Serial I/O II

ECE 3710

I have had a perfectly wonderful evening, but this wasn't it.

- Groucho Marx

serial communication overview

0. set speed (baud rate) at which TX/RX occurs

(the same; assuming one UART)

1. TX:

a. move data to TX register

b. wait for TX to finish

c. goto 1a.

presumably a bit flip, somewhere

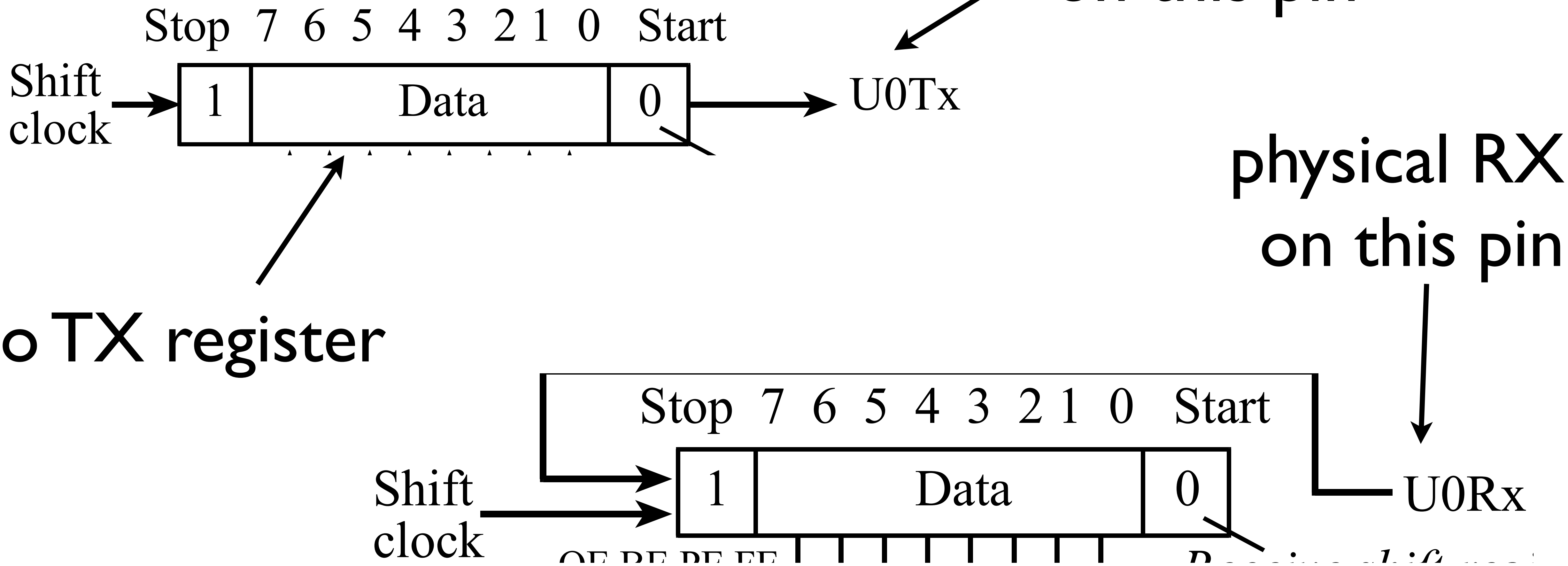
2. RX:

a. wait for RX to finish

b. move data from RX register

c. goto 2a.

the how of serial TX/RX



physical TX on this pin

physical RX on this pin

to TX register

from RX register

serial i/o:

1. knowing when you’ve got it

2. knowing when it’s done

RXFF=1

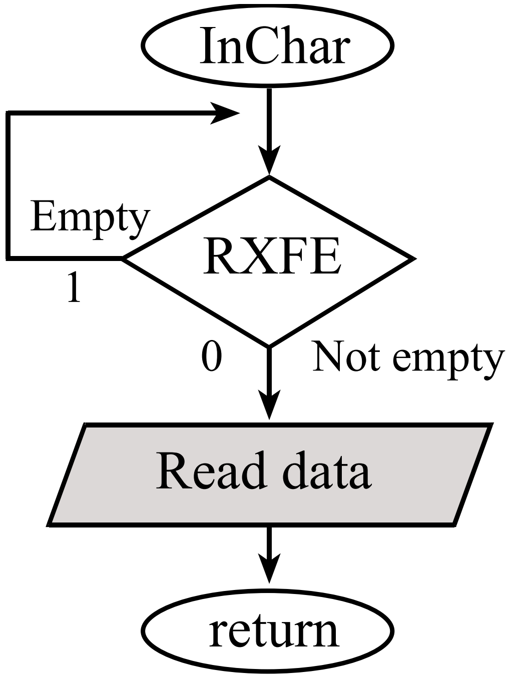
a. rx buffer full

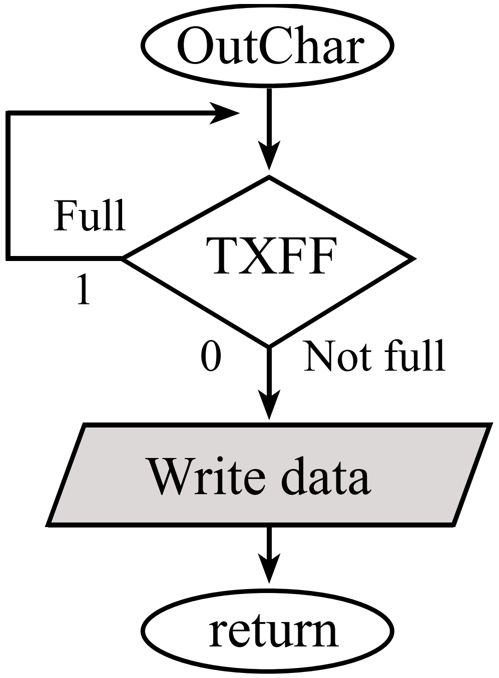
b. rx reg full

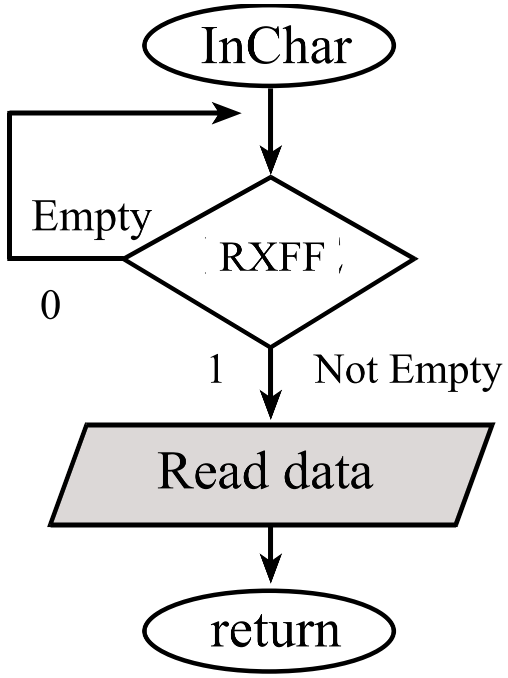
TXFE=1

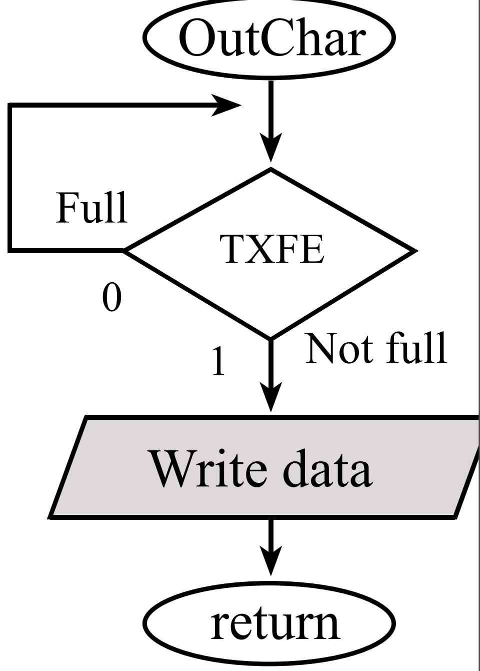
a. tx buffer emtpy

b. tx reg empty









RXFE=1

a. rx buffer empty

b. rx reg empty

TXFF=1

a. tx buffer full

b. tx reg full

note: buffer not full isn’t the same as empty

polling approach: watch these bits

example: RX data and put on stack

```
  ; assume UART0 configured for RX
  mov R0,#0x0
RX
  ; wait for data: RXFE!=1 when we have data
  ldr R2,[R1,#0x18] ;get status
  ands R2,#0x10 ;set Z=1 if result is zero
  bne RX          ;branch if Z=0 (RXFE==1)
  ; got data put in R0 and push to stack
  ldrb R0,[R1,#0x0] ;just a byte
  push {R0}
  b RX ;wait for more data
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		

example: RX data and put on stack

if no FIFO queue

```
  ; assume UART0 configured for RX
  mov R0,#0x0
RX
  ; wait for data: RXFF=1 when we have data
  ldr R2,[R1,#0x18] ;get status
  ands R2,#0x40 ;set Z=1 if result is zero
  beq RX         ;branch if Z=1 (RXFF==0)
  ; got data put in R0 and push to stack
  ldrb R0,[R1,#0x0] ;just a byte
  push {R0}
  b RX ;wait for more data
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		

serial i/o

complicate it:

1. fixed-point arithmetic for baud rate
2. TX/RX register (just one)
3. hardware buffering

this time

1. baud rate calculation

most uC use a baud rate divisor:

we choose these

$$BRD = \frac{\text{SysClk}}{16 \times \text{Baud rate}}$$

may vary by uC

system clock >> I/O bandwidth (bps): need to slow it down

baud rate divisor

assume:

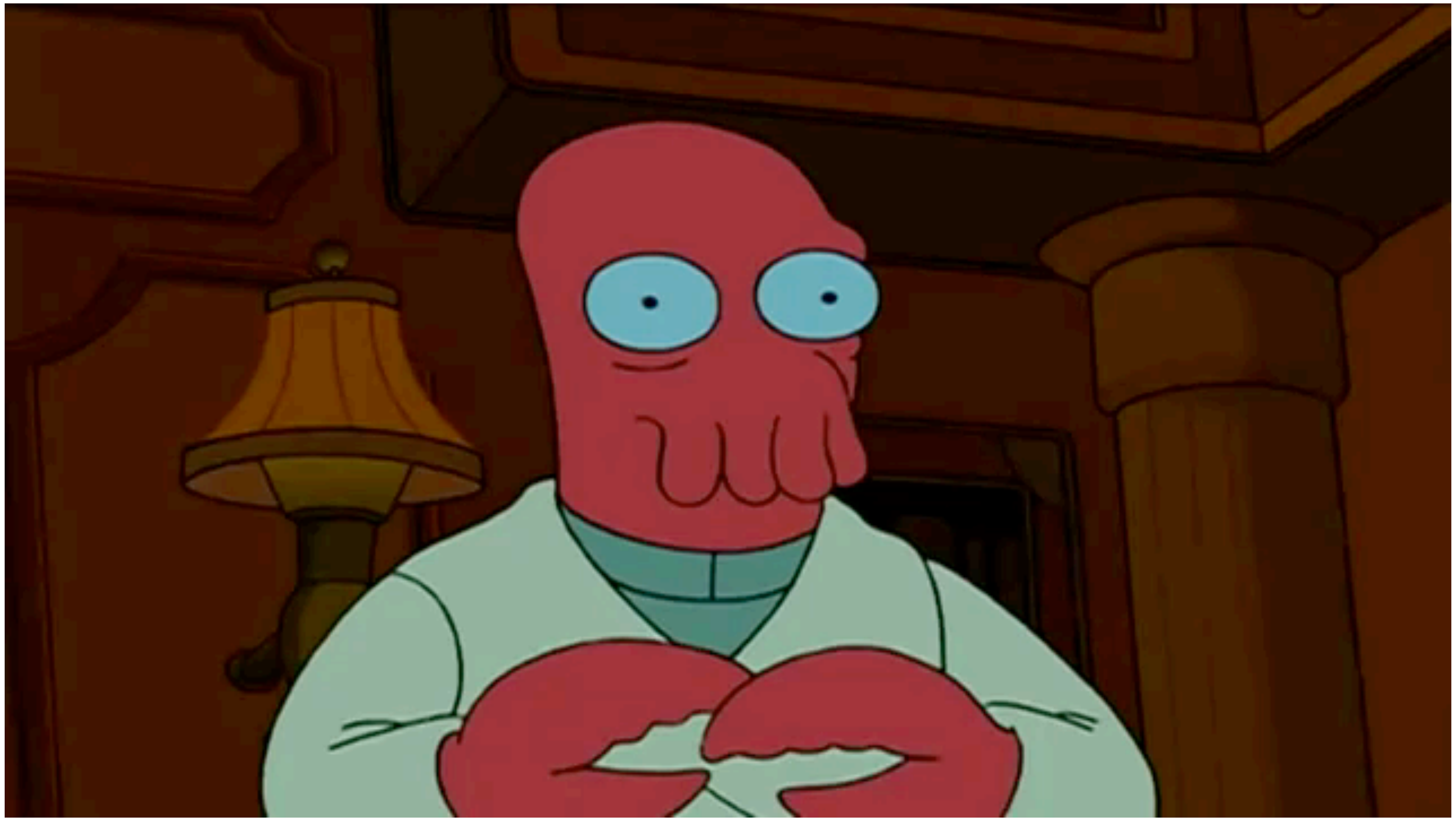
1. SysClk = 5 MHz
2. desire baud rate = 19200

