VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2022 Bc. JAN NOVOTNÝ

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

MODULÁRNÍ ŘÍDICÍ SYSTÉM
MODULAR CONTROL SYSTEM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jan Novotný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Benešl**
ADVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Jan Novotný ***ID:*** 203306

Ročník: 2 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Modulární řídicí systém

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnou a vytvořit funkční vzorek modulárního řídicího systému. Práce je realizována ve spolupráci s firmou ELZACO spol. s r.o. Součástí je elektrické schéma zapojení, návrh desky plošných spojů a obslužná aplikace pro mikrokontroler.

1. Seznamte se s řídicími systémy dostupnými na trhu.
2.Seznamte se s operačními systémy reálného času (RTOS) pro mikrokontrolery.
3. Navrhněte schématické zapojení a desku plošných spojů pro modulární řídicí systém (modul s procesorem a další moduly s IO).
4. Desky plošných spojů realizujte, oživte navržené moduly a naprogramujte jednotlivé moduly.
5. Navrhněte komunikační protokol pro komunikaci jednotlivých modulů.
6. Ověřte celkovou funkčnost zařízení a porovnejte s produkty dostupnými na trhu.

DOPORUČENÁ LITERATURA:

Laplante, P. A.: Real-time Systems Design and Analysis. John Wiley & Sons, Inc., 2004, ISBN 0-471-22855-9.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 18.5.2022

Vedoucí práce: Ing. Tomáš Benešl
Konzultant: Ing. Roman Kubíček

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se věnuje návrhu elektroniky modulárního řídicího systému a všech potřebných atributů. V obecné části seznamuje čtenáře s problematikou průmyslových řídicích systémů a popisuje jejich funkcionalitu. Dále popisuje komunikační sběrnice a protokoly využívané v průmyslu pro přenos dat a popisuje operační systémy reálného času (RTOS). V praktické části se diplomová práce zabývá návrhem a realizací vlastního modulárního řídicího systému složeného z několika modulů. Modulární řídicí systém využívá svůj vlastní komunikační protokol pro výměnu dat mezi jednotlivými moduly. Jsou diskutovány různé možnosti využití tohoto modulárního řídicího systému.

KLÍČOVÁ SLOVA

ARM, Atmel, CAN bus, komunikace, komunikační protokol, MCU, MODBUS, modulární řídicí systém, PLC, RS485, RTOS, sběrnice

ABSTRACT

The work deals with design of electronics for modular control system and all the necessary attributes. In the general part the reader are acquaints with the issues of industrial control systems and describes their functionality. It also describes the communication buses and protocols used in the industrial data transmission and describes real-time operating systems (RTOS). In the practical part, the master thesis deals with the design of its own modular control system consisting of several modules. The modular control system uses its own communication protocol to exchange data between the individual modules. Various possibilities of using this modular control system are discussed.

KEYWORDS

ARM, Atmel, CAN bus, communication, communication protocol, MCU, MODBUS, modular control system, PLC, RS485, RTOS, bus

Vysázeno pomocí balíčku `thesis` verze 4.03; `http://latex.feec.vutbr.cz`

NOVOTNÝ, Jan. *Modulární řídicí systém*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2022, 217 s. Diplomová práce. Vedoucí práce: Ing. Tomáš Benešl

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jan Novotný
VUT ID autora:	203306
Typ práce:	Diplomová práce
Akademický rok:	2021/22
Téma závěrečné práce:	Modulární řídicí systém

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 10.5.2022 .....

.....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Tímto děkuji firmě ELZACO spol. s r.o., která mi nabídla zázemí pro vývoj a materiální podporu. Dále bych rád poděkoval Ing. Tomáši Benešlovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno 10.5.2022 .....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešní době se v průmyslu používají řídicí systémy, mezi něž se řadí PLC (Programmable Logic Controller) pro řízení technologií. Řídicí systémy musí splňovat několik různých standardů a norem. Na řídicí systémy jsou tedy kladeny čím dál větší požadavky jak z hlediska spolehlivosti, tak i výpočetního výkonu.

Trh je celkem nasycen řídicími systémy, jelikož každý významnější výrobce elektrotechnických komponentů má také svoje řídicí systémy. Většina programátorů se zaměřuje pouze na určité systémy, byť všechny systémy podléhají společným standardům a normám, tak jejich vývojová prostředí jsou rozlišná. Dalším specifikem mohou být také komunikační sběrnice a protokoly, které jsou obsaženy pouze na řídicích systémech společnosti, která je vyvinula.

Řídicí systémy lze rozdělit do 2 hlavních skupin a to na kompaktní a modulární řídicí systémy. Kompaktní řídicí systémy bývají převážně jednodušší, ale mají přesně dané periférie a musejí být vybírány na míru výsledné aplikace. Modulární řídicí systémy se skládají z několika modulů, kde hlavním modulem je procesorová jednotka (většinou obsahuje také komunikace) ke které se přidávají vstupně-výstupní periférie a podobu řídicího systému tak lze lépe přizpůsobit výsledné aplikaci.

Tato práce se zabývá návrhem modulárního řídicího systému podobnému PLC. Protože dnešní trh PLC je přesycen, tak cílem této práce není návrh PLC pro následnou výrobu a prodej pro průmyslové použití, ale navrhnout modulární řídicí systém vhodný pro řízení vodních mikrozdrojů. Modulární systém byl především vybrán pro možnost variability a jednodušší výměny poškozených součástí (modulů). Nepředpokládá se, že bude navržen řídicí systém pro výsledné využití, ale pouze pro prozkoumání technologických možností tohoto řešení. Výsledným přínosem má být optimální řídicí systém pro vodní mikrozdroj, který bude mít vhodnou velikost, periférie a cenu.

Práce byla rozdělena do několika dílčích kapitol:

- Popis jednotlivých průmyslových řídicích systému.
- Popis komunikačních sběrnic a protokolů.
- Popis operačních systémů reálného času pro embedded zařízení.
- Výběr vhodných komponentů důležitých pro modulární řídicí systém.
- Návrh jednotlivých modulů řídicího systému.
- Návrh komunikačního protokolu pro vnitřní komunikaci modulů modulárního řídicího systému.
- Realizace a ověření funkčnosti řídicího systému.

Tato práce je realizovaná ve spolupráci s firmou ELZACO spol. s r.o. a je volně navázaná na moji bakalářskou práci[24].

1 Průmyslové řídicí systémy

Pod pojmem průmyslového řídicího systému si většina odborníků okamžitě vybaví pojem PLC. PLC však není jediným názvem pro programovatelný automat, ale v praxi je již tak zaběhnutý, že se používá dodnes. Dnešní průmyslové automaty nejlépe vystihuje název **PAC** (Programmable Automation Controller), avšak norma *ČSN EN 61131–1* používá výhradně pojem **PLC**. Tato kapitola se tedy zabývá základním popisem používaných řídicích systému v průmyslu.

1.1 Rozdělení řídicích systémů

Průmyslové řídicí systémy můžeme rozdělit do následujících kategorií:

- Programovatelné relé.
- Kompaktní PLC.
- Modulární PLC.
- Safety PLC.
- Průmyslové PC – IPC.
- HMI (Human-Machine Interface) – panely pro obsluhu.

1.1.1 Programovatelné relé

Zařízení označená jako programovatelné relé (někdy také *Mikro PLC*) jsou mnohdy velmi primitivní, bez komunikace a s malým výpočetním výkonem v porovnání s PLC. Hlavními periferiemi jsou digitální vstupy a výstupy a jsou někdy vybaveny také analogovými vstupy či výstupy. Většinou se jedná o kompaktní provedení, ale na trhu se můžeme setkat i s modulárním řešením. Oproti PLC mají výhodu v nízké ceně a jednoduchosti (některá lze naprogramovat z integrovaného displeje). Používají se především tam, kde je nutné naprogramovat nějakou logiku či návaznost na čas, která je těžko dosažitelná pomocí diskrétních komponentů (relé, stykače, atd.).

1.1.2 Kompaktní PLC

Jako kompaktní PLC jsou označena zařízení, která spojují v jedno zařízení procesor, komunikaci a vstupně/výstupní periférie. Některá kompaktní PLC mají i HMI (uživatelský panel). Další vstupně/výstupní periférie se připojují pomocí komunikační sběrnice, kde je nutné brát v úvahu, zda rozšiřující periférie jsou od stejného výrobce jako je PLC a vývojové prostředí s nimi umožňuje práci jako s běžnými periferiemi, nebo se jedná o periférie třetích stran, kde je kladen požadavek na programátora, aby programoval komunikaci mezi řídicím systémem a těmito periferiemi.

1.1.3 Modulární PLC

Modulárním PLC lze označit takový řídicí systém, který má zvlášť procesorovou jednotku a periférie, ale propojují se komunikační sběrnicí. Systém komunikační sběrnice bývá různý v závislosti na výrobci a řadě řídicích systémů.

1.1.4 Safety PLC

Pokud se řeší projekt, kde je kladen důraz na bezpečnost, tak se musí vybrat bezpečnostní PLC, nebo takové PLC, které podporuje tyto funkce (standardní modulární PLC s bezpečnostními moduly, které implementují vlastní speciální logiku pro dosažení bezpečnosti). Nejčastějším prvkem dosažení bezpečnosti je zdvojení použitých prvků. Pokud se jedná o bezpečnostní PLC, jsou použity 2 procesory které vykonávají stejný kód a je jim nadřazen třetí procesor, který má za úkol porovnávat jejich výsledky a případně rozhodnout o jejich neshodě a také o případném zásahu (v závislosti na bezpečnostní kategorii zařízení). Existují ale také další řešení pro splnění bezpečnostních požadavků.

1.1.5 Průmyslové PC – IPC

Průmyslové PC je oproti klasickému PC upraveno tak, aby splňovalo průmyslové standarty a normy. Z mechanického hlediska se jedná především o pasivní chlazení, rozšíření pracovního teplotního rozsahu, mechanickou montáž například na DIN lištu a průmyslové sběrnice a konektory. Většinou na nich nenajdeme žádné vstupně/výstupní periférie, jako jsou digitální či analogové, protože se tyto periférie řeší jako vzdálené periférie.

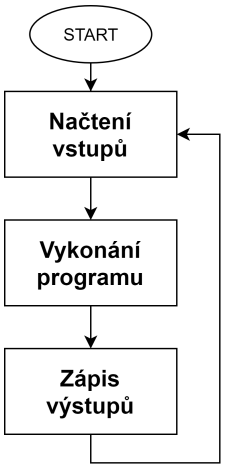
1.1.6 HMI – operátorské panely

Speciálním druhem průmyslových řídicích systémů jsou panely pro obsluhu HMI, nebo také SCADA systémy. Tyto systémy obsahují svůj vlastní program a slouží pro zobrazování parametrů operátorovi, nebo mu umožňují nastavit určité parametry, avšak **neobsahují žádný řídicí algoritmus**. Jako HMI bývá označován přímo operátorský panel (hardwarová součást) a SCADA označuje softwarovou část.

1.2 Funkce PLC

PLC systémy jsou navrženy na vykonávání cyklicky se opakujících úkonů. Program PLC pracuje se vstupně-výstupními perifériemi, které mají *HW* adresu, ale

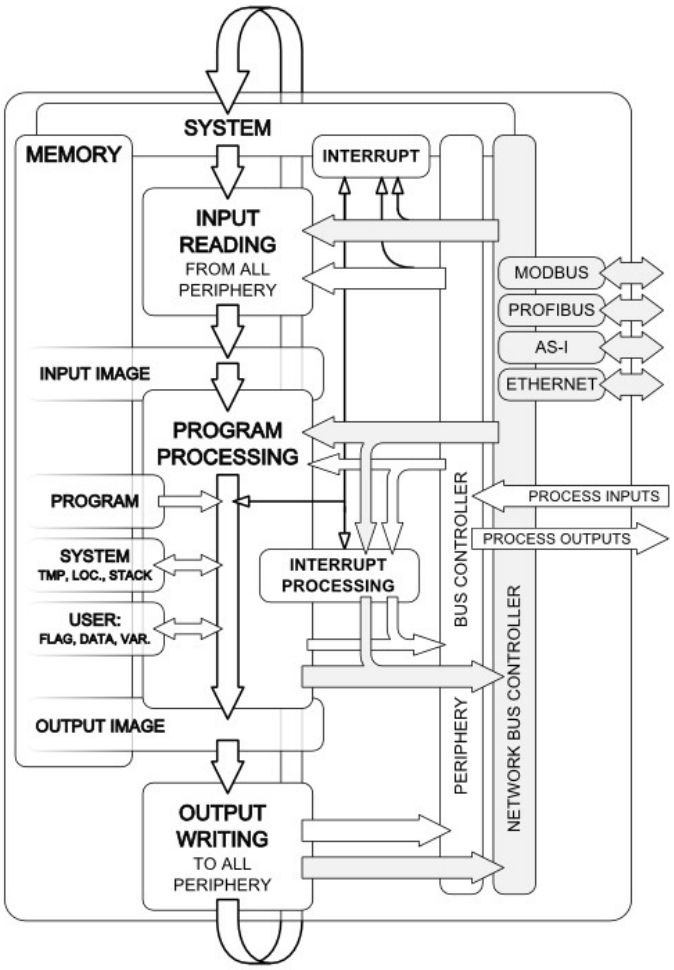
program PLC provede nejdříve načtení vstupů do své vnitřní paměti (dále jako obraz vstupů), se kterými dále může program pracovat, jelikož čtení vstupu při každé potřebě v kódu by zpomalovalo jeho vykonávání a dále je zaručeno, že program bude pracovat se stejnými hodnotami, se kterými je program začal vykonávat. Při vykonávání programu se výstupy zapisují do vnitřní paměti (obraz výstupů) a po dokončení cyklu programu dojde k zapsání všech hodnot na výstup periférií. Po zapsání hodnot na výstup skončil aktuálně vykonávaný cyklus a nadřazený program jej může spustit znovu. Průběh cyklu je znázorněn na obrázku 1.1.



