SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 2021-72

Programska potpora za upravljanje kamerom na CubeSat nanosatelitu

Nikola Gudan

Zagreb, lipanj 2022.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Hannon le.

SADRŽAJ

1. Uvod

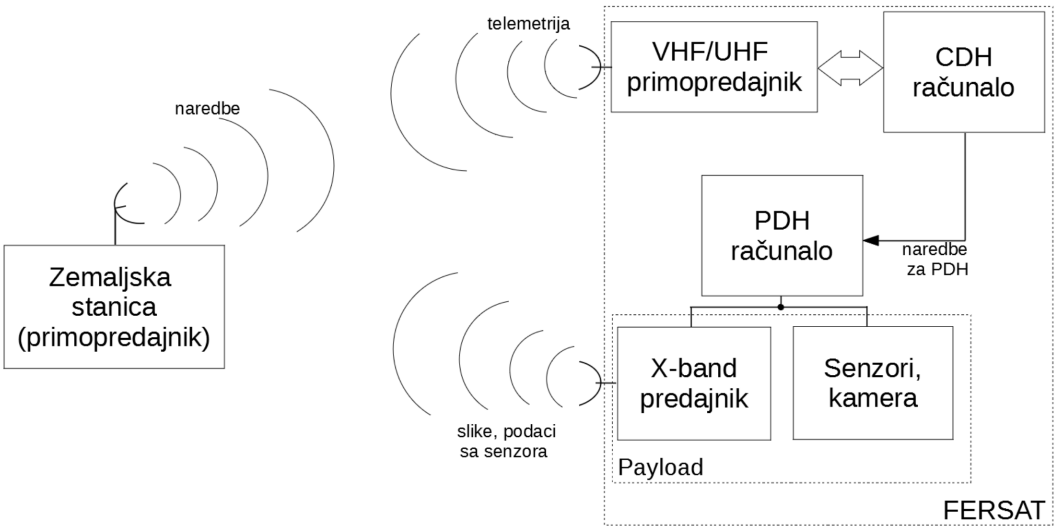
Ovaj završni projekt se izvodi u sklopu projekta FERSAT, koji se od 2018. godine provodi na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu [8]. Cilj projekta je izrada, lansiranje i korištenje jednog nanosatelita u CubeSat formatu, dimenzija 10 cm x 10 cm x 10 cm, volumena jedne litre i težine ne veće od 4/3 kg. Navedene dimenzije satelita odgovaraju formatu CubeSat 1U. Planirana visina orbite satelita je između 500 i 600 km, a očekivano trajanje misije je 3 godine. Korisni teret satelita (engl. *payload*) se sastoji od tri podsustava:

- kamera za snimanje površine Zemlje i zemaljskog horizonta,
- detektori svjetla u vidljivom i ultraljubičastom dijelu spektra za mjerenje svjetlosnog onečišćenja i debljine stupca ozona,
- komunikacijski sustav u radijskom X-pojasu (10.45 GHz) za prijenos podataka na Zemlju.

Kako bi se moglo upravljati radom korisnog tereta, na satelit će biti ugrađeno PDH (engl. *Payload Data Handler*) računalo, čija će zadaća biti prikupljanje podataka s kamere i senzorskog podsustava, pohranjivanje prikupljenih podataka u trajnu memoriju (engl. *non-volatile memory*), te slanje podataka na Zemlju pomoću komunikacijskog sustava. Izabrani mikrokontroler za ulogu PDH računala je STM32L471VGT6 proizvođača ST Microelectronics.

Ostalim podsustavima, koji nisu direktno vezani uz koristan teret, upravlja CDH (engl. *Command and Data Handler*) računalo. CDH računalo može upravljati položajem i orijentacijom satelita, slanjem telemetrijskih podataka na Zemlju, a također upravlja i napajanjem korisnog tereta i šalje naredbe PDH računalu preko CAN (engl. *Controller Area Network*) sučelja. U trenutku pisanja ove dokumentacije, konkretno CDH računalo još nije odabrano.

Slika 1.1 prikazuje blok dijagram cijelog sustava. U okviru ovog projekta razvijena je programska potpora PDH računala za upravljanje kamerom i *flash* memorijom.



Slika 1.1: Blok dijagram FERSAT-a i komunikacija sa zemaljskom postajom [3]

Sustav za upravljanje kamerom se sastoji od Arducam Mini 5MP Plus kamere. Upravljanje kamerom se sastoji od konfiguracije kamere i samog korištenja kamere, odnosno slikanja i spremanja slike. Konfiguracija kamere je nužna kako bi se ispravno podesili parametri trajanja ekspozicije, pojačanje i formata u kojem se slika želi spremiti.

