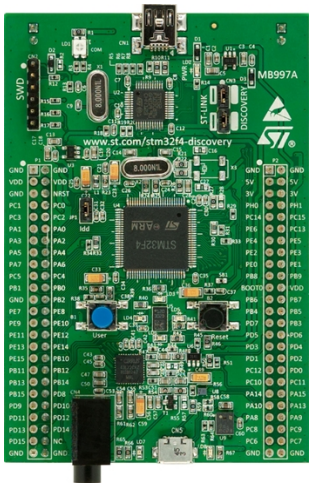
Izdelava knjižnice za delo s splošno namenskim vhodom/izhodom

Na razvojni plošči STM32F4Discovery, ki jo uporabljamo na vajah, je vgrajen mikrokrmilnik STM32F407, le-ta ima 140 pinov, ki jih lahko uporabimo kot digitalni vhod, digitalni izhod ali pa prepustimo da s pini upravljajo druge naprave znotraj mikrokrmilnika. Gre torej za splošnonamenski pine (angl. General Purpose Input Output), ki jih običajno označujemo s kratico GPIO.

V mikrokrmilniku STM32F407 so pini razdeljeni v 9 GPIO naprav (angl. GPIO Port) s črkovnimi oznakami od A do I. Naprave A do H imajo vsaka po 16 pinov, naprava I pa upravlja s preostalimi dvanajstimi. Pini znotraj naprav so označeni s števili od 0 do 15. Posamezne pine mikrokrmilnika tako označujemo s črko P, sledi črka naprave ter številka pina znotraj naprave. Na primer PA0 (pin 0 na napravi A) ali PD12 (pin 12 na napravi D). Takšne oznake lahko vidite tudi na sami razvojni plošči (glej sliko 1).

Slika 1: STM32F4 Discovery.



Vse naprave imajo vnaprej dodeljen pomnilniški prostor. Primer dodeljenih prostorov je prikazan na sliki 2. V tem pomnilniškem prostoru se nahajajo registri naprav. Ti hranijo vse nastavitve naprave, prav tako pa preko njih beremo stanje naprave oziroma jo upravljamo.

Slika 2: Naslovi naprav.

Boundary address	Peripheral	Bus
0x4004 0000 - 0x4007 FFFF	USB OTG HS	AHB1
0x4002 9000 - 0x4002 93FF	ETHERNET MAC	
0x4002 8C00 - 0x4002 8FFF		
0x4002 8800 - 0x4002 8BFF		
0x4002 8400 - 0x4002 87FF		
0x4002 8000 - 0x4002 83FF		
0x4002 6400 - 0x4002 67FF	DMA2	
0x4002 6000 - 0x4002 63FF	DMA1	
0x4002 4000 - 0x4002 4FFF	BKPSRAM	
0x4002 3C00 - 0x4002 3FFF	Flash interface register	
0x4002 3800 - 0x4002 3BFF	RCC	
0x4002 3000 - 0x4002 33FF	CRC	
0x4002 2000 - 0x4002 23FF	GPIOI	
0x4002 1C00 - 0x4002 1FFF	GPIOH	
0x4002 1800 - 0x4002 1BFF	GPIOG	
0x4002 1400 - 0x4002 17FF	GPIOF	
0x4002 1000 - 0x4002 13FF	GPIOE	
0X4002 0C00 - 0x4002 0FFF	GPIOD	
0x4002 0800 - 0x4002 0BFF	GPIOC	
0x4002 0400 - 0x4002 07FF	GPIOB	
0x4002 0000 - 0x4002 03FF	GPIOA	

Vklop ure GPIO naprave

Če želimo uporabiti posamezen GPIO pin, moramo najprej vklopiti napravo, ki ji pin pripada. Za vklop ure skrbi naprava `Reset and Clock Control (RCC)`. Vklopu ur GPIO naprav je namenjen register `RCC_AHB1ENR`. Slika 3 prikazuje izsek dokumentacije mikrokrmilnika o tem registru. Kot vidimo, gre za 32-bitni register, kjer je spodnjih devet bitov namenjenih vklopu in izklopu ure GPIO naprav. Bit 0 ima oznako `GPIOAEN` in skrbi za vklop ure naprave A, bit 1 skrbi za uro naprave B, in tako dalje vse do bita 8, ki skrbi za vklop in izklop ure naprave GPIOI. Edino vprašanje, ki se nam tukaj še poraja je, na katerem naslovu se ta register nahaja. V izseku dokumentacije, ki ga prikazuje slika 3 je namreč podan zgolj odmik naslova znotraj naprave (angl. Address offset).

