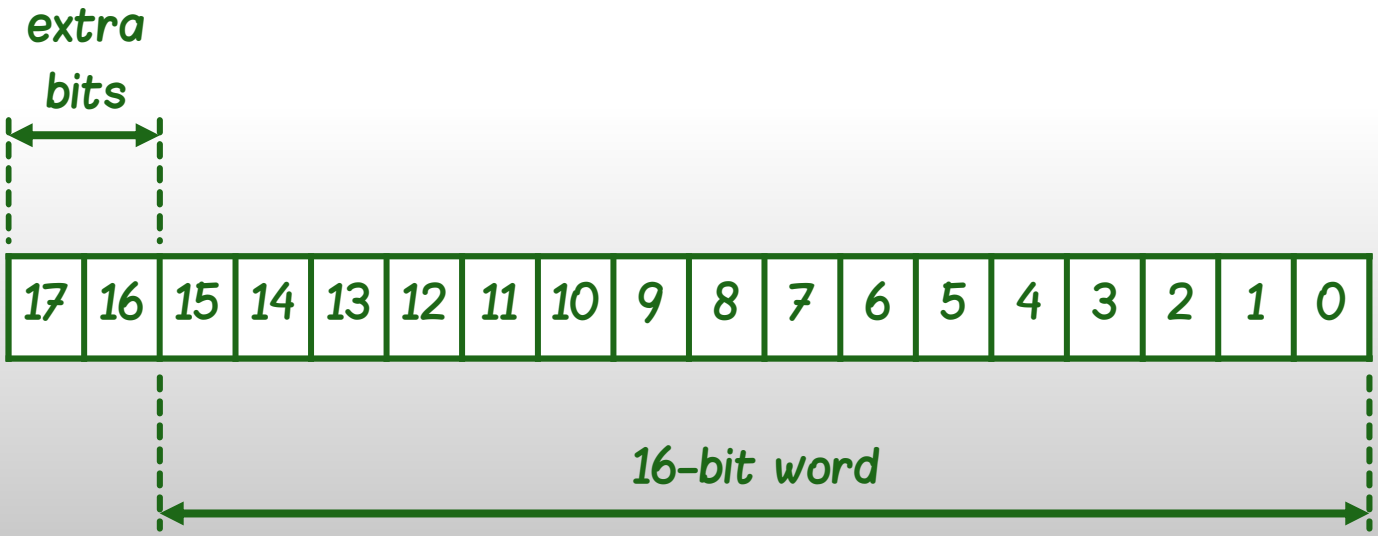
link to arrayForth IDE



data and commands

data as 8-, 16-, 32- or 64-bits chunks

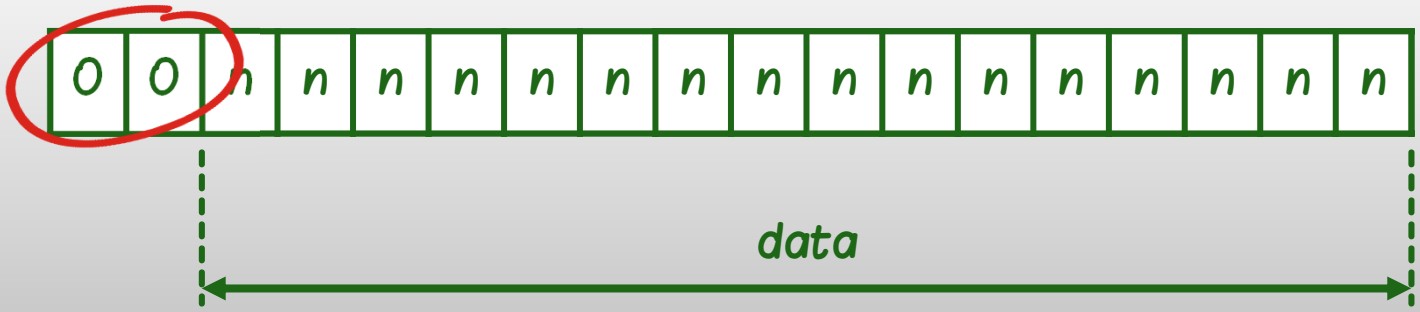
unconventional 18-bit word in GA144



data and commands

data as 8-, 16-, 32- or 64-bits chunks

unconventional 18-bit word in GA144

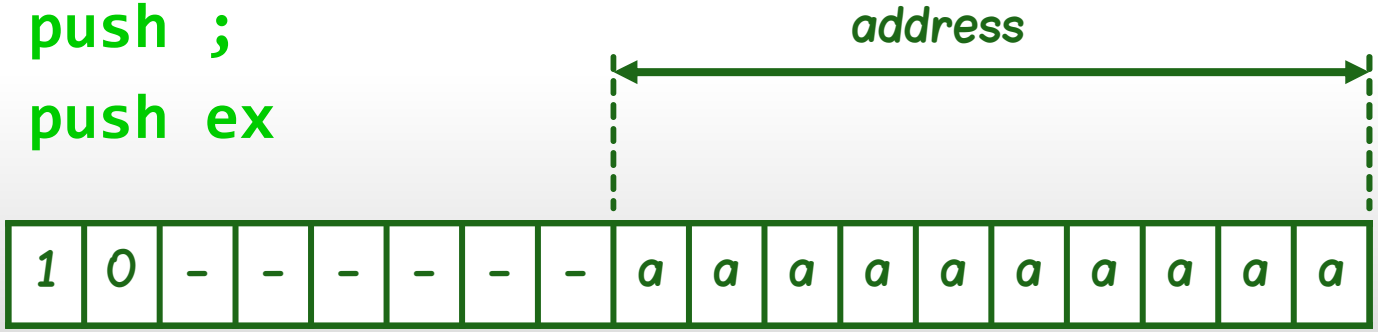


data and commands

data as 8-, 16-, 32- or 64-bits chunks

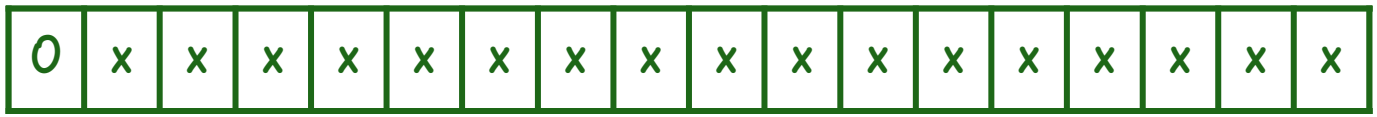
unconventional 18-bit word in GA144

```
push ;
push ex
```

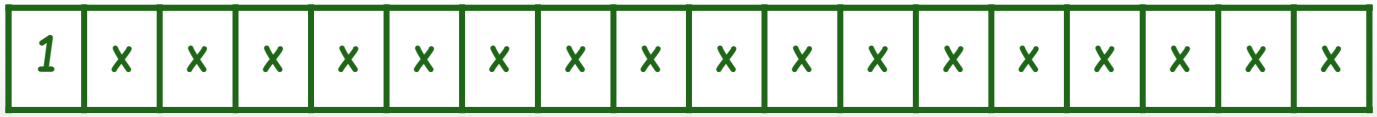


data and commands

data



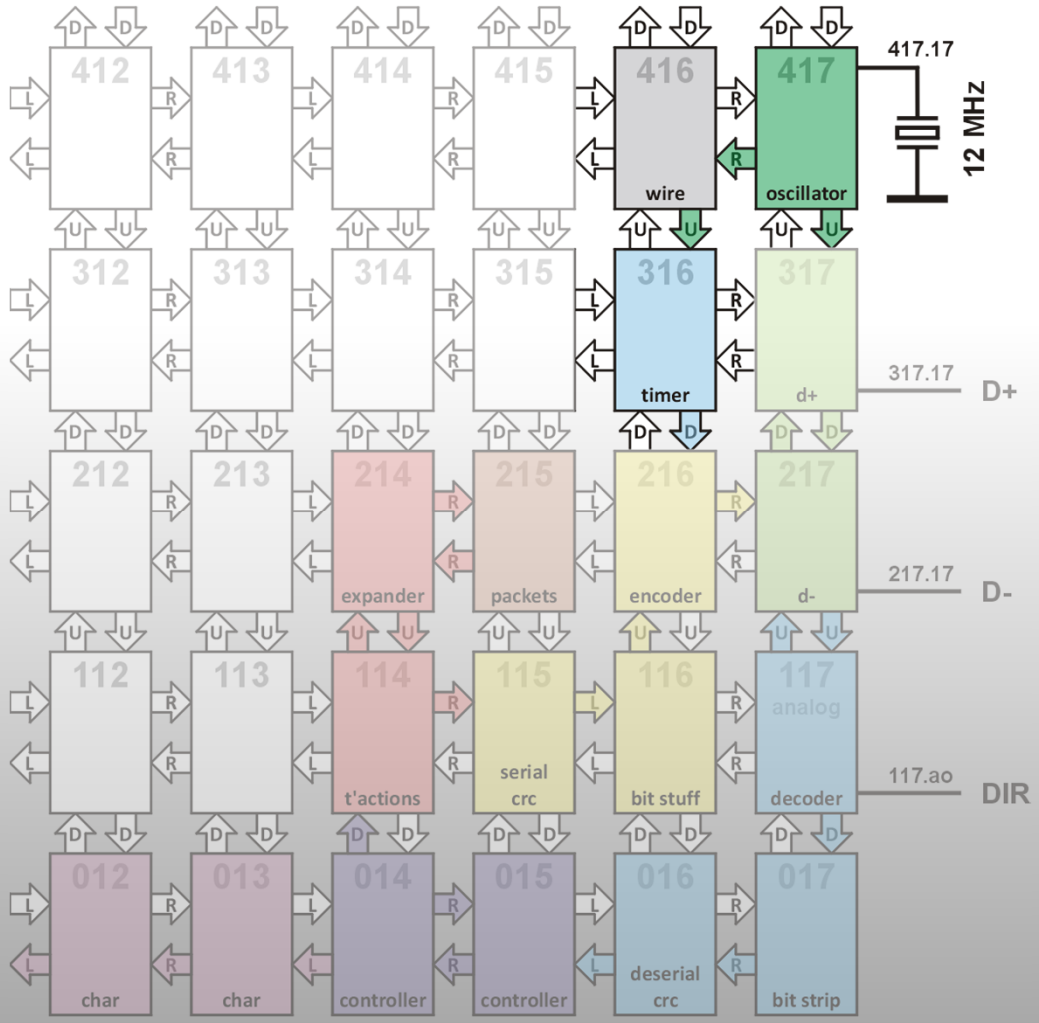
command or message



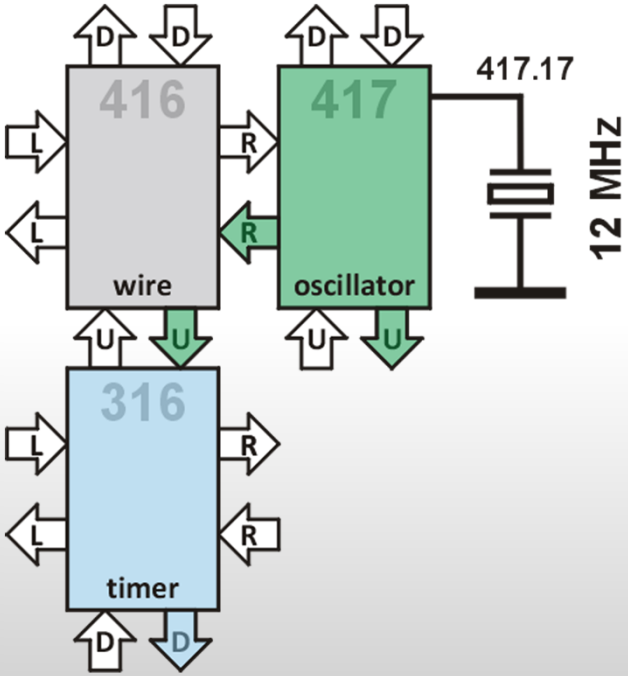
IMPLEMENTATION

part I
serial interface engine

clock


412
413
414
415
416
wire
417
oscillator
417.17
12 MHz
312
313
314
315
316
timer
317
d+
317.17
D+
212
213
214
expander
215
packets
216
encoder
217
d-
217.17
D-
112
113
114
t'actions
115
serial
crc
116
bit stuff
117
analog
decoder
117.ao
DIR
012
char
013
char
014
controller
015
controller
016
deserial
crc
017
bit strip

clock



12 MHz ceramic resonator

double frequency signal to 317

T = 41.7 ns

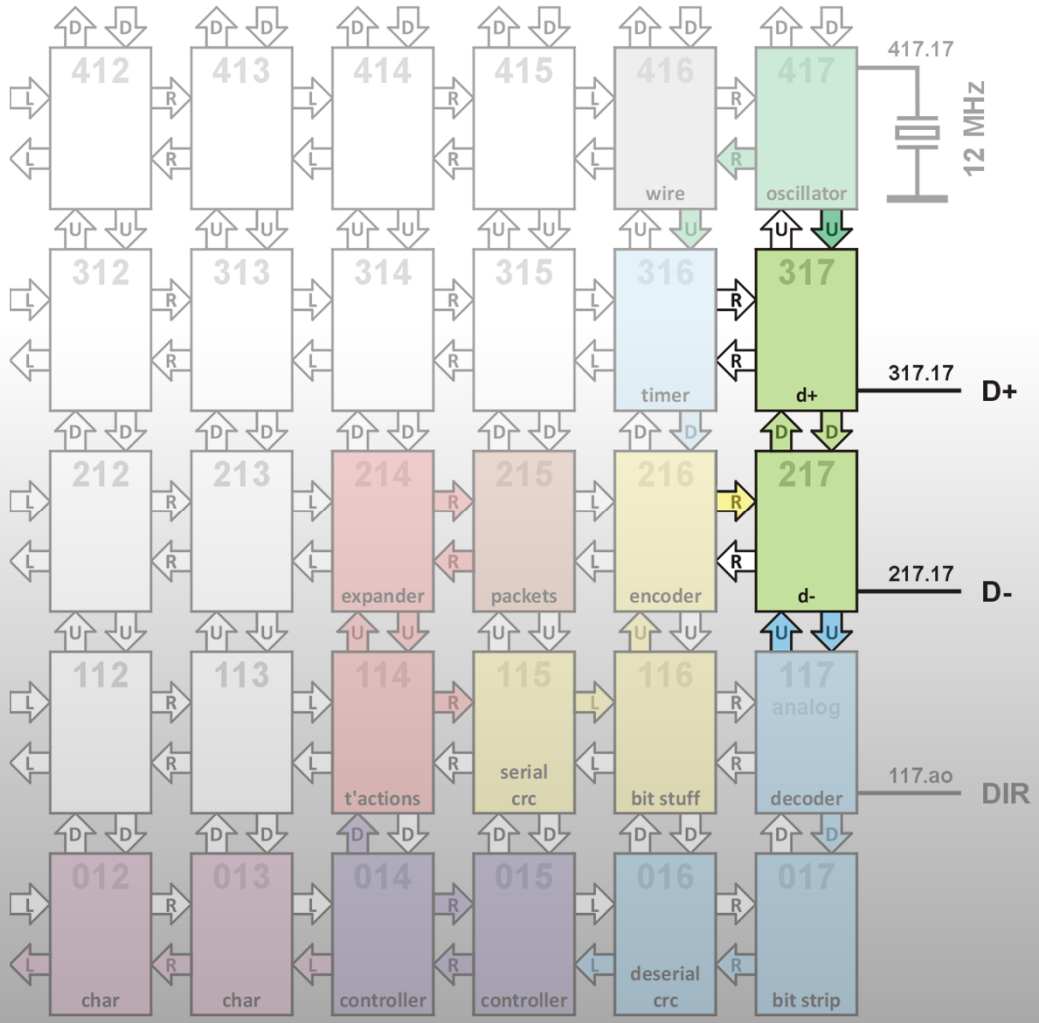
low speed rate 1.5 Mb/s

bit time = 16T

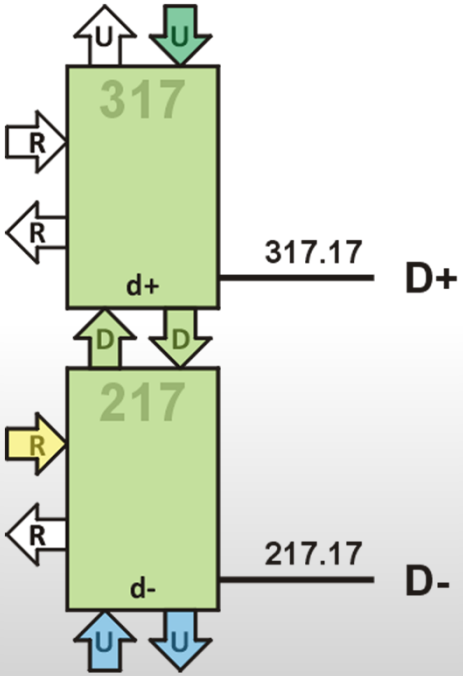
1 ms timer signal to 216

adapted from GreenArrays App Note 012

transceivers



transceivers



GPIO pins to D+ and D-

24 MHz clock via up port of 317

317 controlled via down port when transmitting 217 controlled via right port

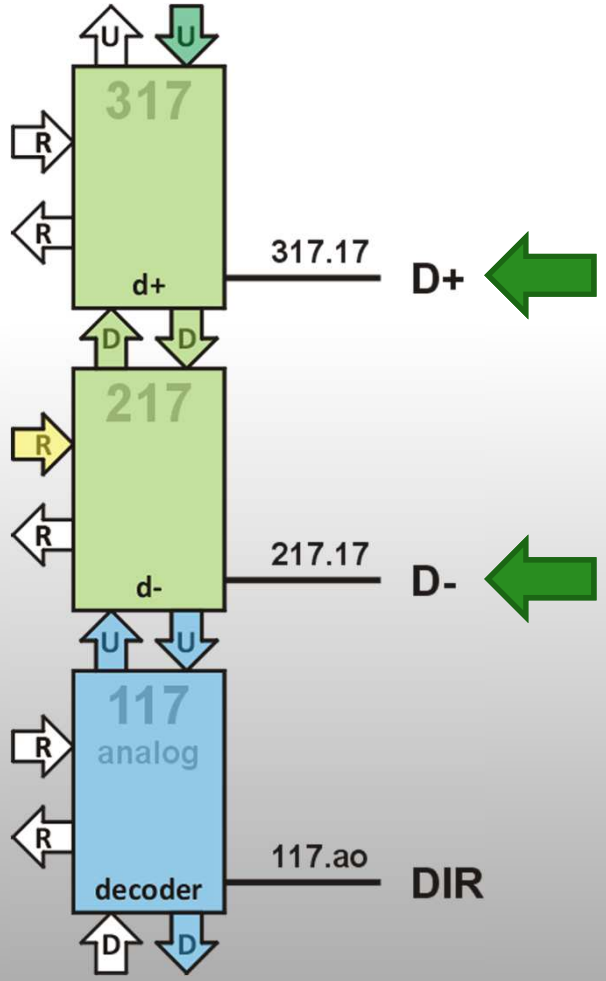
when receiving 217 controlled via up port