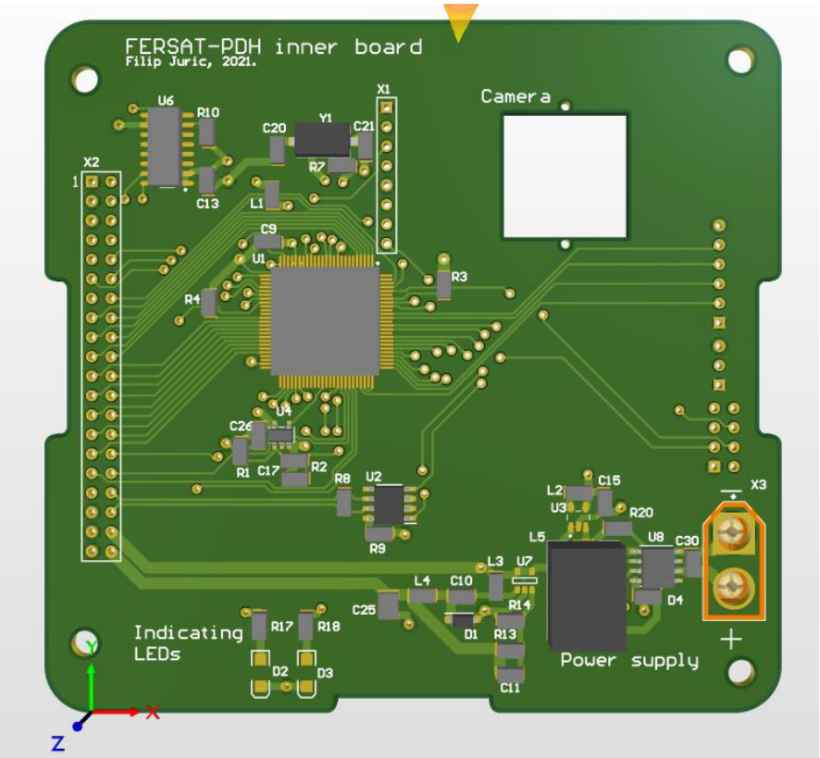
1.1. Arhitektura sustava

Slanjem određenog signala na sklop za kameru može se uslikati slika, a nakon slikanja slika se spremi na vlastiti međuspremnik kamere. Cilj je spremljenu kameru pročitati iz međuspremnika kamere i spremiti ju na *flash* memoriju koja se nalazi na pločici PDH-a, gdje može biti spremljena dok se ne zatraži slanje slike preko X-band predajnika na Zemlju.

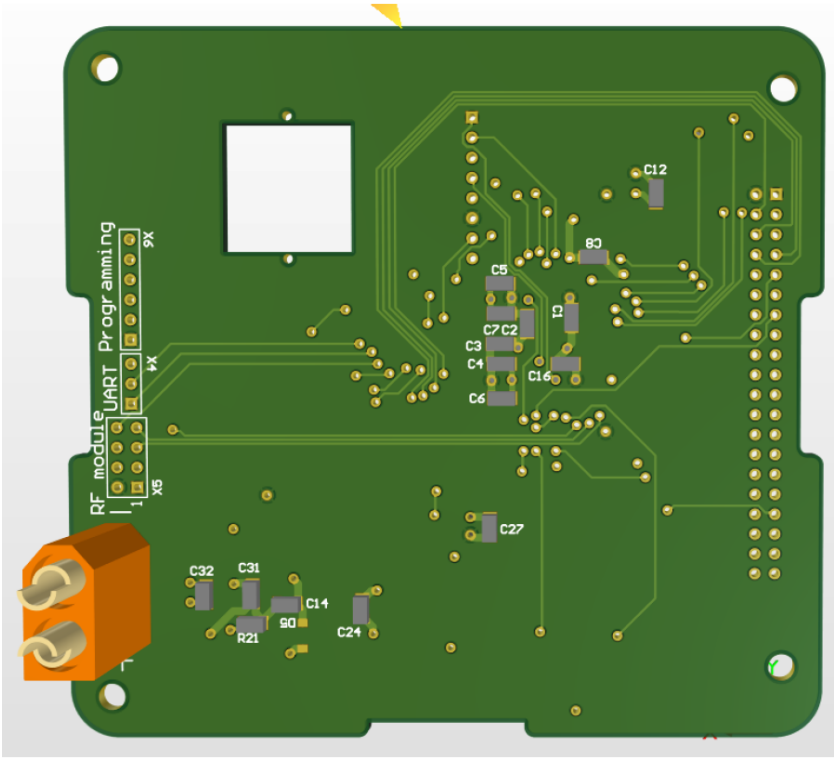
Flash memorija, osim što služi za pohranu slike, služi i za pohranu podataka s drugih senzora. Ona prima i šalje podatke ovisno o poslanoj naredbi putem SPI komunikacije s mikrokontrolerom.

Upravljačko sklopovlje PDH računala se sastoji od STM32F471VGT6 mikrokontrolera, spomenute vanjske *flash* memorije, konektora za povezivanje s ostalim dijelovima sustava (uključujući i konektor za povezivanje s kamerom), sustava za napajanje, upravljačkog sklopovlja za CAN komunikaciju i sklopa za kontrolu izvođenja programa (engl. *watchdog*) [1]. Izgled tiskane pločice upravljačkog sklopovlja PDH računala prikazan je na slikama 1.2 i 1.3. Konektor X1 služi za povezivanje sustava s

kamerom.



Slika 1.2: Prikaz gornje strane tiskane pločice upravljačkog sklopovlja PDH računala [1]



Slika 1.3: Prikaz donje strane tiskane pločice upravljačkog sklopovlja PDH računala [1]

Programska podrška za PDH računalo već je razvijena [3]. Međutim, u međuvremenu je došlo do promjene izbora mikrokontrolera PDH računala, te je stoga postojeću programsku podršku bilo potrebno prilagoditi trenutačnom sklopovlju.

S obzirom na prirodu ovog završnog projekta, gdje je naglasak bio na prilagođavanju postojeće programske podrške, u ovom radu će biti raspravljeni izazovi i izmjene do kojih je došlo tijekom prilagođavanja programske podrške. U poglavlju 2 dan je detaljan opis I^2C komunikacije, te su istaknute razlike između starog i novog sklopovlja koje su bile ključne za prilagođavanje programske podrške. Na isti način opisani su SPI komunikacija i DMA prijenos u poglavlju x. Detaljan pregled razvijene programske podrške dan je u poglavlju y, gdje je opisana integracija programske podrške za kameru i *flash* memorija u FreeRTOS operacijski sustav za rad u stvarnom vremenu.