$$BRD = \frac{5\text{e}6}{16 \times 19200}$$

=13.7061...

that’s no integer

hrm... this is bad:



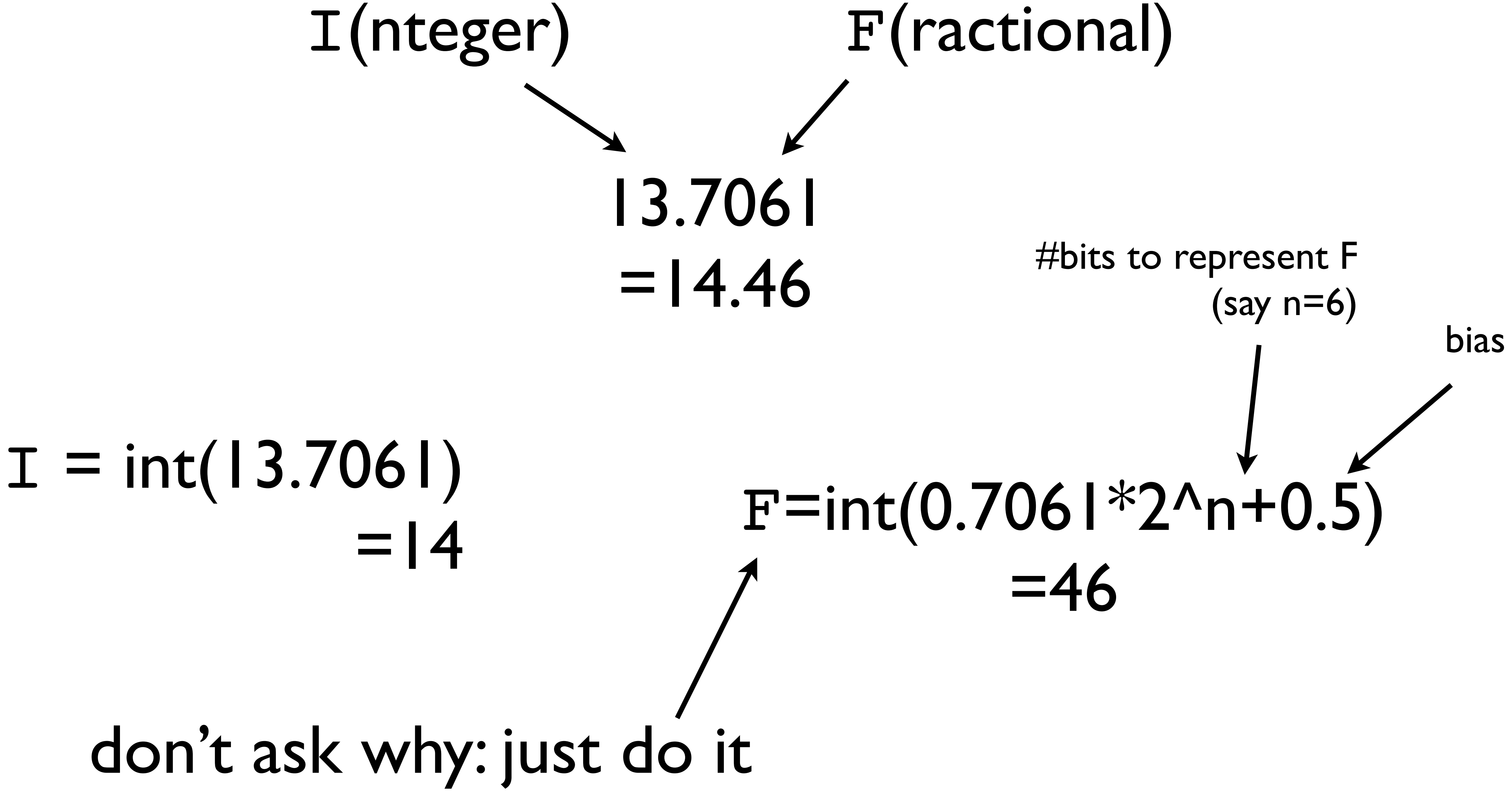
need a way to handle non-integer numbers

enter: fixed point arithmetic

actually, not so bad...

fixed point arithmetic

one register for
each part of number:



I(nteger)

F(ractional)