Kot vidimo na sliki 3 je ta odmik pri registru `RCC_AHB1ENR` 0x30. Dejanski naslov dobimo tako, da odmik prištejemo začetnemu naslovu naprave RCC. Tega lahko najdete na sliki 2. Za napravo RCC je zapisan začetni naslov 0x40023800. Točen naslov registra `RCC_AHB1ENR` je torej 0x40023830. Če preverite rešitve vaših nalog iz pretekle vaje, boste opazili, da ste za vklop ure na tem naslovu postavljali bita 0 in 3. S tem se vklopili uri za napravi GPIOA in GPIOD (gumb integriran na plošči se nahaja na PA0, led diode pa na PD12, PD13, PD14 in PD15).

Slika 3: Register AHB1ENR naprave RCC.

5.3.10 RCC AHB1 peripheral clock register (RCC_AHB1ENR)

Address offset: 0x30

Reset value: 0x0010 0000

Access: no wait state, word, half-word and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reser-ved	OTGHS ULPIEN	OTGHS EN	ETHMA CPTPE N	ETHMA CRXEN	ETHMA CTXEN	ETHMA CEN	Reserved		DMA2EN	DMA1EN	CCMDATA RAMEN	Res.	BKPSR AMEN	Reserved	
	rw	rw	rw	rw	rw	rw			rw	rw			rw		

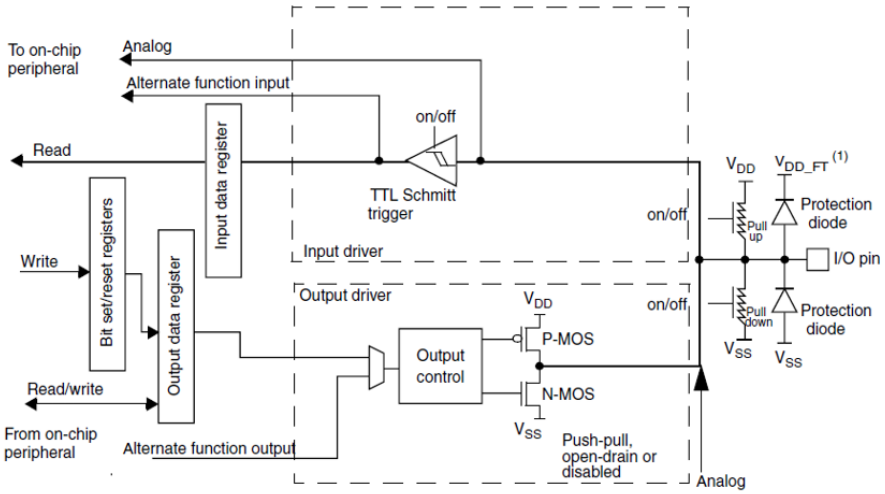
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			CRCEN	Reserved			GPIOIE N	GPIOH EN	GPIOGE N	GPIOFE N	GPIOEEN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
			rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Inicializacija GPIO naprave

Po vklopu ure lahko določamo nastavitve posameznih pinov. Za vsakega izmed šestnajstih pinov ima GPIO naprava enako vhodno/izhodno stopnjo, ki je prikazana na sliki 4. Zgornji del prikazuje vhodni del, spodnji pa izhodnega. Obema deloma je skupna zgolj uporaba zaščitnih diod (angl. Protection diode) ter Pull-up in Pull-down uporov. Uporaba slednjih je opcijska in jo lahko nastavimo tako za vhodne kot izhodne pine. V primeru, da pin uporabimo kot vhod, je to tudi (poleg nastavitve da gre za vhodni pin) edina nastavitev, ki jo je potrebno določiti. Logično stanje na vhodu nato beremo preko registra `Input Data (IDR)`. V primeru, da pin deluje kot izhod, je potrebno nastaviti še način izhoda (Push-pull ali open-drain) ter hitrost osveževanja vrednosti na izhodu. Po izbiri teh nastavitev stanje na izhodu določamo preko registra `Output Data (ODR)` ali preko registrov `Bit set/reset (BSRR)`.

