

GPIO

ATxmega128A1

Networks and Embedded Software

Module 4.3.3

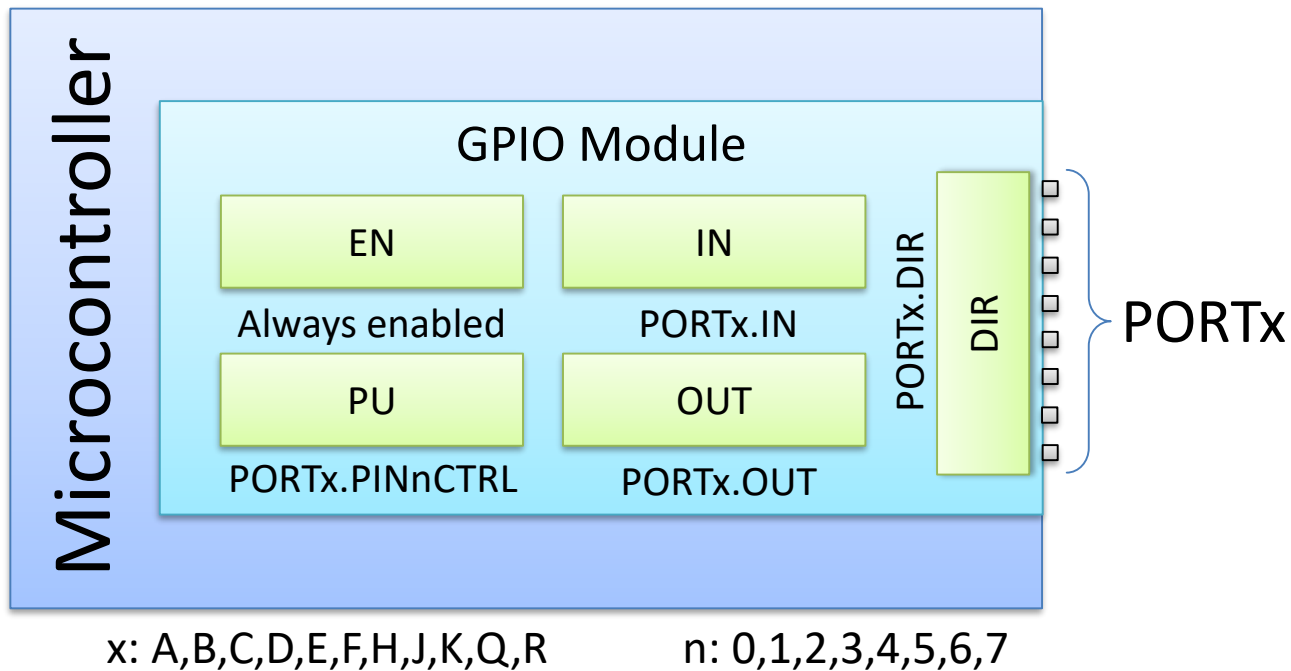
by Wolfgang Neff

GPIO (1)

- Implementation
 - Pins per Port: 8
 - Name of Ports: A, B, C, D, E, F, H, J, K, Q, R
 - Naming Scheme: PORTx.REG
 - Enabled by default
 - Complex Pin Configuration
 - One configuration register per pin
 - Pull-up configuration, pin inversion, etc.

GPIO (2)

- Register Mapping



GPIO (3)

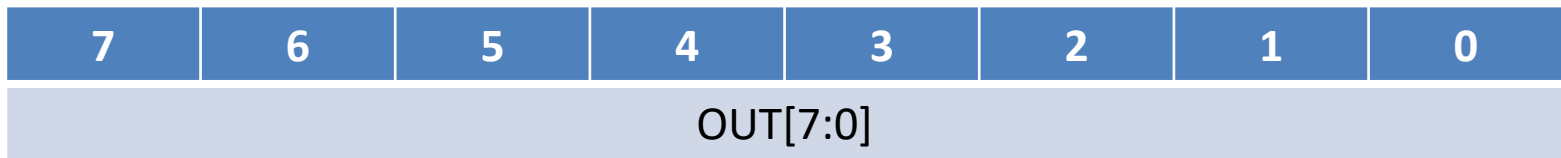
- DIR: Data Direction Register



- Bit 7:0 – DIR: Data Direction
 - 0: pin configured for input
 - 1: pin configured for output
- Naming Scheme
 - `PINn_bm` (n: bit position)

GPIO (4)

- OUT: Data Output Register



- Bit 7:0 – OUT: Data Output Value

- 0: pin is driven low (outputs 0)
 - 1: pin is driven high (outputs 1)

- Naming Scheme

- `PINn_bm` (n: bit position)

GPIO (5)

- IN: Data Input Register



- Bit 7:0 – IN: Data Input Value
 - 0: pin indicates low voltage (input is 0)
 - 1: pin indicates high voltage (input is 1)
- Naming Scheme
 - `PINn_bm` (n: bit position)

GPIO (6)

- PINnCTRL: Pin n Configuration Register

7	6	5	4	3	2	1	0
SRLEN	INVEN	OPC			ISC		

- One Register per Pin
- Bit 5:3 – OPC: Output and Pull Configuration
 - `PORT_OPC_PULLDOWN_gc`: enable pull-down resistor
 - `PORT_OPC_PULLUP_gc`: enable pull-up resistor

GPIO (7)

- Configuration Example
 - Active low LED on pin 0 and 1 of port A
 - Initialization
 - Data direction is output
 - `PORTA.DIR = (PIN0_bm | PIN1_bm);`
 - Initial state: LED 0 -> on, LED 1 -> off
 - `PORTA.OUT = PIN1_bm;`

Initialisation is often done by assignment

GPIO (8)

- Configuration Example (continued)

- Operation

Change is done by bit manipulation

- Switching LED 0 off and LED 1 on

- `PORTA.OUT |= PIN0_bm;`

- `PORTA.OUT &= ~PIN1_bm;`

GPIO (9)

- Configuration Example
 - Active low button on pin 0 of port A
 - Initialization
 - Data direction is input
 - Ports are input by default so nothing to do here
 - Activate pull up resistor
 - `PORTA.PINCTRL = PORT_OPC_PULLUP_gc;`

GPIO (10)

- Configuration Example (continued)
 - Operation
 - Reading the active low button
 - `if (!(PORTA.IN & PIN0_bm)) {...}`



The button is active low. So we have to **test (&)** if the **bit is logically not (!)** set.

GPIO (11)

- Register Summary

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DIR	DIR[7:0]							
DIRSET	DIRSET[7:0]							
DIRCLR	DIRCLR[7:0]							
DIRTGL	DIRTGL[7:0]							
OUT	OUT[7:0]							
OUTSET	OUTSET[7:0]							
OUTCLR	OUTCLR[7:0]							
OUTTGL	OUTTGL[7:0]							

GPIO (12)

- Register Summary (continued)

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IN	IN[7:0]							
INTCTRL	-	-	-	-	INT1LVL		INT0LVL	
INTOMASK	INTOMASK[7:0]							
INT1MASK	INT1MASK[7:0]							
INTFLAGS	-	-	-	-	-	-	INT1IF	INT0IF
PINOCTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]		
...	...							
PIN7CTRL	SRLEN	INVEN	OPC[2:0]			ISC[2:0]		

GPIO (13)

- Interrupt Summary

Source	Description
PORTx_INT0_vect	PORTx interrupt vector 0
PORTx_INT1_vect	PORTx interrupt vector 1