13.7061
=14.46

#bits to represent F
(say n=6)

bias

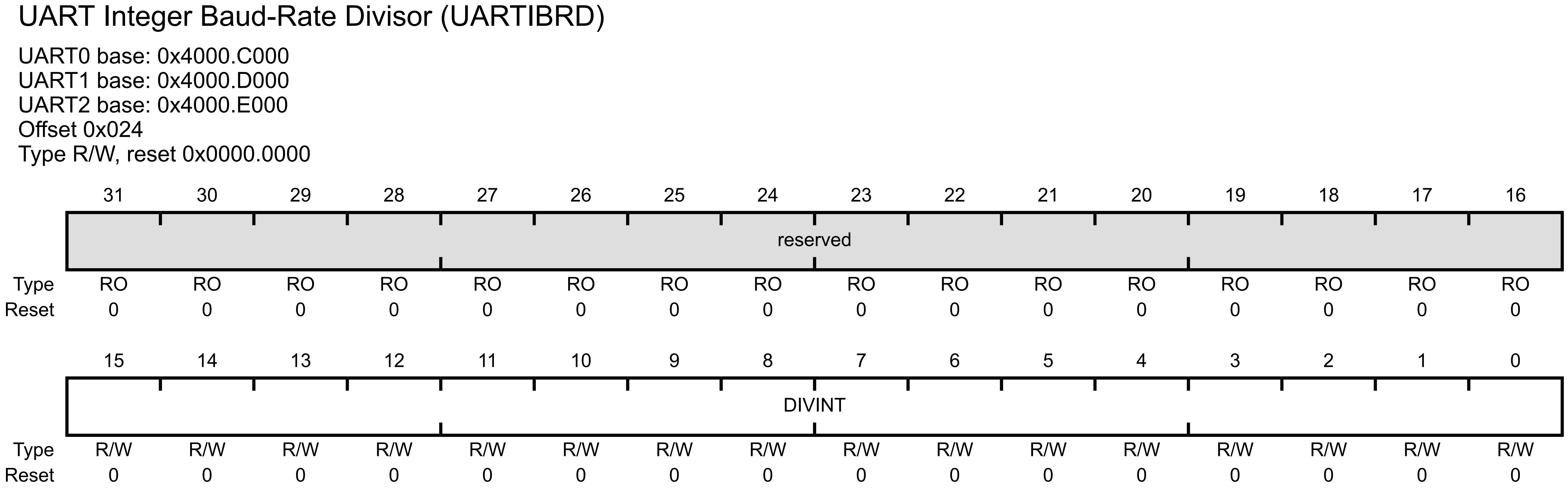
I = int(13.7061)
=14

F=int(0.7061*2^n+0.5)
=46

don’t ask why: just do it

BRD: LM3S1968

16-bits for `I`



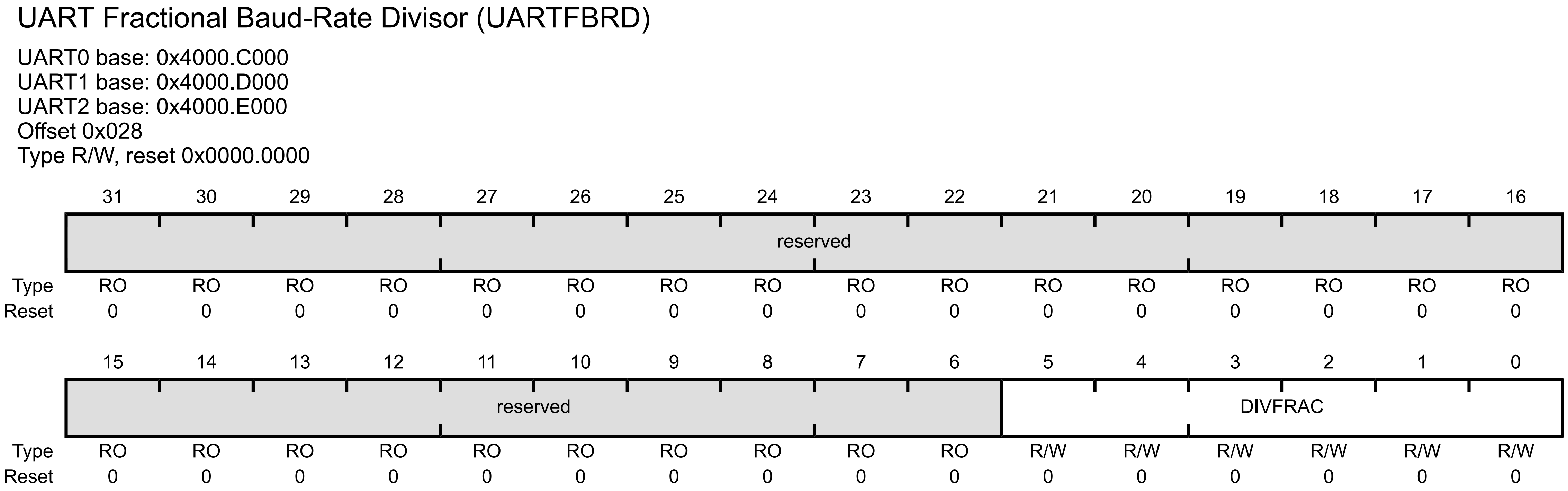
UART Integer Baud-Rate Divisor (UARTIBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x024
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	DIVINT															
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6-bits for `F`



UART Fractional Baud-Rate Divisor (UARTFBRD)

UART0 base: 0x4000.C000
UART1 base: 0x4000.D000
UART2 base: 0x4000.E000
Offset 0x028
Type R/W, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