Način delovanja (angl. mode) pina določimo v registru `Mode (MODER)`. Slika 5 prikazuje izsek dokumentacije, ki opisuje omenjeni register. Kot vidimo na sliki, gre za 32-bitni register, ki je razdeljen v šestnajst dvo-bitnih delov, vsak namenjen enemu pinu. Bita 0 in 1 tako služita za nastavljanje načina delovanja pina 0 izbrane naprave, bita 2 in 3 nastavljata način delovanja pina 1, bita 4 in 5 pina 2 in tako dalje do bitov 30 in 31, ki nastavljata način delovanja pina 15. Na sliki je ravno tako prikazano, da nastavitev bitov na vrednost 00 (oba bita 0) pomeni, da pin deluje kot vhod, vrednost 01 (zgornji bit 0, spodnji bit 1) pa pomeni izhod. Preostali vrednosti določita analogni način delovanja (11) in način alternativne funkcije (10), ki pomeni,

Slika 4: Vhodno/izhodna stopnja.



da s pinom upravlja ena izmed preostalih naprav mikrokrmilnika. Slednja dva načina bomo podrobneje spoznali na vajah v nadaljevanju semestra. Kot vidimo, je odmik naslova 0x00, kar pomeni, da se ta register vedno nahaja na začetku pomnilniškega prostora posamezne naprave. Za napravo A torej na naslovu 0x40020000 in za napravo D na naslovu 0x40020C00. Na prvi naslov ste v zadnji vaji morali na bita 0 in 1 vpisati ničli. S tem ste določili, da bo v napravi A pin 0 deloval kot vhod (na tem pinu najdemo integriran gumb). Na drugi naslov pa ste na zgornjih osem bitov zapisali vrednost 01. Kar pomeni, da ste pine 12, 13, 14 in 15 naprave D nastavili v izhodni način delovanja (na teh štirih pinih najdemo integrirane led diode).

Če želimo pin uporabiti kot vhod, je edina nastavitev, ki jo še moramo nastaviti, uporaba pull-up in pull-down uporov. Pull-up ali pull-down upore najpogosteje uporabljamo, da preprečimo "plavanje" (angl. float) na vhodnih ali izhodnih pinih. V primeru da na pinu ni drugih virov, s pull-up uporom lahko dosežemo, da je na pinu logična enica, medtem ko s pull-downom uporom dosežemo, da je takrat na pinu logična ničla. Ti upori imajo še več drugih primerov uporabe, v katere pa se zaenkrat ne bomo podrobno spuščali. Zaenkrat bomo za vse pine, ki jih bomo uporabljali, nastavljali, da ne uporabljajo omenjenih uporov. To nastavitev določimo v registru `Pull-up, Pull-down (PUPDR)`. Izsek iz dokumentacije je prikazan na sliki 6.

Podobno kot register `MODER`, je tudi `PUPDR` 32-bitni register, kjer sta po

Slika 5: Register MODE naprave GPIO.

6.4.1 GPIO port mode register (GPIOx_MODER) (x = A..I)

Address offset: 0x00

Reset values:

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Slika 6: Register PUPD naprave GPIO.

6.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..I)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O pull-up or pull-down
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

dva bita sta namenjana posameznemu pinu. Vpisana vrednost 00 pomeni, da na pinu ne želimo uporabiti niti pull-up niti pull-down upora. Z vpisom

01 vklopimo pull-up, z vpisom 10 pa pull-down. Vrednost 11 je rezervirana (prepovedana) in se je ne uporablja. Odmik naslova registra `PUPDR` je 0x0C.

Če želimo pin uporabiti kot izhod moramo poleg zgoraj omenjenih registrov nastaviti še registra, ki določata način izhoda ter hitrost osveževanja. Hitrost osveževanja določamo v registru `Output Speed (OSPEEDR)`, ki je prikazan na sliki 7. Po strukturi je register podoben prej opisanima registroma. V primeru registra `OSPEEDR` vpis vrednosti 00 pomeni, da bo hitrost osveževanja 2MHz, vpis vrednosti 01 pomeni, da bo hitrost osveževanja 25 MHz, preostali kombinaciji (10 in 11) pa nastavita hitrost osveževanja na 50 ali 100 MHz. Odmik naslova registra `OSPEEDR` je 0x08. Za vse pine, ki jih bomo uporabljali pri predmetu ORS, bo zadostovala najnižja hitrost osveževanja.

