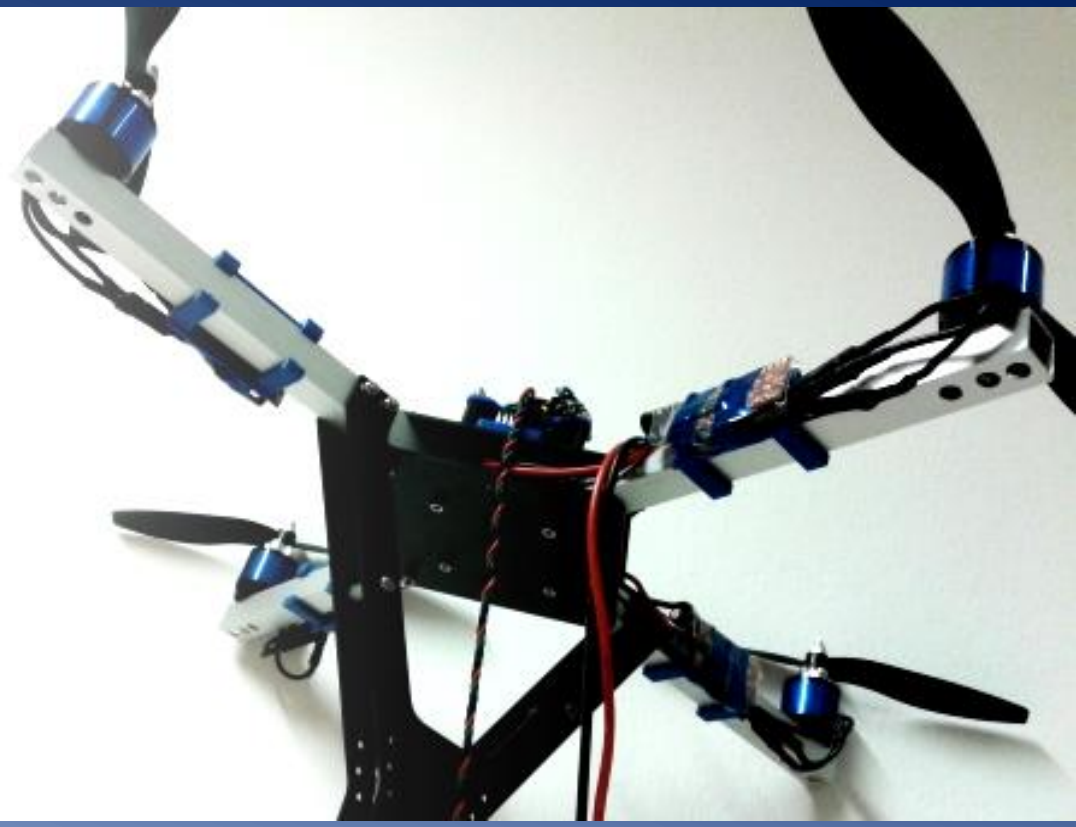
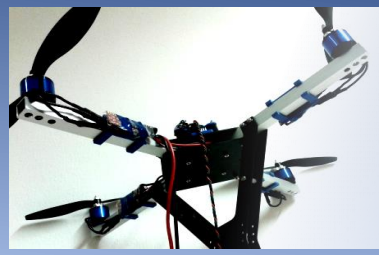
Introduction

DIP & AVR EVK1100

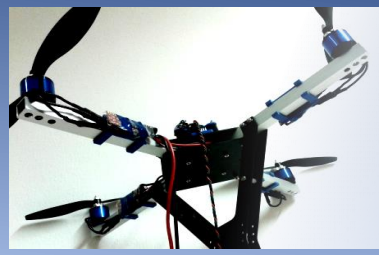




Content

Time scope: 2-4h

- Display and Buttons (DIP)
- DIP (Driver, Code)
- Exercises



Display and Buttons (DIP)

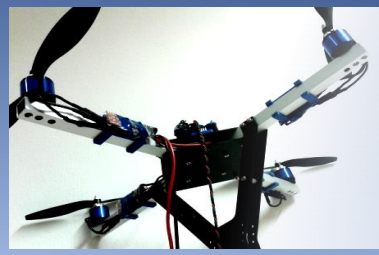
Display:

- DIP204
- 20 Columns x 4 Rows (80 Symbols)
- Display for information and values (variables)
- Example code:

```
char line[20];
int value = 3;
sprint(line,„Value is:%i ", value);
dip204_set_cursor_position(x,y);
dip204_write_string(line);
```

- Sprintf Parameter analog to printf.