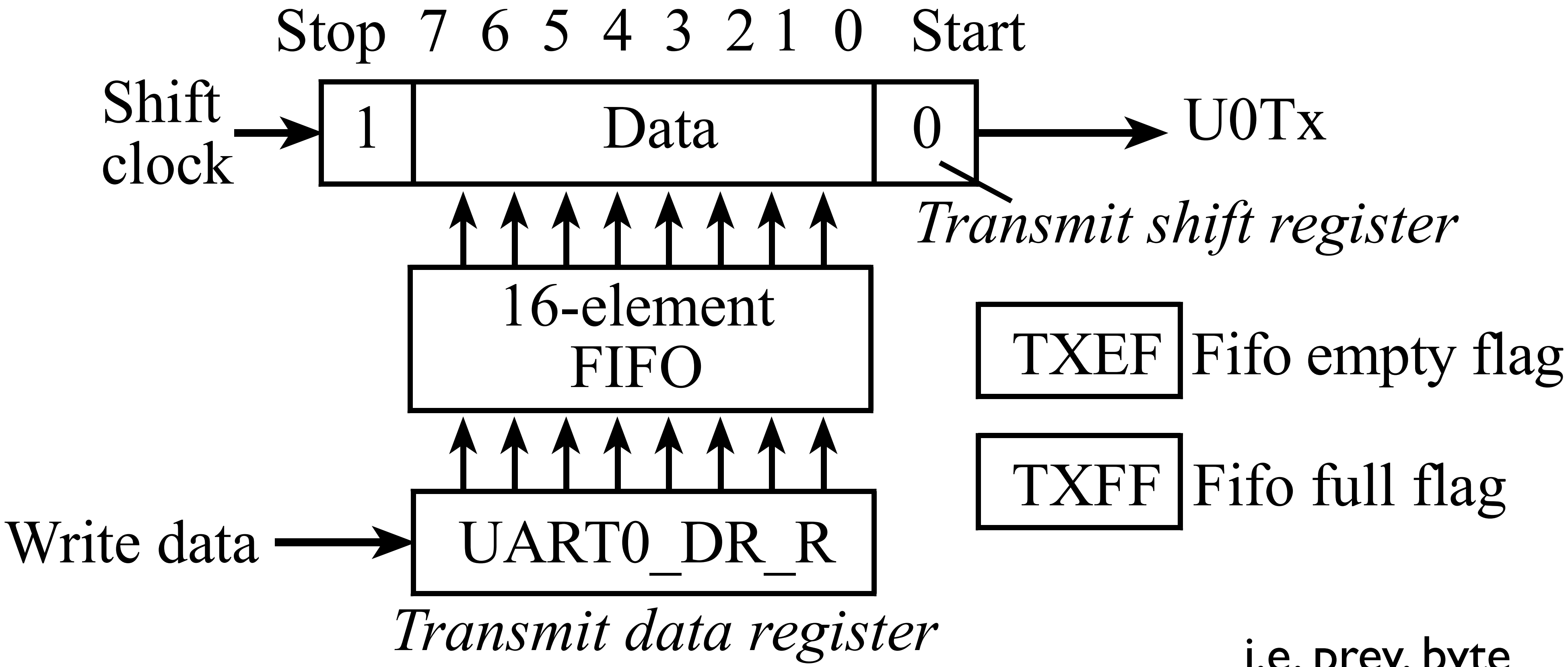
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										DIVFRAC					
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ex: 25 MHz Clock => 115,200 baud

```
; 4. set baudrate divisor
  ; BRD = 25e6/(16*115200)= 13.5634
  ; integer portion:
  ;  int(13.5634)=13=0xD
  mov R0,#0xD
  str R0,[R1,#0x24]
  ; fractional portion:
  ;  int(0.5634*2^6+0.5)=37=0x25
  mov R0,#0x25
  str R0,[R1,#0x28]
```

Q: 50 MHz clock and 1.5e6 baud

TX FIFO buffer



i.e. prev. byte not finished

write to DR register during TX:

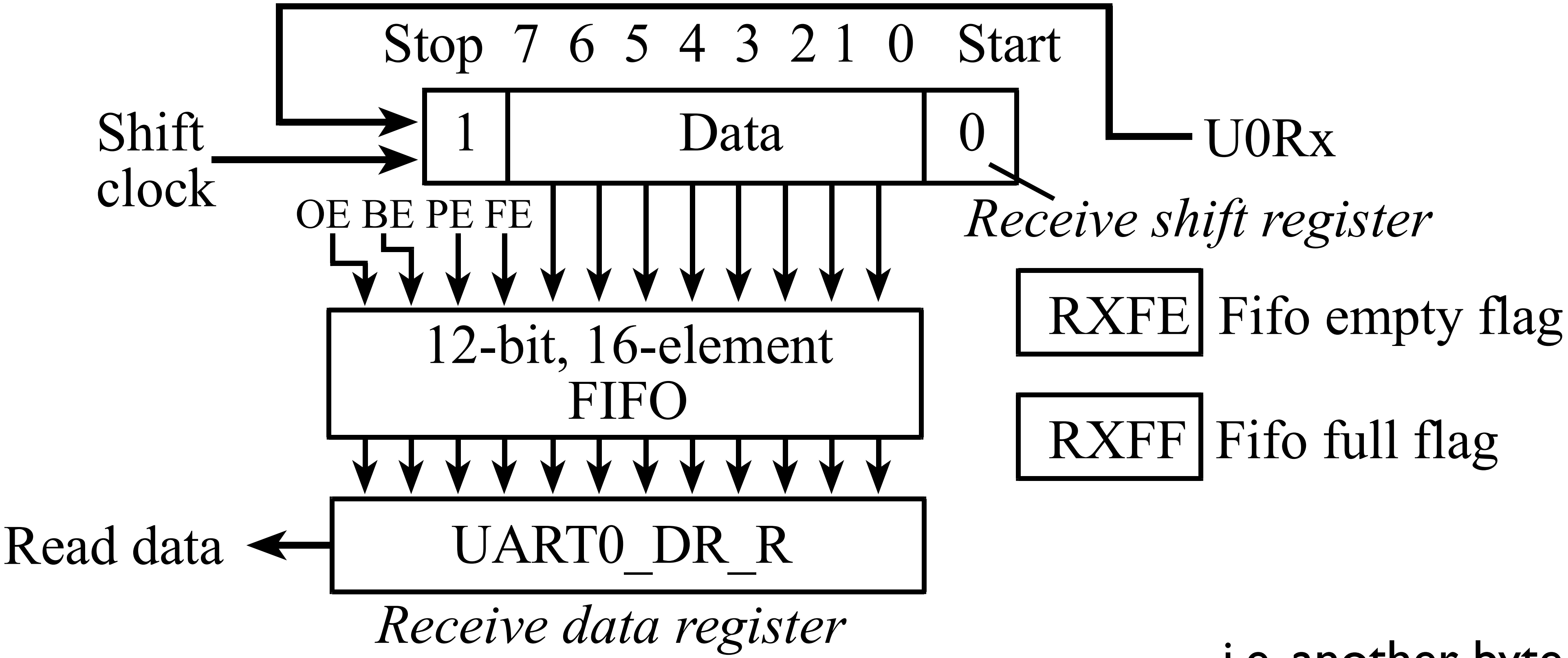
1. new data put in first-in first-out queue
2. can store up 16 bytes (can throw 16 bytes at LM3S1968 UARTs)
3. each tx automatically adjusts queue