Obr. 1.1: Cyklus vykonávání programu PLC.

Obrázek 1.1 ukazuje pouze zjednodušené vykonávání programu, jelikož zde není zahrnuta inicializace, komunikace a systémové potřeby (watchdog, plánování, čítače, přerušení, kontrola paměti, atd.).

Každý výrobce má tuto funkčnost implementovanou jinak a je proto nutné nastudovat dokumentace k programování. Někteří výrobci přistupují k práci s periferiemi, jak je naznačeno na obrázku 1.1, což nese nevýhodu, že celý systém podporuje pouze jednu cyklickou programovou smyčku. Ostatní výrobci, kteří mají implementované programování pomocí priority úloh a jejich nezávislosti (na jednom PLC běží souběžně několik programů), tak s periferiemi pracují v předem nastavené periodě. Hodnoty těchto periférií jsou přepisovány do a z vnitřní paměti. Z této vnitřní paměti si pak jednotlivé spuštěné úlohy načítají hodnoty vstupních periférií z této vnitřní paměti a nevyžadují spuštění čtení vstupních periférií a stejně tak při ukončení úlohy se výstupní hodnoty úlohy zapíší do vnitřní paměti a na výstup jsou přepsány v naplánované době. Obrázek 1.2 zobrazuje složitější model vykonávání programu PLC. V tomto obrázku je zobrazena jak práce s periferiemi, vykonávání programu, tak i komunikace, systémová správa a přerušení.



Obr. 1.2: Složitější cyklus vykonávání programu PLC[28].

1.3 Programování

Tato kapitola popisuje možnosti programování průmyslových řídicích systémů založených na normách a standardech.

1.3.1 Programovací jazyky

Nedílnou součástí programování průmyslových řídicích systémů bývá použití standardizovaného programovacího jazyka. Není stanoveno, které jazyky výrobce musí implementovat a každý výrobce má vlastní výběr. Programovací jazyky IL, ST, LD, FBD a SFC jsou definovány v normě *EN 61131–3*. Při programování složitějších projektů se pak může programátor rozhodovat, který programovací jazyk si vybere pro jakoukoli část programu, pokud mu to vývojové prostředí umožní. Různé problematiky totiž řeší každý programovací jazyk jinak a v jednom programovacím

jazyce může být určitý algoritmus jednodušeji implementován, než v ostatních programovacích jazycích. Výsledkem pak může být například program, který je naprogramován pomocí ladder diagramu a obsahuje funkce, které jsou naprogramované pomocí strukturovaného textu. Ne všechny programovací jazyky mají možnost využívat stejné systémové funkce než ostatní a kód v nich by tak byl mnohem složitější nebo nerealizovatelný.

IL – Instruction List

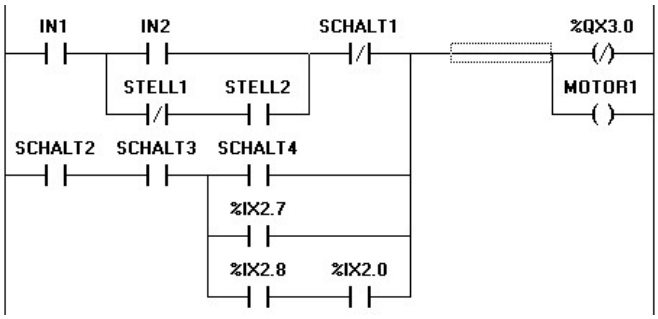
Je nejzákladnějším textovým programovacím jazykem, od kterého se nyní upouští, protože není vhodný pro řešení velmi složitých projektů a program se pak stává nepřehledným. Program je skládán z posloupnosti jednoduchých příkazů a je velmi obdobný asembleru používaném pro mikrokontroléry.

Jeho hlavní výhodou je přesná definice posloupnosti příkazů, jeho rychlost a paměťová nenáročnost. Nevýhodou je délka napsaného kódu a horší přehlednost.

LD – Ladder Diagram

Ladder diagram byl z historického hlediska velmi průlomový pro použití v programovatelných automatech, jelikož se nejvíce podobal elektrickému zapojení a porozuměli mu tedy i elektrikáři, kteří se nevěnovali programování. Jedná se o grafický programovací jazyk, který se vykonává po takzvaných příčkách.

Hlavní výhodou ladder diagramu je jeho přehlednost, jasně definovaná posloupnost zápisu a dobrá práce s logickými funkcemi. Nevýhodu nalezne při programování složitých programových sekvencí, kde kód je velmi rozsáhlý (dlouhý). Další nevýhodou může být složitější práce například s poli či řetězci.



Obr. 1.3: Programovací jazyk "Ladder Diagram"[5].

ST – Structured Text

Strukturovaný text je velice podobný programování v jazyce C a je tak velice vhodný pro práci s poli, řetězci či zpracování analogových hodnot. Nevýhodou může

být jeho horší přehlednost například při zpracovávání velkého množství logických signálů.

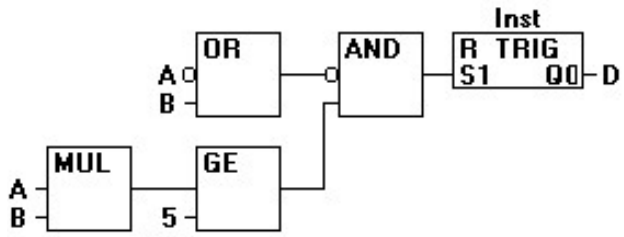
Výpis 1.1: Příklady příkazů jazyku ST[9].

```
1  //Konstrukce příkazu For
2  INT_Var :INT;
3
4  FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <stepsize>} DO
5      <instructions>
6  END_FOR;
7
8  //Konstrukce příkazu Repeat
9  REPEAT
10     <instructions>
11 UNTIL <Boolean expression>
12 END_REPEAT;
13
14 //Příkaz While
15 WHILE <Boolean expression> DO
16     <instruction>
17 END_WHILE;
18
19 //Konstrukce podmínky pomocí příkazu If
20 IF <Boolean_printout1> THEN
21 <IF_instructions>
22 {ELSIF <Boolean_printout2> THEN
23 <ELSIF_instructions1>
24 .
25 .
26 ELSIF <Boolean_printout n> THEN
27 <ELSIF_instructions n-1>
28 ELSE
29 <ELSE_instructions>}
30 END_IF;
31
32 //Konstrukce příkazu Case
33 CASE <Var1> OF
34 <Value 1>:
35     <instruction 1>
36 <Value 2>:
37     <instruction 2>
38     ...
```

```
39 <Value n>:
40     <instruction n>
41 ELSE
42     <ELSE-instruction>
43 END_CASE;
```

FBD – Function Block Diagram

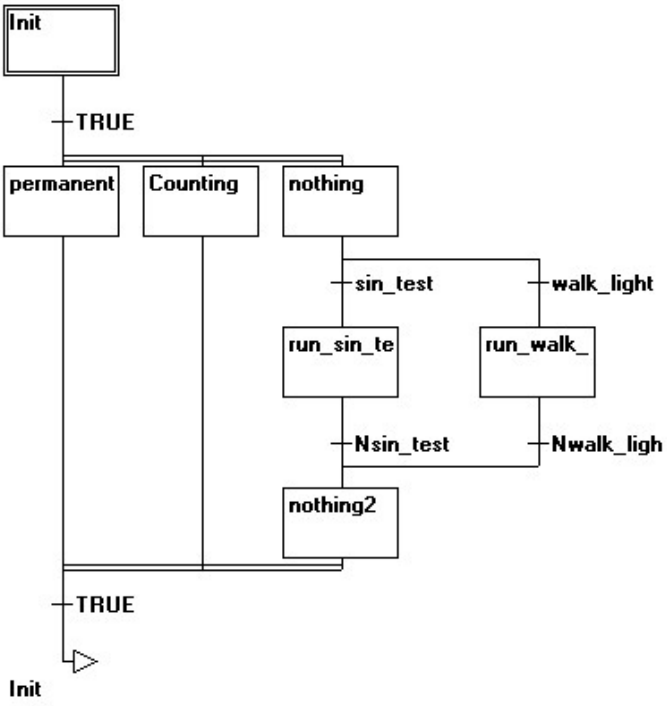
Function block diagram, neboli také schéma funkčních spojů je grafický programovací jazyk, který definuje datový tok mezi jednotlivými bloky (dataflow language). Program je rozčleněn do příček jako u ladder diagramu, kterému je velice podobný, jen s jinou grafickou reprezentací. Tento programovací jazyk je především vhodný pro zpracování logických signálů a nehodí se na zpracování složitých algoritmů, polí, řetězců a komunikací.



Obr. 1.4: Programovací jazyk FBD[7].

SFC – Seguential Function Chart

Také jako vývojový diagram je graficky orientovaný programovací jazyk, který definuje určité stavy a rekce na ně s možností větvení. Je nejvhodnější pro naprogramování sekvenční logiky algoritmu, kde jednotlivé stavy volají funkce napsané v jiných programovacích jazycích, které jsou přehlednější pro detailní popis algoritmu. Je tedy nevhodný pro zpracování velkého množství analogových a logických signálů a složité algoritmy.



Obr. 1.5: Programovací jazyk SFC[8].

C a C++

Někteří výrobci umožňují také programovat v programovacím jazyce C (nejčastěji verze **ANSI C** nebo **C99**) a C++. Tyto programovací jazyky nejsou doporučovány, či jsou označeny za nevhodné podle normy *ČSN EN 61508–3*. Hlavním důvodem jejich nedoporučení je nedefinování chování v určitých stavech, kterých v normě **C99** může být více jak 190.

Výpis 1.2: Příklady nedefinovaného chování v jazyce C[15].

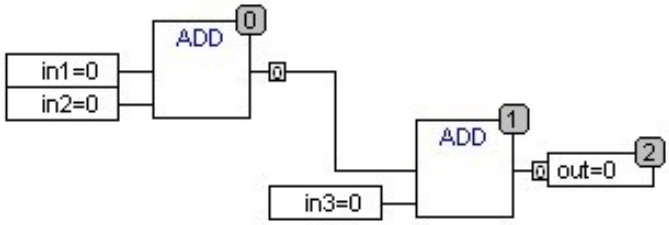
```
1  //Dělení nulou
2  int main()
3  {
4      int x = 25, y = 0;
5      int z = x / y;
6      return z;
7  }
8
9  //Neinicializovaná proměnná
10 int main()
11 {
12     bool val;
```

```
13      if (val)
14          printf("TRUE");
15      else
16          printf("FALSE");
17  }
18
19  //Přístup do paměti mimo pole
20  int main()
21  {
22     int arr[5];
23
24     //Můžeme přistupovat do arr[5] v poslední iteraci,
25     // i když není součástí definování pole.
26     for (int i=0; i<=5; i++)
27        printf("%d ", arr[i]);
28  }
```

Například výrobce **B&R** umožňuje napsat celý program pro PLC v jazyku C a **Unitronics** umožňuje napsat pouze jednoduché funkce, které jsou volány z hlavního programu.

CFC – Continuous Function Chart

Tento programovací jazyk je podobný *FBD*, jen s tím rozdílem, že jednotlivé bloky se vkládají do volné plochy. Výsledný program může připomínat elektronické schéma. Při složitých projektech se může stát program nepřehledným a v případě několika oddělených podprogramů není zaručena jejich posloupnost a záleží na překladači, který kód spustí dříve.



Obr. 1.6: Programovací jazyk CFC[6].

1.3.2 Programovací prostředí

Každý výrobce PLC má svoje vlastní programovací prostředí a většinou nelze používat žádná jiná programovací prostředí. Existují ale *Open-source* projekty, které

využívají možnost více programovacích prostředí, která mají podporu několika výrobců řídicích systémů. Tito výrobci se pak musejí řídit pravidly těchto programovacích prostředí a napsat pro ně takzvané *porty*, které umožňují kompatibilitu.

Hlavní podmínka pro výběr řídicího systému by se neměla zaměřovat pouze na hardware, ale také na možnosti programovacího prostředí. Výrobci řídicích systémů mají také určitou politiku v tomto odvětví, kde programovací prostředí může být pouze jedno pro všechny druhy řídicích systémů nebo mít několik rozdělených aplikací (například pro programování PLC a pro návrh HMI). Dalším rozdílem může být také zpoplatnění těchto programovacích prostředí, či jejich jednotlivých funkcí.

1.3.3 Pokročilé metody průmyslové automatizace

V moderní době se kladou po průmyslové automatizaci vysoké nároky na zefektivňování výroby. Pro větší efektivitu výroby se používají systémy ERP a MES. Systém MES slouží především pro plánování výroby v rámci výrobních kapacit a je úzce spjat s technologií výroby. Systémy ERP slouží pro řízení podnikové výroby a jsou nadřazeny systémům MES, avšak již nejsou spjaty s technologií výroby a soustředí se především na marketingové využití.

