Embedded System Course 2021-22

C4µC

C/C++ Programming for Microcontrollers

Lecturer: Ing. Marco Santic

marco.santic@univaq.it

C4µC

- Introduction
  - Atmega328P
  - Toolchain
  - IDE
  - Examples
  - C language recap
- Basic functionalities
  - GPIO
  - Timers
  - Interrupt
  - Examples
- **Comm. Interfaces**
  - I2C
  - UART
  - SPI
  - Examples
- Other peripherals
  - ADC
  - GPIO bitbanging
  - Examples

C4µC - Comm. Interfaces

- Main issues you have seen:
  - Information » Symbols » Signals
  - Binary simbol: bit
  - Grouping bit in bytes...
    - Characters » Messages
  - Sender and receiver must share a scheme for in space and time dimensions
    - Parallel / Serial communcation
    - Signalizations
  - All this stuff defines a communication protocol

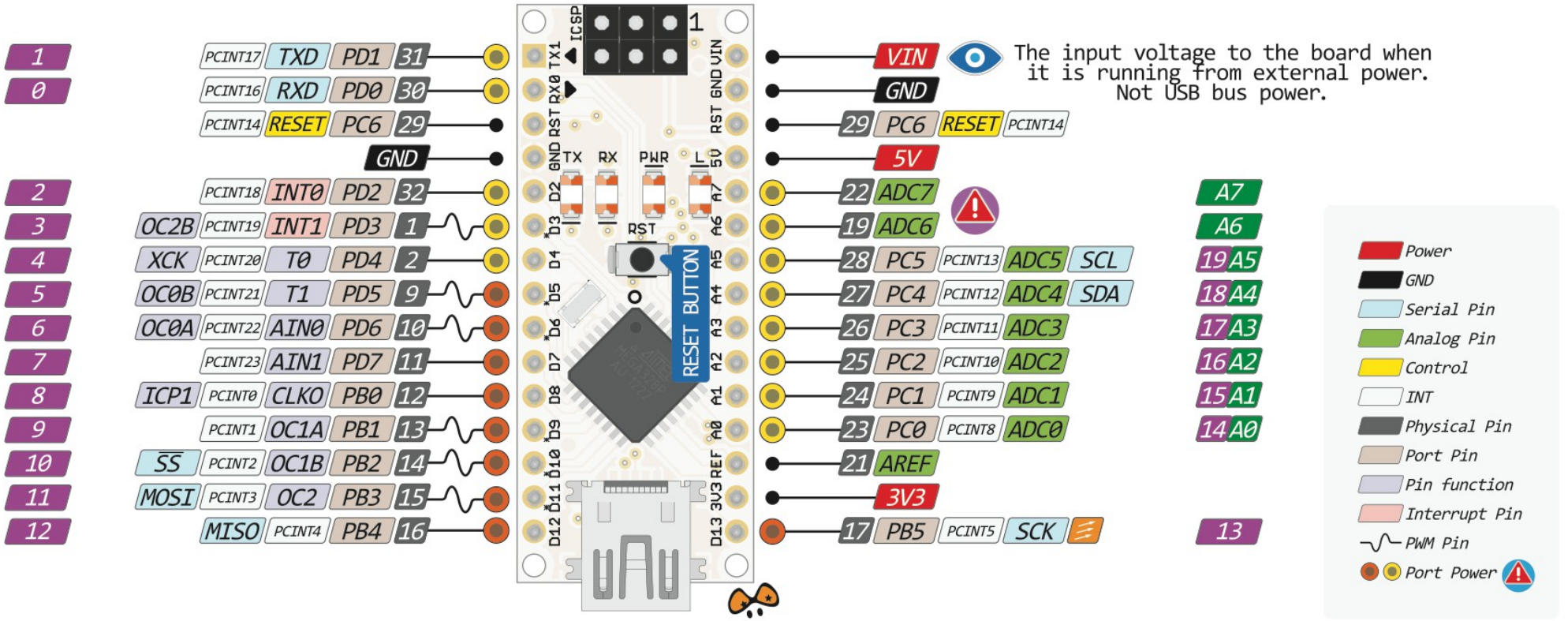
C4µC - Comm. Interfaces

- Does the Atmega328 have a parallel port?
  - Can we have a parallel communication on Arduino?
    - No specific parallel device, but we have GPIO grouped in Ports
    - Each port has a register, usually 8 bit wide
    - Also parallel communication use at least 8 bit (or it uses a multiple of 8)
  - Which ports can we access to make a test?
  - Which lines of the Atmega328 are on the pins of the board?

C4µC - Comm. Interfaces

- Arduino Nano v3 pinout

In the block of previous slides, together with schematic and Atmega328 datasheet



C4µC - Comm. Interfaces

- PORTB
  - PB0 – PB5 (6bits) , but PB5 is on-board led
- PORTC
  - PC0 – PC5 (6bits) , and PC6 is reset
- PORTD
  - PD0 – PD7 (8bits) , but we lose the UART pins Tx Rx

Look at the previous image!!!

- It is possible to use a "narrow" parallel bus? E.g:
  - 4bits , PB0 – PB3 , (4 bits » 16 symbols)
  - What can we code with 4bits instead of 8bits?

C4µC - Comm. Interfaces

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

What can we do to send ASCII characters?