Slika 7: Register OSPEEDR naprave GPIO.

6.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..I)

Address offset: 0x08

Reset values:

- 0x0000 00C0 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15[1:0]		OSPEEDR14[1:0]		OSPEEDR13[1:0]		OSPEEDR12[1:0]		OSPEEDR11[1:0]		OSPEEDR10[1:0]		OSPEEDR9[1:0]		OSPEEDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1[1:0]		OSPEEDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **OSPEEDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.
00: 2 MHz Low speed
01: 25 MHz Medium speed
10: 50 MHz Fast speed
11: 100 MHz High speed on 30 pF (80 MHz Output max speed on 15 pF)

Preostane nam še register v katerem določimo način izhoda. Načina izhoda sta push-pull in open-drain. Več o razliki med njima boste spoznali na predavanjih. Če na povzamemo na kratko, se pri push-pull pri ničli izhod vleče (angl. pull) proti ozemljitvi (GND, angl. ground) in potiska proti viru napajanja (VDD) pri enici. Pri open-drain pa se izhod pri ničli obnaša podobno, pri logični enici pa je izhod v visoki impendanci. V navezavi z open-drain načinom izhoda se zato običajno uporablja pull-up upor. Na vajah bo način izhoda vedno push-pull, razen če bomo izrecno poudarili. Za

LED diode tako izhod vedno nastavimo na push-pull. Register, ki nastavlja način izhoda se imenuje `Output Type (OTYPER)`. V dokumentaciji (slika 8) vidimo, da je tudi ta register 32-bitni, vendar pa se dejansko uporablja zgolj spodnjih 16 bitov. Vsak bit nastavlja način izhoda za en pin. Bit 0 za pin 0, bit 1 za pin 1, in tako dalje do bita 15, ki nastavlja način izhoda za pin 15. Odmik naslova tega registra je 0x04.

Slika 8: Register OTYPER naprave GPIO.

6.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

Branje vhoda GPIO naprave

Za branje vhoda uporabljamo register `Input Data (IDR)`, ki je prikazan tudi na sliki 4. Ta register se znotraj registrov naprave nahaja na odmiku 0x10. Kot prikazuje dokumentacija na sliki 9, je tudi ta register 32-bitni, vendar je podobno kot pri `OTYPER` uporabljenih zgolj spodnjih 16 bitov.

Slika 9: Register IDR naprave GPIO.

6.4.5 GPIO port input data register (GPIOx_IDR) (x = A..I)

Address offset: 0x10

Reset value: 0x0000 XXXX (where X means undefined)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Za razliko od registrov, ki smo jih do sedaj spoznali, se bite tega registra da samo brati (angl. read-only). To je označeno s črko r pod oznakami bitov. Posamezen bit predstavlja logično vrednost na vhodu. Če je na vhodnem pinu visoka napetost, potem bo na bitu, ki predstavlja stanje pina, enica. Če je na vhodnem pinu nizka napetost, potem bo na bitu, ki predstavlja stanje pina, ničla. Ker je gumb, ki ste ga uporabljali na zadnji vaji, vezan na pin PA0, ste stanje gumba brali tako, da ste preverjali stanje bita 0 na naslovu 0x40020010 (0x4002000 zaradi naprave A + 0x10 zaradi odmika registra).

Nastavljanje izhoda GPIO naprave

Za nastavljanje vrednosti na izhodu imamo dve možnosti. Neposredno nastavljanje registra `Output Data (ODR)` ali pa posredno nastavljanje preko para set/reset registrov. Register ODR (slika 10) se znotraj registrov naprave nahaja na odmiku 0x14. Podobno kot pri registru `IDR` tu uporabljamo zgolj spodnjih 16-bitov, le da tu mi nastavljamo posamezne bite na 0 ali 1 in s tem neposredno spreminjamo napetost na istoležnih izhodnih pinih. Primer: vpis enice na bit 5 pomeni, da bo na pinu 5 te naprave visoka napetost. Vpis ničle na isti bit pa povzroči, da je na pinu 5 nizka napetost. Oboje se zgodi le v primeru, da je pin v izhodnem načinu delovanja.