1.4 Průzkum trhu s PLC

Trh s řídicími systémy je velice rozšířený a většinou každý velký výrobce elektroniky má také svoji řadu řídicích systémů. Kvůli velké nabídce budou vypsáni někteří výrobci řídicích systémů známí především na českém trhu.

Výrobci řídicích systémů:

- Amit
- TECO
- SEA
- Siemens
- Unitronics
- Beckhoff
- Omron
- Wago
- Mitsubishi
- B&R
- Allen Breadly
- Schneider electronics
- Phoenix contacts
- Weidmüller

- Panasonic

Na trhu jsou také dostupná řešení "Open-Source". Nejznámější je nejspíše *Controllino*[10], které nabízí hardware založený na *Arduino* a je programovatelné v jeho prostředí (*Arduino IDE*), ale také ve spoustě dalších programovacích prostředích, která to umožňují. Dalším zařízením je *UniPi*[36], které vzniká v ČR a nabízí několik verzí. Nejznámější verze vychází založená na *Raspberry Pi* a další na procesoru od *NXP*. Na trhu jsou i další podobná řešení.

Programovací prostředí mívají větší výrobci většinou vlastní, ale existují také SW řešení nahrazující SW těchto výrobců a umožňují také implementaci do systémů vyvíjených jinou firmou (zabývá se především HW stránkou). Příkladem těchto řešení je *Rexygen*[29], *Mervis*[22] a další. Dalším řešením je *Open-Source* software, který má otevřený kód a může jej implementovat kdokoli. Jedním takovým řešením je *OpenPLC*[27], který řeší především implementaci standardu `EN~61131-3`, ale jeho *runtime* je napsaný pro *Raspberry Pi*.

2 Průmyslová komunikace

Tato kapitola se zabývá popisem komunikačních sběrnic a také komunikačními protokoly. V průmyslu se kladou odlišné nároky na komunikační sběrnice oproti klasickým komunikačním sběrnicím používaným v IT odvětví. Největší rozdíl je v objemu přenášených dat, kde v průmyslu stačí přenášet data o velikostech pár bitů až několik bytů, zato IT odvětví řeší přenos větších dat, která mohou mít i velikost několik *GB*. Ovšem také průmyslové sběrnice kladou mnohem vyšší požadavky na determinismus, chybovost, délku sběrnice, odezvu a spolehlivost. Např. IT odvětví neřeší, že data přijdou o sekundu později, ale v průmyslových aplikacích může být toto fatální.

Existuje velké množství průmyslových sběrnic, tak i protokolů. Některé protokoly jsou spjaty pouze s jednou specifickou komunikační sběrnicí a naopak, některé komunikační protokoly byly navrhovány odděleně od komunikačních sběrnic (využívají toho, co existuje). Určité komunikační protokoly, nebo sběrnice nemají volně dostupné dokumentace a jsou přístupné pouze členům komunity založené výrobcem a proto zde nemohou být popsány. Tato kapitola popisuje z velkého množství průmyslových komunikačních sběrnic pouze výběr, který by byl vhodný pro modulární řídicí systém navrhžený v této práci.

2.1 Komunikační sběrnice

Průmyslové komunikační sběrnice mají vyšší nároky na odolnost proti rušení a vzdálenosti, než v IT odvětví, pokud se nejedná o specializovaná zařízení. Vzdálenosti jednotlivých snímačů od řídicího systému mohou přesahovat i stovky metrů a proto je i cena kabelů rozhodující. Nejčastější je sběrnicová topologie, která přináší značnou úsporu kabeláže.

2.1.1 Sběrnice RS-232

Sběrnice *RS-232* patří k nejstarší sériové komunikační sběrnici, která se částečně používá i v dnešní době. Je určena pro komunikaci pouze mezi dvěma zařízeními. Pro základní komunikaci se využívá jeden vodič pro každý směr a signálová zem. Sběrnice je možné dále rozšířit i o další řídicí signály, které umožňují řízení toku informací[39].

Sběrnice využívá symetrického napájení a to $\pm 5\,V$, $\pm 10\,V$, $\pm 12\,V$ nebo $\pm 15\,V$. Logické úrovně pro datové signály jsou definovány jako:

- log. 0: $3\,V$ až $15\,V$
- log. 1: $-3\,V$ až $-15\,V$

Pro řídicí signály jsou napěťové úrovně invertované. Jejich definování je následující:

- log. 0: $-3\,V$ až $-15\,V$
- log. 1: $3\,V$ až $15\,V$

Maximální rychlost sběrnice *RS-232* je $115,2\,kb/s$ s krátkým kabelem a maximální délka kabelu je $15\,m$ při maximální rychlosti sběrnice $20\,kb/s$. Zajímavostí u této sběrnice je také definování strmosti hran na $30\,V/\mu s$ kvůli historii sběrnice, kde větší strmost hran měla negativní vliv na životnost obvodů (driverů) realizujících sběrnici.

Standard pro tuto sběrnici definuje asynchronní sériovou komunikaci, napěťové úrovně, pořadí dat od *LSB* a konektory (*DE-9* a *DB-25*).

2.1.2 Sběrnice RS-485

Standard **EIA-485** pro sériovou komunikaci *RS-485* nese řadu výhod oproti *RS-232*, avšak specifikuje pouze elektrické požadavky na přijímače a vysílače. Další normy a standardy na ni tedy nahlížejí jako na popis fyzické vrstvy. Tato kapitola čerpá informace především z literatury [19].

Hlavní přednosti sběrnice *RS-485* jsou:

- Možnost napájení z $5\,V$ zdroje napětí.
- Až 32 zařízení na sběrnici (při úpravě driveru sběrnice až 256 zařízení).
- Rychlost sběrnice až $10\,Mbps$ při délce sběrnice do $12\,m$.
- Maximální délka sběrnice $1200\,m$ při rychlosti sběrnice do $100\,kbps$.
- Odolnost sběrnice pro napětí na vstupu od $-7\,V$ do $12\,V$ (může být vyšší).
- Dvouvodičové řešení sběrnice.
- Sběrnicová topologie.

Elektrické vlastnosti

Zásadní změna oproti *RS-232* je v elektrickém principu sběrnice, kde se nevyužívá vyhodnocování signálů proti zemi, ale vzájemné vyhodnocování na dvou vodičích. Pro správnou funkčnost je tedy nutné využít kroucené dvojlinky pro přenos elektrického signálu. Na obrázku 2.1 je znázorněn vysílač a přijímač. Napěťové úrovně pro vysílač jsou $\pm 1,5\,V$ a pro přijímač je vyhodnocovací napětí $\pm 200\,mV$. Jelikož signály jsou vyhodnocovány vzájemně proti sobě, není nutné napájet driver symetrickým napětím a záporná hodnota u napětí má význam pouze pro prohození napěťových úrovní mezi vstupy.

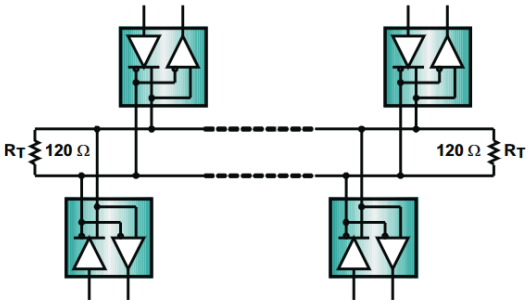
Obr. 2.1: Znázornění realizace driveru pro RS-485[19].

Zatěžování sběrnice

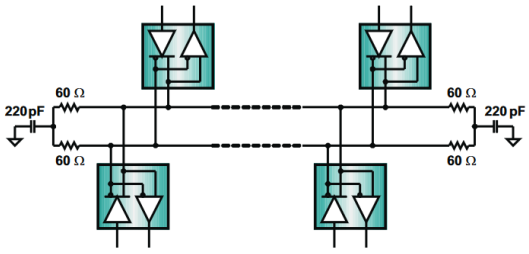
Podle standardu musí mít každý tranciever minimální impedanci odpovídající $12\,k\Omega$ (označována jako *UL - Unit Load*). Při dodržení této vstupní impedance je driver schopný pokrýt až 32 zařízení na sběrnici. V dnešní době existují drivery, které mají záměrně zvýšenou impedanci. Typickou hodnotou je $\frac{1}{8} \cdot UL$, což značí, že impedance zařízení je osmkrát vyšší, než v případě standardu a pokud se použijí na sběrnici tyto drivery, tak je možné mít na sběrnici až 256 zařízení.

Zakončovací odpory

Pro svoji správnou funkčnost sběrnice vyžaduje využití zakončovacích odporů, které slouží k přizpůsobení impedance sběrnice a potlačení odrazů. Zakončovací odpory mají typicky hodnotu $120\,\Omega$. Na obrázku 2.2a je znázorněno klasické zapojení zakončovacích odporů. Pro prostředí, která jsou více zarušená, se může použít zapojení znázorněné na obrázku 2.2b, kde je zakončovací odpor rozdělen na dva s hodnotou $60\,\Omega$ a je vložen kondenzátor jako dolnofrekvenční propust, která sníží rušení na sběrnici.



(a) Normální.

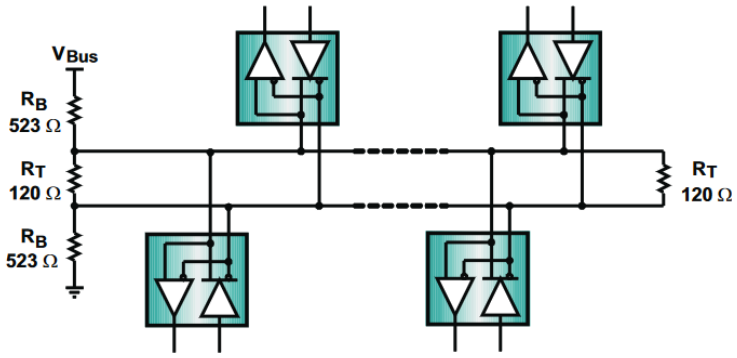


(b) S filtračním kondenzátorem.

Obr. 2.2: Zapojení zakončovacích odporů pro sběrnici RS-485[19].

Jelikož sběrnice je plovoucí a není určen potenciál na sběrnici, využívá se zapojení znázorněné na obrázku 2.3, kde odpory R_B napomáhají definování napětí v klidovém

stavu a toto zapojení je označováno jako *failsafe*. Hodnoty odporů R_B mohou být různé a je nutné brát v potaz jejich existenci, protože mohou mít vliv na zatěžování sběrnice (popsáno v další kapitole). Pokud se použije způsob návrhu, aby hodnota zakončovacího odporu R_T a sériově spojených odporů R_B (pro vysoké frekvence je zdroj napětí jako zkrat) je hodnota rovna 120 Ω, nebude toto zapojení zatěžovat sběrnici.



Obr. 2.3: Znázornění realizace driveru pro RS-485[19].

$$\frac{1}{120} = \frac{1}{2 \cdot R_B} + \frac{1}{R_T} \tag{2.1}$$

Rovnice 2.1.2 vyjadřuje vztah rovnováhy odporů R_B a R_T, aby nedocházelo k zatěžování sběrnice. Z této rovnice lze vyjádřit výpočet odporu R_B dle rovnice 2.1.2 a hodnota odporu R_T musí být zvolena.

$$R_B = \frac{60 \cdot R_T}{R_T - 120} \tag{2.2}$$

Pokud zvolíme hodnotu $R_T = 130\,\Omega$, hodnota R_B bude dle rovnice 2.1.2 780 Ω.

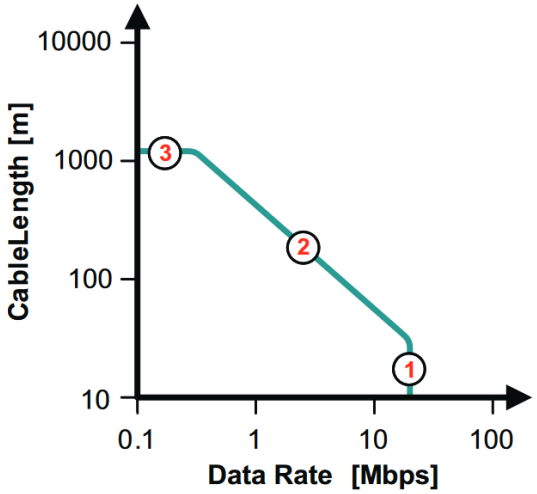
V případě volby $R_T = 120\,\Omega$ musíme brát v potaz přídavné zatěžování sběrnice. Pokud zvolíme $R_B = 523\,\Omega$, jak je tomu na obrázku 2.3, musíme počítat, že toto zapojení způsobí, jako by bylo ke sběrnici připojeno 20 dalších zařízení. Podle vzorce 2.1.2 můžeme vypočítat maximální počet zařízení na sběrnici. Pokud se použijí všechny drivery se jmenovitou impedancí $\frac{1}{8} \cdot UL$, pak je možné mít na sběrnici až 96 zařízení.

$$N = \frac{32 \cdot UL_{STANDARD} - 20 \cdot UL_{FAILSAFE}}{UL_{pertranciever}}[19] \tag{2.3}$$

Délka a rychlost sběrnice

Na obrázku 2.4 je znázorněna závislost rychlosti komunikační sběrnice na její délce. Jsou zde vyznačeny 3 body, které mají tento význam:

1. Znázorňuje část vysokých komunikačních rychlostí na krátkém kabelu a maximální rychlost je zde omezena především drivery. Standard doporučuje rychlost 10 $Mbps$, ale můžeme se setkat i s drivery, které podporují vyšší rychlosti.
2. Vykresluje závislost mezi krátkým a dlouhým kabelem. Snížená rychlost je způsobená ztrátami na vedení. Existuje jednoduché pravidlo (rovnice 2.1.2), pomocí kterého můžeme ověřit návrh rychlosti komunikace v závislosti na délce kabelu.
3. Reprezentuje oblast největší délky kabelů. Tato oblast je omezena především ztrátami na vodiči, kde odpor doporučeného vodiče se blíží k hodnotě odporu zakončovacího odporu, tedy 120 Ω. Toto omezení nastává pro kabel o průřezu 0, 34 mm^2 při délce vodiče 1200 m.