2. I^{2}C sučelje

Za konfiguraciju kamere Arducam 5MP Mini Plus PDH računalo koristi I^{2}C (engl. *Inter-Integrated Circuit*) komunikaciju. S obzirom na to da se za razvoj programske potpore PDH računala koriste *Low-Layer* biblioteke, potrebno je razumijevanje načina rada I^{2}C periferije odabranog mikrokontrolera kako bi se ispravno implementirali upravljački programi. U nastavku slijedi općeniti opis I^{2}C protokola kao i razlike između I^{2}C periferijski sklopova na prethodno korištenom (STM32F407VGT6) i trenutačnom (STM32L471VGT6) mikrokontroleru.

2.1. I^{2}C protokol

I^{2}C je jednostavna dvosmjerna sinkrona serijska sabirnica razvijena od strane *Philips Semiconductors* (sada *NXP Semiconductors*) 1982. godine [6]. Koristi dvije linije:

- serijska podatkovna linija (SDA, *Serial Data Line*),
- serijska taktna linija (SCL, *Serial Clock Line*).

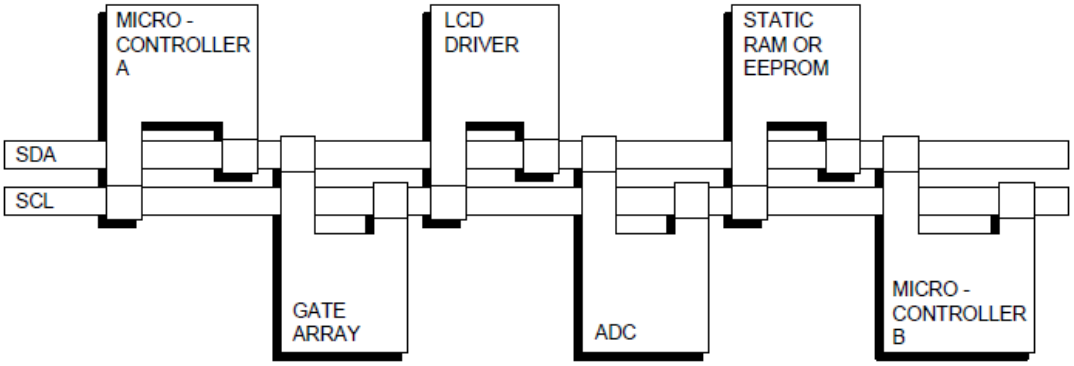
Obje linije su pritegnute na visoku logičku razinu preko *pull-up* otpornika. Moguće brzine prijenosa su:

- do 100 kbit/s u *Standard-mode* načinu rada,
- do 400 kbit/s u *Fast-mode* načinu rada,
- do 1 Mbit/s u *Fast-mode Plus* načinu rada,
- do 3.4 Mbit/s u *High-speed* načinu rada.

Navedene brzine se koriste kod dvosmjernog prijenosa, a moguća je i brzina do 5 Mbit/s u jednosmjernom prijenosu. Više uređaja se može spojiti na jednu sabirnicu, a svaki uređaj je prepoznatljiv po svojoj jedinstvenoj adresi i može se ponašati kao prijamnik ili odašiljač, ovisno o funkciji uređaja [2]. Protokol najčešće, a tako i u ovom slučaju, koristi 7-bitno adresiranje, a moguće je i korištenje 10-bitnog adresiranja. Osim konfiguracije prijamnika i odašiljača, uređaj također može biti *master* uređaj

ili *slave* uređaj tijekom prijenosa podataka. *Master* uređaj je uređaj koji inicijalizira prijenos podataka na sabirnici i generira signal takta kako bi omogućio prijenos. U tom trenutku, bilo koji uređaj koji je adresiran smatra se *slave* uređajem.

Na I^2C sabirnicu se također može spojiti više *master* uređaja, a primjer jednog takvog spoja sa dva mikrokontrolera dan je na slici 2.1. Prijenos podataka bi možda



Slika 2.1: Primjer I^2C sabirnice sa spojena dva mikrokontrolera [2]

mogao izgledati ovako:

1. Mikrokontroler A želi poslati podatke mikrokontroleru B:
    - mikrokontroler A (*master* uređaj) adresira mikrokontroler B (*slave* uređaj)
    - mikrokontroler A (*master*-odašiljač) šalje podatke mikrokontroleru B (*slave* uređaj-prijamnik)
    - mikrokontroler A prekida prijenos
2. Mikrokontroler A želi primiti podatke sa mikrokontrolera B:
    - mikrokontroler A (*master* uređaj) adresira mikrokontroler B (*slave*)
    - mikrokontroler A (*master*-prijamnik) prima podtke sa mikrokontrolera B (*slave*-odašiljač)
    - mikroknotroler A prekida prijenos.

U svakom od navedenih slučajeva mikrokontroler A je generirao takt i prekidao prijenos. Kod prijenosa podataka na I^2C sabirnici, *master* uređaj uvijek generira signal takta. U ovom radu korišten je samo jedan mikrokontroler, odnosno *master* uređaj, pa će se u daljnjem tekstu podrazumijevati samo taj slučaj.