GPIO pins are initialized in weak pulldown mode

GPIO in 217 detects device attachment

transceivers

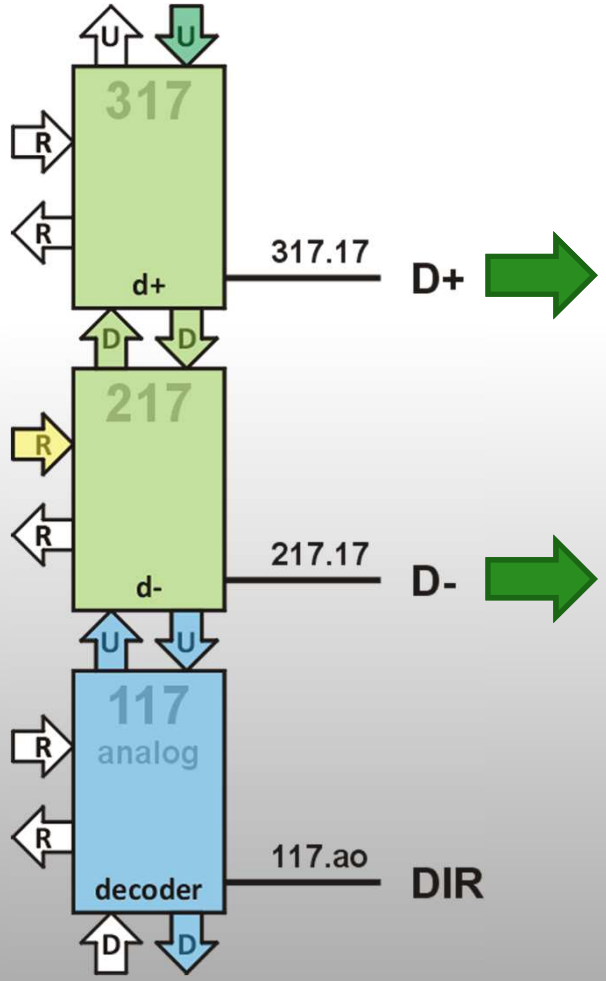
bus direction




```
decoder
117 +node 117 /ram up /a io /b up /p

reclaim 117 node 0 org
...
in 03 155 !b ;
out 05 AA !b ;
```

transceivers

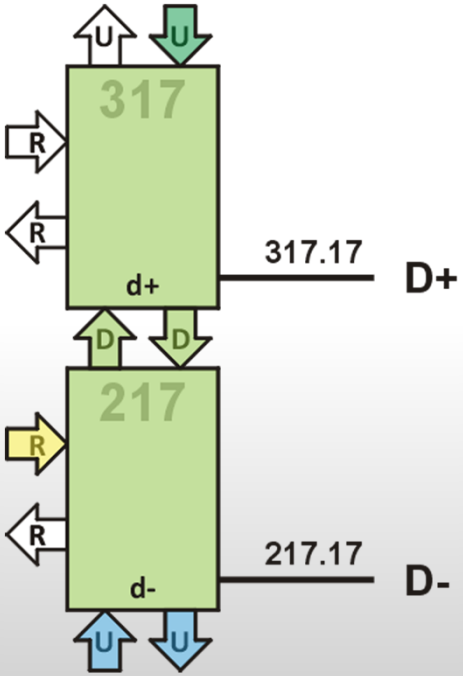
bus direction




```
decoder
117 +node 117 /ram up /a io /b up /p

reclaim 117 node 0 org
...
in 03 155 !b ;
out 05 AA !b ;
```

transceivers

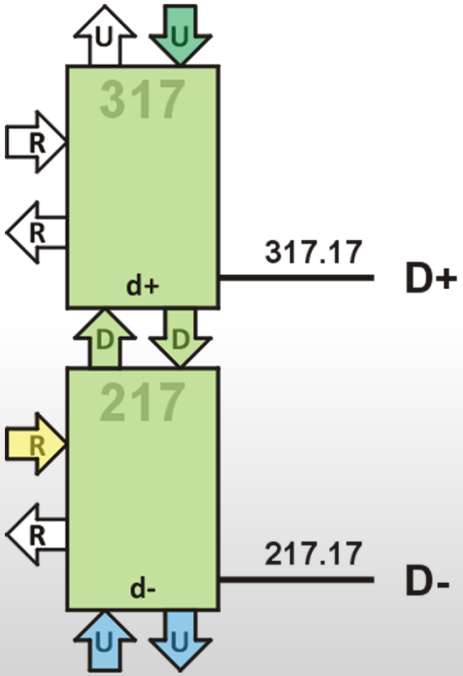
transmit mode

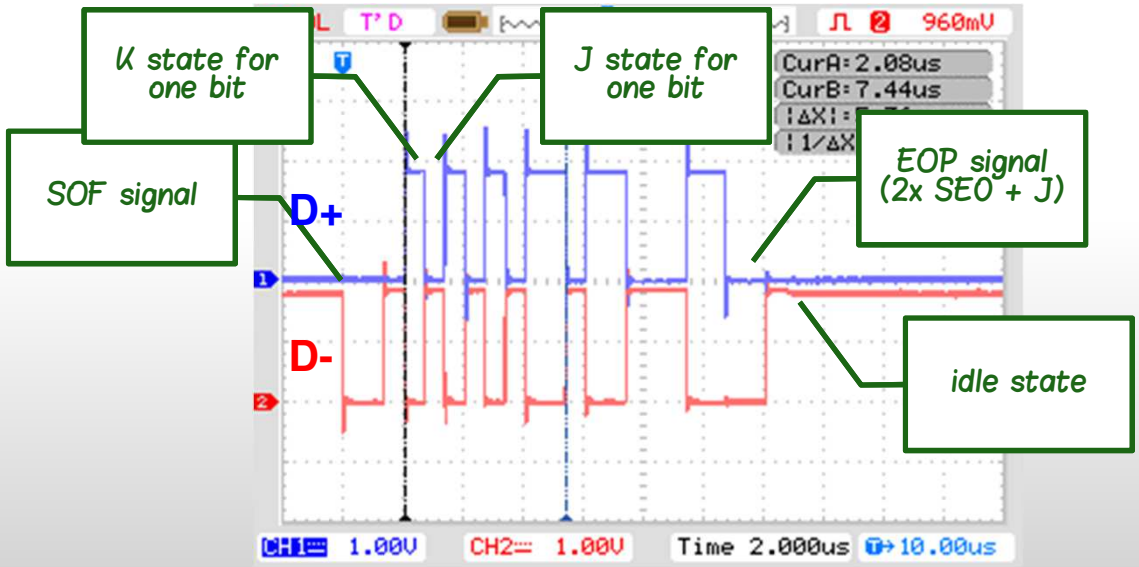



```
d+ 317 +node 317 /ram up /a io /b 2A idl /p

reclaim 317 node 22 org k? 0 org
bit 00 15 for @ unext -d-- ;
+j 04 20000 !b bit ;
+k 06 30000 !b bit ;
...
idl 2A 10000 !b -d-- ;


d- 217 +node 217 /ram io /b 38 go /p

reclaim 217 node 0 org
send 00 a up a! over ! a! ;
j 03 @p ! . . +j 30000 !b 13405 send ;
k 09 @p ! . . +k begin 20000 !b 13405 send ;
se0 0F @p ! . . +j end
idle 12 @p ! . . idl 10000 !b 13403 send ;
```

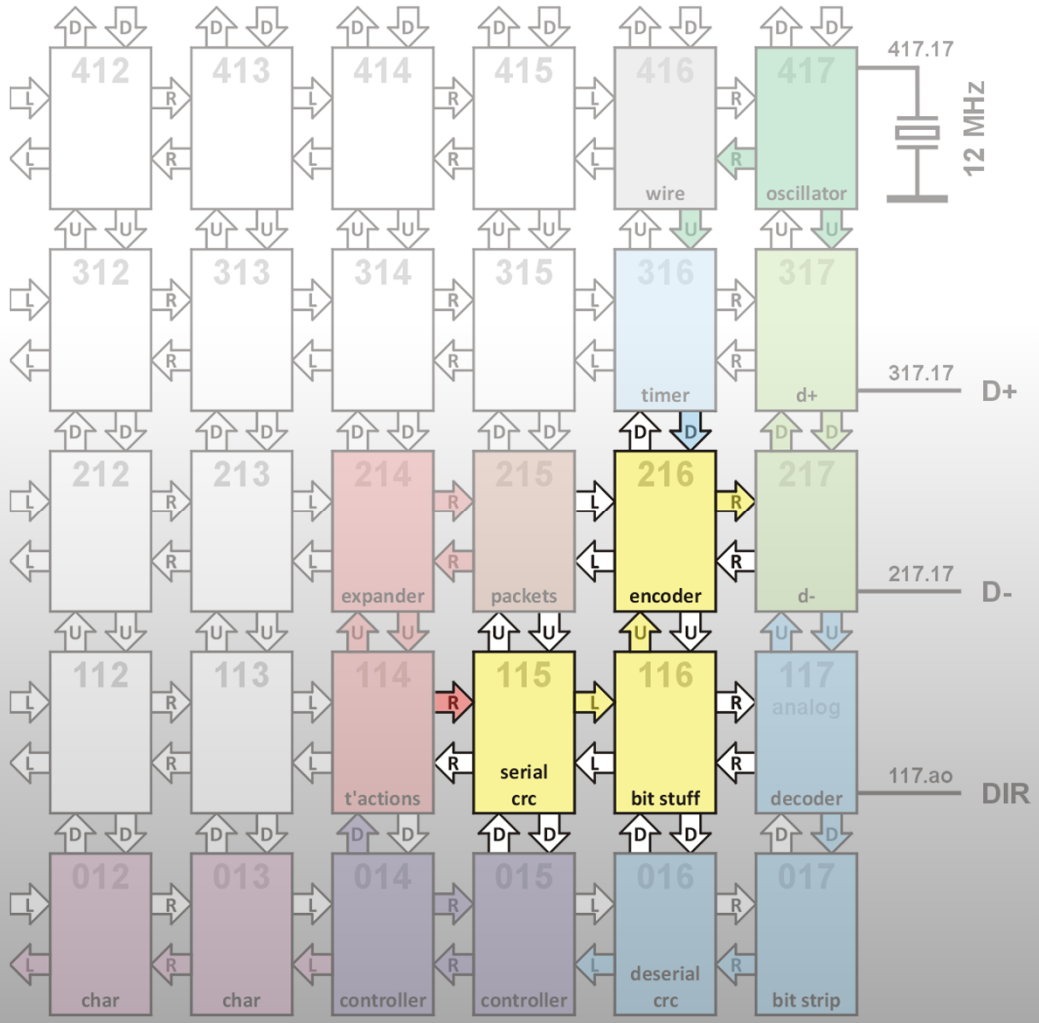
transceivers

transmit mode

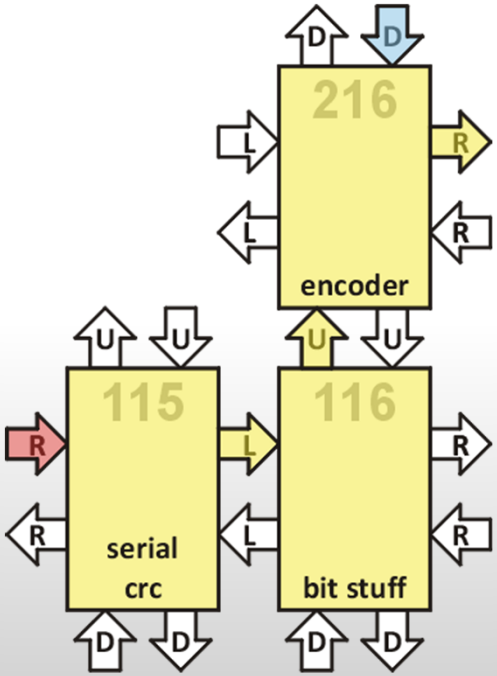




transmit path



transmit path



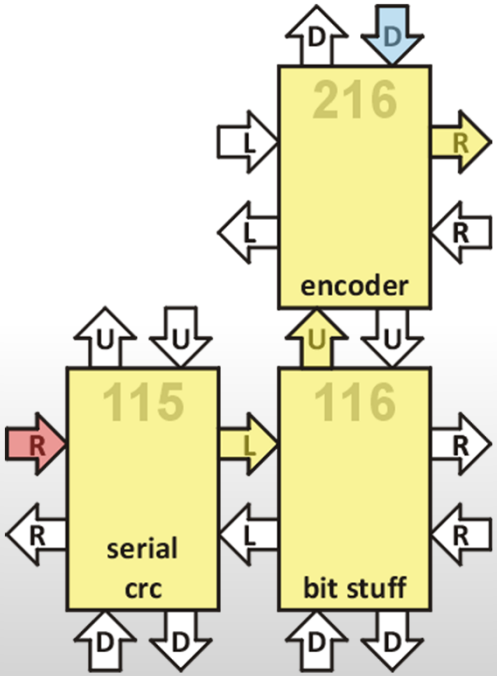
receive stream of bytes

output J and K states

commands intertwined with data

transmit path

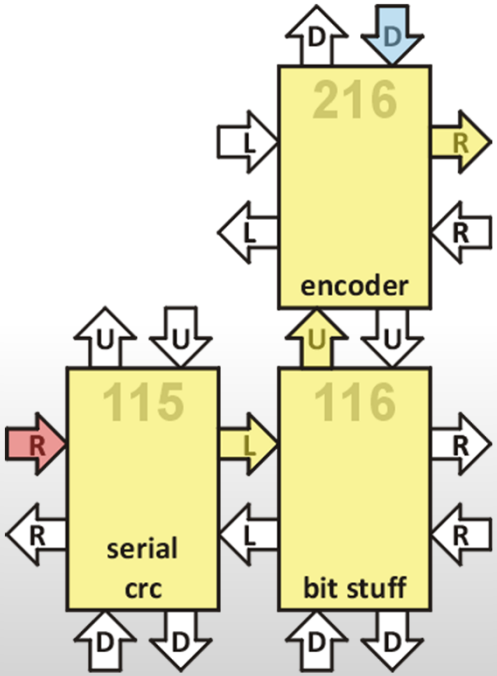
encoder

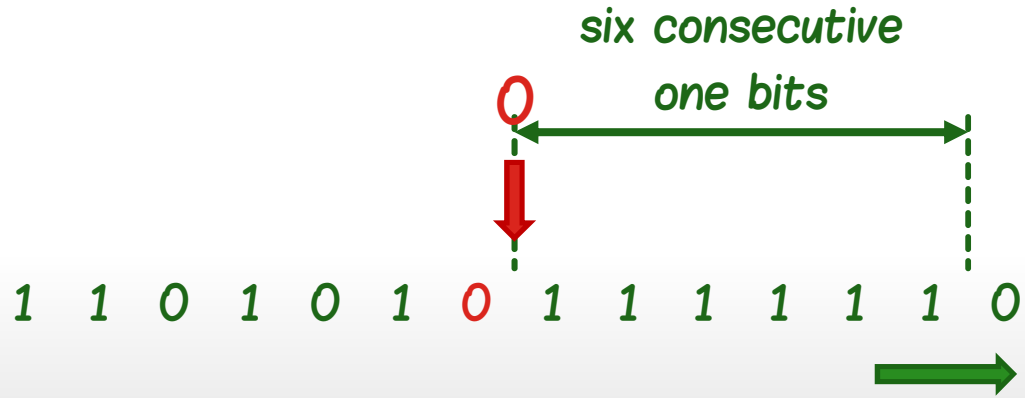



```
encoder
216 +node 216 /ram 195 rd-- /a up /b
16 tmit /p

reclaim 216 node 0 org
stat 00 @p drop @p ; stat! !p ; 0 ,
eop 03 @p ! @p . se0 se0 ! @p ! @p j idle
! dup or stat! ;
enc 0B b stat over - or 1 and
if @p ! . . k else @p ! . . j then stat! ;
alive 15 @ eop ;
tmit 16 @b -if push ex tmit ;
then enc tmit ;
sof 1B @ eop @p ! @p rest atch? ! ;
rst 20 1000 for @ unext ..
@p ! . . se0 25 for @ unext
25 for alive next ;
rx 2D @p ! ; hand 33
```

transmit path

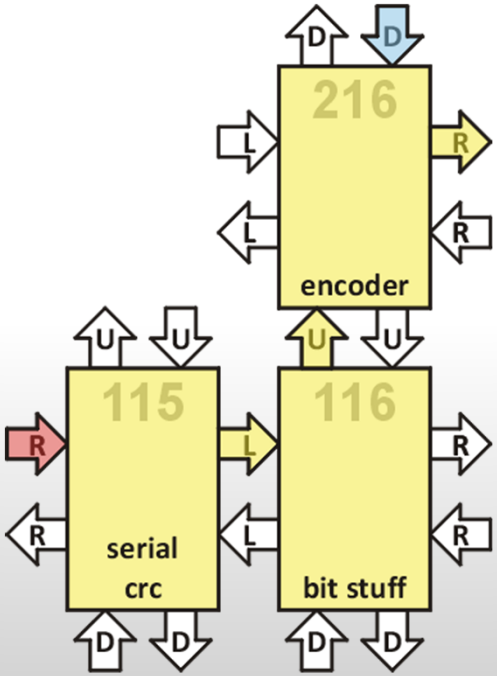
bit stuffing





transmit path

serializer + crc



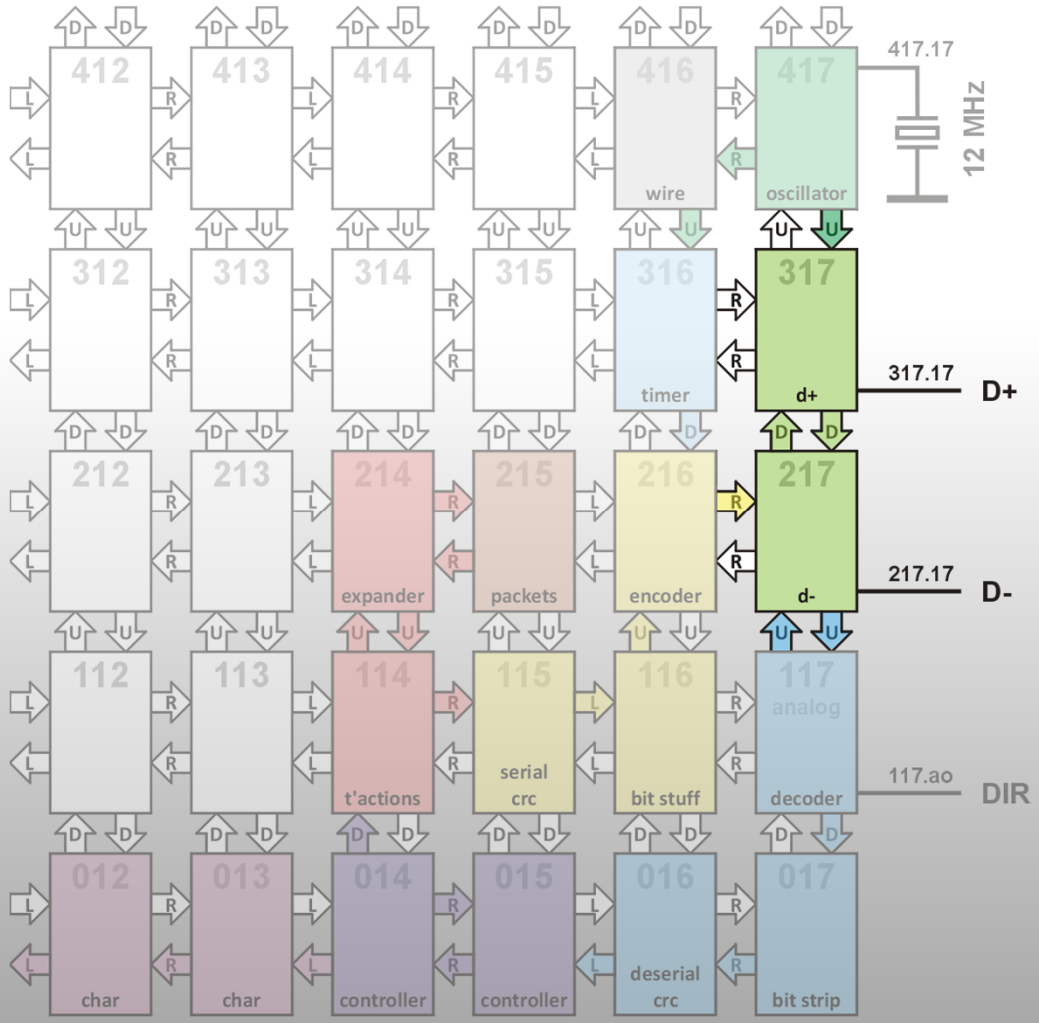
serialize bytes lsb first

calculate crc for all bytes

pass all commands except for

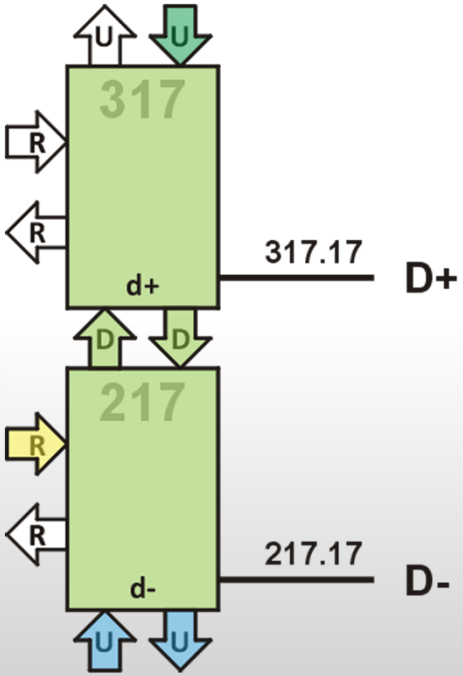
- clear crc register: `clr`
- send 16 crc bits out: `crc!`