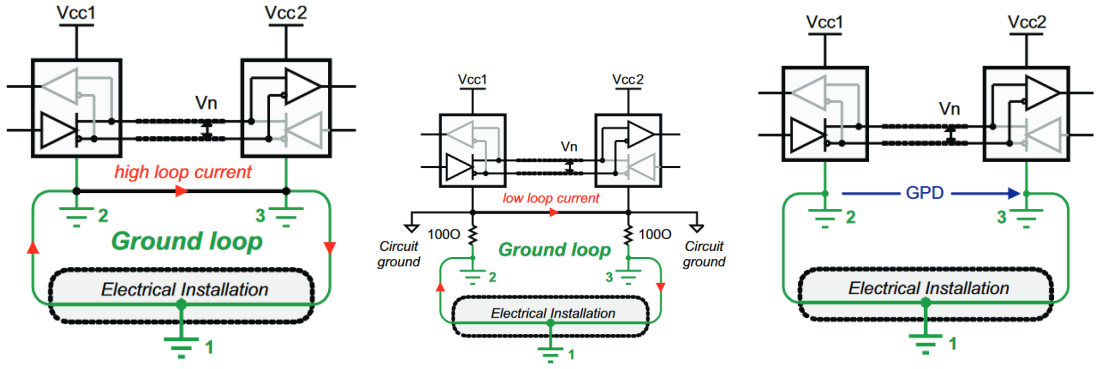


Obr. 2.4: Znázornění závislosti komunikační rychlosti na délce sběrnice RS-485[19].

$$l \cdot f < 10^7 \ [m;\ bps][19] \tag{2.4}$$

Zemnění a galvanické oddělení

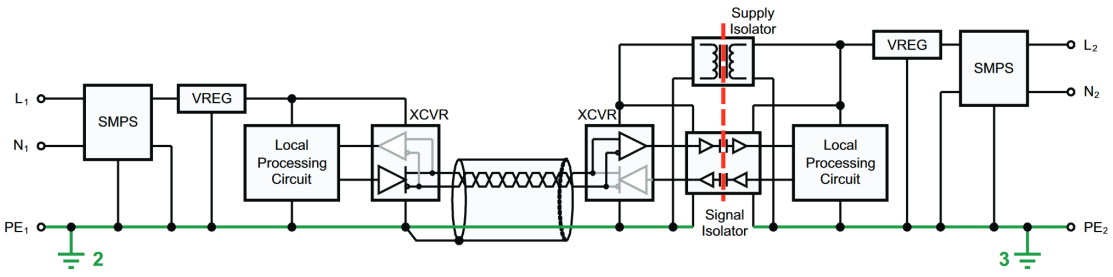
Na obrázku 2.5 jsou znázorněny 3 případy proudových smyček způsobené rozdílem potenciálů. Na obrázku 2.5a jsou země spojeny přímo a pokud by zde byl rozdíl potenciálů, vznikaly by zde velké proudové smyčky, které mají negativní vliv na sběrnici. Zapojení na obrázku 2.5b kde jsou vysoké proudy sníženy pomocí přidaných odporů, ale nebrání jejich vzniku. Nejvhodnější je zapojení zobrazené na obrázku 2.5c, jelikož země jsou propojeny pouze v jednom místě a nevznikají zde proudové smyčky komunikačním kabelem způsobené rozdílem potenciálů zemí.



(a) Spojené země. (b) S přidanou impedancí. (c) Zem v jednom bodě.

Obr. 2.5: Zapojení zemí pro sběrnici RS-485[19].

Nejvhodnější způsob potlačení proudových smyček vzniklých rozdílem potenciálů je využití galvanického oddělení sběrnice. Princip zapojení je znázorněn na obrázku 2.6.

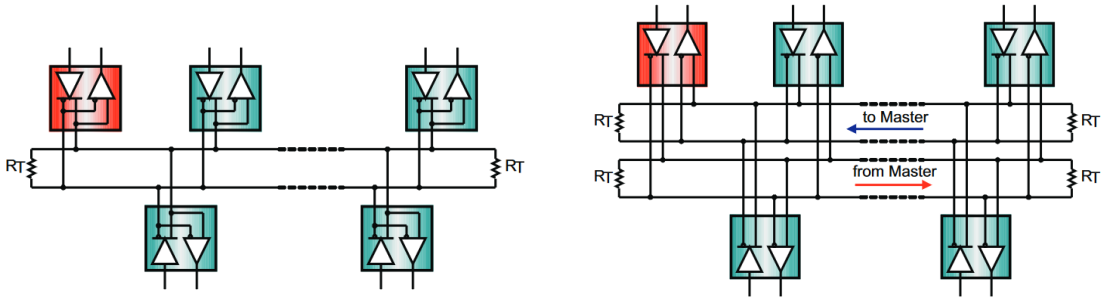


Obr. 2.6: Příklad galvanického oddělení sběrnice RS-485[19].

Zapojení sběrnice

Nejčastější zapojení sběrnice je zobrazeno na obrázku 2.7a, kde může vysílat pouze jedno zařízení (*Half duplex*). Tato sběrnice tedy často využívá komunikaci

typu *master-slave*, kde jedno zařízení řídí tok dat na sběrnici (*master*). Další možnost je vytvořit dva komunikační kanály a zařízení pak bude moci komunikovat dvěma směry najednou. Obrázek 2.7b vyobrazuje zapojení *Full duplex* sběrnice *RS-485*. Sběrnice *RS-485* má vývody pojmenované jako A a B.



(a) Half duplex.

(b) Full duplex.

Obr. 2.7: Připojení zařízení na sběrnici RS-485[19].

2.1.3 Sběrnice CAN bus

Sběrnice *CAN bus* byla vyvinuta společností *BOSCH* jako standard pro automobilový průmysl, ale kvůli svojí spolehlivosti a vlastnostem se rozšířila i do dalších odvětví. Oproti sběrnici *RS-485* má standard sběrnice *CAN bus* také definovanou linkovou vrstvu. Sběrnice má pojmenované signály jako *CANH* (*CAN high*) a *CANL* (*CAN low*).

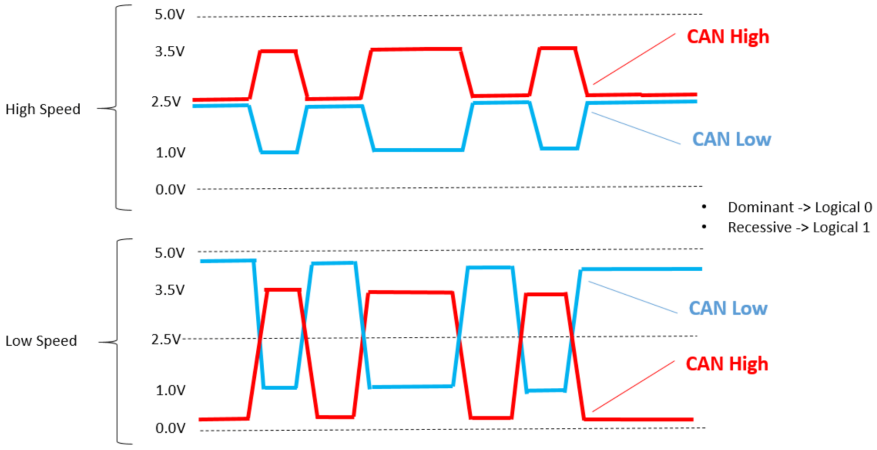
Fyzická vrstva

U sběrnice *CAN bus* bývá opomíjeno, že existuje několik variant fyzické vrstvy. Varianty fyzických vrstev:

- *Low speed*: také někdy označován jako *Fault protect* umožňuje fungování i s jedním vodičem ze dvou. Pokud je jeden vodič přerušen, zkratován na napájení nebo vodiče jsou zkratovány mezi sebou, měla by sběrnice dále fungovat, ale bude hlášeno varování. Jeho maximální rychlost je 125 *kbps* a nepoužívají se terminátory, jelikož jsou brány jako zkrat.
- *High speed*: oproti variantě *Low speed* využívá jiné napěťové úrovně a jeho rychlost může být až 1 *Mbps*.
- *Single wire*: je realizován pomocí jednoho vodiče a optimalizován na nízkou cenu. Použití jednoho vodiče nese omezení v rychlosti, která by neměla přesahovat 100 *kbps*, ale většinou se nastavuje jako nižší. Vyniká také nízkou spotřebou a může probudit zařízení po sběrnici.

Standard *CAN bus* definuje pojmy dominantní a recesivní úrovně namísto log. 0 a log. 1. Lze však říci, že dominantní úroveň odpovídá log. 0 a recesivní úroveň odpovídá log. 1. Je to způsobeno především tím, že na sběrnici může začít vysílat několik zařízení najednou. Avšak na sběrnici se objeví dominantní úroveň oproti recesivní a zařízení zjistí, že se jeho recesivní úroveň neobjevila na sběrnici, proto vysílání rámce odloží na pozdější příležitost. Toto umožňuje determinismus (více popsáno v linkové vrstvě).

Na obrázku 2.8 jsou znázorněny varianty *Low speed* a *High speed*. Hlavní rozdíl je v napěťových úrovních. Verze *Lowspeed* má napěťové úrovně definované okolo $0\,V$ a $5\,V$. *Highspeed* má napěťové úrovně definované pro *CANH* od $2,5\,V$ do $3,5\,V$ a pro *CANL* od $1,5\,V$ do $2,5\,V$.

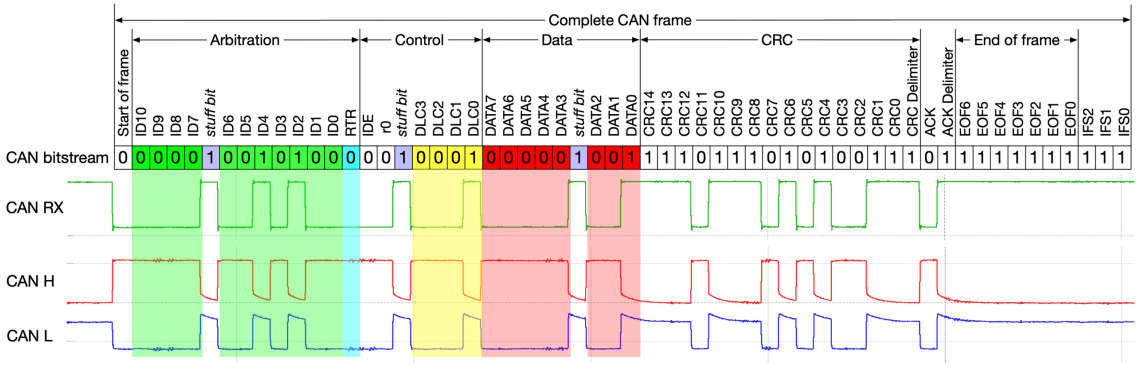


Obr. 2.8: Fyzická vrstva sběrnice CAN bus[31].

Linková vrstva

Aktuálně se nejčastěji využívají dvě verze linkové vrstvy a to *CAN2.0A* a *CAN2.0B*. Jejich rozdíl je ve velikosti arbitráže (identifikátoru). *CAN2.0A*, neboli také standardní *CAN bus*, využívá 11 bitů a verze *CAN2.0B* má 29 bitů identifikátoru. Další používanou verzí je *CANFD*, který se vyznačuje delším datovým blokem (až 64 bytů oproti 8 bytům u verze *CAN2.0*) a vyšší přenosovou rychlostí datového bloku (vše ostatní se odesílá s jinou rychlostí). Různé vysílací rychlosti jednoho rámce jsou způsobeny zpětnou kompatibilitou. Zařízení verze *CAN2.0* a *CANFD* mohou být spolu na stejné sběrnici. Většina moderních driverů pro *CAN bus* právě podporuje *CAN2.0* i *CANFD* a detekuje o jaké rámce se jedná. V případě, že driver nepodporuje *CANFD*, je možné, že bude generovat varování, ale sběrnice by měla být funkční. Další připravovanou verzí *CAN bus* bude *CANXL*, který umožní přenos až 2048 bytů v jednom rámci a také vyšší rychlosti.