example: TX 10101010 then 01010101 by continuously shifting 32-bit value w/buffer

```
  ;assume UART0 configure for TX
  ldr R1,=UART0
  mov R0,#0xAAAAAAAA ;what we’ll TX

TX
  ;loop until TXFF != 1
  ;  (wait until buffer isn’t full)
  ldr R2,[R1,#0x18]
  ands R2,#0x20 ;0b100000 (set Z=1 if result is 0)
  bne TX        ;branch if Z=0 (TXFF==1)

  strb R0,[R1,#0x0] ;TX first byte of R0
  ror R0,#1 ;may as well rotate as
            ;TX will take a while
  b TX
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		

RX FIFO buffer



i.e. another byte incoming

read from DR register during RX:

1. new data put in first-in first-out queue
2. can store up 16 bytes (receive 16 bytes at LM3S1968 UARTs)
3. each read automatically adjusts queue

example: RX data and put on stack w/buffer

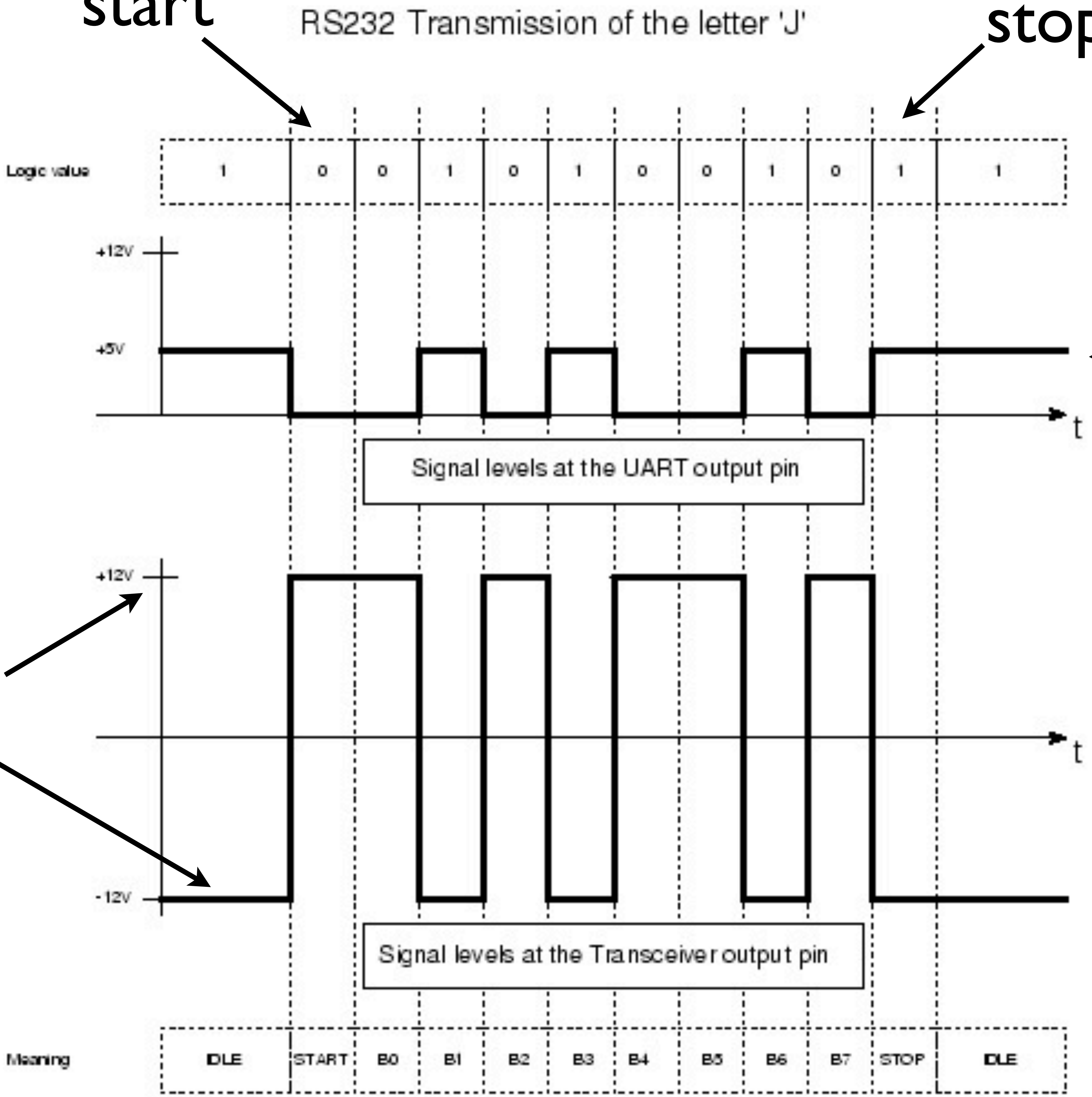
```
; assume UART0 configured for RX
  ldr R1,=UART0
  mov R0,#0x0
; RX data and put on stack
  mov R0,#0x0

RX
  ; wait for data: RXFE!=1
  ;  we have data when buffer isn't empty
  ldr R2,[R1,#0x18]
  ands R2,#0x10 ;0b00010000 (set Z=1 if result is zero)
  bne RX        ;branch if Z=0 (RXFE==1)