2.1.1. Opis komunikacije i vremenski dijagram

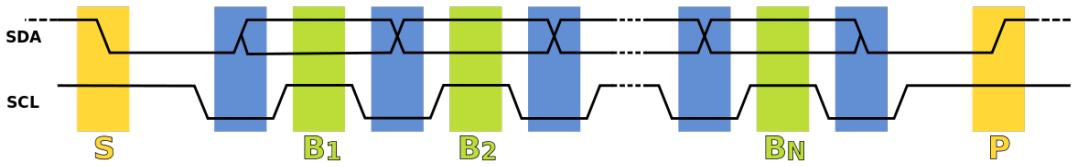
I^2C komunikacija započinje sa *start* simbolom i završava sa *stop* simbolom. Komunikacijom se može čitati ili pisati ovisno o R/W bitu u adresi. Struktura adresiranja kod 7-bitne adrese je prikazana u tablici 2.1.

	Adresno polje							R\W
Pozicija bita u bajtu	7	6	5	4	3	2	1	0
Značenje	MSB						LSB	1=READ, 0=WRITE

Tablica 2.1: Struktura adresiranja kod 7-bitne adrese [6]

Iz tablice je vidljivo da najmanje značajan bit označava želi li se podatak čitati ili pisati.

Imajući na umu izgled adresnog bajta, vremenski dijagram tipčne I^2C komunikacije prikazan je na slici 2.2.



Slika 2.2: Vremenski dijagram I^2C komunikacije [6]

- Prijenos podataka se inicijalizira *start* uvjetom (S) tako da SDA linija prijeđe u nisku logičku razinu dok SCL linija ostaje u visokoj logičkoj razini.
- (Plavo područje) SCL prelazi u nisku logičku razinu i SDA postavlja prvi podatkovni bit dok je SCL u niskoj logičkoj razini.
- (Zeleno područje) Podaci se primaju dok SCL poraste za prvi bit (B_1). Kako bi podaci bili valjani, SDA se ne smije promijeniti između rastućeg brida SCL-a i sljedećeg padajućeg brida.
- Postupak se ponavlja, SDA se postavlja dok je SCL u niskoj razini, a podaci se čitaju dok je SCL u visokoj razini (B_2 do B_n).
- Nakon posljednjeg bita slijedi taktni impuls, tijekom kojeg SDA prelazi u nisku razinu pripremajući se za *stop* uvjet.
- Signalizira se *stop* uvjet kada SCL prijeđe u visoku logički razinu, nakon čega slijedi prelazak u visoku logičku razinu SDA signala.

- (Plavo područje) SCL prelazi u nisku logičku razinu i SDA postavlja

Start i *stop* uvjete uvijek generira *master* uređaj. Nakon svakog bajta prijamnik šalje odašiljaču ACK bit kojim se signalizira uspješno primanje podatka, odnosno NACK bit kojim se signalizira neuspješno primanje podatka. ACK i NACK bitovi se nazivaju signalom potvrde i definiraju se na sljedeći način: odašiljač otpušta SDA liniju tijekom potvrdnog takta kako bi prijamnik mogao spustiti SDA na nisku razinu na kojoj i ostaje tijekom visoke razine takta. Ako SDA ostaje u visokoj razini tijekom devete periode takta, to predstavlja NACK (engl. *Not Acknowledge*) signal, a suprotan slučaj predstavlja ACK (engl. *Acknowledge*) signal. Ako je došlo do NACK signala, *master* uređaj može generirati *stop* uvjet kako bi prekinuo prijenos ili može ponovno generirati *start* uvjet kako bi započeo novi prijenos. Vremenski dijagram cijele komunikacije s potvrdnim signalima prikazan je na slici 2.3.

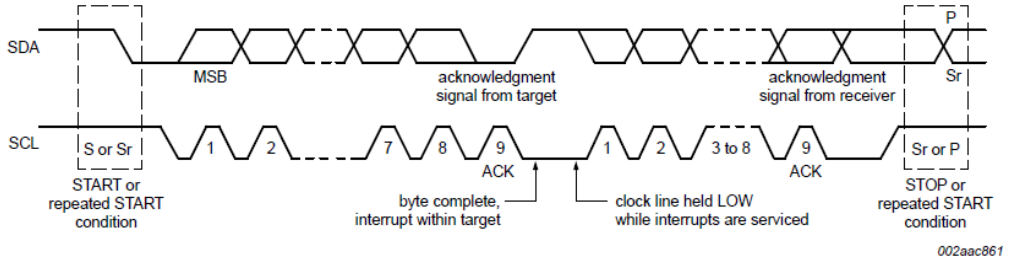


Figure 6. Data transfer on the I²C-bus

Slika 2.3: Prijenos podataka na I^2C sabirnici [2]