Linková vrstva má na starost detekci a potvrzování chyb, vkládání bitů, ověření zprávy, potvrzení přijetí a synchronizaci komunikace. Jak bylo avizováno v předchozí kapitole, tak je možné, aby začalo přistupovat na sběrnici více zařízení najednou a na sběrnici se objeví dominantní úroveň sběrnice. Při vysílání arbitrážní části tedy zařízení mezi sebou provádí souboj o to, kdo bude vysílat. Souboj by neměl trvat déle, než je arbitrážní část. Díky tomuto aspektu je sběrnice deterministická a lze říci, kdo bude na sběrnici vysílat a komunikační protokoly využívají tohoto aspektu pro vysílání přednostních zpráv, jako jsou např. zprávy hlásící poruchu a je pak jistota, že tato zpráva bude vyslána co nejdříve. Kvůli tomuto aspektu se také doporučuje zatížení sběrnice pouze na 60 %. Vkládaní bitů (neboli také *bit stuffing*) vkládá vždy jeden bit opačného významu, pokud je na sběrnici vysíláno 5 za sebou jdoucích stejných bitů. Na obrázku 2.9 je znázorněn přenášený rámec *CAN2.0A* se všemi náležitostmi.

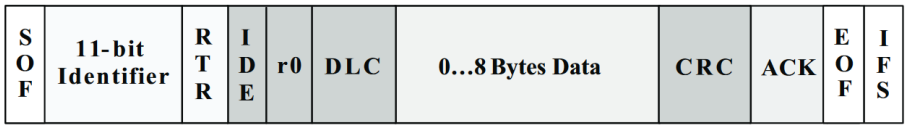


Obr. 2.9: Příklad rámce dat přenášených pomocí CAN bus.

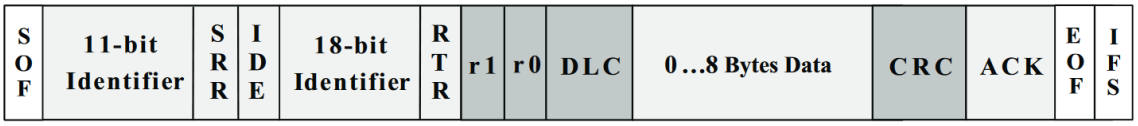
Rámec dat

Na obrázku 2.10 jsou znázorněny datové rámce *CAN2.0A* a *CAN2.0B*.
Jednotlivé bity v rámci mají následující význam[11]:

- **SOF**: *Start Of Frame* označuje začátek vysílání na sběrnici a je vždy dominantní.
- **Identifier**: identifikátor zprávy. Pro rozšířený identifikátor je identifikátor rozdělený na dvě části kvůli vzájemné kompatibilitě.
- **RTR**: *Remote Transmition Request* slouží pro specifikaci typu přenosu, jestli se jedná o zápis nebo čtení. Pokud je tento bit dominantní, tak se jedná o čtení.
- **IDE**: identifikuje rámec přenosu, zda se jedná o rámec verze *CAN2.0A* (dominantní) nebo *CAN2.0B*.



(a) CAN2.0A



(b) CAN2.0B

Obr. 2.10: Rámec dat pro sběrnici CAN bus[11].

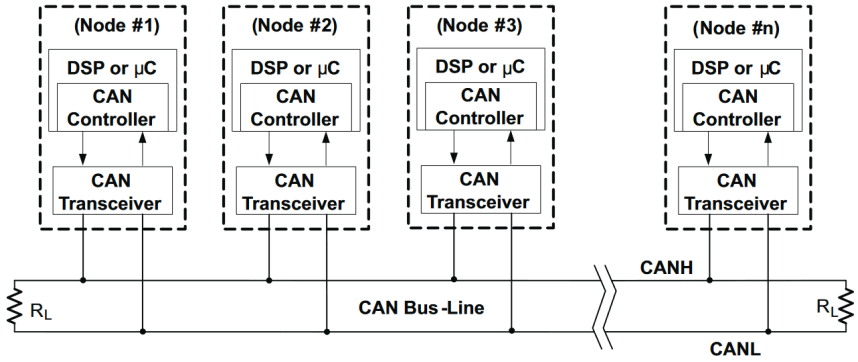
- **r0**: pro *CAN2.0* je rezervovaný, nyní se však používá pro identifikaci rámce *CANFD*.
- **DLC**: *Data Length Code* obsahuje informaci o velikosti přenášených dat (0-8 bytů).
- **Data**: užitečná data. Maximální délka dat závisí na verzi.
- **ACK**: slouží pro potvrzení, zda data obdržel alespoň jeden slave (každý slave vystaví na sběrnici dominantní bit).
- **EOF**: *End Of Frame* je dlouhý 7 bitů a pokud není v rámci chyba, pak je pouze recesivní úroveň. Jestli je v rámci chyba, vystaví se dominantní úroveň. V této oblasti je vynecháno vkládání bitů.
- **IFS**: *InterFrame Space* je vycpávka, která dává čas přijímači na přesun zprávy do vnitřního bufferu. Vycpávka je dlouhá 7 bitů a je recesivní úrovně.
- **r1**: rezervní bit pro budoucí funkce.
- **SRR**: je bit nahrazující bit *RTR* u standardního rámce.
- **CRC**: kontrolní součet sloužící pro detekci chyb při přenosu.

U *CAN bus* se často zmiňuje pojem arbitráž (*Arbitration Field*), což je označení pro data obsahující 11 bitový identifikátor až bit *RTR*. Pomocí arbitrážních bitů lze nastavit filtrace rámců, kde cílem periférie obstarávající *CAN bus* je filtrování příchozích rámců v závisti na jeho hodnotě, aby *MCU* nemusel obstarávat zprávy, které nejsou určené pro dané slave zařízení (rámce ho nezajímají).

Zapojení sběrnice

Stejně jako sběrnice *RS-485* vyžaduje *CAN bus* zakončovací odpory (varianta *High speed*) a kroucenou dvojlinku pro správnou funkčnost. Největší výhodou sběrnice *CAN bus* je definování odboček, tedy kabel nemusí být veden od zařízení k zařízení, ale vytvoří se odbočka přímo k zařízení. Délka těchto odboček je ale ome-

zena jak celkovou délkou, tak i délkou na jednu odbočku v závislosti na rychlosti komunikace. Orientační délka sběrnice a odboček je uvedena v tabulce 2.1. Příklad zapojení sběrnice je na obrázku 2.11.



Obr. 2.11: Příklad zapojení sběrnice CAN bus[11].

Délka a rychlost sběrnice

Rychlost	Délka kabelu	Délka jedné odbočky	Délka všech odboček celkem
1 $Mbps$	25 m	1, 5 m	7, 5 m
800 $kbps$	50 m	2, 5 m	12, 5 m
500 $kbps$	100 m	5, 5 m	27, 5 m
250 $kbps$	250 m	11 m	55 m
125 $kbps$	500 m	22 m	110 m
50 $kbps$	1000 m	55 m	275 m
20 $kbps$	2500 m	137, 5 m	687, 5 m
10 $kbps$	5000 m	275 m	1375 m

Tab. 2.1: Maximální délka sběrnice CAN bus a odboček v závislosti na rychlosti[12].

2.1.4 Deterministické komunikace založené na Ethernetu

Samotný Ethernet (dle *IEEE 802.3*) se v průmyslu k řízení technologií moc neprosadil. Prvním důvodem bylo zanedbatelné zrychlení komunikace u verze s rychlostí přenosu dat 10 $Mbit/s$ a velkou velikostí minimálního rámce. Neprosadil se ale především kvůli algoritmu *CSMA/CD*. Tento algoritmus obstarává přístup na sběrnici, který umožňuje vysílání když je linka volná, jinak čeká na uvolnění linky. Pokud však dojde ke kolizi, odloží se pokus o vysílání a není zajištěn determinismus komunikace. Toto může mít ve výsledku fatální následky při řízení technologie.

Tento problém skoro vymizel při použití aktivních síťových prvků (zařízení typu *switch* a *router*), jelikož jeden port zařízení je propojen s jedním portem druhého zařízení (princip *point-to-point*) a nemohou tak vznikat kolize na lince. Aktivní síťové prvky neřeší problematiku reálného času, jelikož soupeření o vysílání se děje uvnitř těchto zařízení. Proto spousta společností zabývajících se vývojem řídicích systémů vymyslela úpravu Ethernetu s implementací reálného času v podobě změny v linkové vrstvě. Změna linkové vrstvy umožnila použití stejného hardwaru jako pro standardní Ethernet, ovšem použití standardních síťových prvků (*switch* a *router*) není možné, jelikož neumí spolupracovat s touto linkovou vrstvou a tížený determinismus by byl ztracen. Nejznámější jsou tato řešení:

- **Profinet** (*PROFIBUS&PROFINET International*) se nabízí v několika variantách a to:
  - **Profinet RT** s real-time chováním (zaručením odezvy).
  - **Profinet IRT** snižuje proměnlivé zpoždění oproti verzi *RT* a vylepšuje časovou synchronizaci.
- **EtherCat** (*Beckhoff*) je typický kruhovou konfigurací sítě, kde každé zařízení má dva Ethernetové porty. Zprávy se vyměňují stylem nabalování, kdy master zařízení odešle data, první zařízení přečte tato data a odešle tato data dalšímu zařízení s vyměněnými vstupními daty za výstupní data. *EtherCat* je schopný funkce i pokud je přerušen vodič mezi dvěmi zařízeními, ale sběrnice nebude již tak rychlá.
- **Powerlink** (*B&R*) protokol využívá rozdělení komunikace do cyklů, kde jeden cyklus obsahuje inicializační rámec, izochronní rámce a asynchronní rámce. Komunikaci řídí řídicí uzel.

2.2 Průmyslové komunikační protokoly

Existuje velká řada komunikačních protokolů, které jsou postaveny na míru specifikacím sběrnicím, ale také jejich záměna. Příkladem může být komunikační protokol *MODBUS*, který je určen jak pro sériovou sběrnici *RS-485*, tak i pro Ethernet (podporuje protokoly TCP a UDP).

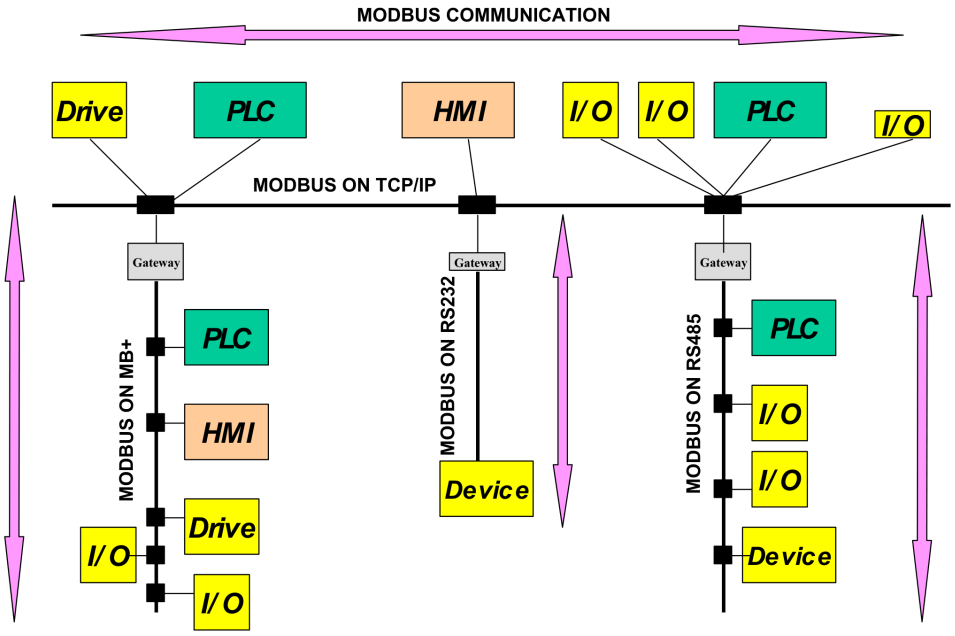
2.2.1 MODBUS

Komunikační protokol *MODBUS* je velice rozšířený, i když se nyní nahrazuje modernějšími technologiemi vyniká nízkou cenou potřebného hardwaru a také jednoduchostí. Standard komunikačního protokolu *MODBUS* je veřejně přístupný a nemusejí se platit žádné poplatky za jeho využívání. Tato kapitola čerpá informace z [23] a [30].

MODBUS existuje v několika variantách a to:

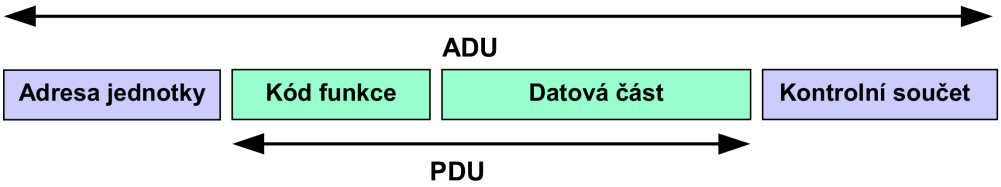
- `MODBUS/RTU`: určen pro sériovou linku (nejčastěji *RS-485*).
- `MODBUS/TCP`: využívá sběrnice *ethernetu* a protokolů **TCP** a **UDP**.
- `MODBUS/ASCII`: je určen pro sběrnice, které podporují přenos *ASCII* znaků.

Existují různá zařízení, která fungují jako převaděč signálů mezi jednotlivými komunikačními sběrnicemi (*gateway*) a lze tak využívat různá rozhraní, dle místa použití. Na obrázku 2.12 je ukázka propojení jednotlivých sběrnic.



Obr. 2.12: Příklad zapojení sběrnice s komunikačním protokolem MODBUS[23].