C4µC - Comm. Interfaces

- We could send 4bits at a time, then compose the character again in 2 data transfers

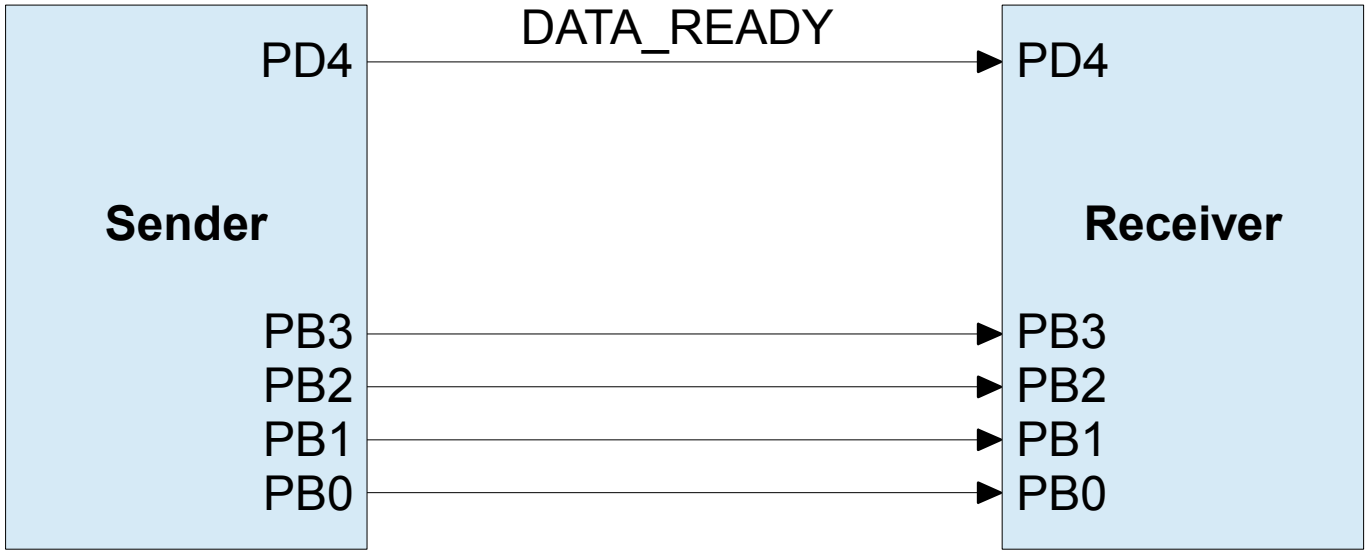
```
// Example sender, it writes

#define BUS_MASK 0x0F

uint8_t stw; //symbol to write

// In setup, set bus as out
DDRB |= BUS_MASK;

// data transfer 1
PORTB &= ~BUS_MASK;
PORTB |= stw & BUS_MASK; //data ready

// data transfer 2
PORTB &= ~BUS_MASK;
PORTB |= (stw >> 4) & BUS_MASK; //d.r.
```

```
// Example receiver, it reads

#define BUS_MASK 0x0F

uint8_t str; //symbol to read

// In setup, set bus as in
DDRB &= ~BUS_MASK;

// data transfer 1, if data ready
str = PINB & BUS_MASK;
str = (str << 4) & ~BUS_MASK;

// data transfer 2, if data ready
str |= PINB & BUS_MASK;
```

Yes, you can

C4µC - Comm. Interfaces

- How sender and receiver agree when data is ready?
- Let's suppose the receiver is always ready...
  - 1 additional signal DATA_READY (set by the sender)



C4µC - Comm. Interfaces

- Open the Arduino IDE (sketch_c4uc_5.1_parallel_sender)

```
// include only avr libraries
#include <avr/io.h>
#include <util/delay.h>

#define BUS_MASK 0x0F

// message to send
uint8_t mts[] = {'H','e','l','l','o',' ','w','o','r','l','d','!'}; // 12 x 2 sym
const uint8_t msg_len = 24; // in symbols
uint8_t wi = 0;  // write index

void send_function();

void setup() {
  PORTD &= ~_BV(PORTD4);    //clear data ready
  PORTB &= ~BUS_MASK; //clear the bus
  DDRD |= _BV(DDD4); // DATA_READY pin at PD4
  DDRB |= BUS_MASK;  // our bus at PB0 to PB3
  DDRB |= _BV(DDB5); // use the LED on PB5
}

void loop() {
  send_function();
  _delay_ms(400); // to see the different transfers on blink
}
```

C4µC - Comm. Interfaces

- Open the Arduino IDE (sketch_c4uc_5.1_parallel_sender)

```
// the function to send symbols...
void send_function() {
  if(wi<msg_len){
    uint8_t mi = wi >> 1;
    uint8_t stw = (wi % 2 == 0) ? mts[mi] : mts[mi] >> 4; //local var symbol
    PORTB &= ~BUS_MASK; //clear the bus
    PORTB |= stw & BUS_MASK; //write the symbol
    PORTD |= _BV(PORTD4);    //signal data ready

    //short blink on led
    PORTB |= _BV(PORTB5);
    _delay_ms(50);
    PORTB &= ~_BV(PORTB5);
    _delay_ms(50);

    wi++;                     //increment write index
    PORTD &= ~_BV(PORTD4);    //clear data ready
  }
}
```

Compile and run: how many blink do you expect?