2.2. Razlika I^2C periferije na STM32L471VGT6 i STM32F407VGT6 mikrokontrolerima

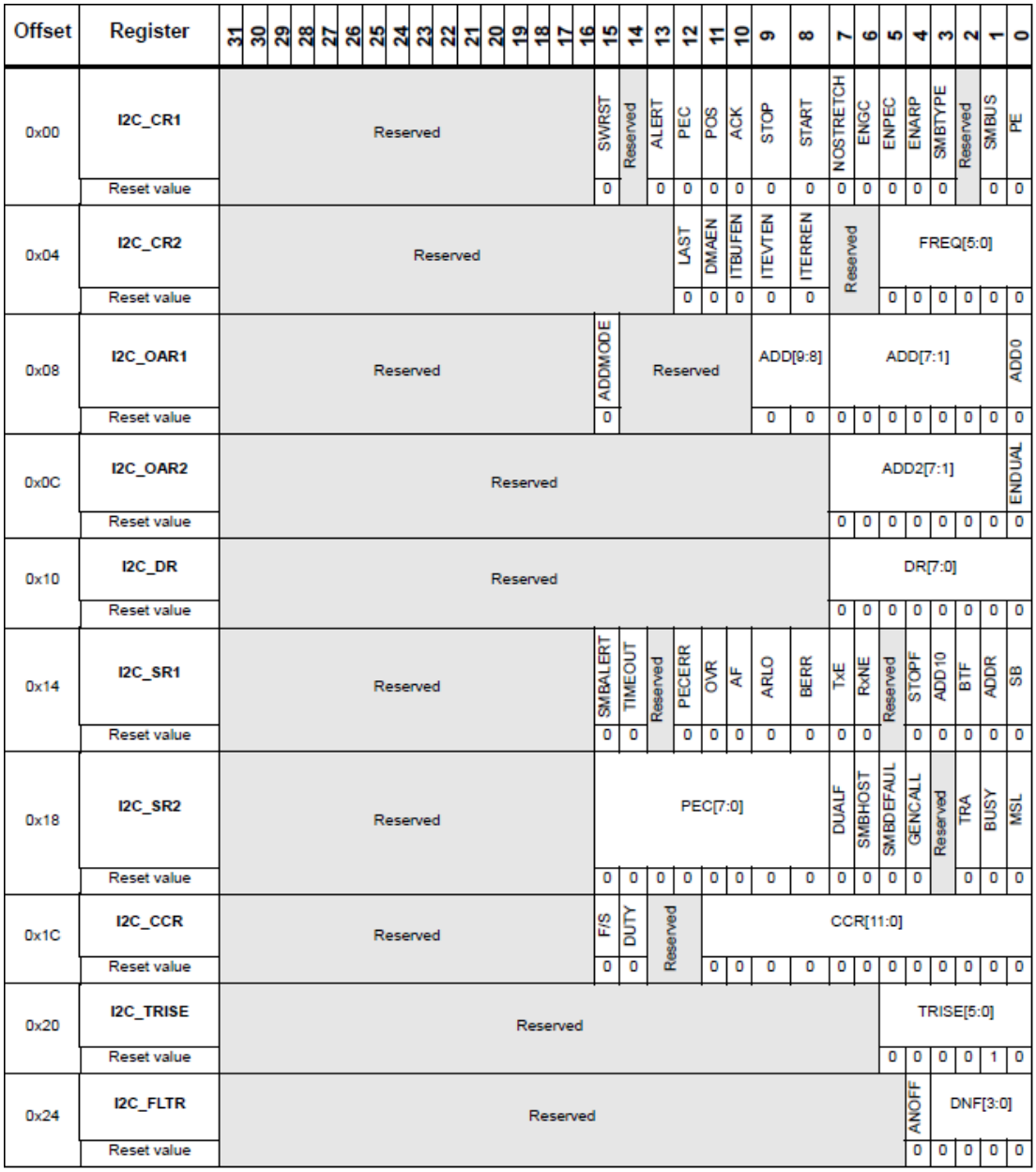
Tijekom prijenosa koda sa starog mikrokontrolera na novi, primjećeno je da postoji razlika između struktura I^2C periferija. Točnije, postoji razlika između registarskih mapa na dvama periferijama, koje su vidljive usporedbom slika 2.4, 2.5 i 2.6.

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	I2C_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN		ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value									0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x4	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0xC	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Slika 2.4: Registarska mapa I^{2}C periferije STM32L471VGT6 mikrokontrolera [5]

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Slika 2.5: Registarska mapa I^{2}C periferije STM32L471VGT6 mikrokontrolera - nastavak [5]



Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	Reserved																SWRST	Reserved	ALERT	PEC	POS	ACK	STOP	START	NOSTRETCH	ENGC	ENPEC	ENARP	SMBTYPE	Reserved	SMBUS	PE
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0		0	0
0x04	I2C_CR2	Reserved																			LAST	DMAEN	ITBUFEN	ITEVTEN	ITERREN	Reserved	FREQ[5:0]						
	Reset value																				0	0	0	0	0		0	0	0	0	0	0	0
0x08	I2C_OAR1	Reserved																ADDMODE	Reserved					ADD[9:8]		ADD[7:1]							ADD0
	Reset value																	0						0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Reserved																								ADD2[7:1]							ENDUAL
	Reset value																									0	0	0	0	0	0	0	0
0x10	I2C_DR	Reserved																								DR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x14	I2C_SR1	Reserved																SMBALERT	TIMEOUT	Reserved	PECERR	OVR	AF	ARLO	BERR	TxE	RxNE	Reserved	STOPF	ADD10	BTF	ADDR	SB
	Reset value																	0	0		0	0	0	0	0	0	0		0	0	0	0	0
0x18	I2C_SR2	Reserved																PEC[7:0]								DUALF	SMBHOST	SMBDEFAUL	GENCALL	Reserved	TRA	BUSY	MSL
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x1C	I2C_CCR	Reserved																F/S	DUTY	Reserved	CCR[11:0]												
	Reset value																	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	I2C_TRISE	Reserved																										TRISE[5:0]					
	Reset value																											0	0	0	0	1	0
0x24	I2C_FLTR	Reserved																										ANOFF	DNF[3:0]				
	Reset value																											0	0	0	0	0	0

Slika 2.6: Registarska mapa I^{2}C periferije STM32F407VGT6 mikrokontrolera[4]

Vidljiva je razlika između količine registara, raspodjele i značenja njihovih bitova, kao i njihovih imena, što implicira različite funkcionalnosti pojedinih registara. Tako, npr. I^{2}C periferija kod STM32F407VGT6 sadržava 2 status registra: I2C_SR1 i I2C_SR2, dok kod STM32L471VGT6 postoji samo jedan status registar I2C_ISR. Ta razlika je bitna zato što se tijekom prijenosa podataka na I^{2}C sabirnici trebaju provjeravati razne zastavice koje se mijenjaju tijekom komunikacije, kao što je npr. zastavica za prazni odašiljački registar (STM32F407VGT6: registar I2C_SR1 bit 7, STM32L471VGT6: bit 0), zastavica za puni prijamnički registar (STM32F407VGT6: registar I2C_SR1, bit 6, STM32L471VGT6: bit 2), zastavica za završetak prijenosa

(STM32F407VGT6: ne postoji, STM32L471VGT6: bit 6) itd.