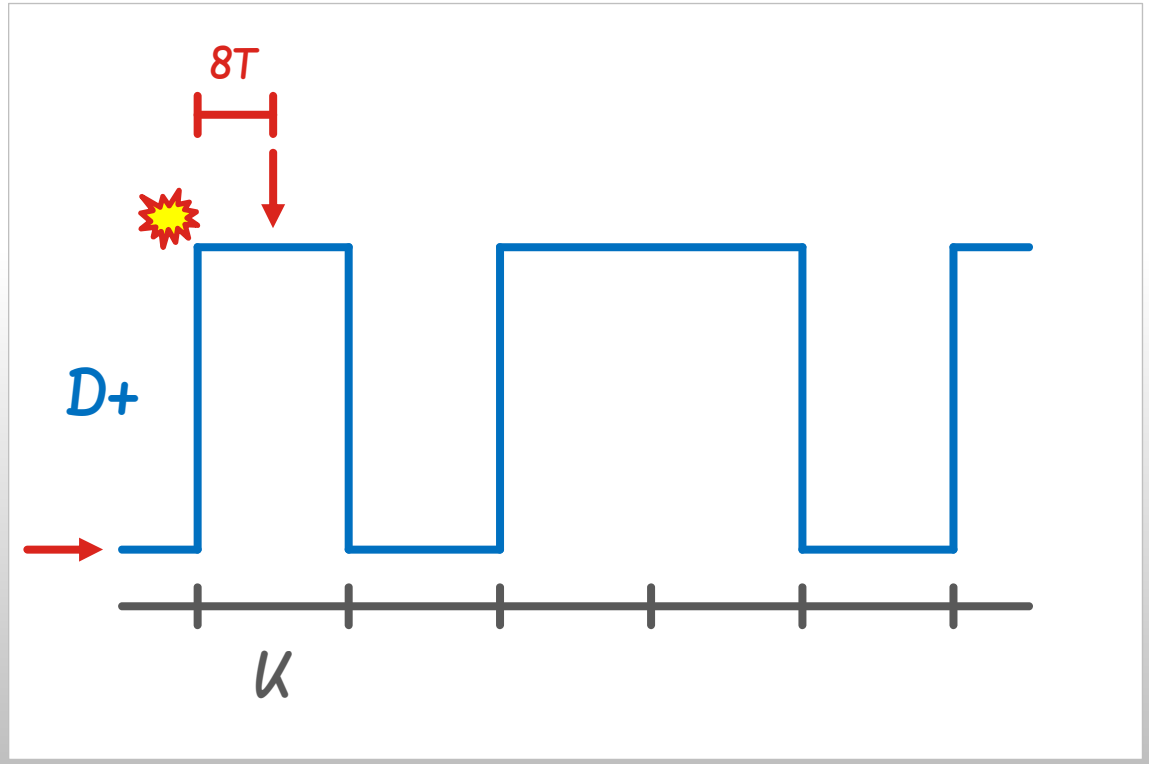
transceivers



transceivers

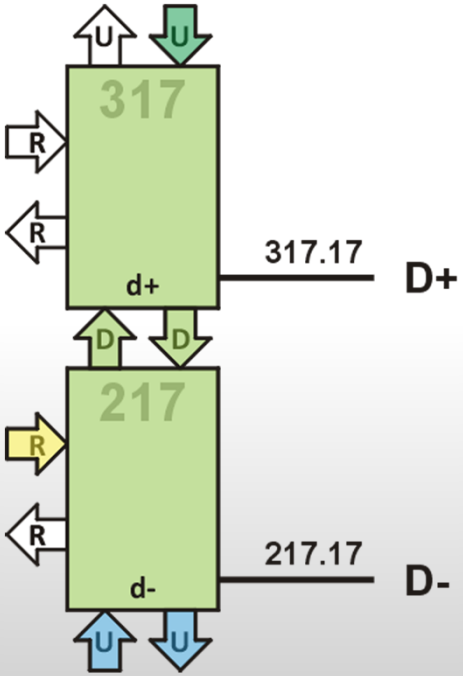
receive mode - self-synchronizing clock

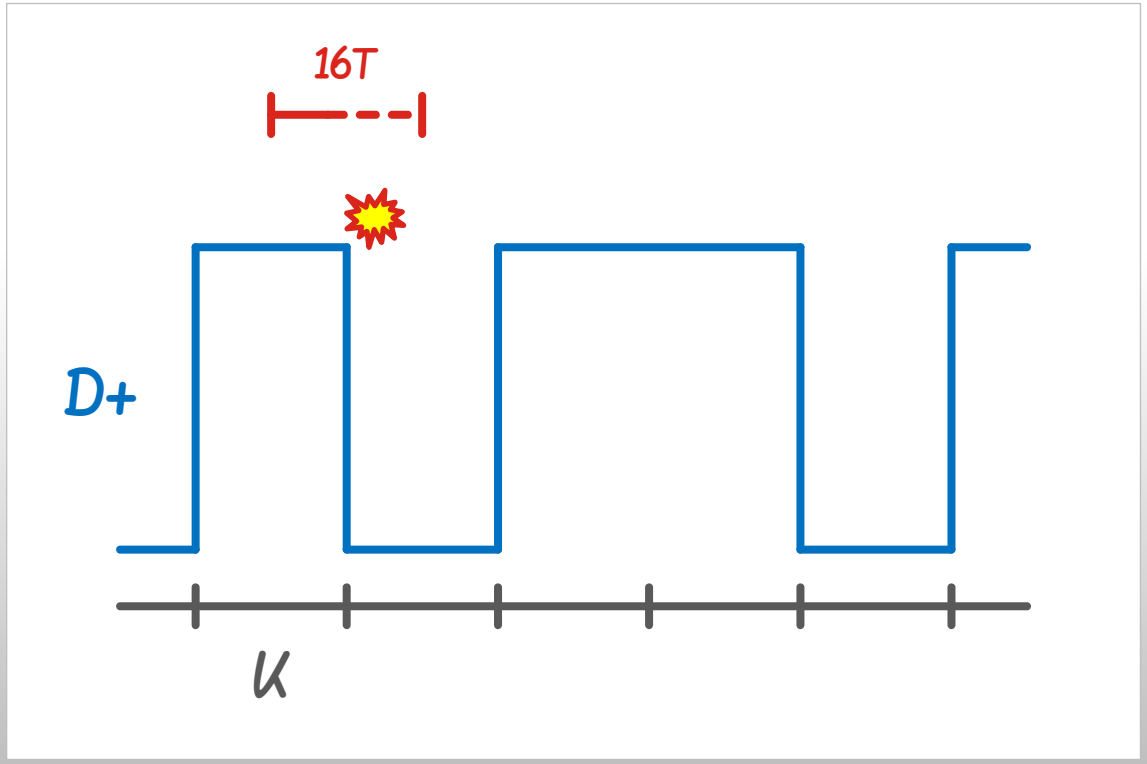




transceivers

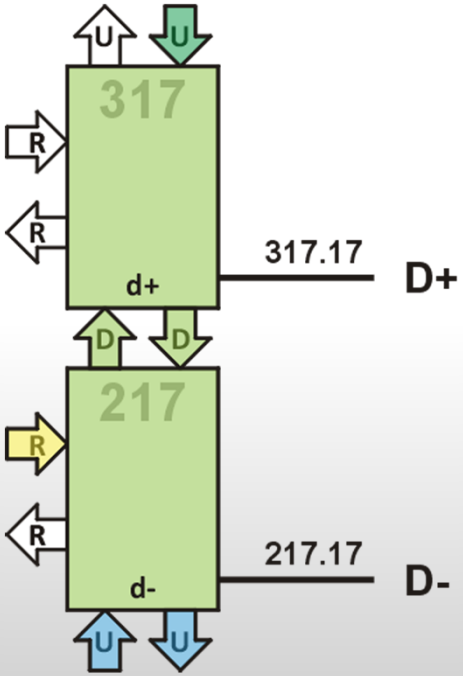
receive mode - self-synchronizing clock

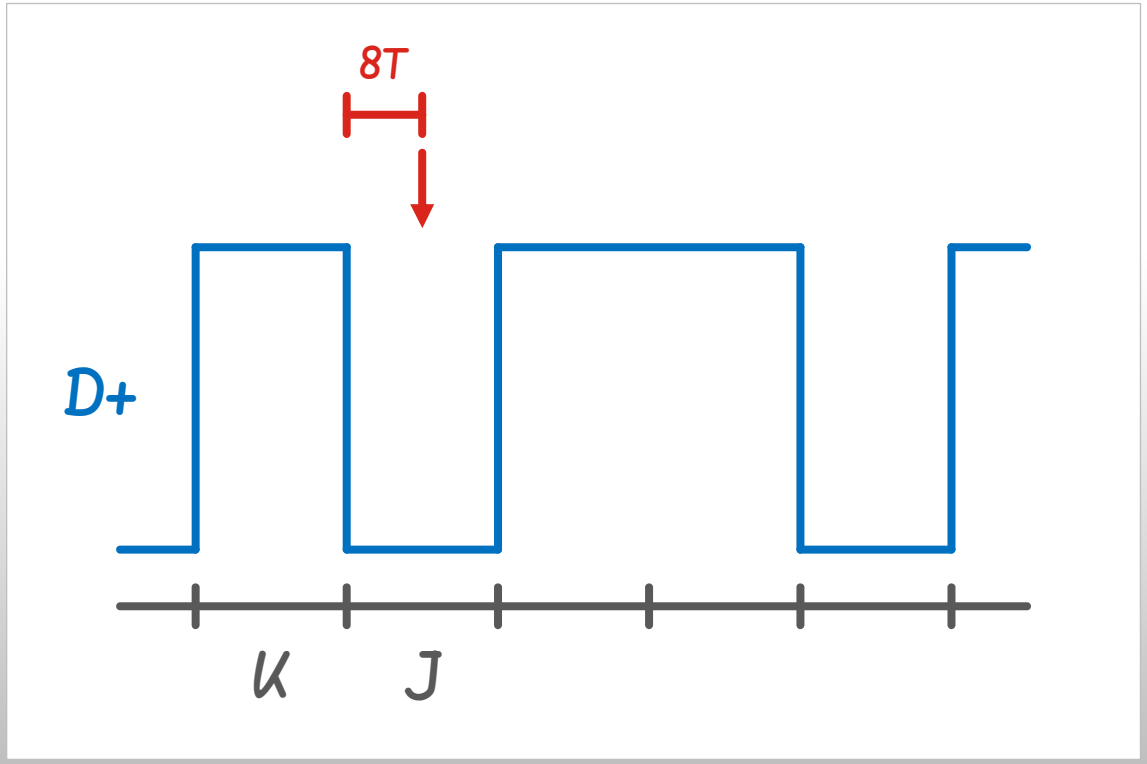




transceivers

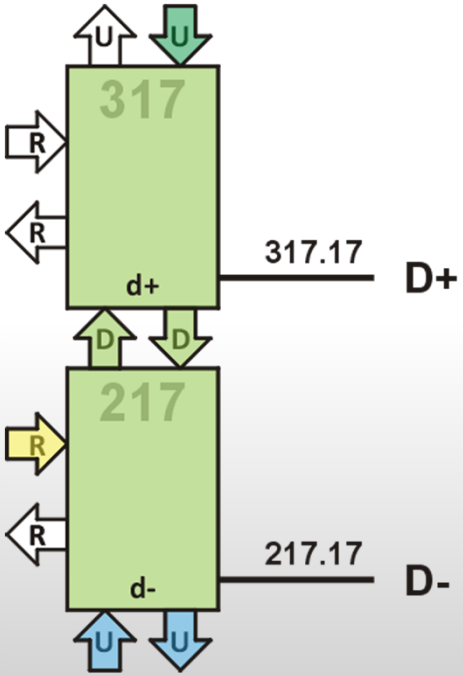
receive mode - self-synchronizing clock

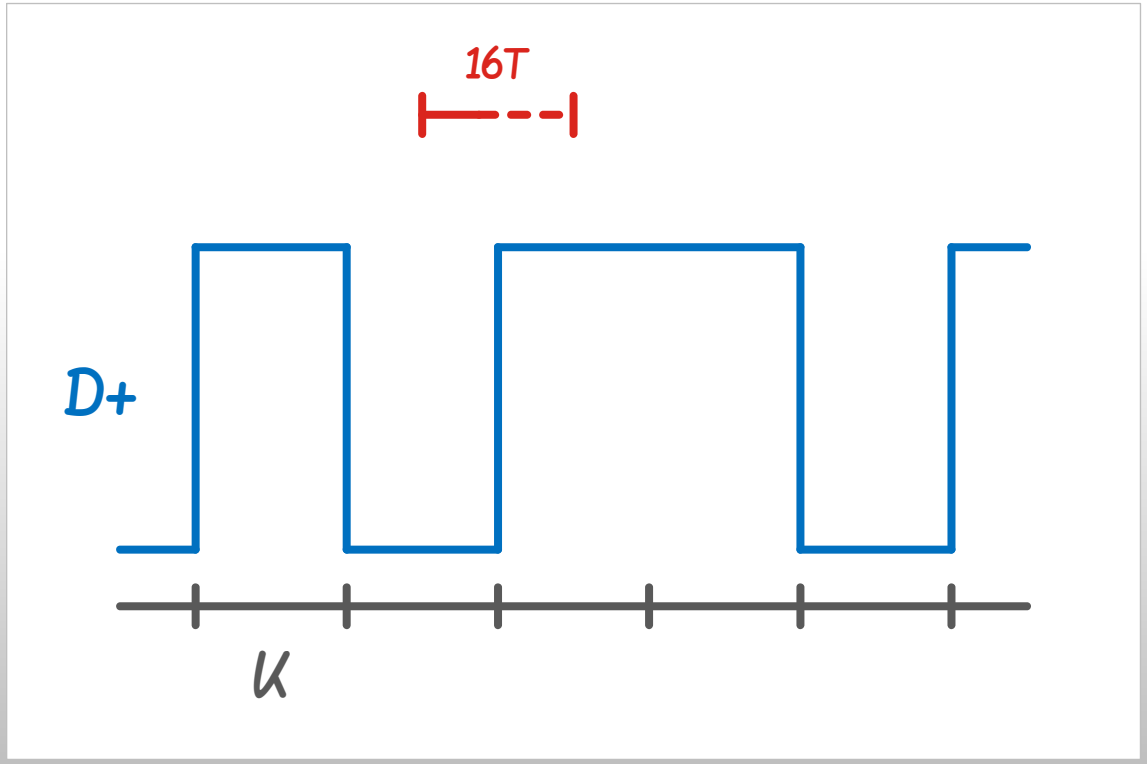




transceivers

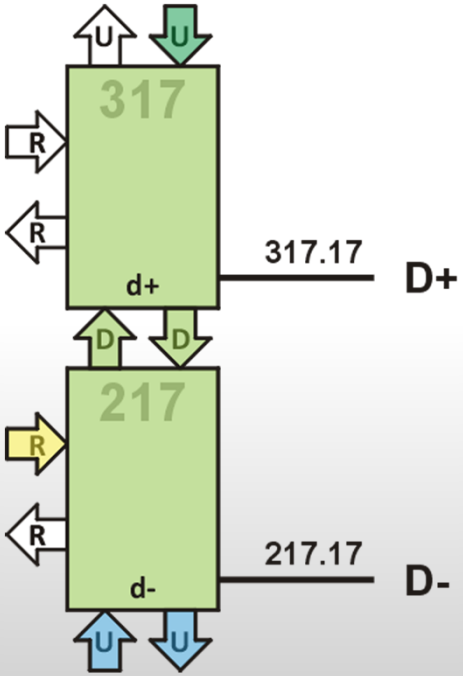
receive mode - self-synchronizing clock

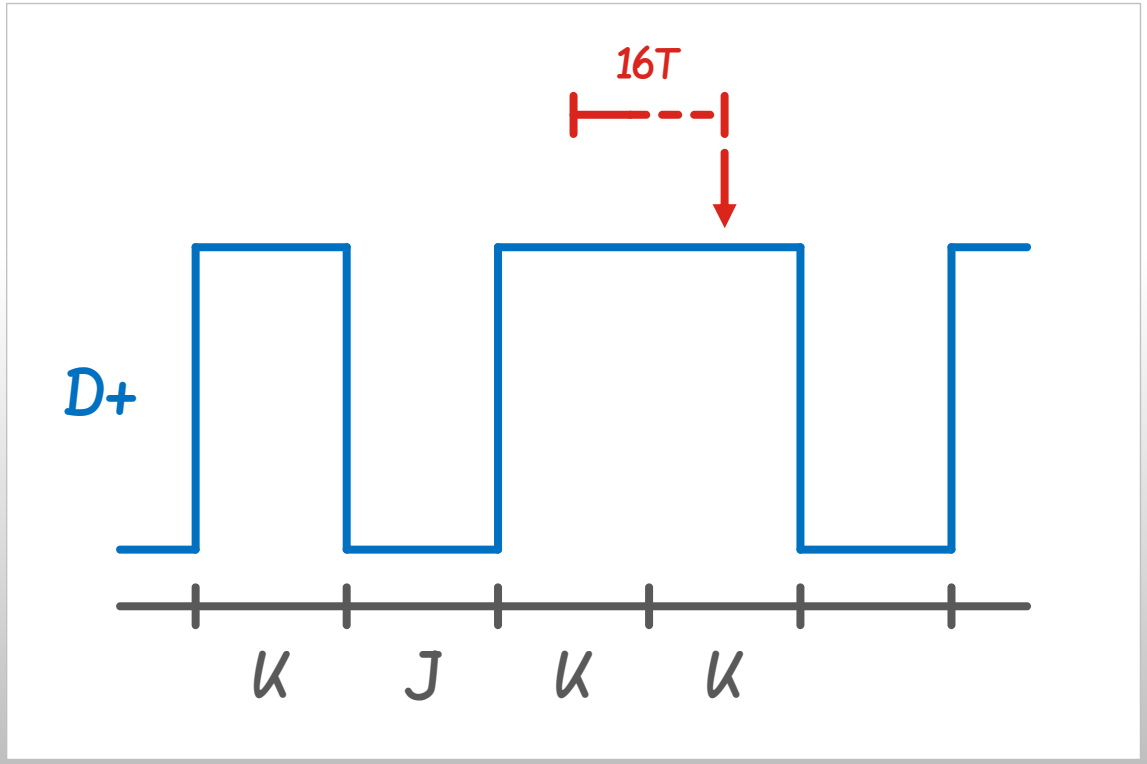




transceivers

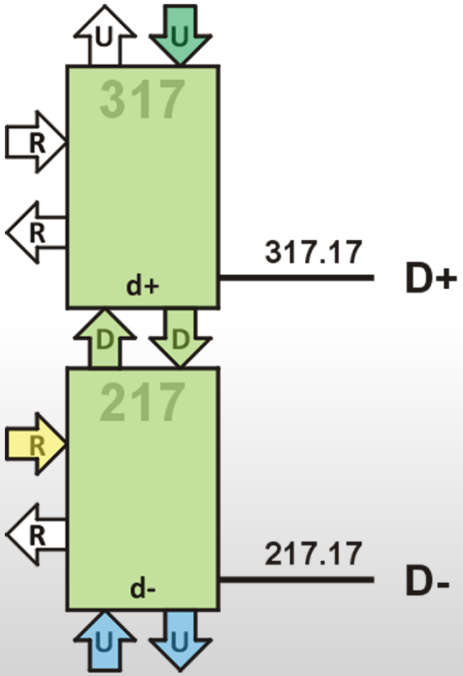
receive mode - self-synchronizing clock

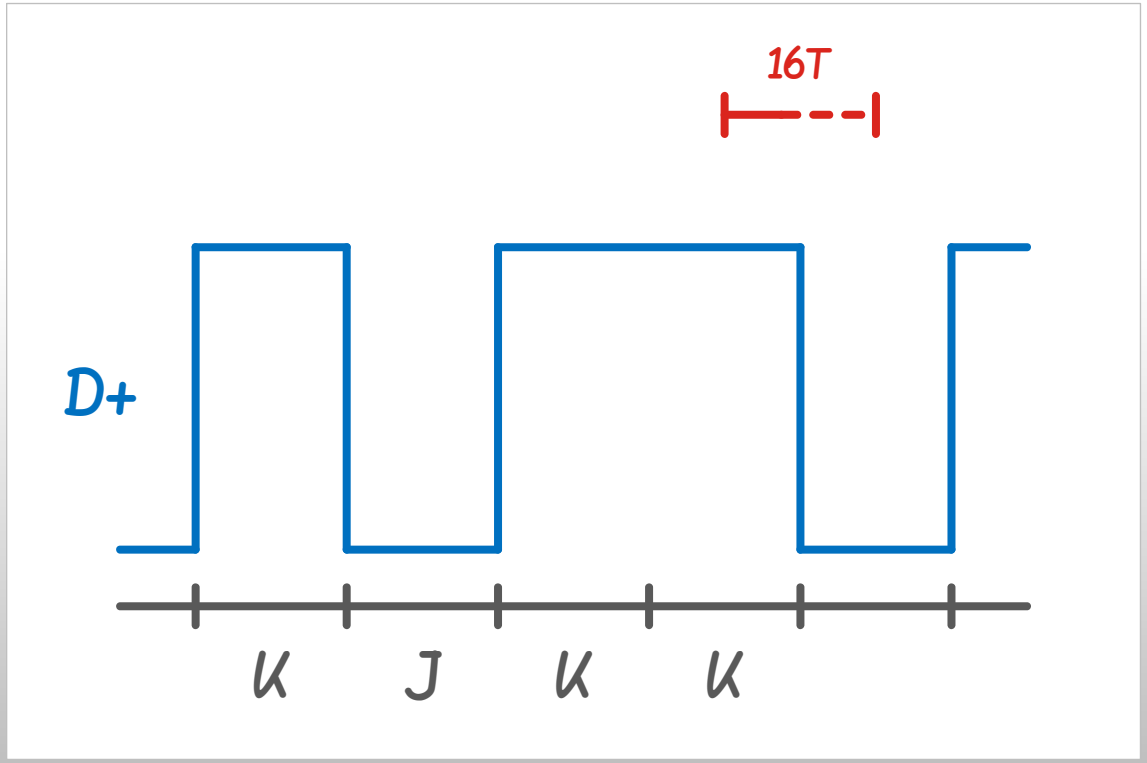




transceivers

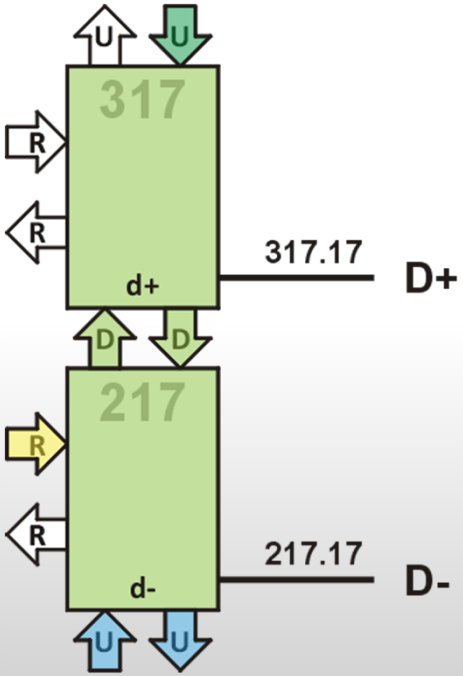
receive mode - self-synchronizing clock





transceivers

receive mode – self-synchronizing clock



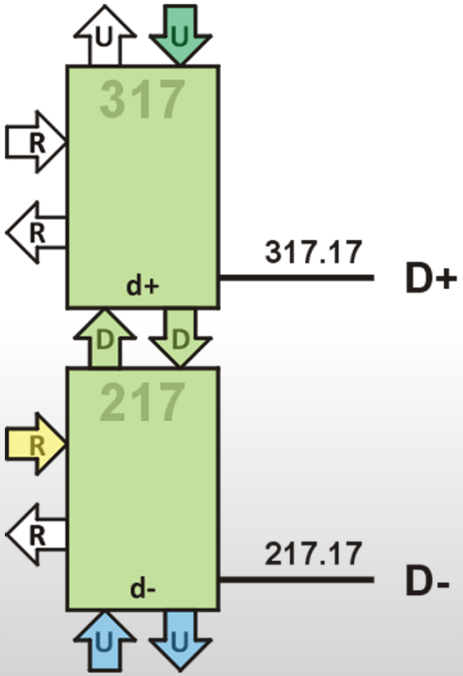
1. detect an edge (start of packet)
2. count 8T, then read D+ state
3. count up to 16T while waiting for an edge
   - if detected stop counting and go to 2.
   - else read D+ state and go to 3.

resynchronized at each edge

bit stuffing – resync at most after 6 bits

transceivers

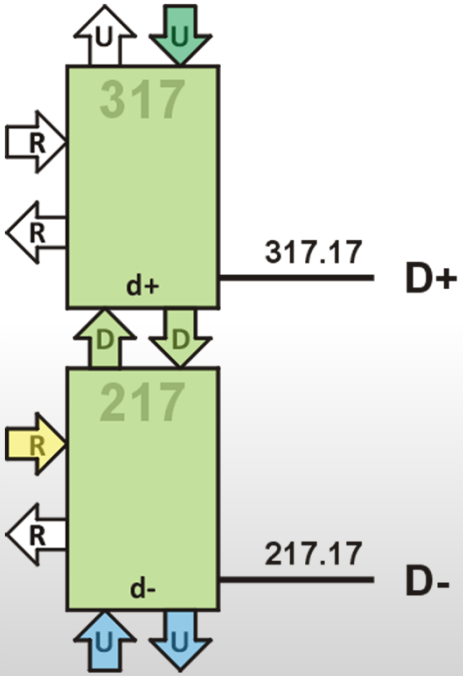
receive mode – node 317




```
d+ 317 +node 317 /ram up /a io /b 2A idl /p

reclaim 317 node 22 org k? 0 org
...
@bit -n 09 @ @ @ @ @ @ @ @b -d-- ;