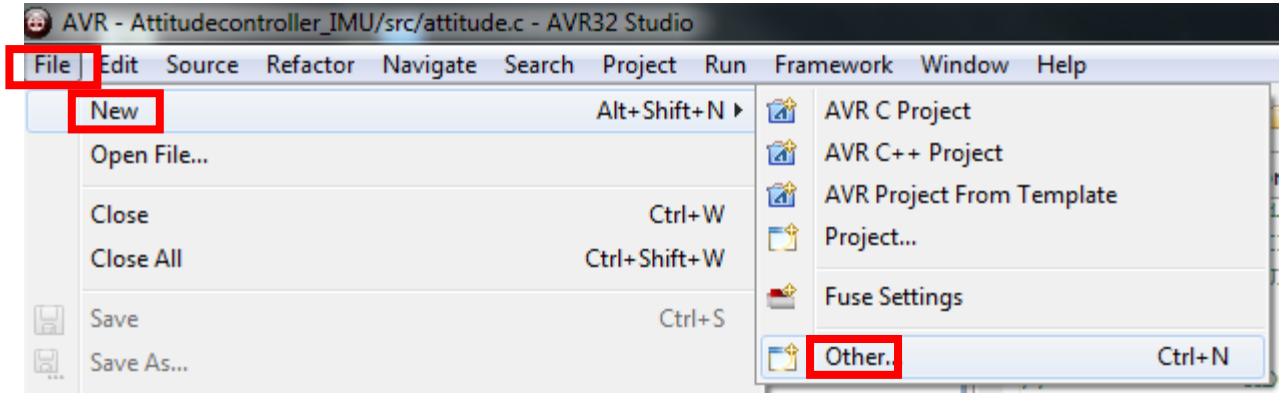
%	Function	Example
d or i	Signed decimal integer	392
u	Unsigned decimal integer	7235
o	Unsigned octal	610
x	Unsigned hexadecimal integer	7fa
X	Unsigned hexadecimal integer (uppercase)	7FA
f	Decimal floating point, lowercase	392.65
F	Decimal floating point, uppercase	392.65
e	Scientific notation (mantissa/exponent), lowercase	3.9265e+2
E	Scientific notation (mantissa/exponent), uppercase	3.9265E+2
g	Use the shortest representation: %e or %f	392.65
G	Use the shortest representation: %E or %F	392.65

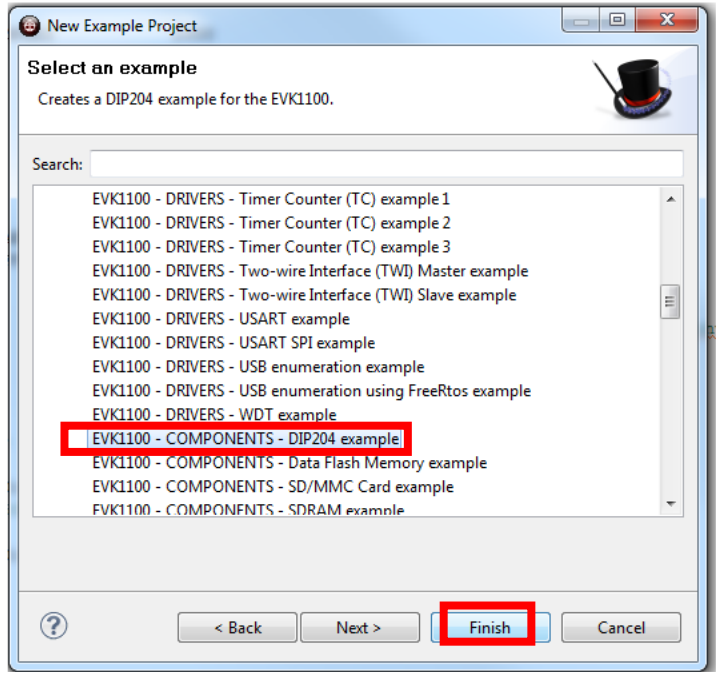


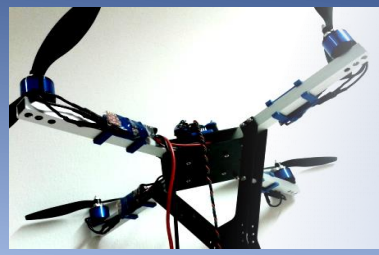
Display and Buttons (DIP)

Lets go!