Vidljivo je također da kod STM32L471VGT6 postoji zastavica ADDR, koja inače kod STM32F407VGT6 signalizira uspješan primitak adrese uređaja mete, a kod STM32L471VGT6 ta zastavica se koristi isključivo u *slave* načinu rada, tako da ta zastavica nije bitna za ovaj projekt. Kako onda mikrokontroler zna da je poslana adresa točna? Naime, STM32L471VGT6 ima poseban registar za pohranu adrese uređaja mete, pa kada mikrokontroler pošalje *start* uvjet on automatski nakon završetka *start* uvjeta pošalje i adresu uređaja mete, a uspješan primitak adrese signalizira zastavica I2C_ISR_TXIS kod slanja podataka, odnosno I2C_ISR_RXNE zastavica kod primitka podataka.

Vidljive su i razlike u raspodjeli zastavica u registrima, kao i razlike u funkcijama koje zastavice signaliziraju. Inače bi te razlike stvarale probleme kod konfiguracije I^2C periferije, no, kako je tu brigu riješio kod generator ugrađen u STM32CubeIDE razvojno okruženje, nije bila posvećena pažnja tim razlikama. Način implementacije spomenutih razlika u programsku podršku opisan je u poglavlju z.

3. SPI sučelje i DMA prijenos

SPI (engl. *Serial Peripheral Interface*) protokol se koristi za prijenos podataka između *flash* memorije i mikrokontrolera, odnosno međuspremnika kamere. Kako bi se oslobodili resursi na mikrokontroleru, za prijenos podataka između kamere i *flash* memorije se, u kombinaciji sa SPI protokolom, koristi i DMA (engl. *Direct Memory Access*) prijenos. S obzirom na to da se koriste *Low-Layer* biblioteke, potrebno je razumjeti način rada SPI i DMA periferija mikrokontrolera. U ovom poglavlju bit će opisan SPI protokol i DMA prijenos i bit će istaknute razlike i problemi kod prilagođavanja programske podrške za STM32L471VGT6 mikrokontroler.

3.1. SPI protokol

SPI je sinkrono serijsko komunikacijsko sučelje koje se koristi za komunikaciju na kratkim udaljenostima, pretežito u ugradbenim računalnim sustavima [7].

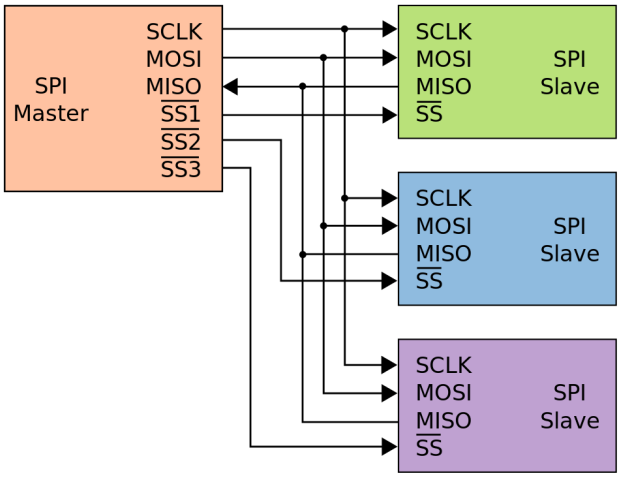
SPI uređaji komuniciraju u *full-duplex* načinu rada koristeći *master-slave* arhitekturu, obično sa jednim *master* uređajem. *Master* (upravljač) uređaj proizvodi okvir za čitanje i pisanje. Više *slave* uređaja može biti spojeno na jedan upravljač tako da se aktivira određeni *chip select* signal za pojedini uređaj.

3.1.1. Opis sučelja

SPI sabirnica se sastoji od četiri signala:

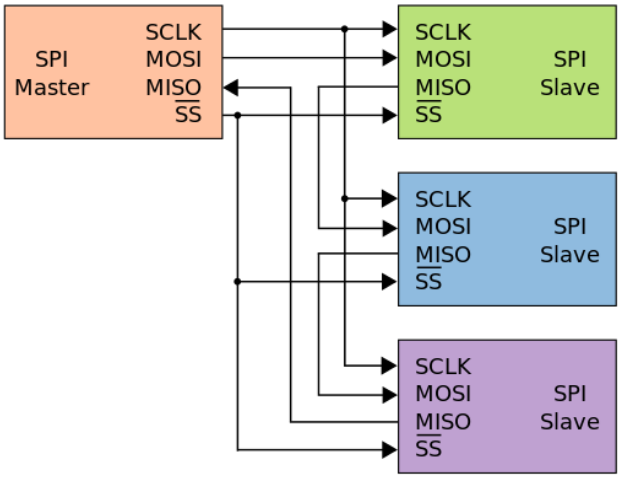
- SCLK: Serijski takt (izvor je *master* uređaj)
- MOSI: *Master Output Slave Input* (izvor podataka iz *master* uređaja)
- MISO: *Master Input Slave Output* (izvor podataka iz *slave* uređaja)
- CS/SS: *Chip/Slave Select* (aktivan nisko, signal iz *master* uređaja, označava da se prenose podaci)

MOSI na *master* uređaju se spaja na MOSI na *slave* uređaju, dok se MISO na *master* uređaju se spaja na MISO na *slave* uređaju. CS/SS se koristi za pokretanje komunikacije između *slave* i *master* uređaja. Za svaki *slave* uređaj postoji zaseban CS/SS priključak na *master* uređaju. Takav način spajanja se naziva neovisni *slave* uređaj. Primjer spajanja tri *slave* uređaja na jedan *master* uređaj u konfiguraciji neovisnog *slave* uređaja prikazan je na slici 3.1.