edge 0D 271 for @ @b -if pop ;
then next dup or -d-- -d-- ;
sync 15 edge @bit
j? 18 15 for @ @b - -if @bit k? ;
then next - -d-- j? ;
k? 22 15 for @ @b -if @bit j? ;
then next -d-- k? ;
```

transceivers

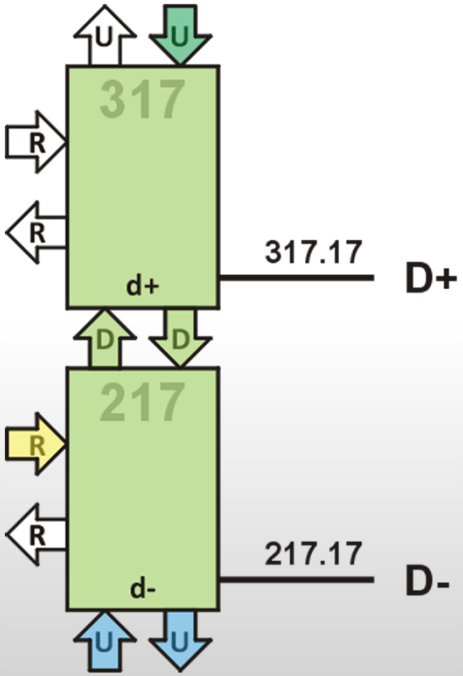
receive mode – node 217

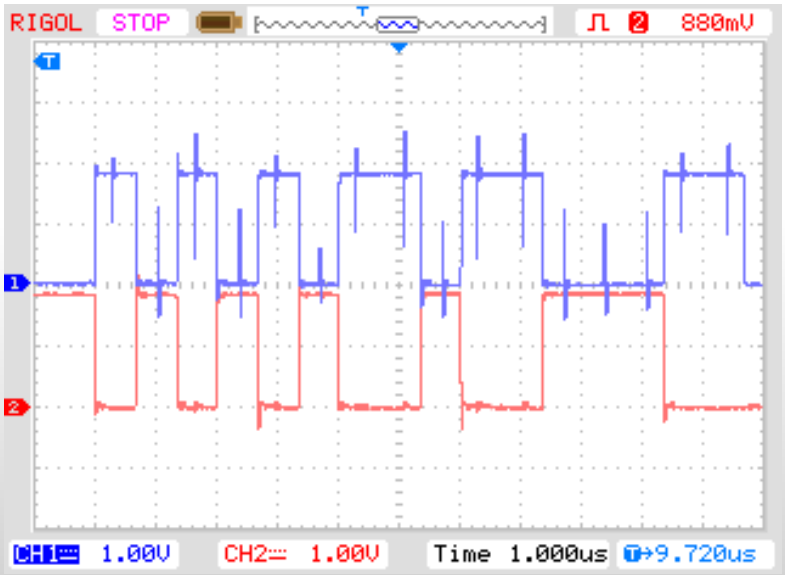



```
d- 217 +node 217 /ram io /b 38 go /p

reclaim 217 node 0 org
send 00 a up a! over ! a! ;
...
hand 23 13427 send ---u ;
read -n 26 @p ! @ ; !p ;
nxt 28 read ahead *
inp 2A @p ! . . sync read
if dup send swap then
* @b send drop send ---u ; then send r--- ;
```

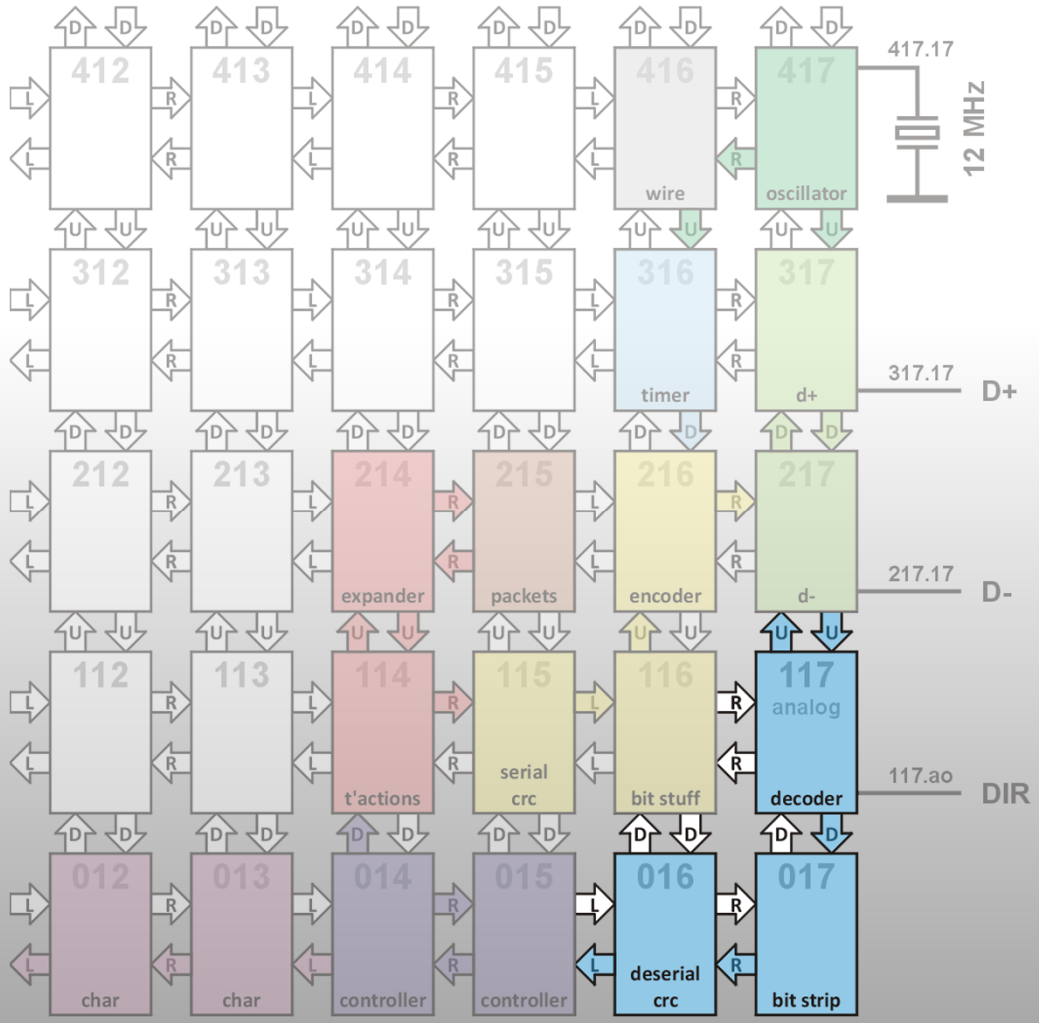
transceivers

receive mode

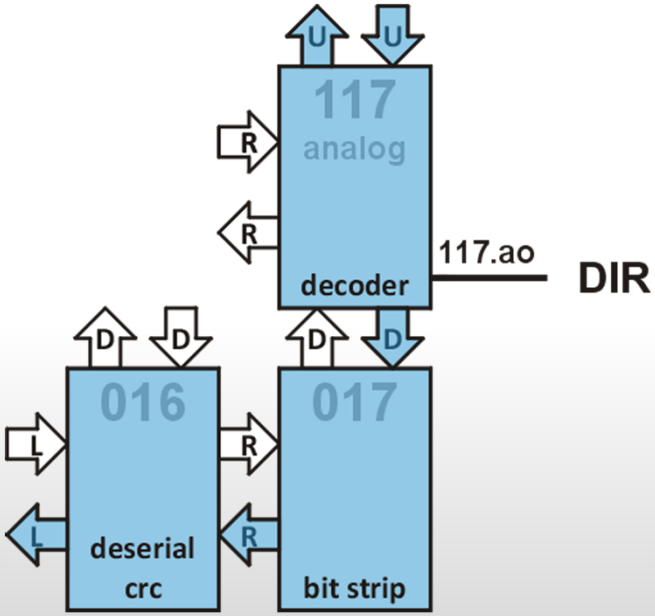




receive path



receive path



start reception of packet and check for timeout

receive stream of D+ and D- line states

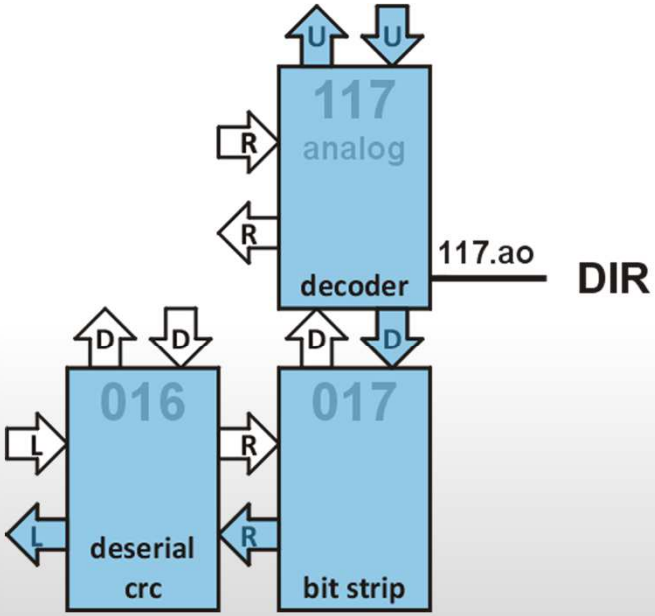
output decoded bytes

inform controller about state of the line

switch direction of voltage level shifter

receive path

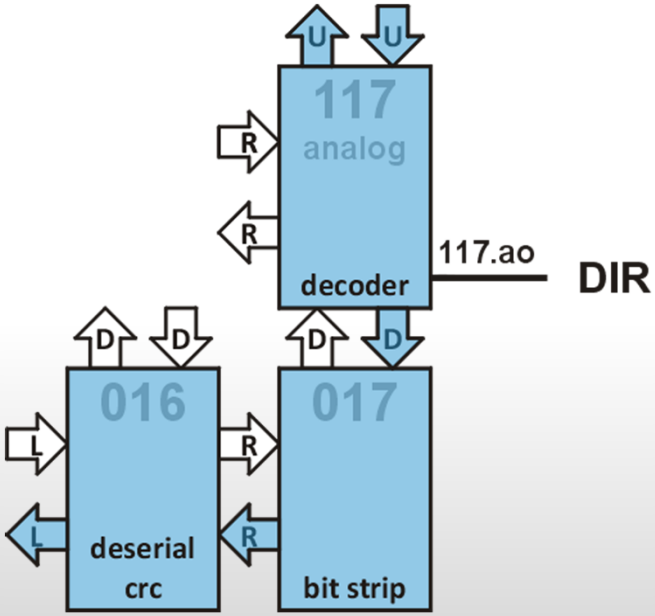
decoder




```
decoder 117 +node 117 /ram up /a io /b up /p

reclaim 117 node 0 org
d- 00 @p drop @p ; d-! !p ; 0 ,
...
tx 10 io b! @p . rest ! @p ! . r--- ; ---u ;
bit 16 sf-sb drop d- over or . -if dup or ;
then drop 80 ;
rcv 1C begin @ @ over or
-if bit !b d-! @p ! . . nxt swap end ;
then 20000 !b tx ;
rx 27 down b! 20000 d-! @p ! @ . inp if rcv ;
then - !b io b! ---u ; 33
```