- Create example project: File -> New -> Other -> AVR Example Project
- Choose DIP 204 example: -> AT32UC3A0512 -> EVK 1100 COMPONENTS DIP204 example





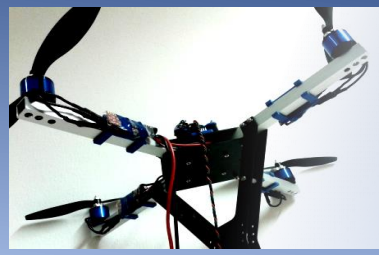


Display and Buttons (DIP)

Buttons

- EVK1100 has 3 Push-Buttons (PB0,PB1, PB2) and 1 Joystick button (Joy)
- Buttons generate Interrupt
- Code in Interrupt-Service-Routine (ISR)
- ISR should be kept as short as possible
- „Short“ could be the setting of a flag
- Implement more complex functions in the main program
- ISR for Joy: dip204_example_Joy_int_handler()

```
/*!
 * \brief The joystick interrupt handler.
 */
#if __GNUC__
__attribute__((__interrupt__))
#elif __ICCAVR32__
__interrupt
#endif
static void dip204_example_Joy_int_handler(void)
{
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_UP))
  {
    dip204_set_cursor_position(19,1);
    dip204_write_data(0xDE);
    display = 1;
    /* allow new interrupt : clear the IFR flag */
    gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_UP);
  }
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_DOWN))
  {
    dip204_set_cursor_position(19,3);
    dip204_write_data(0xE0);
    display = 1;
    /* allow new interrupt : clear the IFR flag */
    gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_DOWN);
  }
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_LEFT))
  {
    dip204_set_cursor_position(18,2);
    dip204_write_data(0xE1);
    display = 1;
    /* allow new interrupt : clear the IFR flag */
    gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_LEFT);
  }
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_RIGHT))
  {
    dip204_set_cursor_position(20,2);
    dip204_write_data(0xDF);
    display = 1;
    /* allow new interrupt : clear the IFR flag */
    gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_RIGHT);
  }
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_PUSH))
```

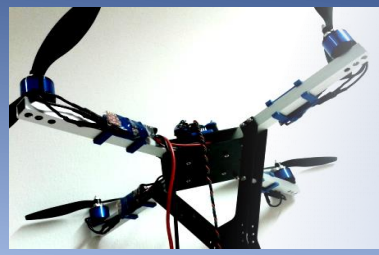


DIP (Driver, Code)

DIP Introduction (Driver)

Required for:
- Both -> Black
- Only Display -> Blue
- Only Buttons -> Green

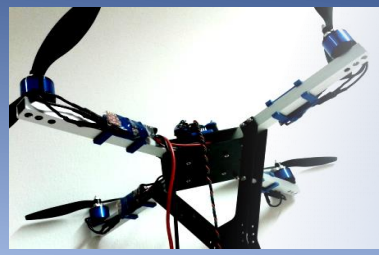
- We want: dip204_write_string(const char* string);
- We want: Click button -> execute Code
- Therefore the following is required:
  - Include-Files (contain structures, functions, drivers, libraries, etc.)
  - IRQ -> configure interrupts and IDs for buttons:
    - Activate PIN Interrupt
    - Initialize Interrupt Vector Table (IVT) (reset, execute only once!)
    - Register ISR
    - Deactivate IVT Interrupts while working
  - Initialize SPI (HW Init): SPI is a HW-Interface used to control the Display DIP204.
    - Init SPI Master (EVK1100 as Master)
    - Set Mode (SPI Modus: There are 4 modes)
    - Activate SPI
    - Set Chip frequency
  - Mapping-> Assignment of PINs to functions? (Buttons, SPI Pins for Display)
  - Create SPI Options struct -> used by the init function
  - Start oscillator -> Measure for for SPI (and others)
  - Delay Init (used by DIP Init)
  - DIP Init -> Initializes the Display