Slika 3.1: Spoj tri *slave* uređaja na jedan *master* uređaj u konfiguraciji neovisnog *slave* uređaja. Vidljivo je da *master* uređaj ima tri SS priključka, a svaki odgovara jednom *slave* uređaju, dok se SCLK, MOSI i MISO linije međusobno dijele između *slave* uređaja [7]

Moguće je još spojiti uređaje u konfiguraciju ulančavanog *slave* uređaja. U toj konfiguraciji *slave* uređaji dijele isti CS/SS, a ulančavanjem preko MISO/MOSI linija podaci se prenose prema načelu posmačnog registra, koji je objašnjen u sljedećem potpoglavlju. Prikaz spajanja tri *slave* uređaja se nalazi na slici 3.2.



Slika 3.2: Spoj tri *slave* uređaja na jedan *master* uređaj u konfiguraciji ulančavanog *slave* uređaja [7]

3.1.2. Način rada

SPI sabirnica radi s jednim *master* uređajem i jednim ili više *slave* uređaja. Ako se koristi jedan *slave* uređaj, onda CS signal može biti postavljen u nisku logičku razinu, ako *slave* uređaj to dopušta. Neki *slave* uređaji zahtijevaju padajući brid CS signala kako bi započela komunikacija. Ako se koristi više *slave* uređaja potreban je zaseban CS signal *master* uređaja za svaki *slave* uređaj.

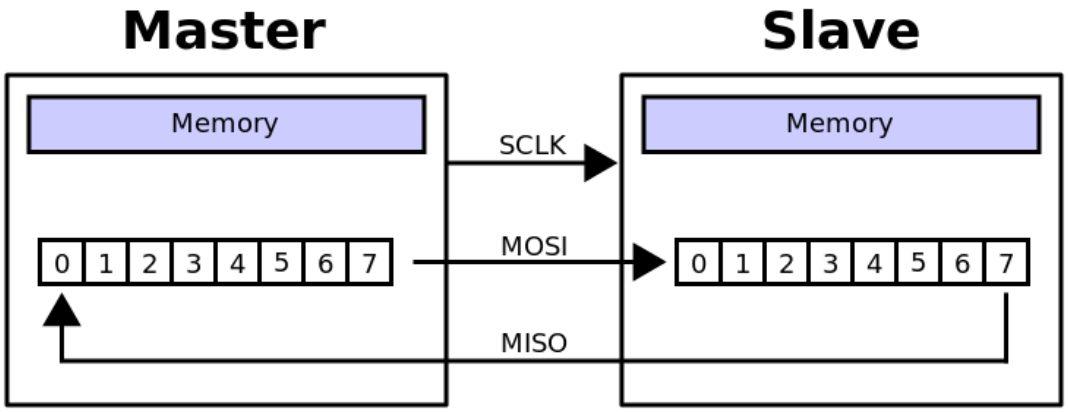
Prijenos podataka

Za početak komunikacije *master* uređaj konfigurira takt koristeći frekvenciju koju podržava *slave* uređaj, obično do nekoliko MHz. *Master* uređaj zatim odabire *slave* uređaj postavljanjem CS linije u nisko logičko stanje. Ako je potreban period čekanja, npr. za AD (analogno-digitalnu) pretvorbu, *master* uređaj mora pričekati minimalno taj period vremena prije puštanja takta.

Tijekom svakog perioda takta obavlja se prijenos podataka u *full-duplex* načinu rada. To znači da *master* uređaj pošalje jedan bit na MOSI liniju, koji *slave* uređaj pročita, dok u isto vrijeme *slave* uređaj šalje jedan bit na MISO liniju, koji *master* uređaj pročita. Takva sekvenca se održava čak i kada se izvodi jednosmjerni prijenos podataka.

Prijenosi podataka uključuju dva posmačna registra zadane veličine, npr. 8 bitova, jedan u *master* i drugi u *slave* uređaju. Registri su spojeni u topologiji virtualnog

prstena (slika 3.3).



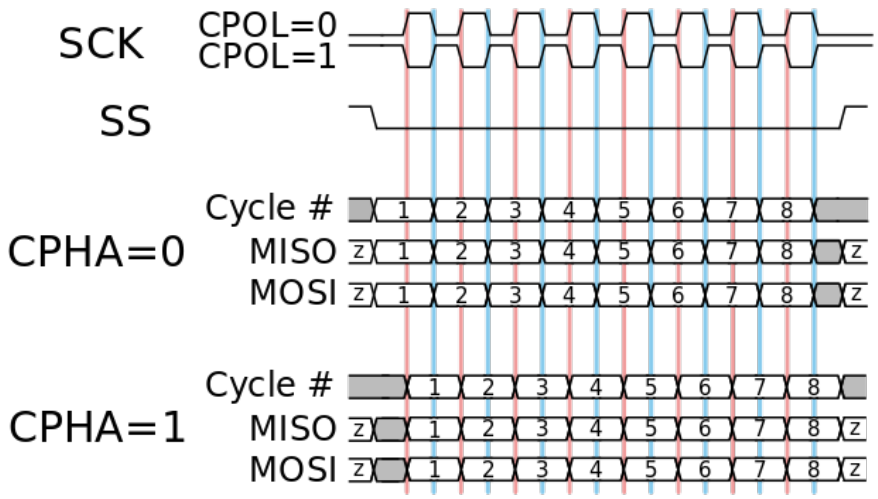
Slika 3.3: Tipičan spoj dvaju posmačna registra koji formiraju kružni međuspremnik [7]