Slika 10: Register ODR naprave GPIO.

6.4.6 GPIO port output data register (GPIOx_ODR) (x = A..I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy[15:0]:** Port output data (y = 0..15)
These bits can be read and written by software.
Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I).

Če želimo spremeniti vrednost določenega bita v registru `ODR`, moramo vrednost registra ODR najprej prebrati, nato z logično operacijo in/ali spremeniti njegovo vrednost ter vrednost zapisati nazaj na naslov. Kot smo

spoznali na prvih vajah, bi programsko postavljanje p-tega bita zapisali kot `ODR = ODR | (1 << p)`, brisanje pa kot `ODR = ODR & ~(1 << p)`. Vsaka izmed teh operacij potrebuje tri korake za izvedbo: branje vrednosti, izračun ter pisanje vrednosti (angl. read-modify-write).

Mikrokrmilnik STM32F407 omogoča, da obe operaciji izvedemo bolj učinkovito, in sicer preko registrov set in reset. Izsek iz dokumentacije teh dveh registrov je prikazan na sliki 11. Na sliki sta prikazan kot en register, dejansko pa gre za dva 16-bitna registra. Oba registra omogočata zgolj vpis (angl. write-only). In sicer vpis enice na bit p registra `Bit Set (BSR)` pomeni, da se na bit p v registru `ODR` vpiše enica. Vpis **enice** na bit p registra `Bit Reset (BRR)` pa pomeni, da se na bit p v registru `ODR` vpiše **ničla**. Ko se sprememba vrednosti v registru `ODR` izvede, se registra `BSR` in `BRR` ustrezno počistita. Register `BSR` se nahaja na odmiku 0x18, register `BRR` pa na odmiku 0x1A. Na zadnji vaji ste LED diode prižigali z vpisovanjem enic na naslov 0x40020C18, ugašali pa z vpisovanjem enic na naslov 0x40020C1A. Prva vrednost predstavlja naslov `BSR` registra naprave GPIOD, druga pa naslov `BRR` registra iste naprave.

Slika 11: Register BSRR naprave GPIO.

6.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)
These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Resets the corresponding ODRx bit
Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)
These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODRx bit
1: Sets the corresponding ODRx bit

Registri GPIO naprave, ki smo jih pravkar spoznali, si v pomnilniškem prostoru ene naprave sledijo tako kot prikazuje tabela 1.

Tabela 1: Odmiki registrov GPIO naprave.

Odmik naslova	Register
0x00	Mode (MODER)
0x04	Output Type (OTYPER)
0x08	Output Speed (OSPEEDR)
0x0C	Pull-up/Pull-down (PUPDR)
0x10	Input Data (IDR)
0x14	Output Data (ODR
0x18	Bit Set (BSR)
0x1A	Bit Reset (BRR)

Funkcije pinov

Kot smo že zapisali, se gumb na razvojni plošči STM32F4 Discovery nahaja na pinu PA0, medtem ko so štiri LED diode povezane na pine PD12, PD13, PD14 in PD15. Verjetno se sprašujete, kako bi ta isti podatek lahko izbrskali sami. Podatki o tem kam so povezani posamezni na razvojni plošči so zapisani v uporabniškem priročniku (angl. User manual), ki ga najdete na učilnici ali na spletni strani proizvajalca (povezava). V priročniku lahko na straneh od 20 do 28 najdete tabelo, katere izsek je prikazan na sliki 12. Na sliki vidimo potrditev, da je zelena LED dioda vezana na pin PD12, oranžna na PD13, rdeča na PD14 ter modra na PD15. Prav tako na desni strani vidite, da je uporabniški (angl. User) gumb vezan na pin PA0.

Slika 12: Izsek iz uporabniškega priročnika.