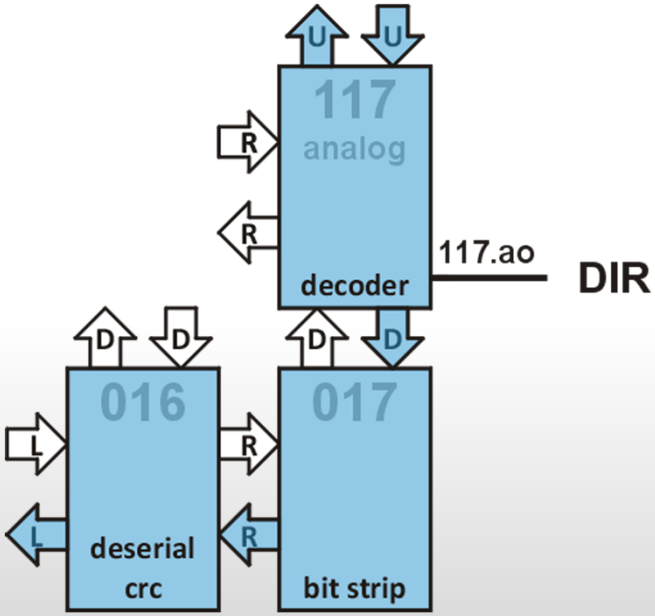
receive path

bit striping



receive path

deserializer + crc



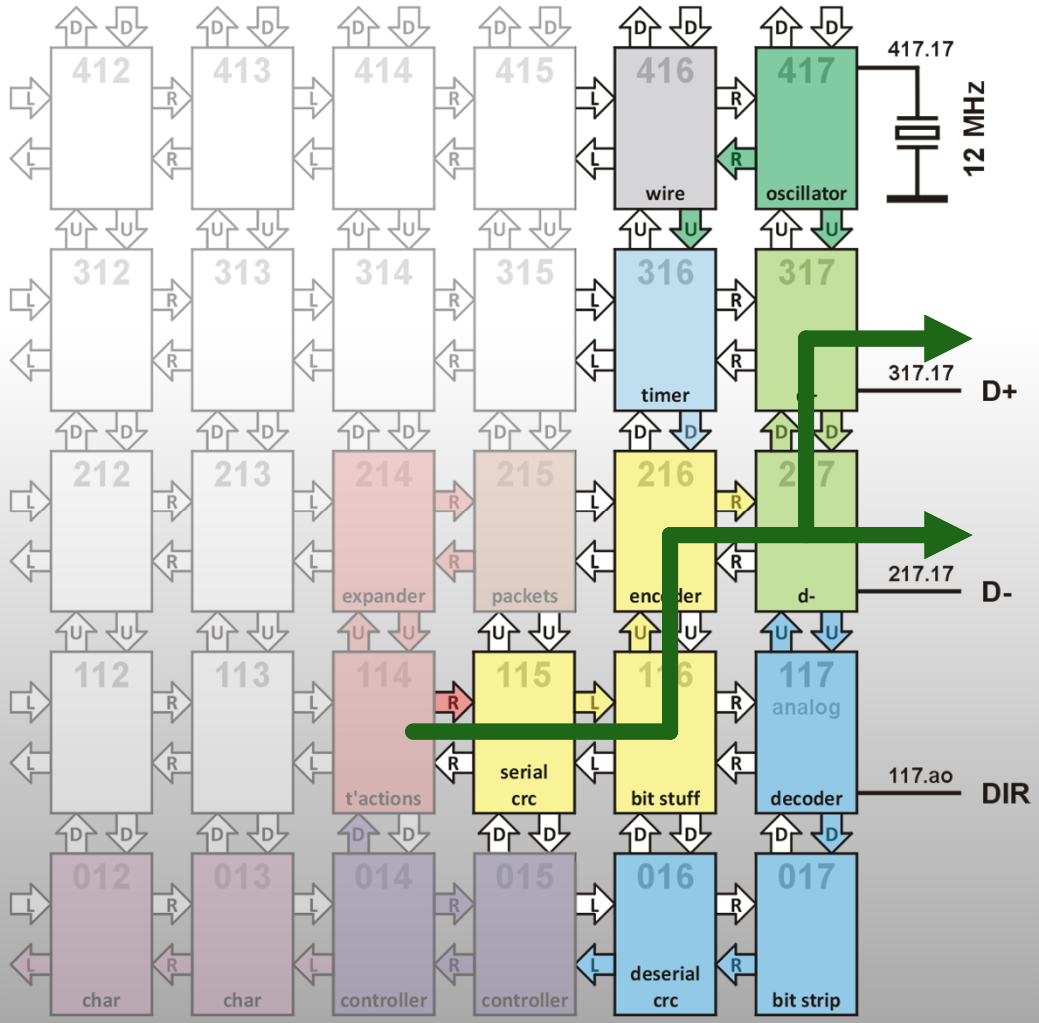
converts serial data into bytes

starts crc for data packets

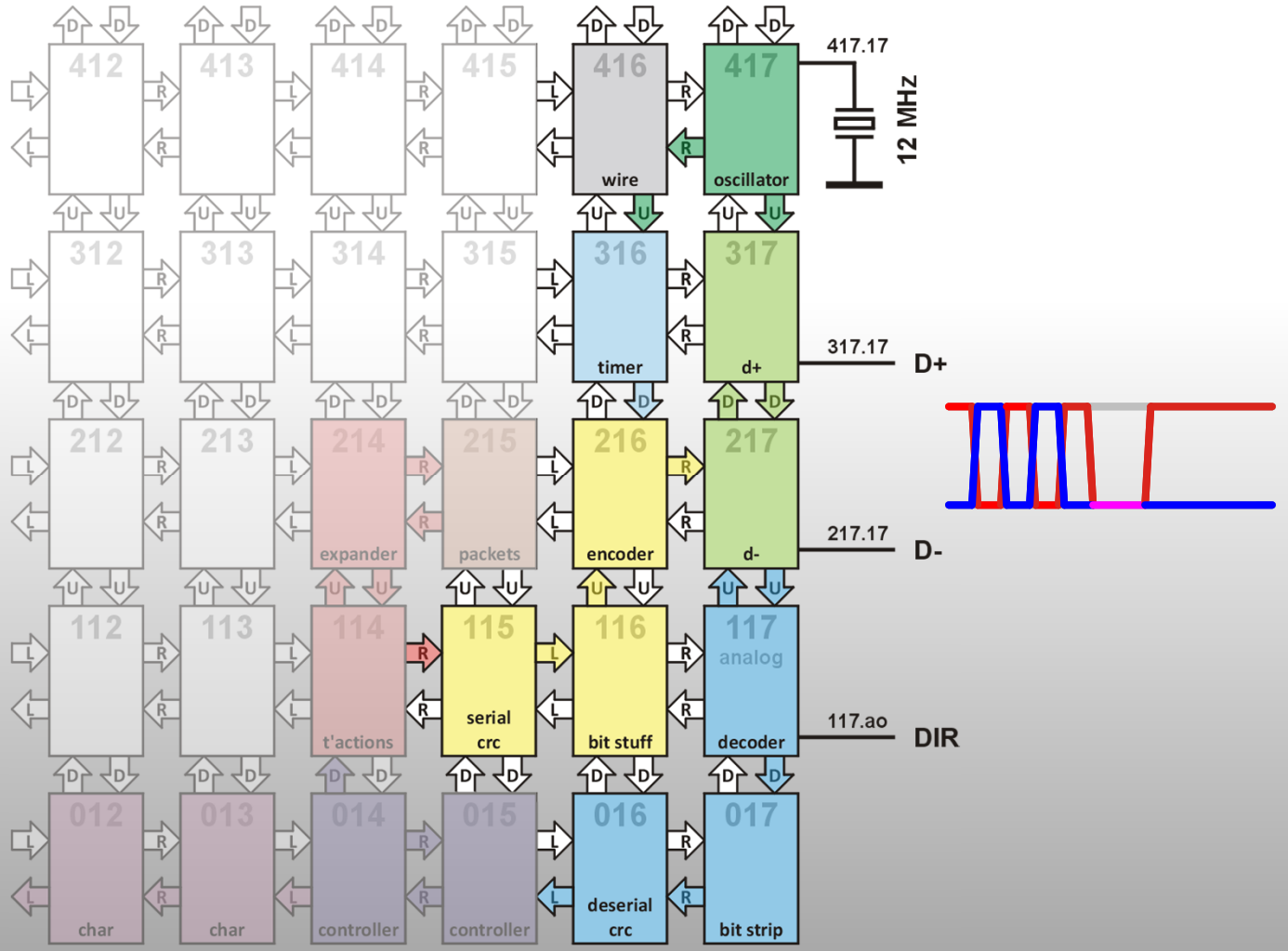
checks crc at the end of data packet

passes handshake packets and messages without crc calculation

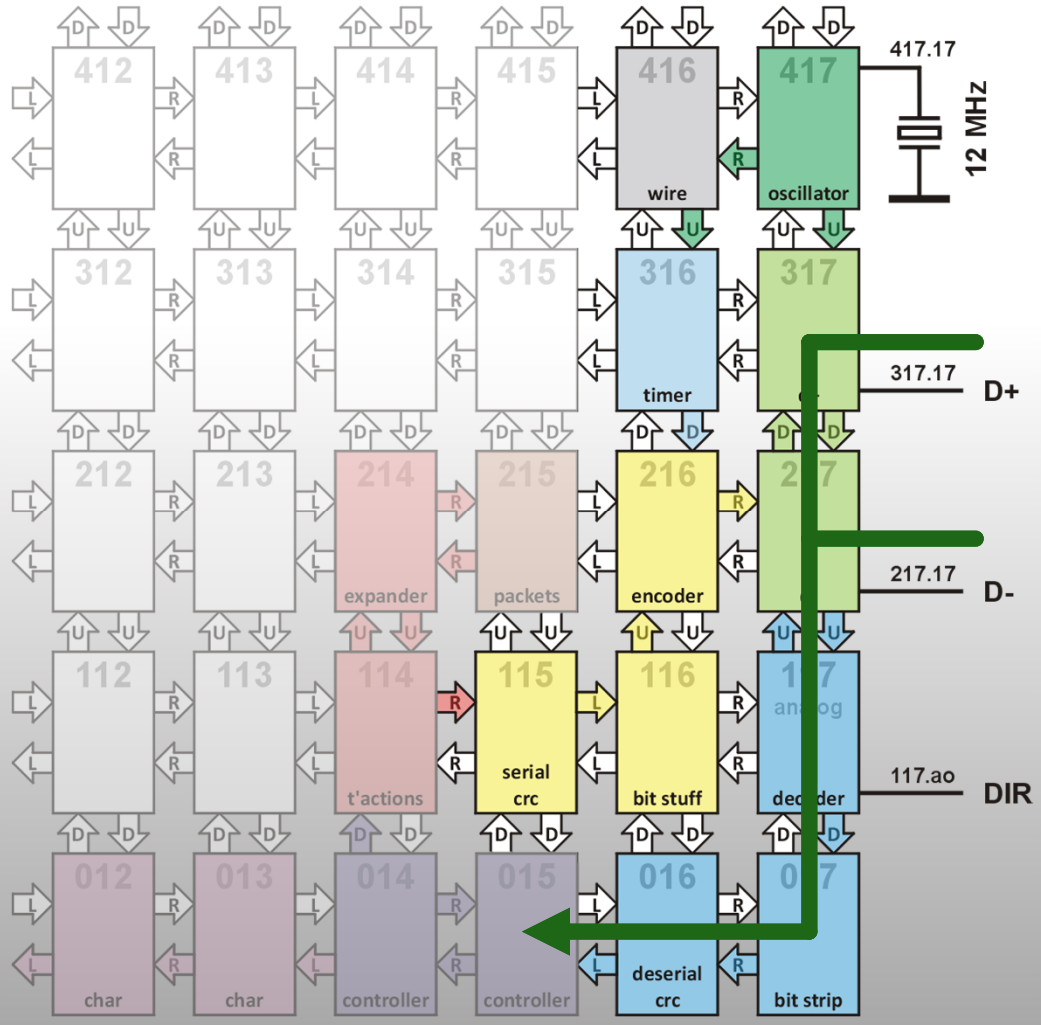
serial interface engine



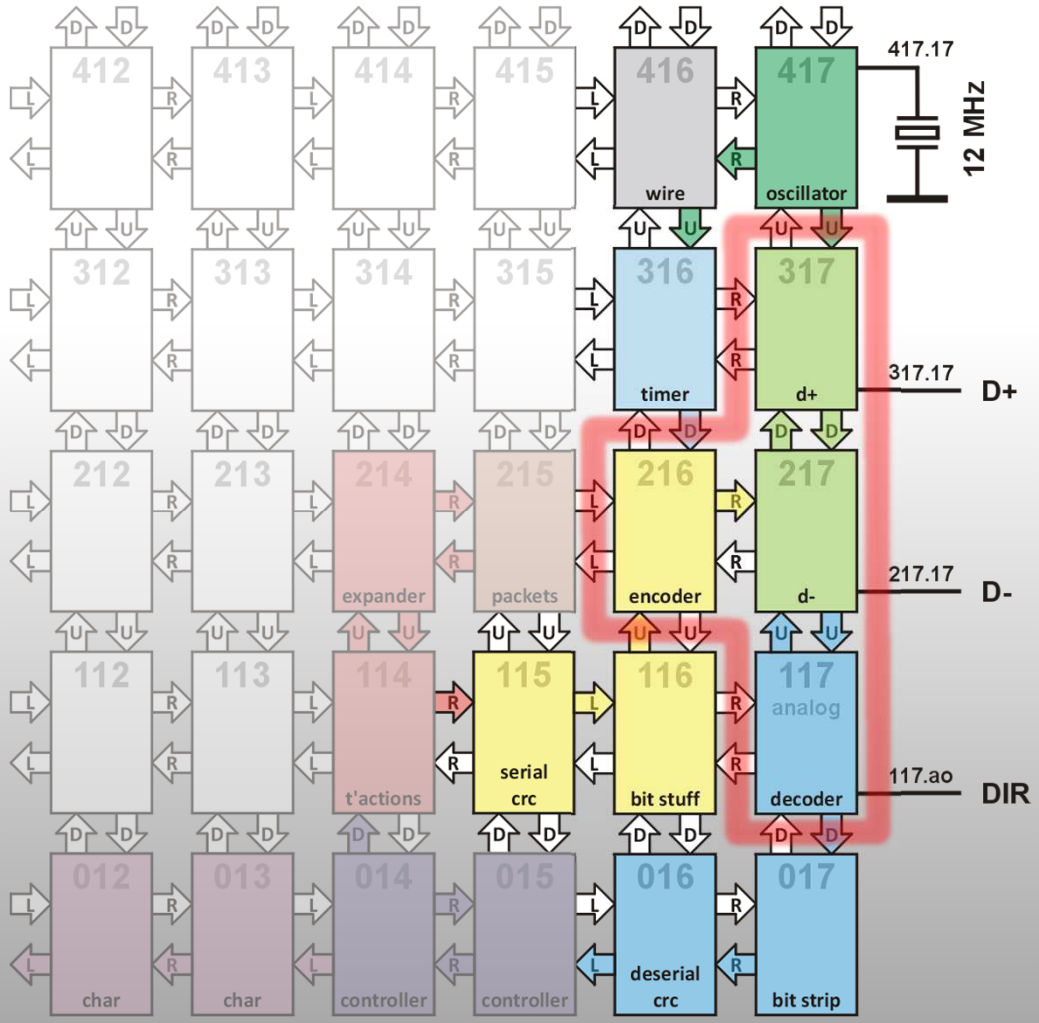
serial interface engine



serial interface engine



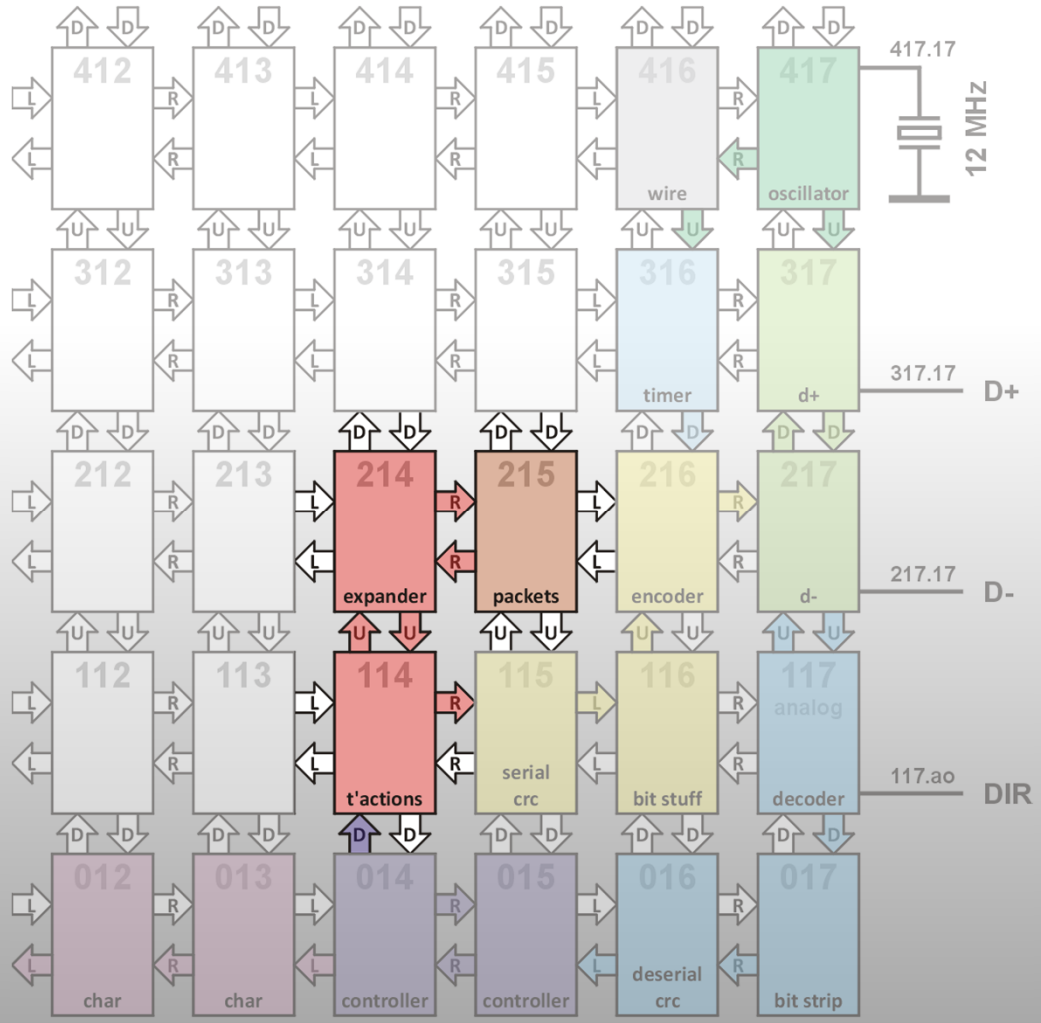
serial interface engine



IMPLEMENTATION

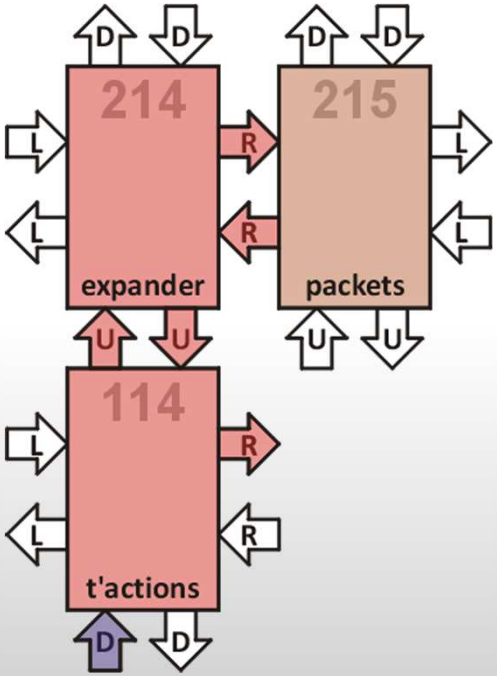
part II
keyboard controller

packets and transactions



packets and transactions

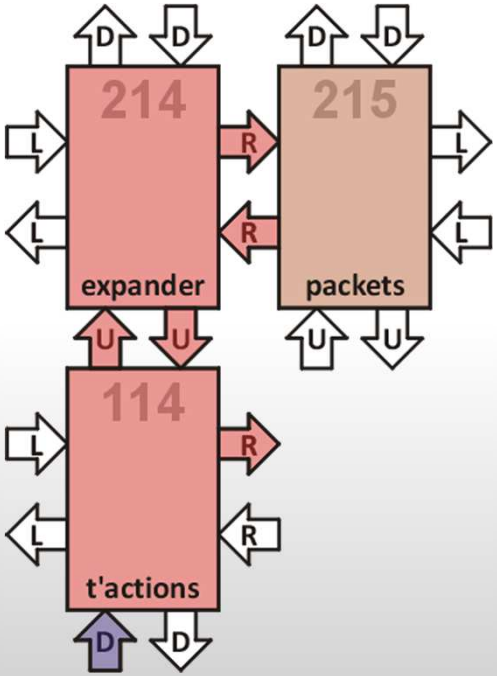
packets




```
packets 215 +node 215 /ram right /a right /p

reclaim 215 node 0 org
setup 00 602 , 802D , 10 , 38024 ,
in00 04 702 , 8069 , 10 , 38029 ,
in01 08 702 , 8069 , 80A0 , 38029 ,
ack 0C 401 , 80D2 , 38024 ,
conf 0F F07 , 80C3 , 20000 , 13416 ,
9 , 100 , 0 , 0 , 3802C ,
prot 18 F07 , 80C3 , 20000 , 13416 , 210B ,
0 , 0 , 0 , 3802C ,
idle 21 F07 , 80C3 , 20000 , 13416 , 210A ,
0 , 0 , 0 , 3802C ,
rept 2A F07 , 80C3 , 20000 , 13416 , 2109 ,
2 , 0 , 100 , 3802C ,
led 33 804 , 804B , 20000 , 13416 , 30005 ,
3802C ,
out00 39 602 , 80E1 , 10 , 38024 , 3D
```

packets and transactions

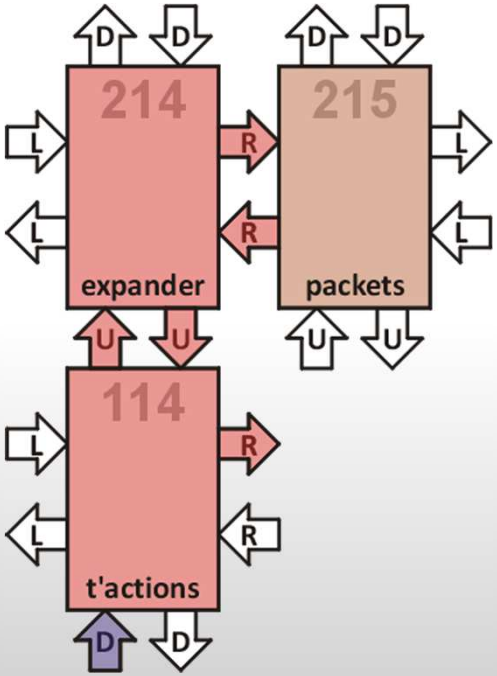
expander




```
expander 214 +node 214 /ram up /a right /b
1D send /p

reclaim 214 node 0 org
...
stp 24 20000 ! 13416 ! 20003 ! ;
stp@ 29 eop 20033 ! ;
crc@ 2C 20000 ! 1141C ! 20003 ! 2002D ! ; 33
```