C4µC - Comm. Interfaces

- Open the Arduino IDE (sketch_c4uc_5.2_parallel_receiver)

```
// include only avr libraries
#include <avr/io.h>
#include <util/delay.h>
#define BUS_MASK 0x0F
#define dbg

uint8_t mtc[] = {'H','e','l','l','o',' ','w','o','r','l','d','!'}; // to check
uint8_t msg[13]; // message received (note: 13 for null terminator)
const uint8_t msg_len = 24; // in symbols
uint8_t ri = 0;  // read index
uint8_t reading = 1;  // start in reading state

void receive_function();
void check_function();

void setup() {
#ifdef dbg
  Serial.begin(115200);
  while(!Serial);
  msg[12] = '\0';
#endif
  DDRD &= ~_BV(DDD4); // DATA_READY at pin PD4
  DDRB &= ~BUS_MASK;  // our bus at PB0 to PB3
  DDRB |= _BV(DDB5);  // use the LED on PB5
}
```

Note: #define dbg

C4µC - Comm. Interfaces

- Open the Arduino IDE (sketch_c4uc_5.2_parallel_receiver)

```
void loop() {
  receive_function();
  check_function();
}

// the function to receive symbols...
void receive_function() {
  if(ri<msg_len){
    if(PIND & _BV(PIND4)){ //read data ready signal
      uint8_t str = PINB & BUS_MASK;  //read the symbol
#ifdef dbg
      Serial.println(str,HEX);
#endif
      uint8_t mi = ri >> 1;
      msg[mi] |= (ri % 2 == 0) ? str : str << 4;

      // 3 short blinks on led
      PORTB |= _BV(PORTB5); _delay_ms(25);  PORTB &= ~_BV(PORTB5); _delay_ms(25);
      PORTB |= _BV(PORTB5); _delay_ms(25);  PORTB &= ~_BV(PORTB5); _delay_ms(25);
      PORTB |= _BV(PORTB5); _delay_ms(25);  PORTB &= ~_BV(PORTB5); _delay_ms(25);

      ri++;                          //increment read index
    }
  }
}
```

C4µC - Comm. Interfaces

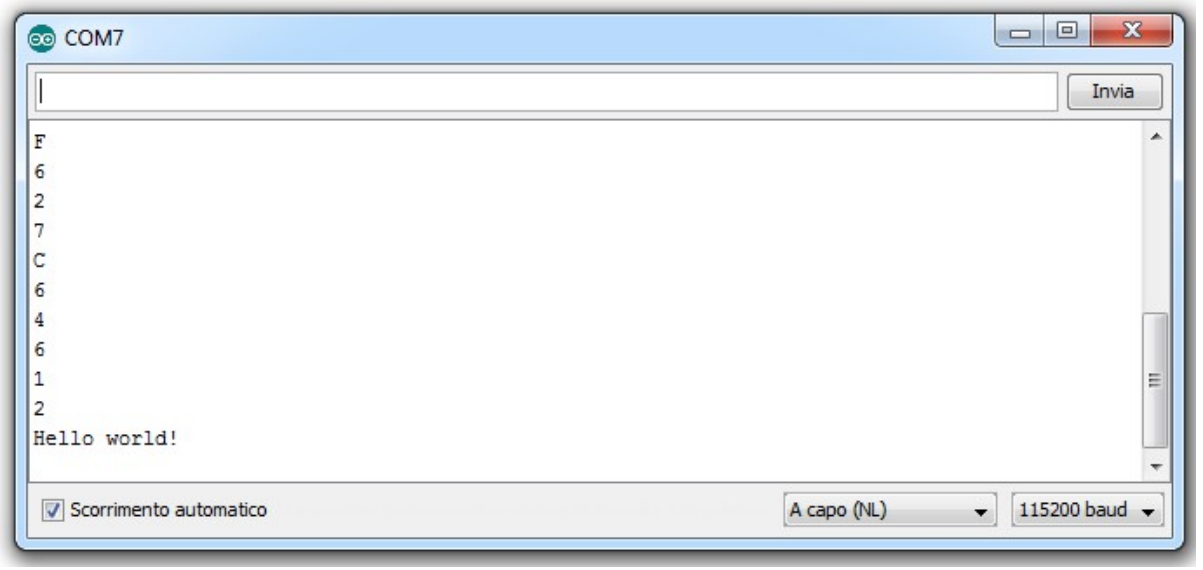
- Open the Arduino IDE (sketch_c4uc_5.2_parallel_receiver)

```
void check_function() {
  if(ri==msg_len){
#ifdef dbg
    Serial.println((char *)msg);
#endif
    _delay_ms(500);
    uint8_t ok = 1;
    for(uint8_t i; i<(msg_len>>1); i++){
      ok &= (msg[i] == mtc[i]);
    }
    if(ok){
      while(1){ // slow blinks forever
        PORTB |= _BV(PORTB5); _delay_ms(1000); PORTB &= ~_BV(PORTB5); _delay_ms(1000);
      }
    } else {
      while(1){ // ERROR: fast blinks forever
        PORTB |= _BV(PORTB5); _delay_ms(25);  PORTB &= ~_BV(PORTB5); _delay_ms(25);
      }
    }
  }
}
```

Compile and run: how many blink do you expect?
Connect all the wires (Bus+data_ready+GND) and test
Test commenting //#define dbg

C4µC - Comm. Interfaces

- Connect 2 devices with wires
- Reset first Receiver, then Sender
- If dbg is defined, reset the Receiver closing and re-opening the serial monitor
  - We should see an output like the one in figure



C4µC - Comm. Interfaces

- Exercise: let's write the same apps with Arduino API
  - sketch_c4uc_5.3_parallel_sender_AAPI
  - sketch_c4uc_5.4_parallel_receiver_AAPI