Main function	Alternate functions	LQFP100	CS43L22	MP45DT02	LIS302DL	Pushbutton	LED
PD12	FSMC_A17/ TIM4_CH1/ USART3_RTS	59					GREEN
PD13	FSMC_A18/ TIM4_CH2	60					ORANGE
PD14	FSMC_D0/ TIM4_CH3	61					RED
PD15	FSMC_D1/ TIM4_CH4	62					BLUE

Main function	Alternate functions	LQFP100	CS43L22	MP45DT02	LIS302DL	Pushbutton
BOOT0	VPP	94				
NRST		14				RESET
PA0-WKUP	USART2_CTS/ USART4_TX/ ETH_MII_CRS/ TIM2_CH1_ETR/ TIM5_CH1/ TIM8_ETR/ ADC123_IN0/ WKUP	23				USER

Uporaba konstant in struktur za delo s specifičnimi naslovi

Z načinom pisanja kode za delo z GPIO, kot smo jo pisali na prejšnji vaji, bi težko naredili kak večji projekt. Kot vas je večina ugotovila, je takšno kodo, ki vsebuje polno neposredno vpisanih naslovov in konstant, težko brati in razhroščevati. Najmanj, kar bi lahko naredili je, da vsem konstantam in naslovom dajemo opisna imena in jih zberemo na enem mestu (na vrhu datoteke ali v ločeni .h datoteki), saj jih bo tako lažje preverjati in vzdrževati.

Za primer vzemimo Bit Set (`BSR`) register naprave GPIOD, ki se nahaja na naslovu 0x40020C18. Namesto, da v funkcijah uporabljamo direktno vrednost naslova, lahko temu naslovu dodelimo neko ime, ki nam pove kaj ta naslov predstavlja (na primer `GPIOD_BIT_SET_REG`). Prav tako lahko dodelimo smiselna imena konstantam, ki jih uporabljamo za delo z registrom. Primer takih konstant lahko vidite spodaj:

```
#define GPIOD_BIT_SET_REG 0x40020C18
#define PIN_0 0x0001
#define PIN_1 0x0002
#define PIN_2 0x0004
#define PIN_3 0x0008
```

S takšnimi konstantami bi vklop pinov zapisali kot

```
uint32_t *p = (uint32_t *) GPIOD_BIT_SET_REG;
*p = PIN_0;
// vklopimo lahko tudi vec pinov hkrati
*p = PIN_1 | PIN_2;
```

Zgornjo uporabo lahko še dodatno optimiziramo tako, da konstanto, ki predstavlja naslov registra, spremenimo v kazalec. S tem se izognemo prvi vrstici v zgornjemu primeru.

```
#define GPIOD_BIT_SET_REG ((uint16_t*)0x40020C18)
#define PIN_0 0x0001
#define PIN_1 0x0002
#define PIN_2 0x0004
#define PIN_3 0x0008

*GPIOD_BIT_SET_REG = PIN_0;
*GPIOD_BIT_SET_REG = PIN_1 | PIN_2;
```

Kot smo videli na primeru GPIO naprav, imajo vse naprave mikrokrmnilnika množico registrov, ki jih moramo nastavljati, ko te naprave uporabljamo. Registri posamezne naprave se vedno nahajajo na zaporednih naslovih v pomnilniku. Z uporabo zgornjega pristopa bi 4 zaporedne registre ene naprave lahko nastavili na sledeč način:

```
#define NAPRAVA_REG1  ((uint32_t *) 0x40020C00)
#define NAPRAVA_REG2  ((uint16_t *) 0x40020C04)
#define NAPRAVA_REG3  ((uint16_t *) 0x40020C06)
#define NAPRAVA_REG4  ((uint32_t *) 0x40020C08)

// registrom dolocimo vrednosti
*NAPRAVA_REG1 = 0x8000;
*NAPRAVA_REG2 = 0x4000;
*NAPRAVA_REG3 = 0x6000;
*NAPRAVA_REG4 = 0x5000;
```

Z uporabo strukture, ki opisuje registre naprave lahko omenjeno kodo še dodatno izbojšamo. Vemo namreč, da se vsi elementi strukture nahajajo na zaporednih naslovih. S tem se izognemu ponavljanju celotnih naslovov za vsak register posebej. Navedemo namreč zgolj začetni naslov naprave. Če je struktura pravilno sestavljena (predstavlja pomnilniško sliko naprave) bodo potem vsi elementi strukture kazali na pravilne registre.

```
struct naprava_x {
  uint32_t REG1;
  uint16_t REG2;
  uint16_t REG3;
  uint32_t REG4;
};

#define NAPRAVA 0x40020C00

struct naprava_x * x;
x = (struct naprava_x *) NAPRAVA;

// registrom dolocimo vrednosti
x->REG1 = 0x8000;
x->REG2 = 0x6000;
x->REG3 = 0x4000;
x->REG4 = 0x5000;
```