  ; got data put in R0 and push to stack
  ldrb R0,[R1,#0x0]
  push {R0}
  b RX ;wait for more data
```

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved								TXFE	RXFF	TXFF	RXFE	BUSY	reserved		

RS-232 vs UART

1. both serial
2. different levels
3. active low vs active high

RS-232, physically


RS232 Transmission of the letter 'J'
Logic value
1
0
0
1
0
1
0
0
1
0
1
1
+12V
+5V
t
Signal levels at the UART output pin
+12V
-12V
t
Signal levels at the Transceiver output pin
Meaning
IDLE
START
B0
B1
B2
B3
B4
B5
B6
B7
STOP
IDLE

start

stop

UART output:

idle is logic high

notice:

1. levels
2. 1 is -
   0 is +

signal on :wire

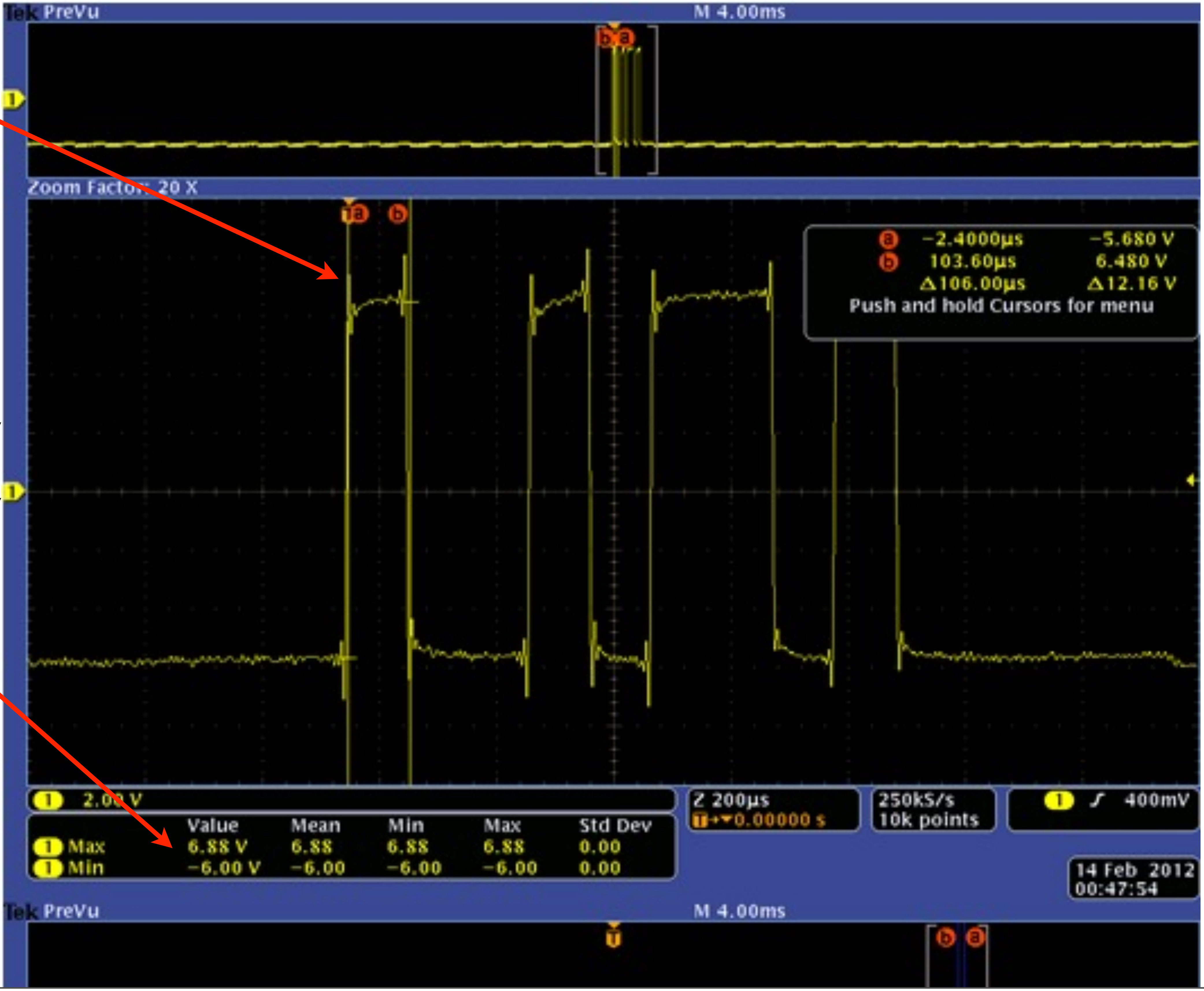
Transmitting ‘K’ (0b01001011)

(LSB first)

start bit