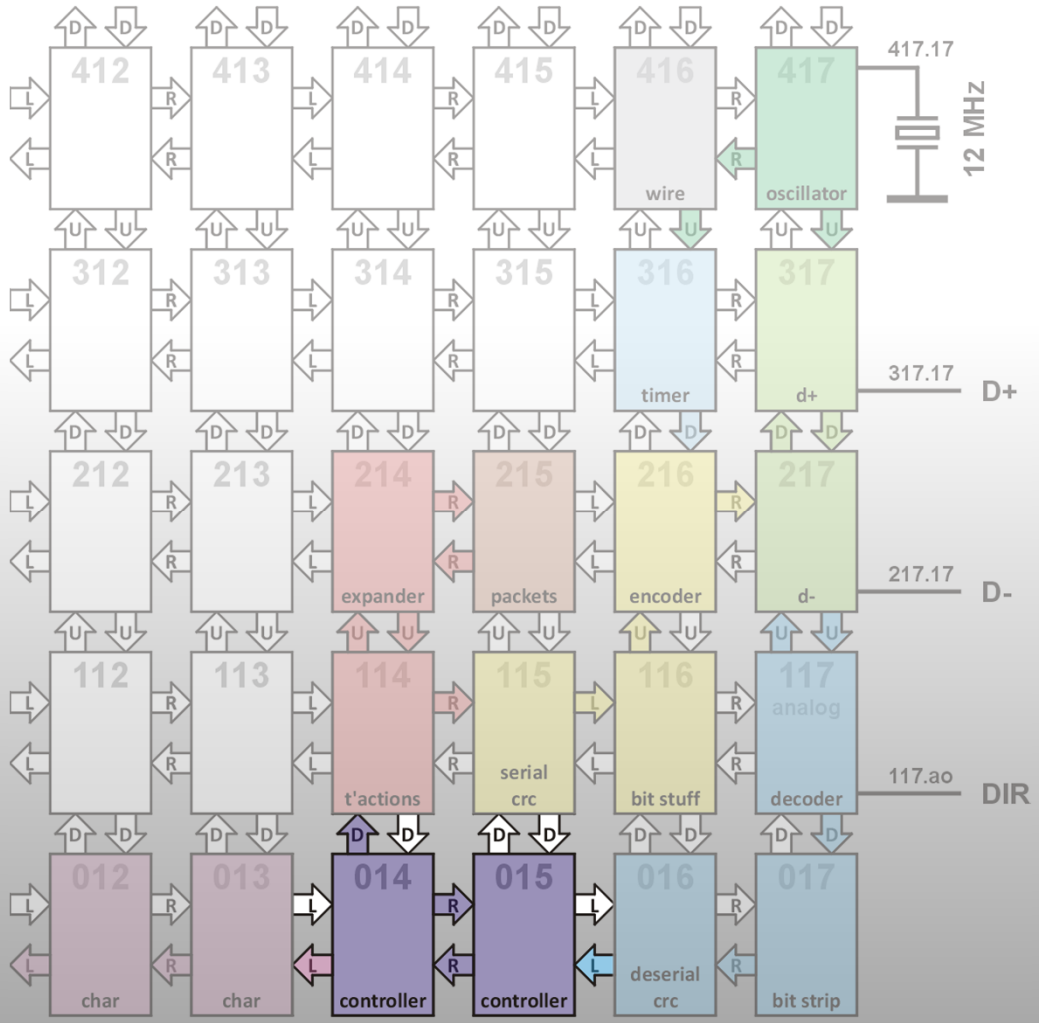
packets and transactions

transactions



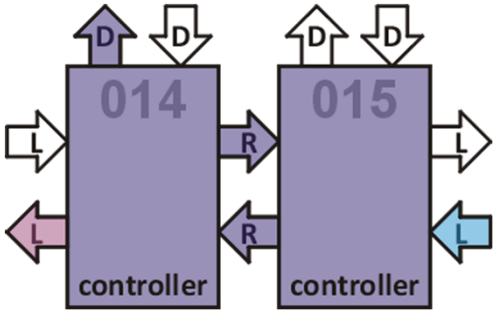

```
t'actions
114 +node 114 /ram right /a up /b down /p

reclaim 114 node 0 org
alive 00 20015 ! ;
sof 02 2001B ! ;
rst 04 20020 ! ;
pckt n 06 !b @b for @b ! unext ;
ack 08 C pckt ;
in00 0A 4 pckt ;
in01 0C 8 pckt ;
setc 0E 0 pckt F pckt ;
setp 12 0 pckt 18 pckt ;
seti 16 0 pckt 21 pckt ;
setr 1A 0 pckt 2A pckt ;
led 1E 39 pckt 33 pckt ;
wait 22 7 for alive next ; 27
```

controller


412
413
414
415
416
417
wire
oscillator
417.17
12 MHz
312
313
314
315
316
317
timer
d+
317.17
D+
212
213
214
215
216
217
expander
packets
encoder
d-
217.17
D-
112
113
114
115
116
117
analog
t'actions
serial
crc
bit stuff
decoder
117.ao
DIR
012
013
014
015
016
017
char
char
controller
controller
deserial
crc
bit strip

controller

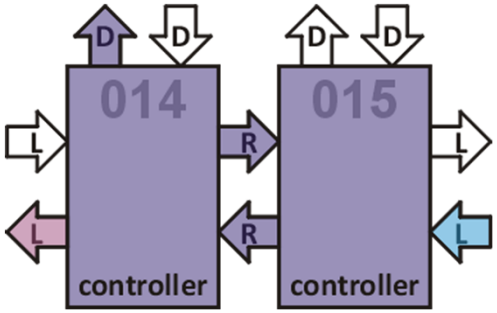
initialization




```
controller
15 +node 15 /ram left /a right /b 28 go /p

reclaim 15 node 0 org
...
go 28 ...
13404 rst x!
1340E setc rqst stat
13412 setp rqst stat
13416 seti rqst stat
1341A setr rqst 1341E led rqst stat
13425 keys !b r--- ; 3D
```

controller

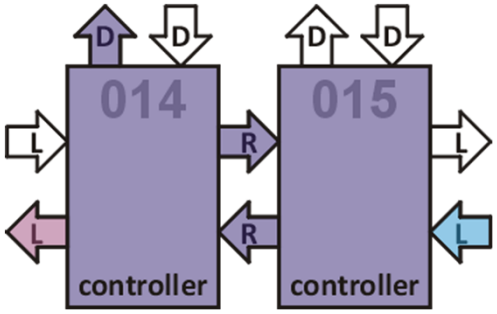
report




```
controller
14 +node 14 /ram down /a right /b right /p

reclaim 14 node 0 org
...
keys 24 begin 13422 wait !
sof 1340C in01 ! report end 2B
```

controller

report

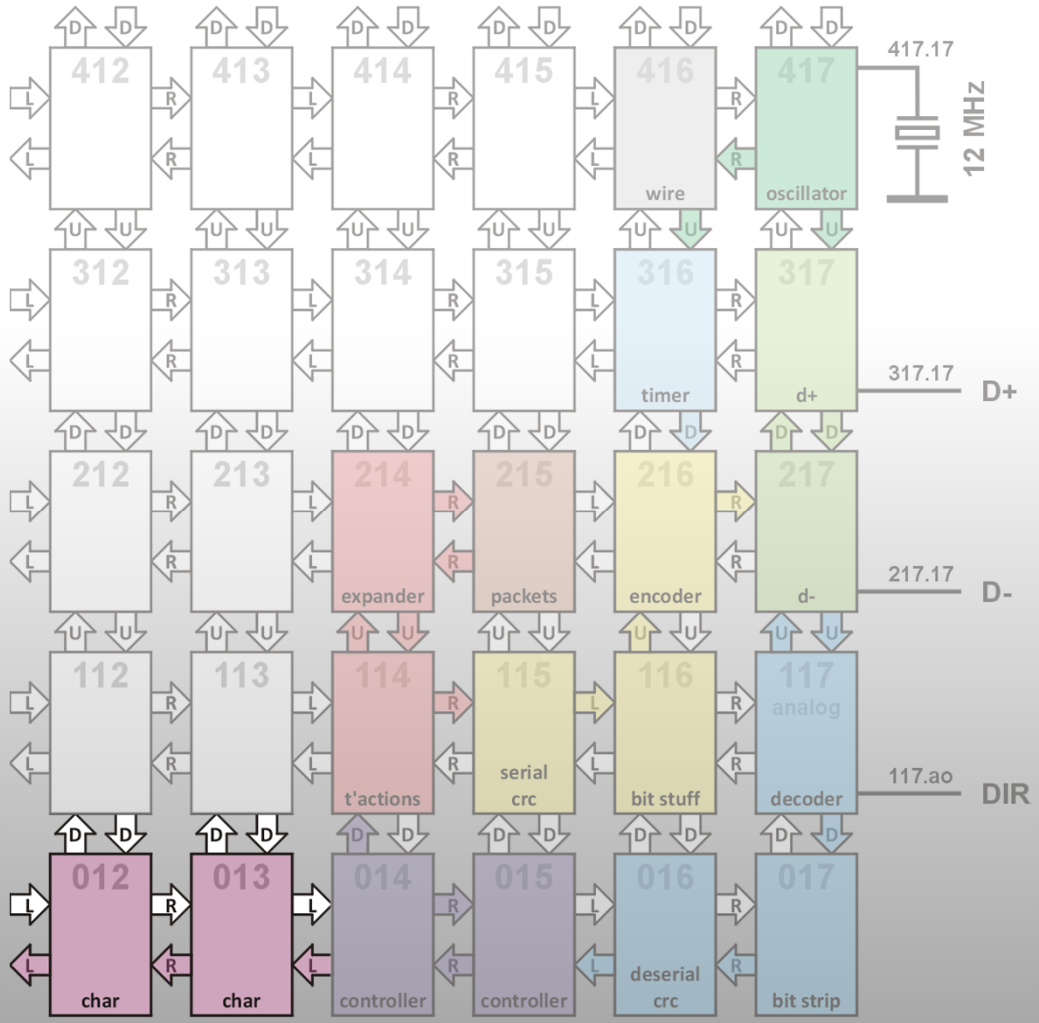


byte	description
0	modifier keys
1	reserved
2	keycode 1
3	0
4	0
5	0
6	0
7	0

bit	modifier key
0	left CTRL
1	left SHIFT
2	left ALT
3	left GUI
4	right CTRL
5	right SHIFT
6	right ALT
7	right GUI

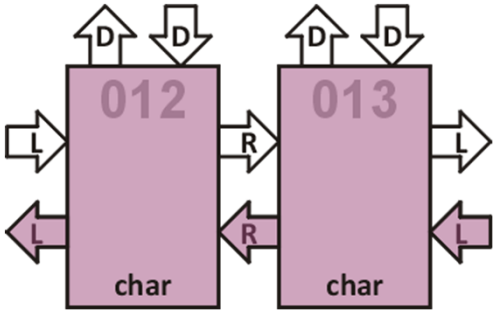
Device Class Definition for HID, version 1.11, USB Implementers' Forum, 2001

character decoder



character decoder

keycodes

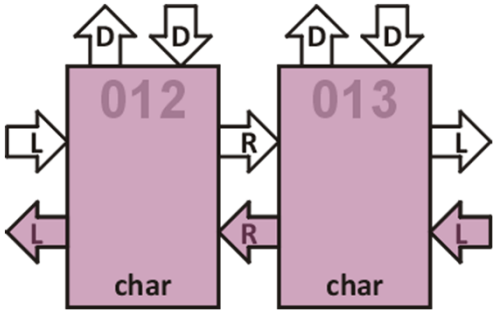


hex	keycode	hex	keycode	hex	keycode	hex	keycode
00	no event	12	o O	24	7 &	36	, <
01	roll over	13	p P	25	8 *	37	. >
02	POST fail	14	q Q	26	9 (	38	/ ?
03	error	15	r R	27	0)	39	Caps Lock
04	a A	16	s S	28	Enter	3A	F1
05	b B	17	t T	29	Esc	3B	F2
06	c C	18	u U	2A	Backspace	3C	F3
07	d D	19	v V	2B	Tab	3D	F4
08	e E	1A	w W	2C	Spacebar	3E	F5
09	f F	1B	x X	2D	- _	3F	F6
0A	g G	1C	y Y	2E	= +	40	F7
0B	h H	1D	z Z	2F	[{	41	F8
0C	i I	1E	1 !	30	] }	42	F9
0D	j J	1F	2 @	31	non-US #	43	F10
0E	k K	20	3 #	32	\ |	44	F11
0F	l L	21	4 $	33	; :	45	F12
10	m M	22	5 %	34	' "	46	PrintScr
11	n N	23	6 ^	35	` ~	47	ScrLock

HID Usage Tables, version 1.12, USB Implementers' Forum, 2004

character decoder

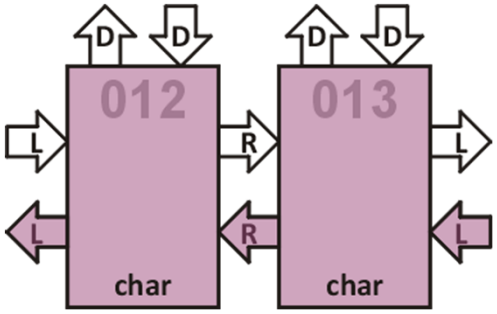
etherForth character set



hex	char	hex	char	hex	char
00	0	10	g	20	w
01	1	11	h	21	x
02	2	12	i	22	y
03	3	13	j	23	z
04	4	14	k	24	*
05	5	15	l	25	/
06	6	16	m	26	@
07	7	17	n	27	!
08	8	18	o	28	.
09	9	19	p	29	,
0A	a	1A	q	2A	;
0B	b	1B	r	2B	'
0C	c	1C	s	2C	#
0D	d	1D	t	2D	-
0E	e	1E	u	2E	?
0F	f	1F	v	2F	+

character decoder

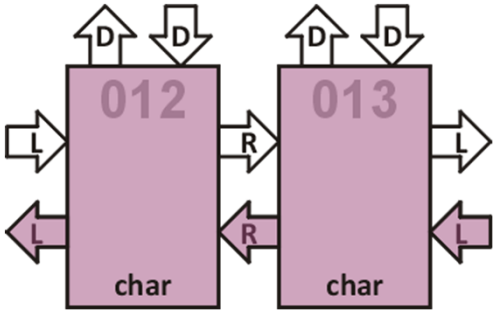
converting keycodes to etherForth character set



hex	keycode	hex	keycode	hex	keycode	hex	keycode
		12	o O	24	7 &	36	. <
		13	p P	25	8 *	37	. >
		14	q Q	26	9 (	38	/ ?
		15	r R	27	0)	39	Caps Lock
04	a A	16	s S	28	Enter	3A	F1
05	b B	17	t T	29	Esc	3B	F2
06	c C	18	u U	2A	Backspace	3C	F3
07	d D	19	v V	2B	Tab	3D	F4
08	e E	1A	w W	2C	Spacebar	3E	F5
09	f F	1B	x X	2D	- _	3F	F6
0A	g G	1C	y Y	2E	= +	40	F7
0B	h H	1D	z Z	2F	[{	41	F8
0C	i I	1E	1 !	30	] }	42	F9
0D	j J	1F	2 @	31	non-US #	43	F10
0E	k K	20	3 #	32	\ |	44	F11
0F	l L	21	4 $	33	; :	45	F12
10	m M	22	5 %	34	' "	46	PrintScr
11	n N	23	6 ^	35	` ~	47	ScrLock

character decoder

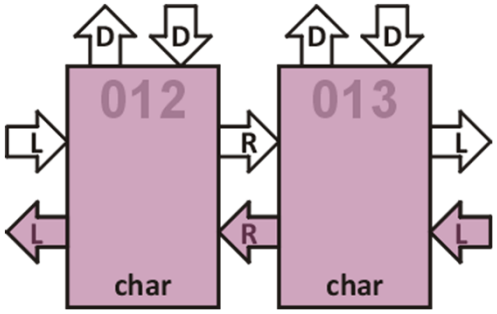
converting keycodes to etherForth character set



hex	keycode	hex	keycode	hex	keycode	hex	keycode
				24	7	36	,
				25	8 *	37	.
				26	9	38	/ ?
				27	0		
				2D	-		
				2E	= +		
		1E	1 !				
		1F	2 @				
		20	3 #				
		21	4	33	;		
		22	5	34	'		
		23	6				

character decoder

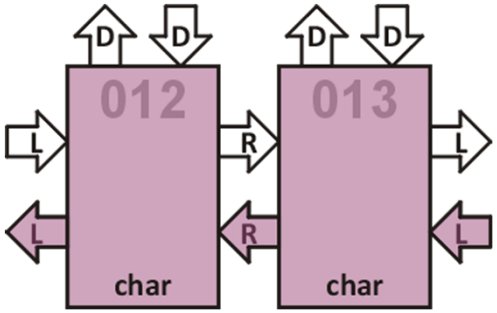
converting keycodes to etherForth character set




```
chars
reclaim 12 node 0 org
chrs 3F009 0 , 27040 !1 , 26081 @2 ,
2C0C2 #3 , 3F103 4 , 3F144 5 , 3F185 6 ,
3F1C6 7 , 24207 *8 , 3F248 9 , 3FB4F - ,
2FFD0 + , 3FA95 ; , 3FAD6 ' , 3FA58 , ,
3FA19 . , 2E95A ?/ ,
...
```

character decoder

function keys and non-printable characters



message – bit 17 set

F1 – 20001, F2 – 20002 …

convert to call instruction

Esc – 1341C, Enter – 13408 …

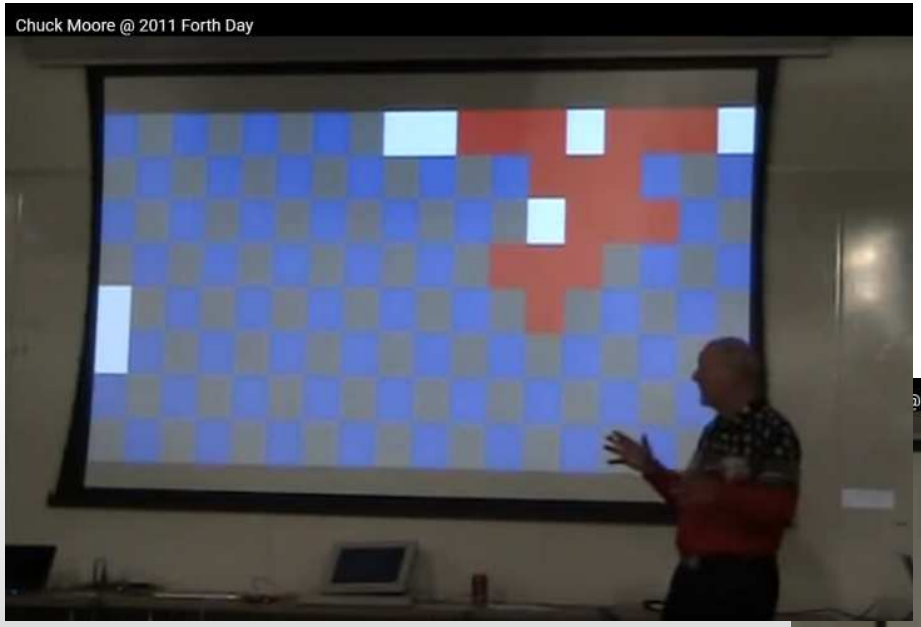
DEMO I

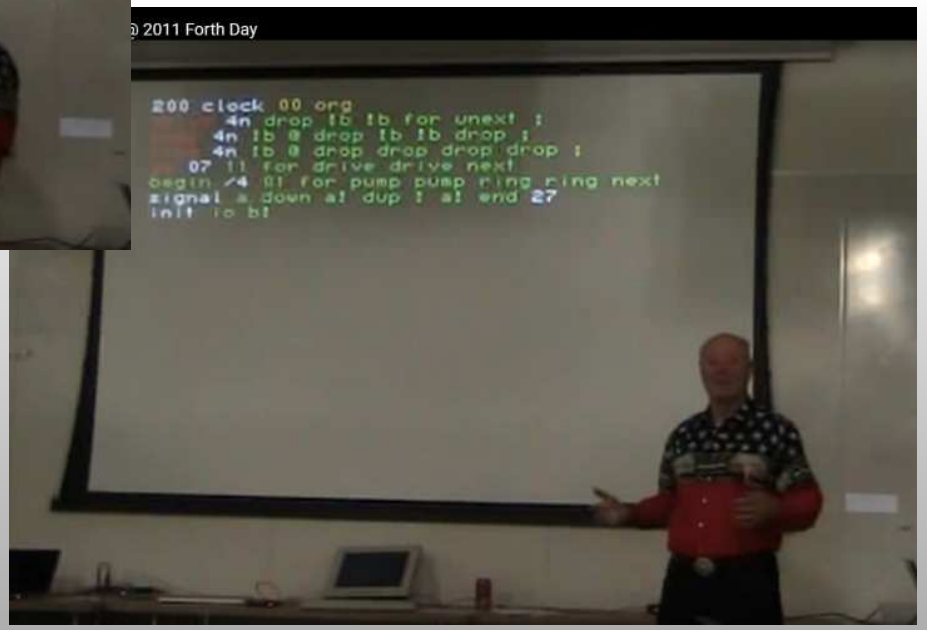
USB host and keyboard controller

http://www.youtube.com/watch?v=Ftot1N9Jaas&t=41m10s

etherForth

Chuck Moore – Forth Day 2011



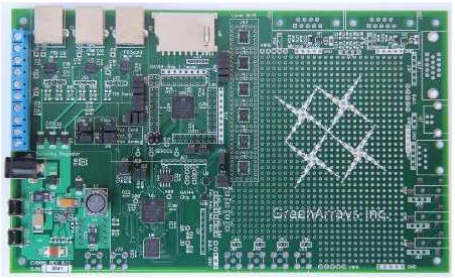


etherForth

Eval Weblog

Eval Weblog

Rationale



This blog will document my experience with GreenArrays' Evaluation Board. It has 2 GA144 multi-computer chips, each with 144 f18a computers. Total of 288 computers running at 650 Mips or 194 Gips. Each chip has 5 A/Ds and 5 D/As. Total of 10 of each. I'll use some for video output and some for audio (command) input.