Let's code together step-by-step

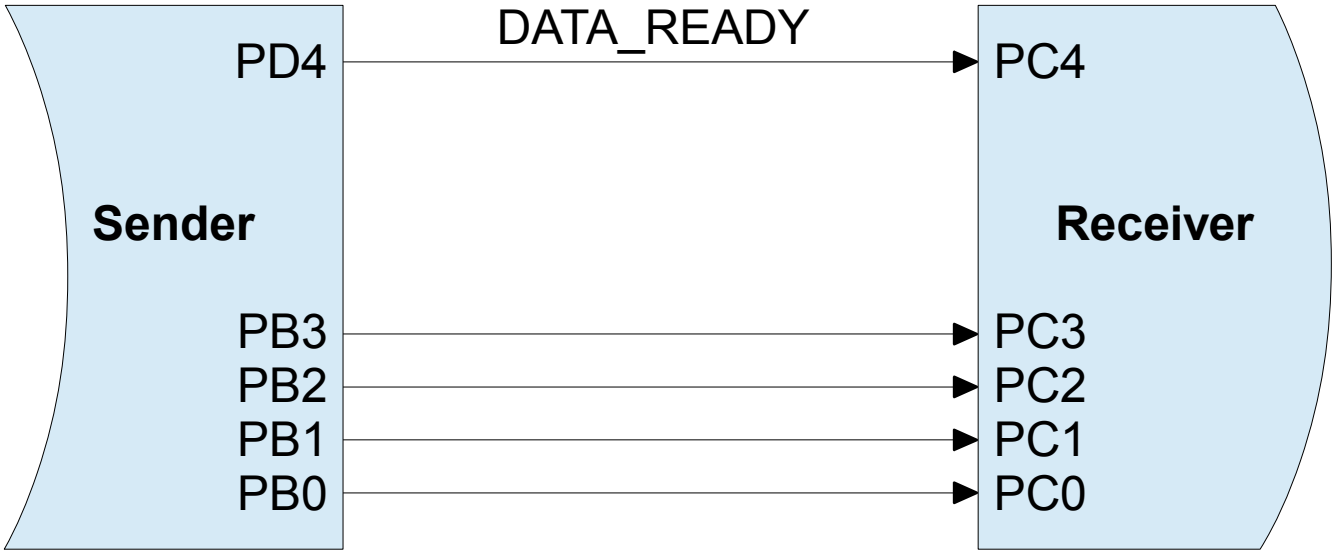
C4µC - Comm. Interfaces

- We have put these 3 functions (send, receive, check) on 2 devices
- What if we have only one Arduino?
- Can we put these 3 function on it?
  - Let's change the function name... in task (only nomenclature)

```
void loop() {
  send_task();
  receive_task();
  check_task();
}
```

C4µC - Comm. Interfaces

- We have to choose other pins for the receiver
  - Port C has some free pins
    (verify that they can be digital on datasheet)



C4µC - Comm. Interfaces

- Exercise: let's write the app with Arduino API
  - sketch_c4uc_5.5_parallel_loopback_AAPI

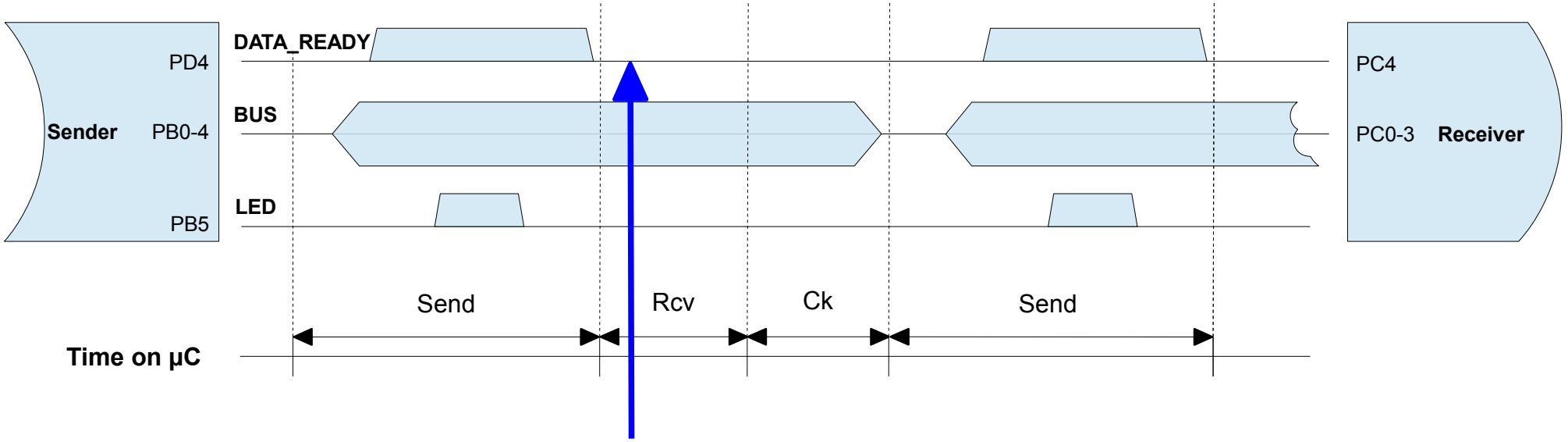
Let's code together step-by-step

```
void loop() {
  send_task();
  receive_task();
  check_task();
}
```

How many blinks we can expect?
Does the check is OK or KO?