Če zopet uporabimo trik, da je konstanta kazalec, lahko kodo naredimo bolj berljivo ter jo še dodatno skrajšamo:

```
1   struct naprava_x {
2     uint32_t REG1;
3     uint16_t REG2;
4     uint16_t REG3;
5     uint32_t REG4;
6   };
7
8   #define NAPRAVA ((struct naprava_x *) 0x40020C00)
9
10  // registrom dolocimo vrednosti
11  NAPRAVA->REG1 = 0x8000;
12  NAPRAVA->REG2 = 0x6000;
13  NAPRAVA->REG3 = 0x4000;
14  NAPRAVA->REG4 = 0x5000;
```

Omenjen zapis morda res ni krajši kot tisti, iz katerega smo začeli, a se je bistveno zmanjšalo število fiksno zapisanih naslovov v naši kodi. To pa posledično pomeni bistveno manjšo možnost za vnos napak. Če upoštevamo še dejstvo, da lahko zapisano strukturo uporabimo na več mestih (isto strukturo za GPIO lahko uporabimo za vseh devet GPIO naprav) pa smo s tem zapisom tudi močno skrajšali našo kodo.

Naloga

Iz učilnice naložite izhodiščno main.c datoteko za vaš projekt. Rešite naslednje podnaloge:

- Sestavite strukturo, ki predstavlja GPIO napravo. Elementi strukture naj imajo smiselna imena iz katerih se enostavno razbere njihov pomen.
- Napišite funckijo `clock_on`, ki vklopi uro podane naprave. Primera uporabe: `clock_on(GPIOAd); clock_on(GPIOCd);`
- Napišite funckijo `init_GPIO`, ki izbran pin nastavi na poljubne nastavitve. Primera uporabe:
  `init_GPIO(GPIODd, 12, OUT, NO_PULL, PUSH_PULL, S2MHz);`
  `init_GPIO(GPIOAd, 0, IN, NO_PULL, PUSH_PULL, S2MHz);`

Prvi argument funckije je naprava, ki ji pin pripada (od `GPIOAd` do `GPIOId`), drugi argument je številka pina, sledijo način delovanja (`IN`, `OUT`, `AF` ali `ANALOG`), kjer kratica `AF` predstavlja alternativno funkcijo, uporaba pull-up/pull-down uporov (`NO_PULL`, `PULL_UP` ali `PULL_DOWN`), način izhoda (`PUSH_PULL` ali `OPEN_DRAIN`) ter na koncu hitrost osveževanja (`S2MHz`, `S25MHz`, `S50MHz` ali `S100MHz`).

- Napišite funkcijo za vklop/izklop izhodnega pina, ki nastavi izbrani pin na podano vrednost: `GPIO_pin_write(naprava, pin, vrednost)`.
- Napišite funkcijo za branje stanja vhodnega pina, ki vrne enico, če je stanje podanega pin na vhodu ena oziroma ničlo, če je stanje podanega pina ničla: `GPIO_pin_read(naprava, pin)`.
- Uporabite zgoraj implementirane funkcije za proženje led sekvence ob pritisku na gumb, led sekvenca naj bo enaka kot pri prejnji vaji. Torej ob pritisku na uporabniški gumb (gumb modre barve), naj se najprej z zamikom prižge vsaka ledica posebej, ko so vse led diode prižgane, se po kratkem zamiku ugasnejo.

Pomoč in namigi

- Gumb se nahaja na pinu PA0. Pin nastavite kot vhod brez pull-up ali pull-down uporov.
- LED diode se nahajajo na pinih PD12, PD13, PD14, PD15. Vse pine LED diod inicializirajte kot izhod, brez pull-up ali pull-down uporov in z načinom izhoda push-pull. Hitrost osveževanja lahko nastavite na najnižjo.