The Host chip has a 256 Mword flash memory and a 1Mword 16-bit SRAM. It has a 700 Mbps serial link to the Target chip.

I plan to use it for a variety of experiments, which I'll describe here. The goal is to port colorForth and OKAD to this platform and use it to design the next multi-computer chip.

Here are some pages of enduring interest:

- c18 instruction use
- Arithmetic code
- etherCode
- The Map is Not the Territory
- video output

2012 August 13 Monday

I continue to add more etherCode as it gets written and documented.

If you have code you like, consider sharing it with GreenArrays.

I'm starting an ambitious new project on the Eval board. It will generate spin-off publishable code

2012 July 19 Thursday

I continue to add etherCode examples. And I've edited the instruction and arithmetic pages. It's always best to use the latest advice.

2012 July 13 Friday

I've quit listing time-of-day since I no longer have a clock. I'm keeping time by the sun.

Earlier I posted etherCode about random numbers. I've reposted it with corrections. My initial impression of randomness was refuted when I tried to produce normally-distributed numbers. I now use the high-order bits of the product, instead of the seed.

2012 July 9 12:00 Monday

Every now and then an idea occurs that should have been obvious sooner. I actually did this accidently, then realized the value.

In order to compare 2 functions on the monitor, I've drawn one red and another green. A third can be blue. But there's another way. Alternate 2 frames:

- Draw one function with 768 scan lines

github.com/colorforth/colorforth.github.io

etherForth

introduction

- variant of colorForth, resident in GA144
- ether – software for routing messages about the chip*
- source code with pre-parsed words
  - 6-bit tags
  - 6-bit characters and tokens

* GreenArrays' app notes 11 and 17; Ganglia: A Dynamic Message Routing Surface

etherForth

tags, characters, tokens

hex	tag	color
30	token	
31	token	
32	char	
33	char	
34	char	
35	char	
36	decimal	
37	decimal	
38	address	
39	eol	
3A	space	
3B	cursor	
3C	eob	
3D	char	
3E	hex	
3F	hex	

hex	char	token	hex	char	token	hex	char	token
00	0	;	10	g	+*	20	w	right
01	1	ex	11	h	2*	21	x	down
02	2	begin	12	i	2/	22	y	left
03	3	end	13	j	-	23	z	up
04	4	unext	14	k	+	24	*	io
05	5	next	15	l	and	25	/	-till
06	6	if	16	m	or	26	@	then
07	7	-if	17	n	drop	27	!	else
08	8	@p	18	o	dup	28	.	till
09	9	@+	19	p	pop	29	,	while
0A	a	@b	1A	q	over	2A	;	for
0B	b	@	1B	r	a	2B	'	zif
0C	c	!p	1C	s	.	2C	#	data
0D	d	!+	1D	t	push	2D	-	ldata
0E	e	!b	1E	u	b!	2E	?	ahead
0F	f	!	1F	v	a!	2F	+	leap

etherForth

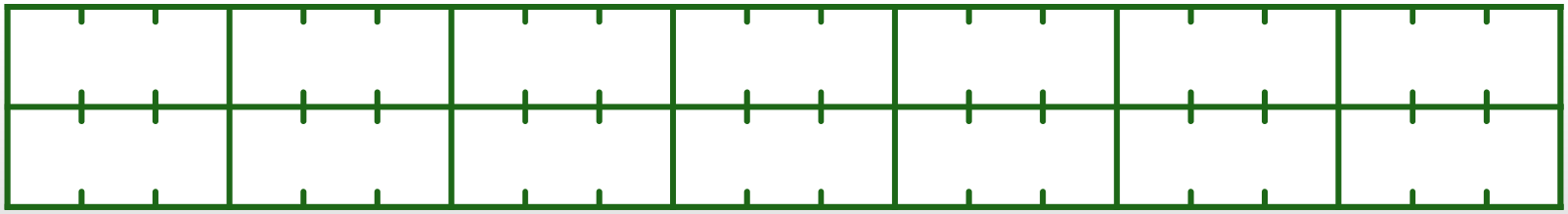
tags, characters, tokens

```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```

etherForth

tags, characters, tokens

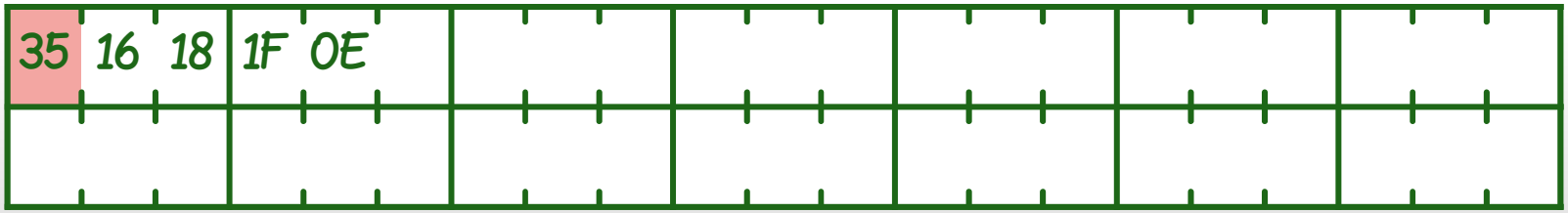
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

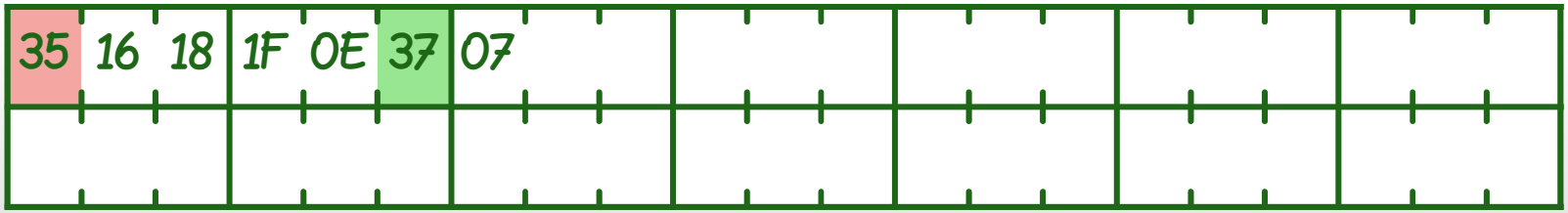
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

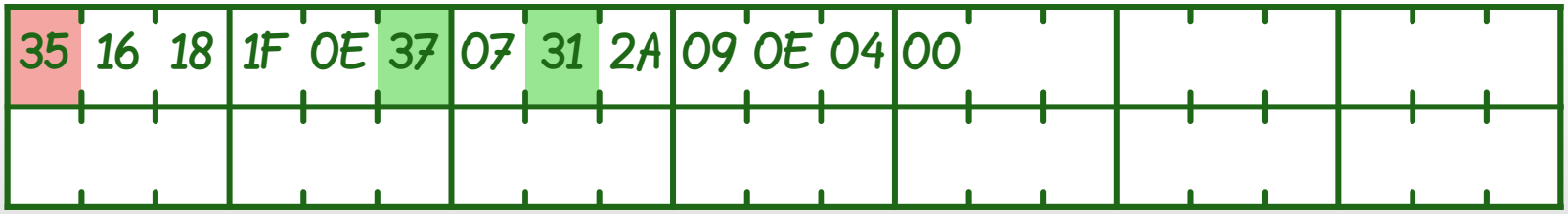
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

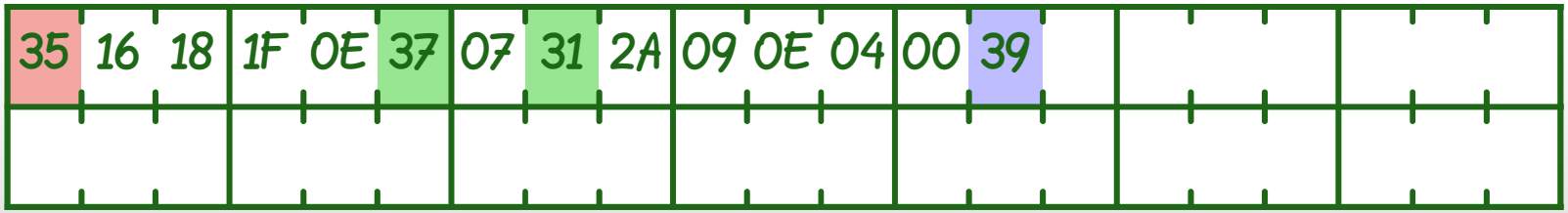
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

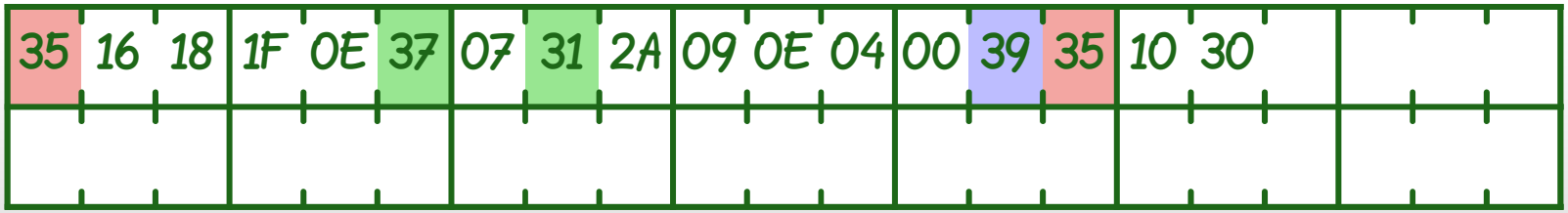
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

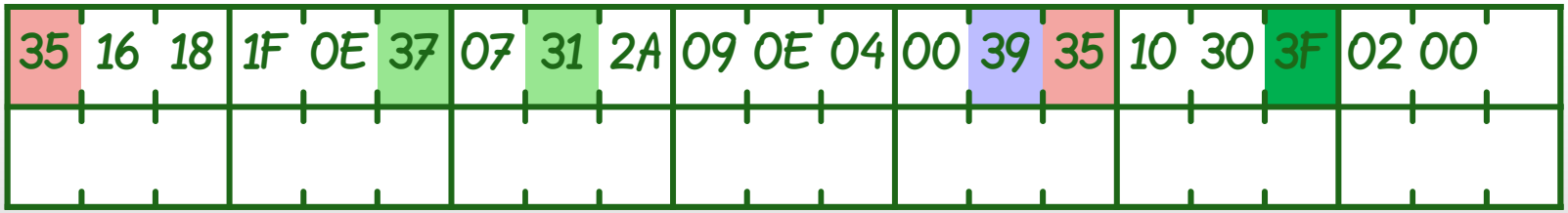
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

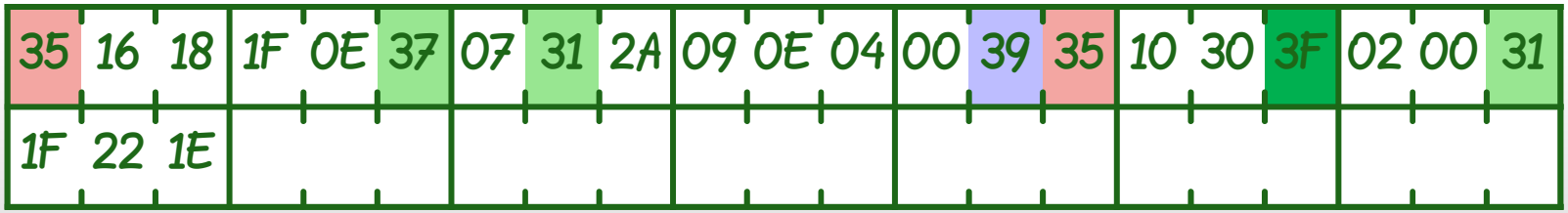
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

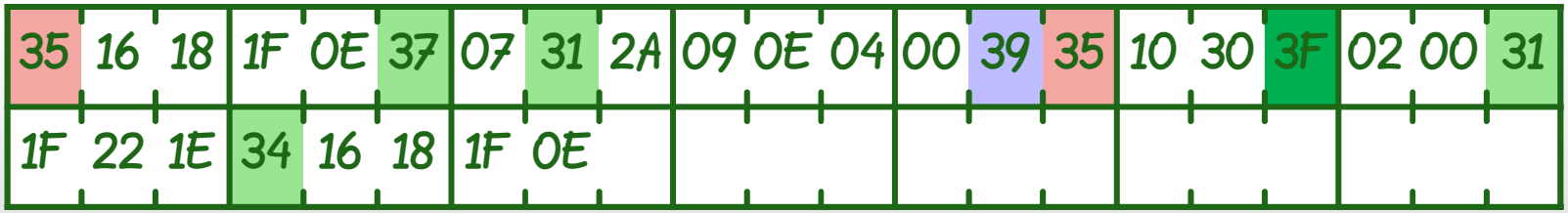
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

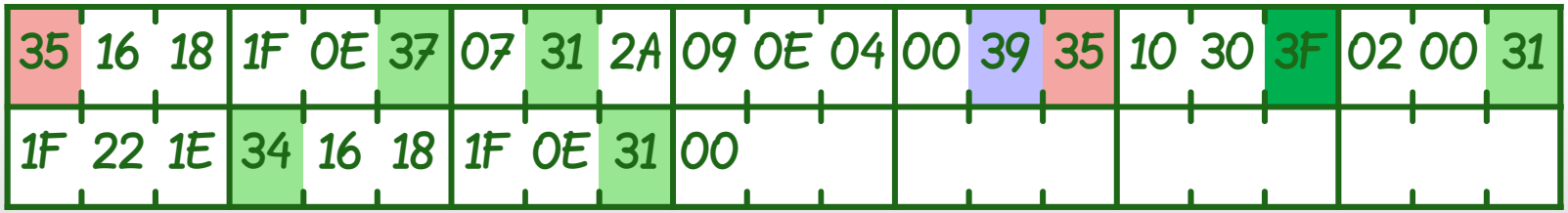
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

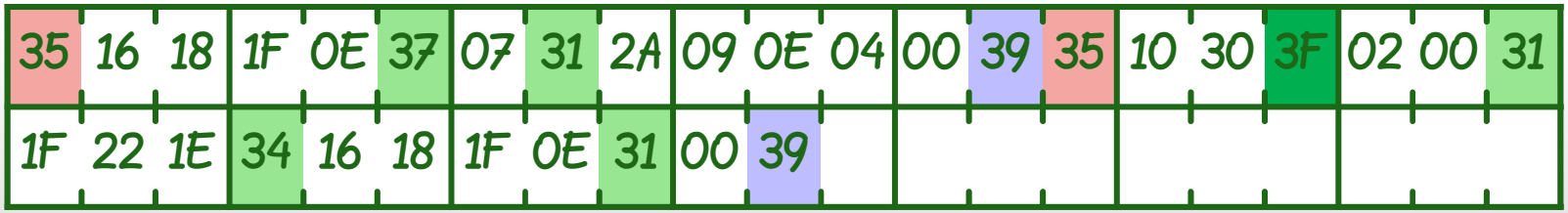
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

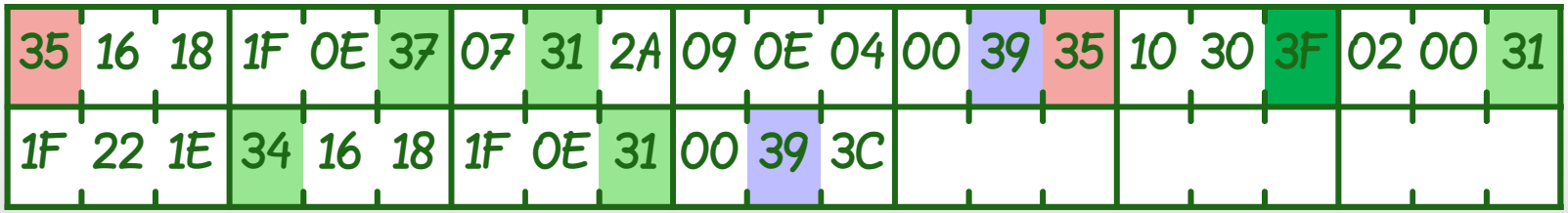
```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



etherForth

tags, characters, tokens

```
move 7 for @+ !b unext ; ,
go 20 a! left b! move ; ,
```



51 characters = 11 18-bit words (33 6-bit characters)

etherForth

introduction

variant of colorForth, resident in GA144

ether – software for routing messages about the chip*

source code with pre-parsed words

- 6-bit tags
- 6-bit characters and tokens

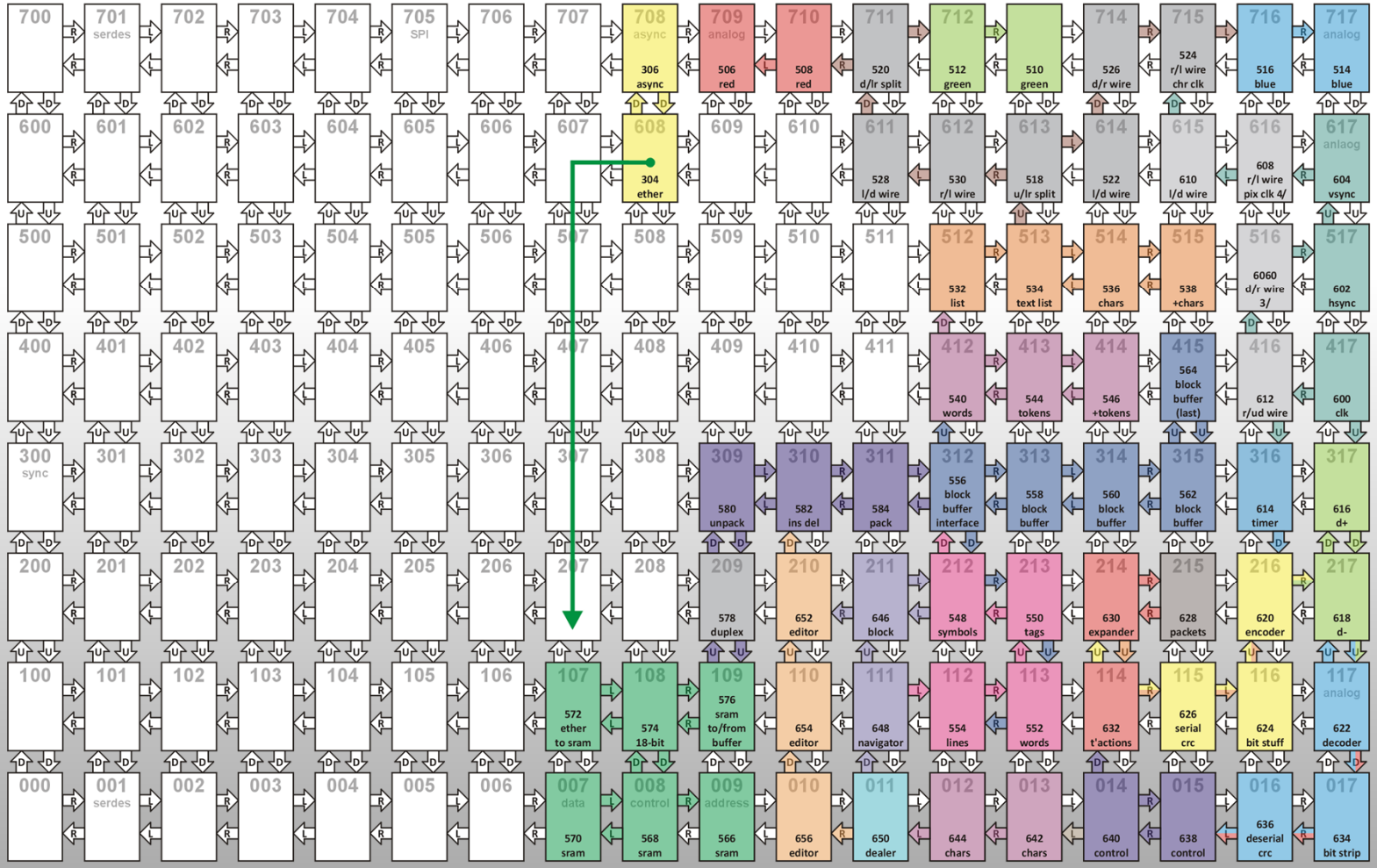
standalone operating system, booting from Flash