C4µC - Comm. Interfaces

- The code cannot work: we are on wrong assumptions
  - The receiver is not ready (not "active"), not when the sender is transmitting....
  - Have a look at timing diagrams



The receive_task will always find DATA_READY LOW

Exercise: use PD2 instead of PC4 with interrupt INT0 to call receive_task (three lines of code to modify...): sketch_c4uc_5.6_parallel_loopback_AAPI_INT

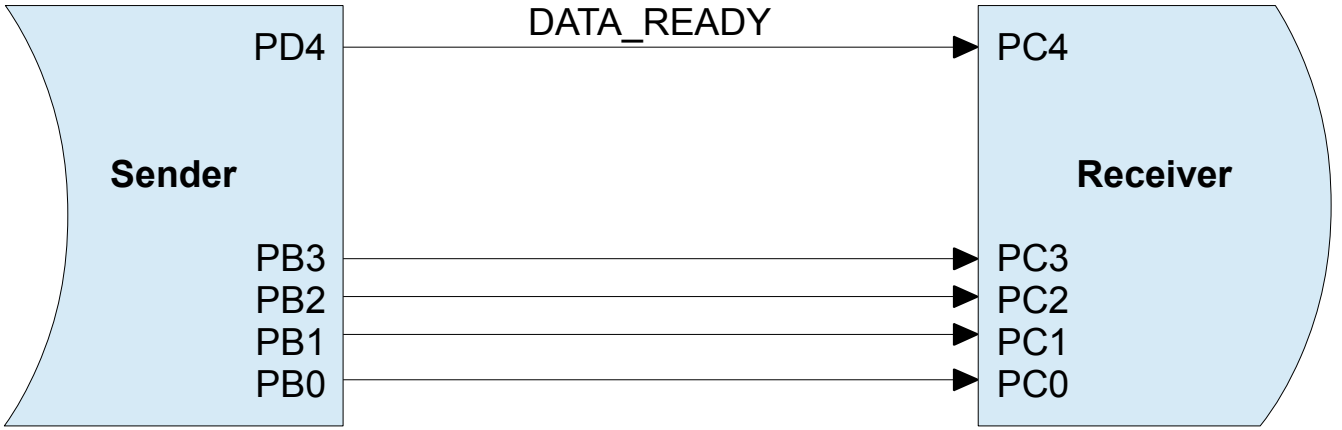
C4µC - Comm. Interfaces

- Other issues can be analised:
  - On 2 different devices, if we cannot assume that receiver is ready, we have to check it
  - Or again on the same device... how can we solve without interrupt?
  - What if tasks:
    - are not in order?
    - are called randomly and repeatedly?
      E.g.: rcv, rcv, ck, snd, rcv, snd, snd, ...

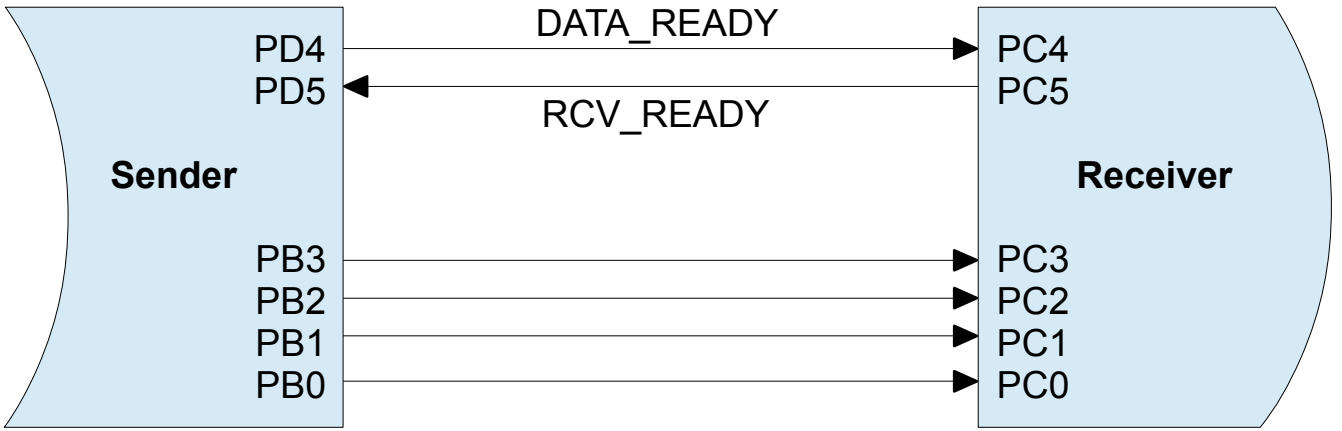
- What could fail?
  - Discuss the assumptions

C4µC - Comm. Interfaces

- We have assumed "*receiver is always ready*"