DIP (Driver, Code)

Header/Include Files:

▪ Processor IO defines (UC3A0512)	#include <avr32/io.h>
▪ Puplic Compiler defines	#include "compiler.h "
▪ Board defines (EVK1100): LEDs, Buttons, etc.	#include "board.h"
▪ Clock Defines and functions: pcl_switch_to_osc()	#include "power_clocks_lib.h "
▪ GPIO (PINS) Driver functions	#include "gpio.h"
▪ USART Header with functions	#include "usart.h"
▪ SPI Interface for DIP control	#include "spi.h"
▪ DIP module with functions	#include "dip204.h"
▪ Interrupt driver	#include "intc.h"
▪ Power management for oscillator	#include "pm.h"
▪ Delay functions for display	#include „delay.h"
▪ Standard Input Output for sprintf, printf	#include "stdio.h"



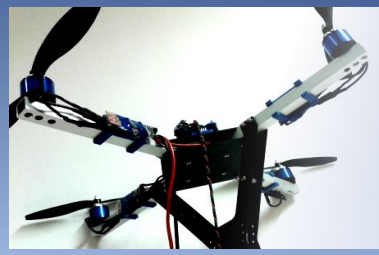
DIP (Driver, Code)

JOY ISR Code Example:

```
int counter = 0;
__attribute__((__interrupt__))
static void Joy_int_handler(void)
{
  if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_UP))
  {    counter++;
gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_UP); }
  if
(gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_DOWN))
  {   counter--;
gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_DOWN)
; }
if (gpio_get_pin_interrupt_flag(GPIO_JOYSTICK_PUSH))
  {   counter = 0;
gpio_clear_pin_interrupt_flag(GPIO_JOYSTICK_PUSH);
  }
}
```

IRQ Init Code Example:

```
Disable_global_interrupt();
INTC_init_interrupts();       // IVT being initialized
gpio_enable_pin_interrupt(GPIO_JOYSTICK_UP , GPIO_FALLING_EDGE);
gpio_enable_pin_interrupt(GPIO_JOYSTICK_DOWN , GPIO_FALLING_EDGE);
gpio_enable_pin_interrupt(GPIO_JOYSTICK_PUSH , GPIO_FALLING_EDGE);

// Register Interrupts | Priority: 3. Parameter (AVR32_INTC_INT1)
INTC_register_interrupt
(&Joy_int_handler, AVR32_GPIO_IRQ_0 + (GPIO_JOYSTICK_UP/8) ,  AVR32_INTC_INT1);
INTC_register_interrupt
(&Joy_int_handler, AVR32_GPIO_IRQ_0 + (GPIO_JOYSTICK_DOWN/8) ,  AVR32_INTC_INT1);
INTC_register_interrupt
(&Joy_int_handler, AVR32_GPIO_IRQ_0 + (GPIO_JOYSTICK_PUSH/8), AVR32_INTC_INT1);

Enable_global_interrupt();
```



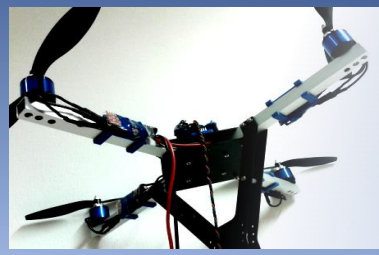
DIP (Driver, Code)

DIP Init Code Example (part 1/2):

```
static const gpio_map_t DIP204_SPI_GPIO_MAP =
  {   {DIP204_SPI_SCK_PIN,  DIP204_SPI_SCK_FUNCTION },  // SPI Clock.
    {DIP204_SPI_MISO_PIN, DIP204_SPI_MISO_FUNCTION},  // MISO.
    {DIP204_SPI_MOSI_PIN, DIP204_SPI_MOSI_FUNCTION},  // MOSI.
    {DIP204_SPI_NPCS_PIN, DIP204_SPI_NPCS_FUNCTION}   // Chip Select
};                  // Map SPI Pins
pm_switch_to_osc0(&AVR32_PM, FOSC0, OSC0_STARTUP);  // Activate Oscillator
spi_options_t spiOptions =
  {   .reg          = DIP204_SPI_NPCS,
    .baudrate     = 1000000,
    .bits         = 8,
    .spck_delay   = 0,
    .trans_delay  = 0,
    .stay_act     = 1,
    .spi_mode     = 0,
    .modfdis      = 1
  }; // add the spi options driver structure for the LCD DIP204
```