standard mandates:

1. max 25 V
2. +/- 3V invalid



a	−2.4000µs	−5.680 V
b	103.60µs	6.480 V
	Δ106.00µs	Δ12.16 V

Push and hold Cursors for menu

	Value	Mean	Min	Max	Std Dev
1 Max	6.88 V	6.88	6.88	6.88	0.00
1 Min	−6.00 V	−6.00	−6.00	−6.00	0.00

14 Feb 2012
00:47:54

RS-232 w/UART: how we make it happen

1. build inverter and amplifier for UART

(Op Amps...)

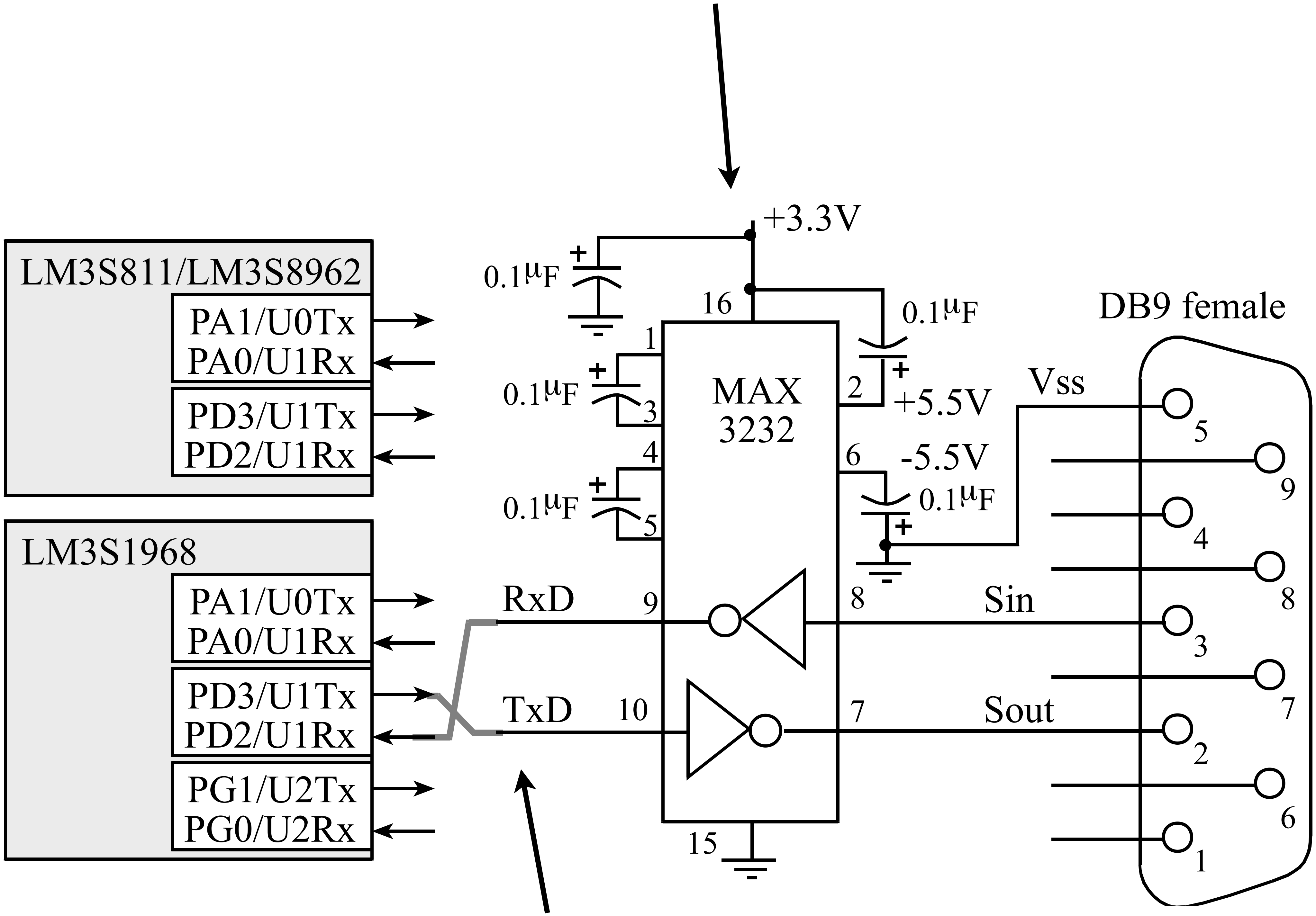
2. let someone else do it

option two →

RS-232 w/UART...

UART w/ RS-232 line driver

line driver

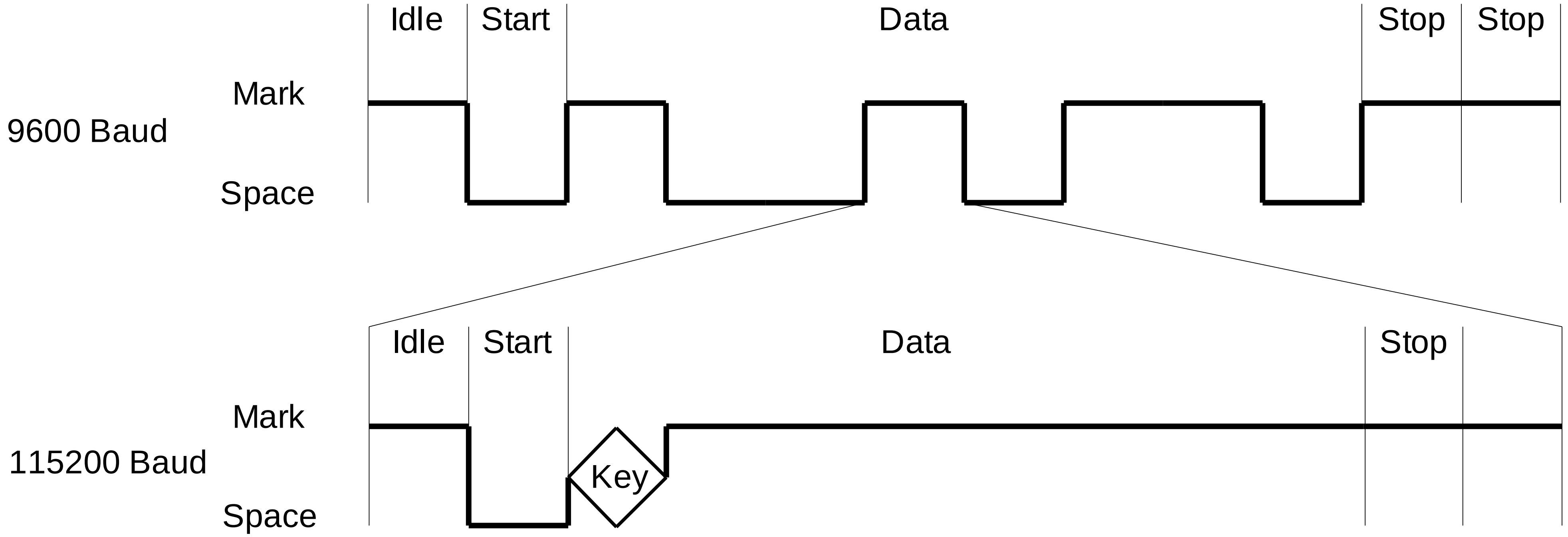


one-to-one mapping

fun with RS-232:

1. have device w/serial interface
2. want to get information off of it without anyone knowing

modifying RS232



insert data by running transmitter at higher rate