Podaci se obično pomiču tako da se prvo pomakne najznačajniji bit. Na brid takta, *master* i *slave* uređaj pomaknu bit i pošalju ga na prijenosnu liniju. Na sljedeći brid takta, na svakom prijamniku bit se uzorkuje s prijenosne linije i postavlja se kao novi najmanje značajni bit u posmačnom registru. *Master* i *slave* uređaji u potpunosti razmjene podatke u registrima nakon što se svi bitovi u registrima prebace. Ako je potrebno razmijeniti još podataka, posmačni registri se ponovno napune te se postupak ponavlja, a prijenos se može obavljati za bilo koji broj perioda takta. Kada je prijenos dovršen, *master* uređaj prestaje davati takt i obično isključi CS signal, odnosno postavi ga na visoku razinu.

Prijenos se obično obavlja u riječima širine 8 bitova, no moguća je i širina riječi od 12 bita, ili čak 12 bitova, koji se koristi za digitalno-analogne i analogno-digitalne pretvornike.

Polaritet takta i faza

Osim što mora podesiti frekvenciju takta, *master* uređaj mora isto tako podesiti polaritet takta (CPOL, engl. *Clock Polarity*) i fazu (CPHA, engl. *Clock Phase*) ovisno o podacima. Vremenski dijagram je prikazan na slici 3.4.



Slika 3.4: Vremenski dijagram koji pokazuje polaritet takta i fazu. Crvene linije označuju vodeće bridove, a plave linije označavaju prateće bridove.

CPOL određuje polaritet kanala. Polaritet može biti invertiran jednostavnim inverterom.

- Ako je CPOL = 0, onda takt miruje u niskom logičkom stanju, a svaki period se sastoji od impulsa visokog logičkog stanja. To znači da je vodeći brid rastući brid, a prateći brid padajući brid.
- Ako je CPOL = 1, onda takt miruje u visokom logičkom stanju, a svaki period se sastoji od impulsa niskog logičkog stanja. To znači da je vodeći brid padajući brid, a prateći brid rastući brid.

CPHA određuje fazu podatkovnih bitova u odnosu na takt.

- Ako je CPHA = 0, "izlazna" strana mijenja podatak na prateći brid prethodnog perioda takta, dok "ulazna" strana prihvaća podatak na (ili ubrzo nakon) vodeći brid perioda takta. Izlazna strana zadržava valjani podatak sve do pojave pratećeg brida trenutnog perioda takta.
- Ako je CPHA = 1, "izlazna" strana mijenja podatak na vodeći brid trenutnog perioda takta, dok "ulazna" strana prihvaća podatak na (ili ubrzo nakon) pratećeg brida perioda takta. Izlazna strana zadržava valjani podatak do pojave vodećeg brida sljedećeg perioda takta. Na zadnji period, *slave* uređaj zadržava valjani podatak na MISO liniji sve dok *slave* uređaj ne bude deselektiran.

MOSI i MISO signali su obično stabilni za vrijeme pola perioda takta, sve do sljedeće promjene takta. SPI *master* i *slave* uređaji mogu uzorkovati podatke u bilo kojem vremenu unutar te polovice periode takta.

Kombinacije različitih konfiguracija CPOL i CPHA bitova predstavljaju načine rada. Konvnecija je da CPOL predstavlja viši bit, dok CPHA predstavlja niži bit. Načini rada kod ARM-ovih mikrokontrolera su prikazani u tablici 3.1

SPI način rada	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tablica 3.1: SPI načini rada kod ARM-ovih mikrokontrolera [7]

3.2. DMA prijenos

4. Zaključak

Zaključujem ovaj projekt.

LITERATURA

[1] Filip Jurić. Upravljačko sklopovlje za prikupljanje i obradu senzorskih signala na CubeSat nanosatelitu. Završni rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2021.

[2] *UM10204 I^2C-bus specification and user manual*. NXP Semiconductors, 2021. Rev. 7.0.

[3] Goran Petrak. Programska potpora ugradbenog računalnog sustava za udaljeno prikupljanje fotografija putem satelita. Diplomski rad, Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2021.

[4] *RM0090 Reference manual STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs*. ST Microelectronics, 2021. Rev. 9.

[5] *RM0351 Reference manual STM32L47xxx, STM32L48xxx, STM32L49xxx and STM32L4Axxx, advanced Arm®-based 32-bit MCUs*. ST Microelectronics, 2021. Rev. 9.

[6] Wikipedia. I^2 c, 2022. URL `https://en.wikipedia.org/wiki/I%C2%B2C`. Preuzeto: 30. 05. 2022.

[7] Wikipedia. Serial peripheral interface, 2022. URL `https://en.wikipedia.org/wiki/Serial_Peripheral_Interface`. Preuzeto: 20. 06. 2022.

[8] FER ZKIST. FERSAT - opis projekta, 2022. URL `https://www.fer.unizg.hr/zkist/FERSAT/projekt`. Preuzeto: 18. 06. 2022.

Programska potpora za upravljanje kamerom na CubeSat nanosatelitu

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

Software for Camera Control on CubeSat Nanosatellite

Abstract

Abstract.

Keywords: Keywords.