expects RAM, RGB display, and keyboard

* GreenArrays' app notes 11 and 17; Ganglia: A Dynamic Message Routing Surface

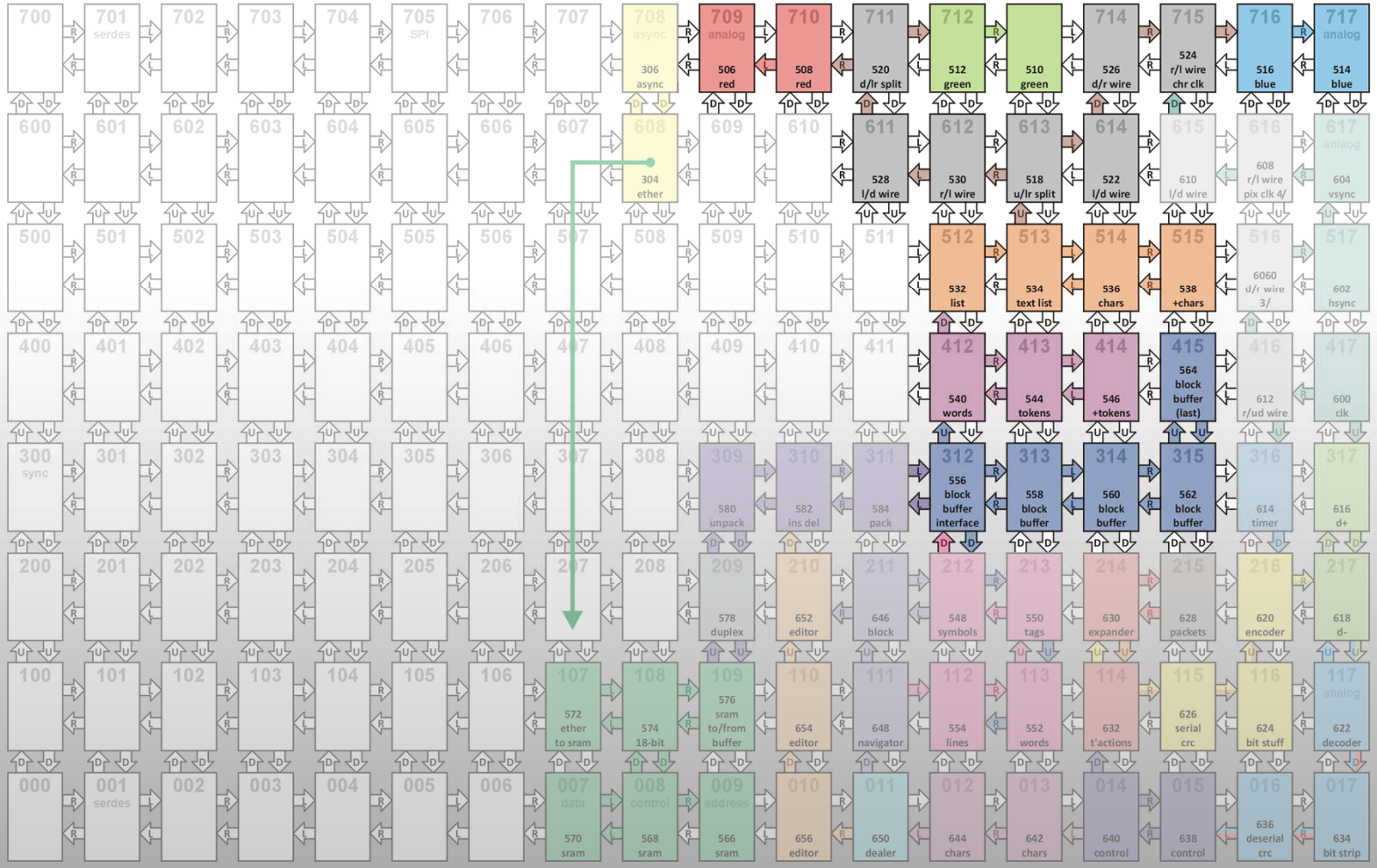
floorplan

etherForth



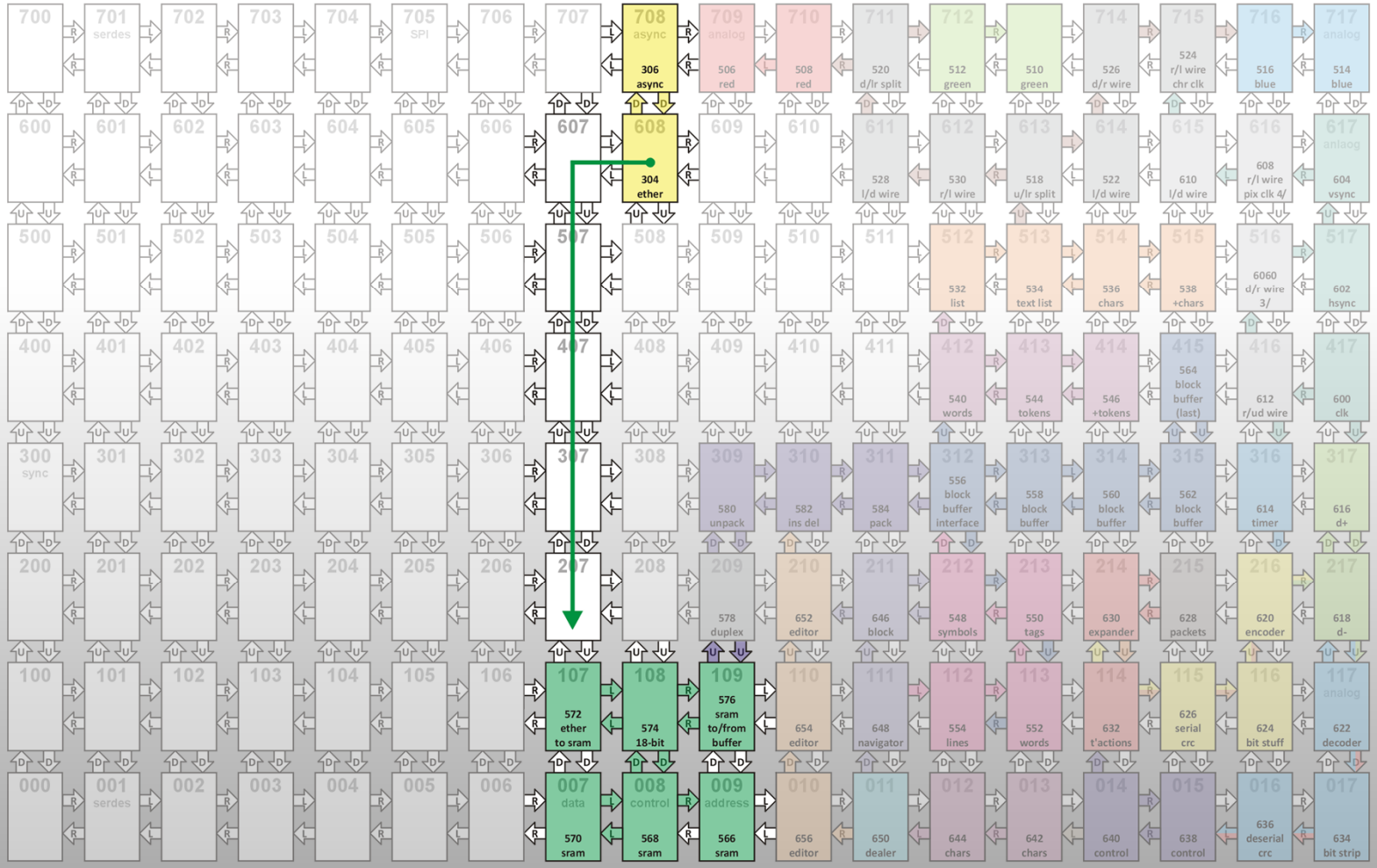
floorplan

etherForth – VGA display



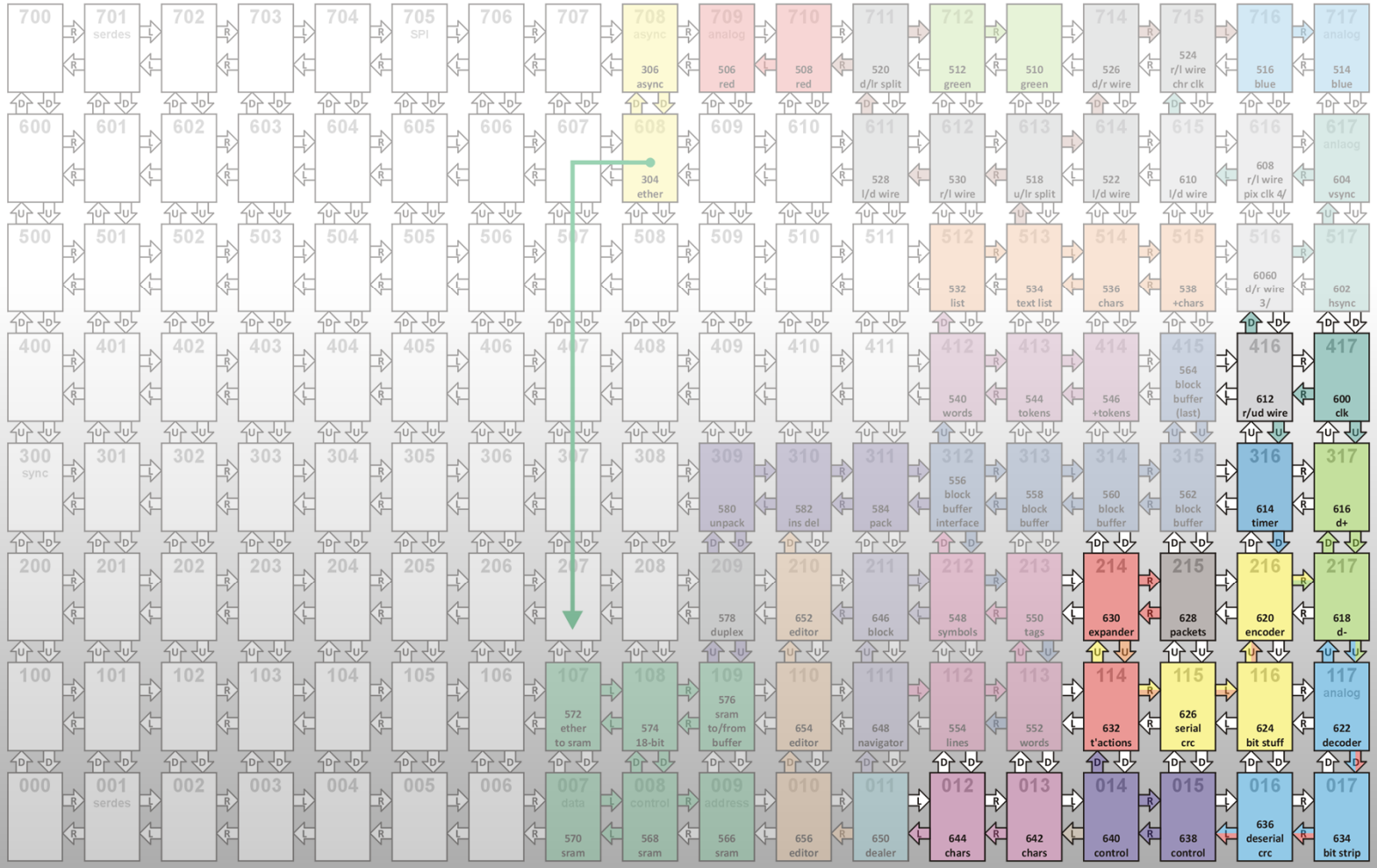
floorplan

etherForth - SRAM



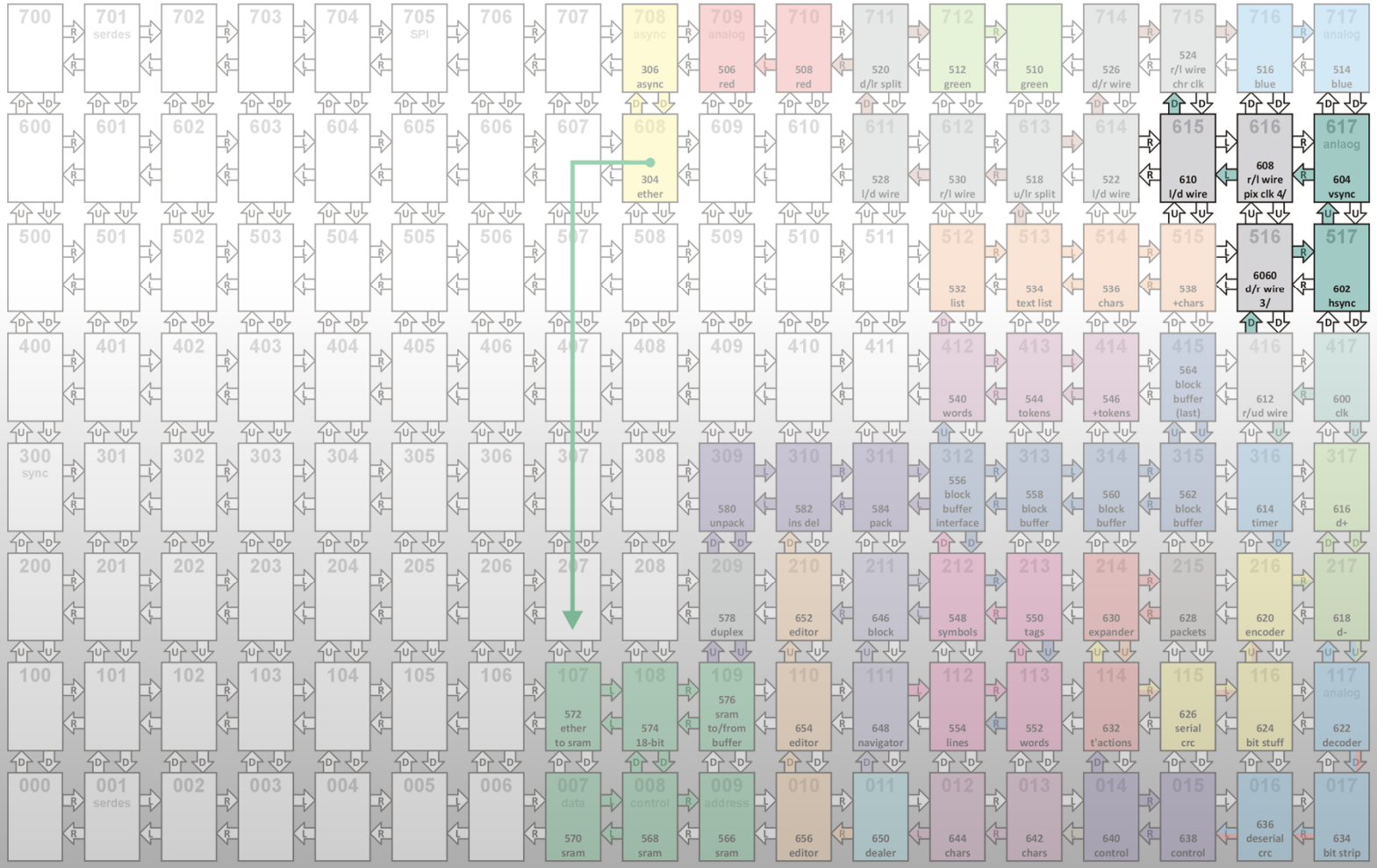
floorplan

etherForth – USB keyboard



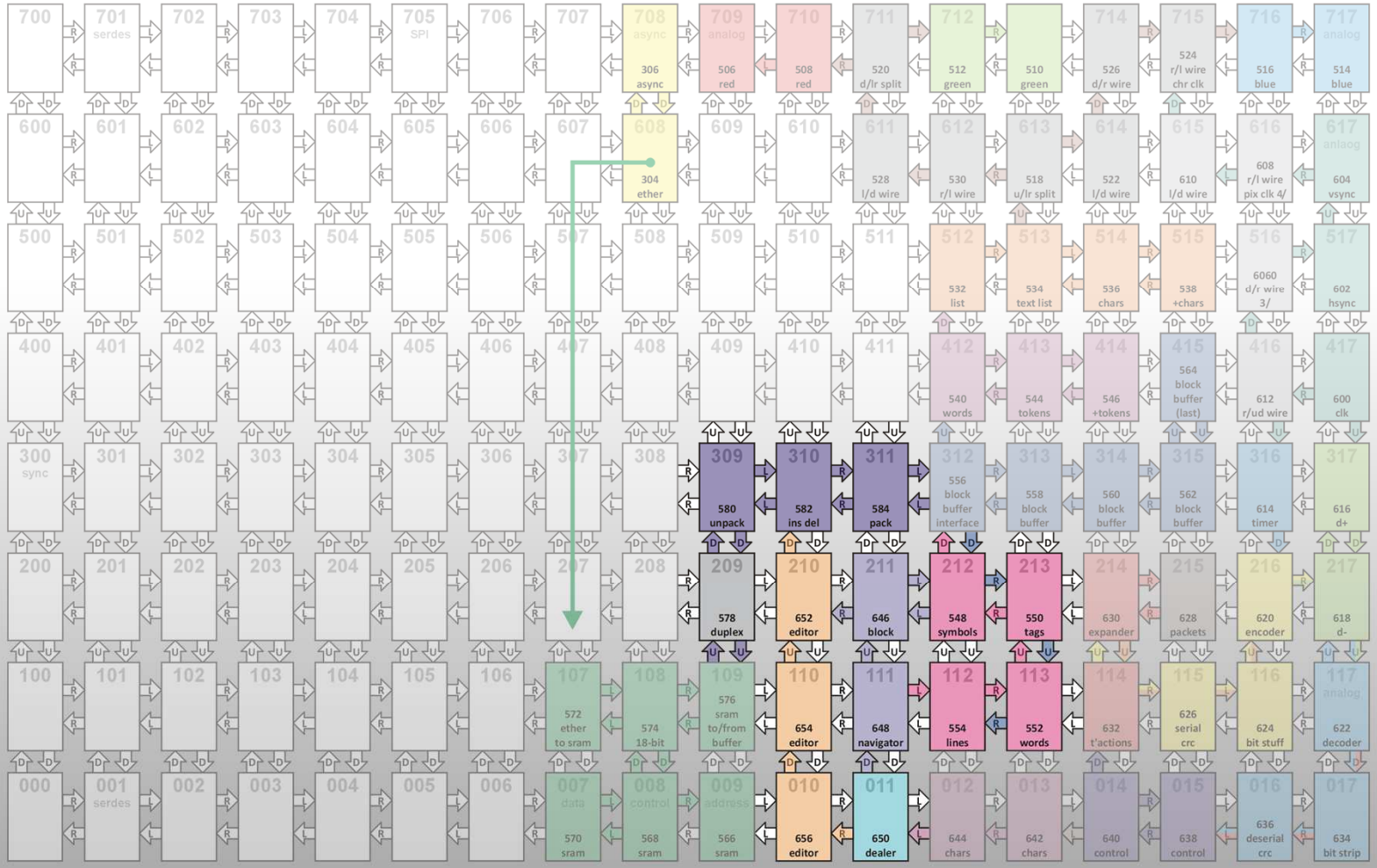
floorplan

etherForth - video syncs & char clock



floorplan

etherForth - editor



DEMO II

etherForth

http://www.youtube.com/watch?v=Ftot1N9Jaas&t=49m56s

CONCLUSION

Can GA144 speak USB-ish?

YES! (at low speed, with one device only)

acknowledgements

GreenArrays, Inc.

Greg Bailey

Chuck Moore

MQP Electronics

contact information

Email: dkalny@seznam.cz Skype: live:dkalny