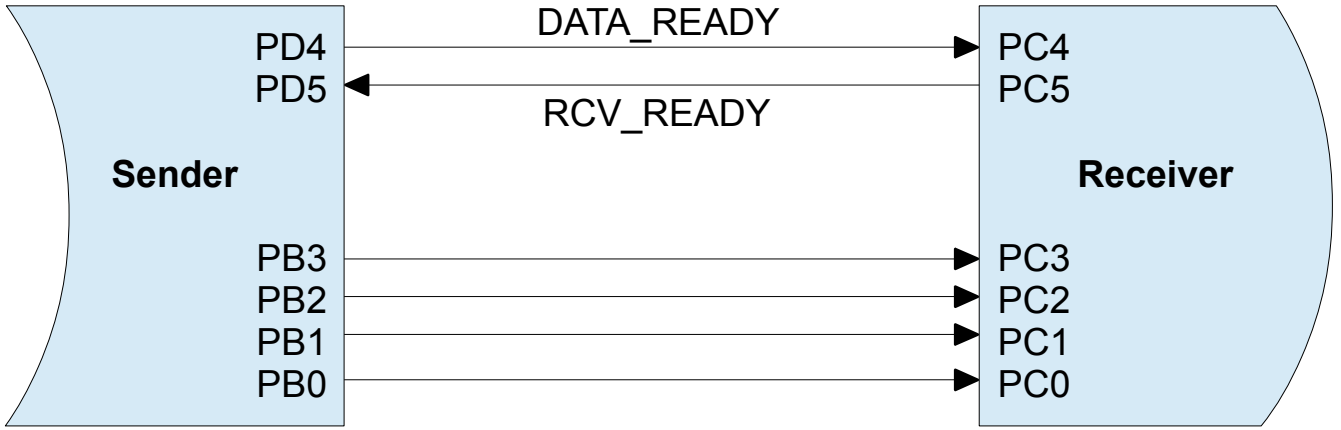


- We need a 2nd signal to tell that the rcv is ready (or just has done reading)



C4µC - Comm. Interfaces

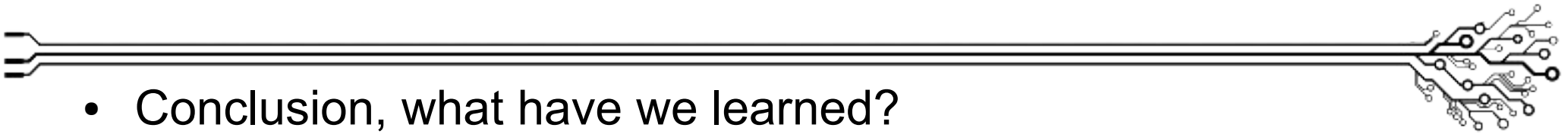
- This is a synchronization scheme



- We have named **DATA_READY** and **RCV_READY** signals,
  - They are often named **REQ** and **ACK**
- The *must* is to have the documentation, with description, possibly operation flowchart and with timing diagrams
- Let's make a timing diagram of our parallel device...

Exercise: try to implement the protocol with REQ and ACK

C4µC - Comm. Interfaces

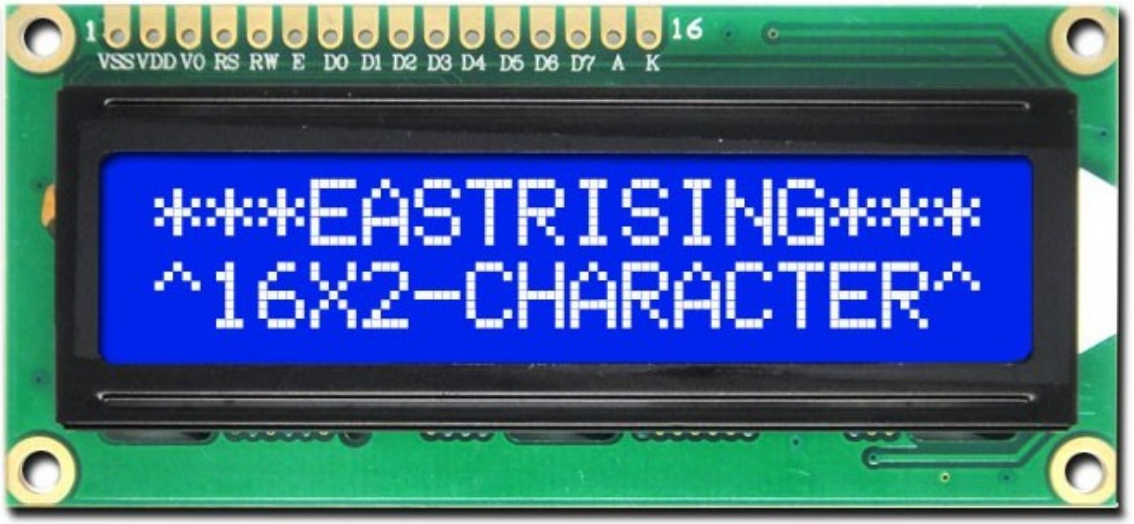


- Conclusion, what have we learned?
  - Coding know-how, understanding bit and mask operation on registers
  - New small comm protocol
    - Synchronization issues
    - Data buffering (our `msg[]` is de facto a rcv buffer)
  - Multitasking??
    - The µC is monothread, but we can split the tasks and functionalities, when they have different priorities... (this is only a preview of an advanced topic)

- What can we do with this 4-bit bus?
  - Coming soon: HD44780

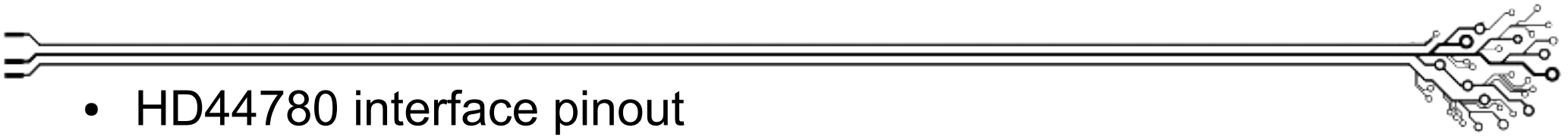
C4µC - Examples

- HD44780 is the driver chip at the base of LCD displays like:



- The Hitachi HD44780 LCD controller is an alphanumeric dot matrix liquid crystal display (LCD) controller developed by Hitachi; it was made commercially available around year 1987. (wikip-eng)
- Due to its spread the HD44780 can be considered a de facto standard to control LCD. It is used often in embedded systems to implement displays for electrical and hobbystic equipment, thanks mainly to its wide versatility and ease of interfacing. (wikip-it)

C4µC - Examples



- HD44780 interface pinout

1) Ground
2) VCC (+3.3 to +5V)
3) Contrast adjustment (VO)
4) Register Select (RS). RS=0: Command, RS=1: Data
5) Read/Write (R/W). R/W=0: Write, R/W=1: Read (This pin is optional due to the fact that most of the time you will only want to write to it and not read. Therefore, in general use, this pin will be permanently connected directly to ground.)
6) Clock (Enable). Falling edge triggered
7) Bit 0 (Not used in 4-bit operation)
8) Bit 1 (Not used in 4-bit operation)
9) Bit 2 (Not used in 4-bit operation)
10) Bit 3 (Not used in 4-bit operation)
11) Bit 4
12) Bit 5
13) Bit 6
14) Bit 7
15) Backlight Anode (+) (If applicable)
16) Backlight Cathode (-) (If applicable)

IT CAN BE USED IN 4bit DATA-MODE!!!

C4µC - Examples

- HD44780: let's analise some libraries...
  - Examples --> LiquidCrystal --> HelloWorld

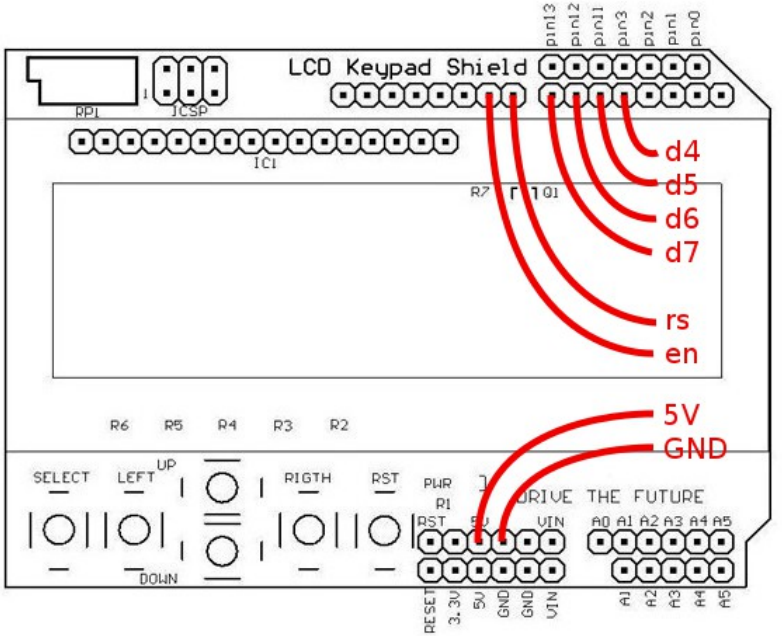
```
/* This sketch prints "Hello World!" to the LCD and shows the time.

  The circuit:
 * LCD RS pin to digital pin 12
 * LCD Enable pin to digital pin 11
 * LCD D4 pin to digital pin 5
 * LCD D5 pin to digital pin 4
 * LCD D6 pin to digital pin 3
 * LCD D7 pin to digital pin 2
 * LCD R/W pin to ground
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 * 10K resistor:
 * ends to +5V and ground
 * wiper to LCD VO pin (pin 3)
 ...
*/

// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
```

Explore the sketch's code and library

C4µC - Examples

- HD44780 and keypad (http://linksprite.com/wiki/index.php5?title=16_X_2_LCD_Keypad_Shield_for_Arduino)
  - Our connections

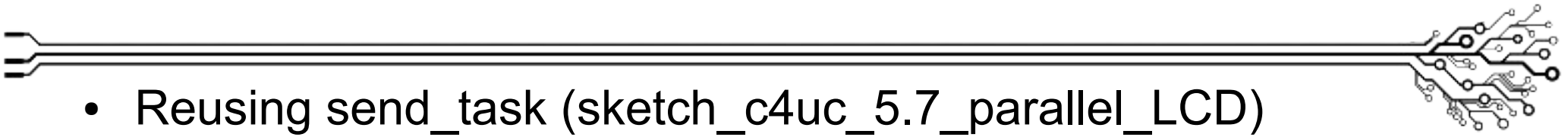
```
/* This sketch ...

  The circuit:
 * LCD RS pin to digital pin 5
 * LCD Enable pin to digital pin 4
 * LCD D4 pin to digital pin 8
 * LCD D5 pin to digital pin 9
 * LCD D6 pin to digital pin 10
 * LCD D7 pin to digital pin 11
 * LCD VSS pin to ground
 * LCD VCC pin to 5V
 ...
*/

// include the library code:
#include <LiquidCrystal.h>

//               (rs, e, d4, d5, d6, d7);
LiquidCrystal lcd( 5, 4,  8,  9, 10, 11);
```



Test the sketch with our connections

The sketch uses 2.330 byte (7%)...