Na obrázku 2.13 je zobrazen základní tvar *MODBUS* zprávy, která se skládá z datové a aplikační části. Datová část (*PDU* - Protocol Data Unit) obsahuje kód funkce a přenášená data. Aplikační data (*ADU* - Application Data Unit) obsahují datovou část rozšířenou o adresu zařízení a kontrolní součet pro detekci chyb v rámci.



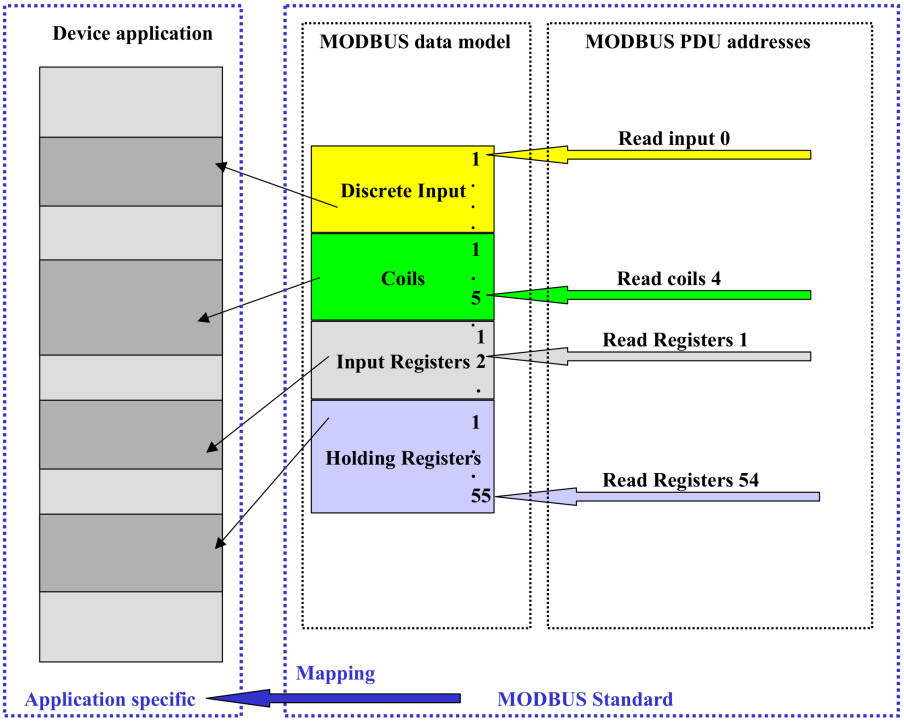
Obr. 2.13: Tvar *MODBUS* zprávy pro sériovou linku[30].

Paměťová orientace

Na obrázku 2.14 je znázorněn adresovací model pro *MODBUS*. Jeho paměť je rozdělena na 4 části:

- **Diskrétní vstupy**: bitové proměnné určené pouze pro čtení. Vhodné pro čtení digitálních vstupů zařízení.
- **Cívky**: bitově orientovaná paměť určená pro zápis a čtení. Vhodné pro digitální výstupy zařízení.
- **Vstupní registry**: registry určené pro čtení. Vhodné pro realizaci analogových vstupů.
- **Registry**: standardní registry vhodné pro čtení a zápis. Vhodné pro realizaci analogových výstupů, či parametrů k nastavování parametrů zařízení.

MODBUS registry mají velikost 16 *bit*ů a vždy musí být operováno s celými registry. Pokud jsou proměnné velikosti typu byte, tak využívají pouze část registru, nebo pokud jsou větší, využívají několik registrů.

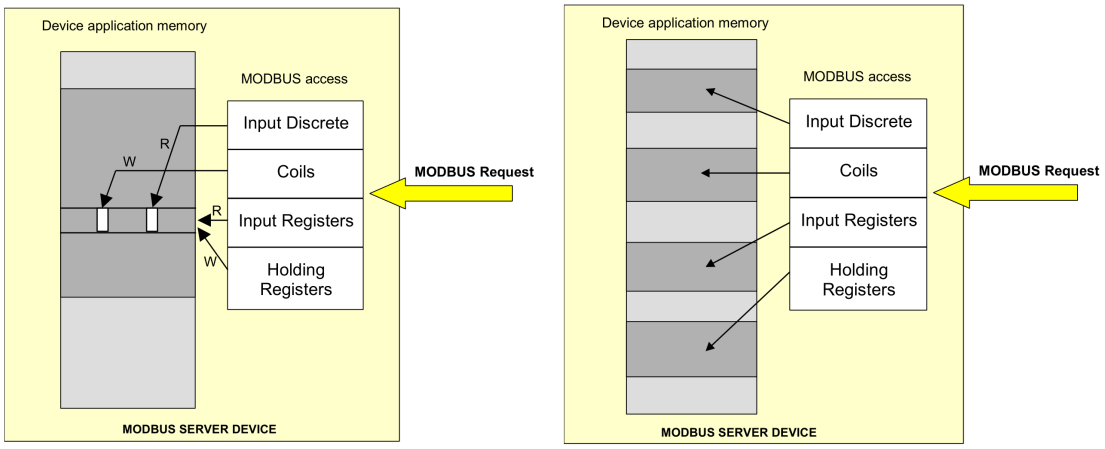


Obr. 2.14: Adresovací model protokolu MODBUS[23].

Paměťová alokace jednotlivých bloků musí být určena výrobcem a standard *MODBUS* ji neurčuje, jelikož její požadavky se mohou měnit v závislosti na výsledné aplikaci. Na obrázku 2.15 jsou znázorněny dva příklady této alokace, ale jsou možné i další. Prvním případem je (obrázek 2.15a) překrývající se paměťový prostor a je tedy možné k němu přistupovat přes jakékoli funkce. Druhým případem

(obrázek 2.15b) je umístění každého bloku samostatně, což nese vyšší nároky na paměť zařízení. Jednotlivé paměťové bloky nemusejí být implementovány všechny a výrobce si může vybrat, kolik jich bude k dispozici, ale je umožněno adresování pomocí 16 *bit* proměnné.

Adresy paměťových bloků musí určit výrobce zařízení. Každý paměťový blok má svoje adresy začínající na nule a přistupuje se k nim pomocí rozdílných funkčních kódů.



(a) Společné (překrývající se). (b) Oddělené.

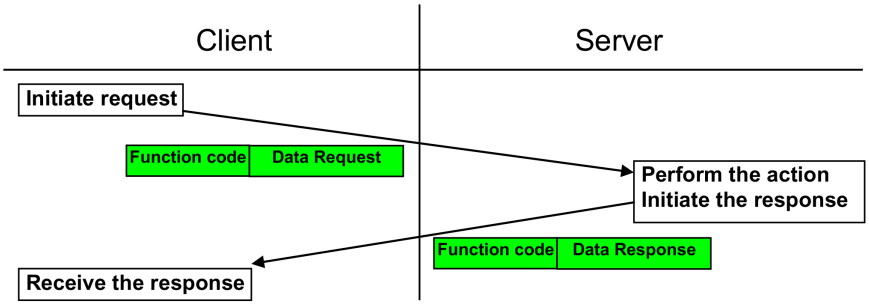
Obr. 2.15: Možnosti paměťových alokací MODBUS[23].

Typ komunikace

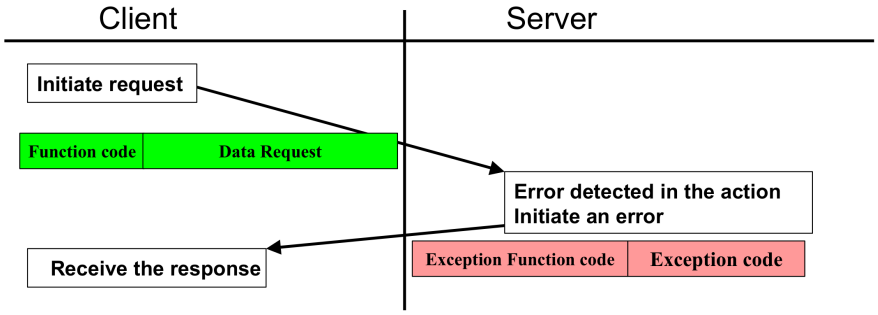
Komunikace *MODBUS* je založena na principu *master-slave* komunikace. *Master* (*Client*) je inicializátorem komunikace, tedy odesílá požadavek a *slave* (*Server*) mu vrací odpověď. *Slave* není oprávněn komunikovat, pokud mu *master* neodešle požadavek. Na obrázku 2.16 je znázorněn ideální stav komunikace, kdy tedy nenastane žádná porucha. Pokud nastane nějaká porucha přijatelná slave zařízením, tak slave zařízení odešle nazpět zprávu informující o přijetí chyby. Výměna dat s chybou je znázorněna na obrázku 2.17.

Možné chyby jsou:

- Kód chyby **0x01**: Nepodporovaný, či chybný kód funkce.
- Kód chyby **0x02**: Neplatná adresa dat.
- Kód chyby **0x03**: Neplatná hodnota dat.
- Kód chyby **0x04**: Chyba *slave* zařízení.
- Kód chyby **0x05**: Potvrzení sloužící pro prodloužení času na zpracování požadavku.



Obr. 2.16: Transakce komunikace MODBUS bez chyby[23].



Obr. 2.17: Transakce komunikace MODBUS s chybou detekovatelnou slave zařízením[23].

- Kód chyby **0x06**: *Slave* zařízení je zaneprázdněno a *Master* může zkusit odeslat požadavek znovu až za nějaký okamžik.
- Kód chyby **0x08**: Špatná paměť *slave* zařízení. Typické pro funkce *20* a *21*.
- Kód chyby **0x0A**: Cesta brány (*gateway*) není dostupná.
- Kód chyby **0x0B**: Cílové zařízení přes bránu není dostupné, nebo nezaslalo odpověď v požadovaném čase.

Možnosti výměny dat

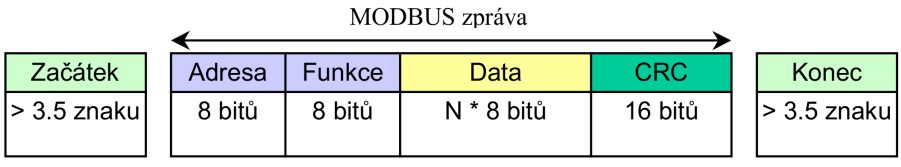
Pro rozlišení jednotlivých požadavků se využívá takzvaných funkčních kódů. Tyto kódy jsou uvedeny na obrázku 2.18. Není nutné aby zařízení mělo implementované všechny funkční kódy, ale pouze potřebné pro správnou funkčnost zařízení. Kód funkce je zakódován do jednoho bytu, kde 7 *LSB* bitů je využito pro funkční kód (tedy až 128 funkcí) a *MSB* bit označuje chybu. Pokud se tedy vyskytne chyba v datech, *slave* zařízení odpovídá zpětně kódem funkce s hodnotou tohoto bitu 1.

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Server ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19
			CANopen General Reference	**43**	**13**	**2B**	6.20

Obr. 2.18: Jednotlivé funkční kódy pro MODBUS[23].

MODBUS/RTU

V režimu *RTU* se přenášejí celé byty a mezi vysíláním jednotlivých bytů nemůže být mezera větší jak $1,5\,bytu$ a vysílání musí být souvislé. Konec vysílání je ukončen klidem na lince větší jak $3,5\,znaku$. Toto platí do rychlosti $115200\,bps$, jinak jsou ještě uměle vkládány mezery mezi jednotlivé byty a konec přenosu je také definován pevně, aby přijímací procesor měl čas zprávy zpracovat. Formát rámce je znázorněn na obrázku 2.19.



Obr. 2.19: Formát rámce MODBUS/RTU[30].

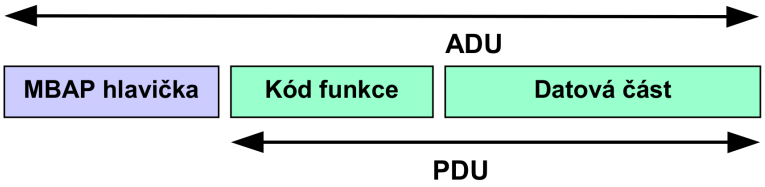
Požadavky na sériovou komunikaci jsou následující:

- 1 start bit.

- 8 datových bitů.
- 1 bit sudé parity. Pokud není zařízením podporován, je nahrazen druhým stop bitem.
- 1 stop bit.

MODBUS/TCP

Na obrázku 2.20 je znázorněn rámec pro komunikaci založenou na Ethernetu. Oproti sériové lince zde není kontrolní součet, který je součástí samotného Ethernetového rámce. Součástí *ADU* je hlavička zprávy (*MBAP* - MODBUS Application Protocol Header). Komunikační protokol *MODBUS* má registrovaný port 502 v rámci Ethernetové komunikace.



Obr. 2.20: Tvar *MODBUS* zprávy pro verzi TCP[30].

MODBUS/ASCII

MODBUS/ASCII rozděluje jeden byte na dva *ASCII* znaky. Na obrázku 2.21 je zobrazen rámec přenášených dat. Oproti verzi *RTU* je definován začátek a konec komunikace pomocí znaků, nikoli klidem na sběrnici. Začátek přenosu je určen znakem „:“ a konec řídicími znaky *CR*(Carriage Return) a *LF*(Line Feed), jež odpovídají sekvenci nového řádku. K detekci chyb pak slouží *LRC*, které má délku dvou znaků.