DIP Init Code Example (part 2/2):

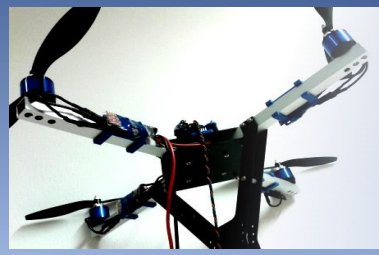
```
// Assign I/Os to SPI
gpio_enable_module(DIP204_SPI_GPIO_MAP,
                   sizeof(DIP204_SPI_GPIO_MAP) / sizeof(DIP204_SPI_GPIO_MAP[0]));

// Initialize EVK1100 as SPI master
spi_initMaster(DIP204_SPI, &spiOptions);

// Set SPI mode
spi_selectionMode(DIP204_SPI, 0, 0, 0);

// Enable SPI
spi_enable(DIP204_SPI);

// Setup Chip Registers (Frequenzy)
spi_setupChipReg(DIP204_SPI, &spiOptions, FOSC0);

// Initialize Delay Driver: required by dip204_init()
delay_init( FOSC0 );

// Initialize LCD
dip204_init(backlight_PWM, TRUE);
```



DIP (Driver, Code)

DIP functionality:

Create variable	int counter = 13;
Reserve memory space	char my_string[80];
Remember to reserve enough memory space!	
Write variable into String	sprintf(my_string, "C is %d ", counter);
Set cursor to column 8, row 1	dip204_set_cursor_position(8,1);
Display has 20 columns in 4 rows	
Write String to display	dip204_write_string("BANANARAMA");
Set cursor column 1, row 2	dip204_set_cursor_position(1,2);
Write String to display	dip204_write_string(my_string);



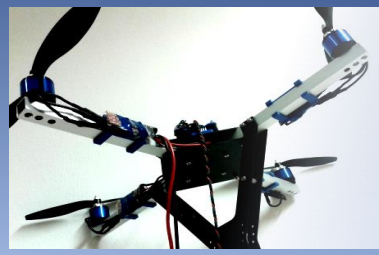
Exercises

Required Hardware:

- EVK1100
- Micro USB cable for power and flashing

Required Software:

- AVR Studio 32 (with Tool Chain and FLIP Driver)
- Example Code: main_dip.c



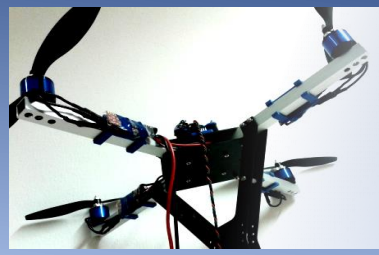
Exercises

Exercise 1:
Create a new project based on the DIP-Example. Replace the code with the lines from slides 7 to 10 or from file „main_dip.c“. Have a look at the DIP-Example and compare. Get familiar with the code. The code will stop. Find the reason for the break down and resolve the problem!

Hint: To be able to continue the work with the project you have to copy your code into a DIP example project.

Exercise 2:
Write a program to implement a 3 page display using a button for scrolling. Make it possible to use the other buttons to change the displayed variables (increment, decrement). Each display page is then used to show a text together with two variables. All variables should be modifiable using the buttons. Use at least two integers and at least two floating point variables.

Hint:
Have a look at the DIP Example to check out how the button interrupts work. Basically the joystick button should be everything you need to solve this exercise.



Exercises

Exercise 3:
Make the DIP project from exercise 2 work with USART functionality, so that the variables are shown both on the display and on the PC. Use the relevant code from the USART introduction exercises.

Remember to add the required drivers for USART under Framework / Select Drivers.

Check your results using Hterm.