C4µC - Examples



- Reusing send_task (sketch_c4uc_5.7_parallel_LCD)

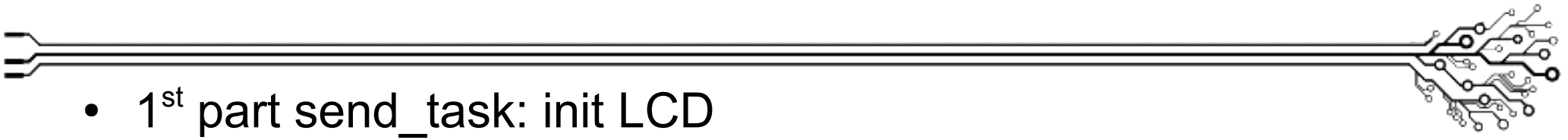
```
// include only avr libraries
#include <avr/io.h>
#include <util/delay.h>

#define BUS_MASK 0x0F
//#define NO_API

uint8_t mts[] = {'E','m','b','e','d','d','e','d',' ',
'S','y','s','t','e','m','s','\n','C','4','u','C'}; //21 x 2 sym
uint8_t minit[] = {0x33, 0x32, 0x28, 0x0C, 0x06, 0x01}; //6 x 2 sym

//              DB: 7654 3210
//                  |||| ||||
// $33 8-bit mode   xx11 0011  <-- Init sequence to switch to 4bit mode...
// $32 8-bit mode   xx11 0010  <-- ... three times 0x11 and one 0x10
// $28 8-bit mode   xx10 1000  <-- Function set: 4bit bus, 2-line, 5x8 dots
// $0C 8-bit mode   xx00 1100  <-- Display ctrl: display ON, cursor OFF, blink OFF
// $06 8-bit mode   xx00 0110  <-- Entry mode: cursor moves to right
// $01 8-bit mode   xx00 0001  <-- Clear display

const uint8_t msg_len = 42; // in symbols
const uint8_t init_len = 12; // in symbols
uint8_t wi = 0;  // write index
uint8_t ii = 0;  // init index
```

C4µC - Examples

- Try to exclude API setup() and loop()...

```
void send_task();

#ifndef NO_API
void setup() {
#else
int main() {
#endif
  DDRD |= _BV(DDD4); // Enable
  DDRD |= _BV(DDD5); // Register Select
  DDRB |= BUS_MASK;
  PORTB &= ~BUS_MASK; //clear the bus

#ifdef NO_API
  while(1){
    send_task();
  }
#endif
}

#ifndef NO_API
void loop() {
  send_task();
}
#endif
```

C4µC - Examples



- 1st part send_task: init LCD

```
// the function to send symbols...
void send_task() {
  if(ii<init_len){
    uint8_t mi = ii >> 1;
    uint8_t stw = (ii % 2 == 0) ? minit[mi] >> 4 : minit[mi]; //local var symbol

    PORTD &= ~_BV(PORTD5);    // cmd_mode rs=0
    PORTB &= ~BUS_MASK;       //clear the bus
    PORTB |= stw & BUS_MASK; // write the symbol

    //short blink on led
    PORTB |= _BV(PORTB5); _delay_ms(50); PORTB &= ~_BV(PORTB5); _delay_ms(50);

    PORTD |= _BV(PORTD4);     // signal data on bus, enable=1
    _delay_ms(1);
    PORTD &= ~_BV(PORTD4);    // enable=0

    ii++;                     // increment init index
    return;
  }
```

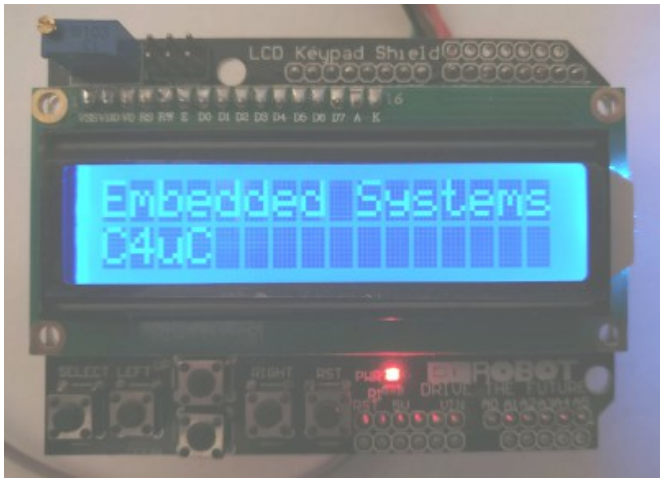
C4µC - Examples

- 2nd part send_task: send message

```
 if(wi<msg_len){
   uint8_t mi = wi >> 1;

   if(mts[mi]=='\n'){ // small workaround to write next line cmd during chars tx
     wi += 2;
     ii -= 2;
     minit[ii>>1] = 0xC0; // next line command
     return;
   }

   uint8_t stw = (wi % 2 == 0) ? mts[mi] >> 4 : mts[mi]; //local var symbol

   PORTD |= _BV(PORTD5);     // char_mode rs=1
   PORTB &= ~BUS_MASK;       // clear the bus
   PORTB |= stw & BUS_MASK; // write the symbol

   //short blink on led
   PORTB |= _BV(PORTB5); _delay_ms(50); PORTB &= ~_BV(PORTB5); _delay_ms(50);

   // short pulse on enable...
   PORTD |= _BV(PORTD4); _delay_ms(1); PORTD &= ~_BV(PORTD4);

   wi++;                     // increment write index
 }
} //end of send_task
```

C4µC - Examples

- Result



- Our sketch occupies only 772 bytes (2%)
- Uncomment **`#define NO_API`**
- Recompile and see:
  - *The sketch uses 474 byte (1%)*
  - The code can be further optimized and organized

Ex.: try to organize functions as in the library