Začátek	Adresa	Funkce	Data	LRC	Konec
znak „:“	2 znaky	2 znaky	0 až 2*252 znaků	2 znaky	2 znaky CR, LF

Obr. 2.21: Formát rámce MODBUS/ASCII[30].

Požadavky na sériovou komunikaci jsou následující:

- 1 start bit.
- 7 datových bitů.
- 1 bit sudé parity. Pokud není zařízením podporován, je nahrazen druhým stop bitem.
- 1 stop bit.

2.3 Teorie detekce chyb přenosu dat

Aby byla komunikace spolehlivá, je nutné zjistit, zda přijatá data jsou správná, tedy jsou taková jaká se vyslala. Pro detekci chyb v datech se do přenášených dat přidávají sekce s informací navíc, která je odvoditelná od základních dat. Existují také metody pro opravu dat, kde není nutné odesílat data znovu a jsou založeny na pravděpodobnostním principu. Jsou např. odeslány 3 bity místo 1 bitu a pokud je v přenosu chyba, je ji možné s určitou pravděpodobností opravit. Metody pro opravu chyb jsou schopné opravit pouze určitý počet chyb a tyto opravy mají svá omezení. Pro přenášená data lze podle vzorce 2.3 vypočítat, kolik zpráva obsahuje užitečných informací v procentech, kde n_P vyjadřuje počet užitečných bitů zprávy a n určuje celkový počet přenášených bitů zprávy.

$$P = \frac{n_P}{n} \cdot 100 \tag{2.5}$$

Žádná detekce chyb není dokonalá natolik, aby odhalila všechny chyby, ale u každé metody je uvedeno pro jaké chyby je navržena a kolik chyb v přenášených datech může pokrýt. Může tedy nastat situace, kdy data přijdou natolik pozměněná, že je detekce chyb neodhalí, ale v reálu tyto metody možnosti výskytu snižují na minimum.

Metody určené pro detekci chyb jsou známé a nemají žádný význam pro šifrování dat na sběrnici a je snadné upravit zprávu, aby odpovídala přenášenému rámci s vypočítanou chybovou sekcí.

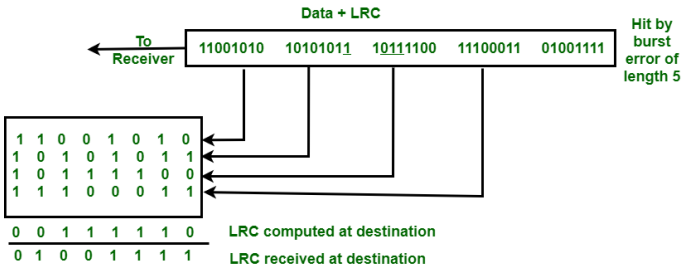
2.3.1 Paritní bit

Paritní bit využívá redundantního bitu, tedy ke zprávě je přidán jeden bit. Tento bit doplňuje zprávu tak, aby počet bitů nabývající hodnoty 1 odpovídal lichému nebo sudému počtu. Je tak tedy možné odhalit pouze lichý počet chyb ve zprávě. Z matematického hlediska se jedná o speciální případ 1 bitového CRC (popsáno v kapitole 2.3.4) s polynomem $x + 1$[41]. Můžeme se také setkat s názvem **VRC** (*Vertical Redundancy Check*)[16].

Implementace výpočtu a kontroly paritního bitu bývá realizována pomocí hardwarového vybavení zařízení (periférie sériové komunikace) a nemá výpočetní vliv na zařízení (případná chyba paritního bytu bývá řešena pomocí přerušení). Paritní bit se většinou vkládá za každý byte zprávy.

2.3.2 Podélný redundantní součet – LRC

Podélný redundantní součet (*Longitudinal Redundancy Check*), nebo také dvourozměrná parita. Dle délky výsledného *LRC* je zpráva rozdělena po této délce (nejčastěji jeden byte). Následně se tyto části naskládají nad sebe a vypočítá se paritní bit pro každý sloupec bitů. Zařízení na druhé straně provede stejný výpočet a porovná přijaté *LRC* s vypočítaným *LRC*. Pokud je zaznamenána pouze jedna chyba, lze ji v kombinaci s *VRC* (paritním bitem) opravit[17]. Na obrázku 2.22 je znázorněn příklad výpočtu a přenosu dat s chybou v přenosu dat (chyba je znázorněna podtržením). *LRC* je schopný odhalit pouze lichý počet chyb ve sloupci.



Obr. 2.22: Příklad LRC s chybou v přenosu.[17]

2.3.3 Kontrolní součet – checksum

Kontrolní součet detekuje jak sudý, tak lichý počet chyb, ale může nastat taková situace, kdy zpráva bude porušena tak, že změna nebude detekována. Postup výpočtu je jednoduchý a skládá se z následujících kroků[14]:

1. Rozdělení zprávy do bloků po n bitech, kde délka kontrolního součtu je n bitů.
2. Sečtení všech bloků zprávy. Při výskytu přenosu bitů do vyšších řádů (výsledek je větší, než je maximální hodnota n bitového čísla) dojde k ořezání této hodnoty a k opětovnému přičtení k ořezanému výsledku (před přičtením dojde k binárnímu posunu bitů o n bitů doprava).
3. Z výsledné hodnoty je proveden 1. doplněk (binární negace), čímž je hodnota kontrolního součtu.

Přijímač pro výpočet správnosti postupuje obdobně, jen kontrolní součet je vypočítán ze všech přijatých dat (včetně přijatého kontrolního součtu) a výsledek je porovnáván, zda je roven nule. Pokud je výsledek roven nule, tak jsou data správná, jinak byla data přijaty s chybou.

2.3.4 Cyklický redundantní součet – CRC

Cyklický redundantní součet je matematická metoda, jak dosáhnout čísla, které se zásadně mění při změně jednoho bitu ve zprávě (mění více bitů ve svém výsledku). Výsledné *CRC* bývá oproti zprávě velice krátké a tedy nelze přidělit každé zprávě jedinečný výsledek *CRC*[42].

Pro výpočet *CRC* je nutné znát zprávu, pro kterou se má CRC vypočítat, délku *CRC* a řídicí polynom. Někdy je také nutné znát počáteční podmínku pro řídicí polynom. Například *MODBUS/RTU* využívá *CRC* o délce 16 bitů (označované jako *CRC16*), řídicí polynom je tvaru $x^{16}+x^{15}+x^2+1$ a počáteční nastavení řídicího polynomu je nastaveno na $0xFFFF$. Zde je dobré zmínit, aby byl algoritmus úspěšný, přidává se jeden bit s hodnotou 1 na začátek, ale při jeho hodnotě v hexadecimální hodnotě se neudává (například u uvedeného *MODBUS/RTU* se udává $0x8005$, ale ve skutečnosti je polynom tvaru $0x18005$).

Existuje několik standardizovaných variant CRC, které udávají jeho délku, tak řídicí polynom.

3 Operační systémy reálného času

Operační systémy reálného času (*RTOS*) jsou speciální podskupinou operačních systémů. Oproti klasickým OS kladou nárok na vykonání jednotlivých úloh v závislosti na čase. *RTOS* se nejčastěji využívají na *embedded* zařízeních. Tato kapitola popisuje jejich funkce, dělení, vlastnosti, proč se používají a porovnání vybraných *RTOS*. Tato kapitola čerpá informace především z literatury [20], [13] a [3].

3.1 Definice reálného času

Systém reálného času, je takový systém, který musí splňovat omezení doby odezvy nebo riziko vážných následků, včetně selhání. Z obecného hlediska lze *RTOS* definovat jako: „Operační systém reálného času je takový operační systém, který je schopen provádět výpočty a reagovat na události v předem definovaných časových intervalech (angl. deadline)“[20]. Další definicí *RTOS* je dle teorie informace, kde operační systém reálného času je takový systém, který zaručuje, že informace jsou zpracovány v systému ve správném pořadí a s omezenou časovou lhůtou.

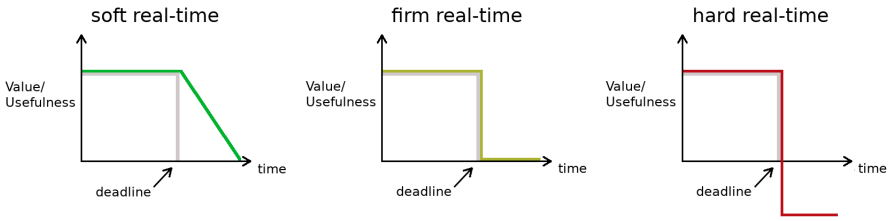
3.2 Dělení

V závislosti na práci v čase můžeme operační systémy reálného času rozdělit do tří skupin[20]:

- **Soft real–time**: Stačí, když vlákna jsou zpracována v průměrném časovém intervalu a pokud vlákna jsou zpracována později, systém s nimi může s nějakými omezeními pracovat.
- **Firm real–time**: Vlákna musejí být vykonána v přiděleném časovém intervalu, či s minimálním zpožděním.
- **Hard real–time**: Vlákna se musejí bezpodmínečně splnit. Pokud by se vlákno nestihlo vykonat, znamenalo by to selhání systému. Pozdní data jsou neplatná, bezcenná nebo nebezpečná.

Na obrázku 3.1 jsou znázorněny funkce užitečnosti pro jednotlivé varianty *RTOS*.

Dalším dělením *RTOS* je dle implementace a to na **micro-kernel**, který má oddělené určité dílčí funkce a **monolytic kernel**, který se chová jako zapouzdřené jádro.



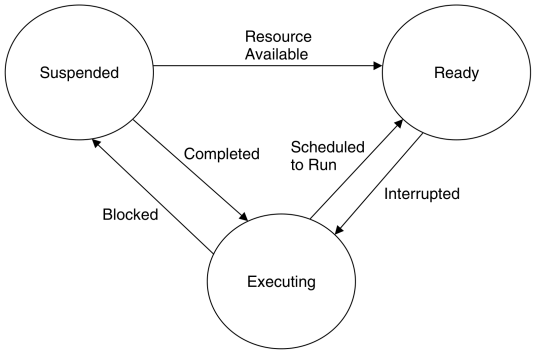
Obr. 3.1: Funkce užitečnosti pro klasifikaci RTOS[33].

3.3 Plánovač úloh

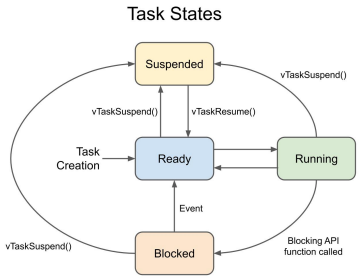
Plánovač úloh (angl. *scheduler*) je srdcem operačních systémů a umožňují přepínání jednotlivých procesů. Pro řízení úloh využívá v základu tři stavy úloh a to:

- **Suspended**: úloha je zastavena a není prováděna. Tento stav také označuje čekání na periferii, která blokuje běh úlohy. Některé OS přidávají další stav označený jako **Blocked**, kteý má význam čekající úlohy na periferii.
- **Ready**: úloha je připravena pro vykonávání a je ji možné spustit.
- **Executing**: úloha se provádí.

Obrázek 3.2 vyobrazuje stavové diagramy plánovače úloh `RTOS`. Obrázek 3.2a vyobrazuje obecné stavy úloh a obrázek 3.2b je stavový diagram RTOS *FreeRTOS*.



(a) Obecný stavový diagram[20].

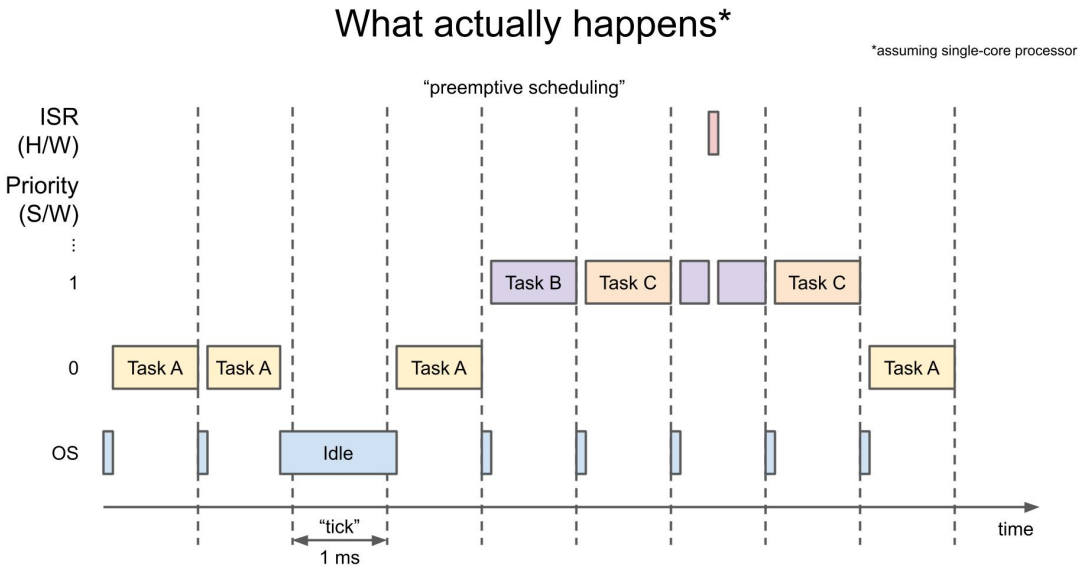


(b) Stavový diagram FreeRTOS[18].

Obr. 3.2: Stavový diagram plánovače úloh RTOS.

Existují různé metody pro přepínání více úloh které jsou aktivní. Nejčastěji používaným algoritmem je tzv. „*Round-robin*“ s nastavenou prioritou úloh. Příklad tohoto plánovače je zobrazen na obrázku 3.3. Úkoly (*task*) se primárně přepínají v závislosti na jejich prioritě. Pokud je však více než jedna aktivní úloha se stejnou prioritou, použije se metoda „*Round-robin*“. Tato metoda přidělí každé úloze určitý čas, po který může být spuštěna a přepíná mezi nimi. V uvedeném příkladu je

také znázorněna úloha *Idle*, která je spuštěna, pokud není naplánována žádná jiná úloha. Dále je vyobrazena také doba plánovače pro vykonání jeho funkcí. Nejčastěji s plánovačem jsou zmiňovány tzv. „*Tick*“, které udávají chod systému. Některé procesory jsou přímo vybaveny generátorem těchto pulsů a není nutné využívat externích časovačů.



Obr. 3.3: Příklad funkce plánovače úloh[18].

3.4 Služby systému

Jádro systému neobsahuje pouze plánovač úloh, ale také služby, které lze využít pro správu (řízení) úloh. Nejčastěji to bývají:

- **Systémové funkce** spjaté přímo s daným OS. *FreeRTOS* nabízí např.:
  - **Vytvoření úlohy**.
  - **Pozastavení úlohy**.
  - **Spuštění úlohy**.
  - **Odstranění úlohy**.
  - **Spuštění plánovače**: speciální funkce, která zavolá funkci plánovače úloh, jako jiná událost. Největší význam má spuštění úlohy s nejvyšší prioritou okamžitě v závislosti na přerušení a je třeba tuto úlohu vykonat co nejdříve.

  - **Kritická sekce** pro označení kódu, který je nutné vykonat okamžitě, jelikož přerušení vykonávání tohoto kódu by mohlo znamenat poškození dat. Tato sekce by měla být vykonána co nejrychleji.
- **Semafory** (někdy také *mutex*) pro správu zdrojů. Nejčastějším případem použití může být komunikační periferie, kterou využívá několik úloh a je zde jeden *token*, který může mít pouze jedna úloha. Speciálním případem může být čítací semafor, kdy může jeden zdroj využívat více jak jedna úloha.
- **Časovače** s cílem snížení potřebných hardwarových časovačů pro vykonání určitých akcí. Tyto časovače jsou však pro pomalé děje a jsou závislé na frekvenci *RTOS*.
- **Fronty zpráv** pro komunikaci mezi jednotlivými úlohami.
- **Časové prodlevy** umožňují dát úlohu do stavu *Blocked* na určitou dobu, po které se opět přesune do stavu *Ready*.
- **Paměťové operace** jako je alokování paměti.
- **Softwarové nadstavby** např. ulehčující komunikaci, práci se soubory, práci s přerušením, *stream*, *CLI* atd..

3.5 Problémy s RTOS

Problémy při běhu RTOS mohou nastat za různých podmínek a některé jsou problémem návrhu vlastního programu vykonávaný RTOS.

Poruchou operačního systému může být:

- Překročení limitu pro **deadline** vlákna.
- **Deadlock** - vzájemné čekání dvou úloh na dokončení, které nikdy neskončí.
- **Race condition** - výsledek operace záleží na pořadí spuštění jednotlivých úloh, jelikož data pracují se stejnými (sdílenými) zdroji.
- **Livelock** - zacyklení dvou a více úloh v čekacích smyčkách. Tyto úlohy jsou živé, ale nemohou postupovat dále ve vykonávání svého kódu.
- **Priority inversion** je způsobena blokováním společných zdrojů úlohou s nízkou prioritou úloze s vysokou prioritou, kde se tento blokující čas zvyšuje o vykonávání úloh se střední prioritou. Tento problém lze např. vyřešit dočasnou změnou priority s blokujícím zdrojem na vyšší prioritu a po odblokování zdroje se priorita změní opět nazpět.

Pro hledání těchto poruch lze použít speciální software, který sleduje dění v procesoru a zobrazuje jej v grafické podobě na počítači. Následně lze pak tyto poruchy snadno odhalit a zacílit se na danou část.

U multiprocesorových systémů mohou nastat ještě další problémy, kde např. ke sdílenému zdroji chtějí přistupovat oba procesy najednou.

3.6 RTOS vs tradiční způsob

Tradičním způsobem je myšlena nekonečná programová smyčka, ve které jsou řešeny všechny funkce. Největší výhodou tradičního řešení je rychlost vykonávání programu, jelikož zde není režie RTOS, ale celkový kód může být nepřehledný a jednotlivé změny ve vykonávání kódu jsou náročné. Práce se správou vykonávání jednotlivých funkcí (vláken) a jejich paralelismus je v RTOS mnohem snazší. U složitých projektů je velice těžké dosáhnout synchronizace vykonávání kódu tradičním řešením. Další výhodou RTOS může být nezávislost vykonávání jednotlivých úloh, jelikož selhání jedné úlohy nemusí vést k selhání celého systému.

3.7 RTOS pro embedded systémy

Existuje mnoho RTOS vhodných pro použití na MCU. Jejich rozdíly jsou jednak v ceně, ale také v přístupu k jednotlivým problémům. Důležitým faktorem je také certifikace systému pro bezpečnostní aplikace, podpora různých platforem (procesorů), možnosti rozšíření atd.. Mezi nejznámější patří:

- **FreeRTOS** [13] (*Amazon*) je zdarma i pro komerční účely, což z něj činí oblíbený RTOS pro malé nebo začínající projekty. Je možné si pořídit také placenou verzi, která nese název *OpenRTOS* a zahrnuje záruku a právní ochranu. Dalším derivátem je *SAFERTOS*, který je certifikovaný do průmyslu (*IEC 61508 SIL 3*), lékařství, automobilového průmyslu a ostatních odvětví. Lze jej použít také na 8-bit procesory
- **MbedOS**[4] (*Arm*), který vychází z **RTX5** (*Keil*) jsou zdarma i pro komerční účely a jako u *FreeRTOS* lze pořídit záruky a certifikace systému. Systém je zaměřen především na komunikace a nabízí také spoustu ovladačů (přidaných softwarových knihoven). Tyto systémy jsou zaměřeny na 32-bit procesory architektury *ARM*.
- **μC/OS**[37] je *open-source* operační systém a lze jej použít i zdarma pro komerční účely. Jeho certifikovanou verzí je **Cesium RTOS**, která je placená.
- **embOS**[32] (*Segger*) je zdarma pouze pro nekomerční účely, jinak je placený. Výrobce nabízí také další verze, jako je např. **embOS-Safe**.
- **VxWorks**[38] (*Wind River*) je placený RTOS zaměřující se především na bezpečnost.

Jako vhodný RTOS byl zvolen *FreeRTOS* především kvůli jeho licenci (možnost použití pro komerční použití zdarma), osobním zkušenostem a obsahem všech potřebných funkcí.

4 Návrh vlastního modulárního řídicího systému

Jak je již z názvu *Modulární řídicí systém* patrné, že hlavní předností tohoto řídicího systému musí být jeho modulárnost. Tato modulárnost spočívá v možnosti volby jednotlivých vstupně/výstupních modulů. Aby bylo toto možné, je nutné mít dobře promyšlenou nejen elektrickou část návrhu řídicího systému, ale také jeho mechanickou část.

4.1 Mechanická konstrukce

Na trhu jsou dostupná řešení pro modulární elektronické systémy, jako jsou krabičky a konektory. Byl vybrán výrobce *Phoenix Contact*[26], který nabízí tyto systémy:

- **Modulární pouzdra BC** – jsou vhodná pro automatizaci budov, protože mají optimalizované rozměry pro domovní rozvaděče (mají stejný bokorys, jako jističe).
- **Modulární systém ICS** – nabízejí flexibilní hranatý tvar.
- **Modulární pouzdra ME** – pouzdra kalíškového tvaru.
- **Modulární systém ME–MAX** – pouzdra pro moderní průmyslovou automatizaci.
- **Modulární systém ME–IO** – je vhodný především pro řešení V/V modulů.
- **Modulární systém ME–PLC** – vhodný pro řešení řídicích systémů s vyšším počtem kontaktů na sběrnici v nosné liště (prodává se jako stavebnice a musí být navržena vlastní propojovací DPS do konektoru). Využívá ale speciální nosnou lištu místo standardní DIN lišty.

Většina těchto krabiček je uzpůsobena pro použití na DIN lištu, kromě systému krabiček *ME–PLC*, která využívá vlastní tvar lišty. Systém *ME–PLC* je z těchto systémů nejdražší a proto byl zamítnut. Jako nejvhodnější systém byl zvolen modulární systém *ICS*, který nabízí dostačující konektory sběrnice, větší plochu DPS a také konektory na vyšší napětí pro připojení síťového napětí ($230\,V\,AC$).

4.1.1 Krabička pro moduly

Na obrázku 4.1 je ukázka možnosti tohoto systému přímo od výrobce.

Obr. 4.1: Modulární systém ICS[26].

Krabičky *ICS* se skládají z následujících dílů:

- Tělo krabičky.
- Konektory, případně záslepky.
- Horní kryt krabičky.
- Konektor pro vnitřní sběrnici (obrázek 4.2).

Většinu dílů nabízí výrobce ve více variantách, takže je možné zvolit například jinou barvu, jiné provedení, šířku modulu atd..

4.1.2 Konektor sběrnice

Konektor sběrnice je vyráběn pro připevnění přímo na DPS, takže není nutné osazovat desku protikusem, ale na DPS jsou vytvořené plošky, které po zasunutí do konektoru tvoří spojení. Tento postup vyžaduje speciální technologii výroby DPS, jelikož vyžaduje pozlacení, a v místě konektoru je nanášená vrstva zlata vyšší (*goldfingers*) než v případě koncové úpravy měděných ploch na DPS (*finish surface*).

Phoenix Contact nabízí několik různých druhů konektorů, které jsou kompatibilní s vybraným druhem krabičky. Hlavním parametrem je celkový počet vodičů v konektoru a následně možnost vložení sériových kontaktů. Vybraným konektorem se tedy stala možnost s 8 spoji, z nichž jsou 2 sériové (spoj je pouze mezi nejbližšími moduly).

Obr. 4.2: Konektor sběrnice modulárního systému ICS[26].

4.2 Výběr komunikačního rozhraní

Pro komunikaci mezi jednotlivými moduly byla vybrána sběrnice `RS485` s vlastním komunikačním protokolem. Výběr této komunikace je popsán podrobněji v kapitole 12, která se zabývá návrhem vlastního komunikačního protokolu.

4.3 Rozdělení modulů

U modulárního řídicího systému je vhodné logicky rozdělit jeho jednotlivé moduly, aby bylo co nejmenší množství jejich druhů, ale také aby neobsahovaly zbytečné periférie, které koncový uživatel nechce.

Moduly modulárního řídicího systému byly rozděleny následovně[1]

- **Procesorová jednotka** – kapitola 6 na straně 76.
- **Analyzátor sítě** – kapitola 7 na straně 80.
- **Digitální vstupy** – kapitola 8 na straně 85.
- **Digitální výstupy** – kapitola 9 na straně 86.
- **Analogové vstupy a výstupy** – kapitola 10 na straně 87.
- **HMI (displej)** – kapitola 11 na straně 92.

Procesorová jednotka obsahuje řídicí algoritmus a zajišťuje také komunikaci s ostatními moduly a nese také uživatelský program. Dále je vhodné do procesorového modulu umístit komunikace, protože hlavní procesor bude uchovávat jednotlivé proměnné.

[1] Jejich podrobný návrh je popsán v následujících kapitolách.

Jelikož byl kladen požadavek na implementaci analyzátoru sítě do řídicího systému, byl vytvořen tento modul, který může fungovat jako samostatný analyzátor sítě s komunikací, jako V/V modul nebo také jako procesorová jednotka.

Dále jsou moduly digitálních periférií rozděleny do dvou modulů, kde jsou samostatně digitální vstupy a výstupy, protože každá aplikace může obsahovat specifický počet těchto periférií. Například v projektu může být několikanásobně více digitálních vstupů, než výstupů a pokud by byl jeden modul rozdělen na stejný počet digitálních vstupů a výstupů, byl by pak problém s nevyužitými perifériemi.

Analogové vstupy a výstupy byly sjednoceny v jednom modulu a jsou SW konfigurovatelné, což snižuje výsledný počet modulů.

Posledním modulem je zobrazovací jednotka, která není tvořena jako modul řídicího systému, ale jako samostatná jednotka komunikující na samostatné komunikační sběrnici.

4.4 Výběr MCU

Protože se jedná o modulární systém, který se skládá z několika nezávislých modulů, které komunikují po sběrnici, je vhodné vybrat procesor spíše univerzální, než pro každý modul jiný. Jelikož je však kladen rozdílný výpočetní výkon pro každý modul, neubráníme se výběru rozdílných MCU pro každý modul.

Z hlediska zkušenosti byl výběr MCU zaměřen na MCU společnosti *Atmel* (nyní *Microchip*) a to na řadu SAM, která je založena na architektuře jádra ARM. Důležitým aspektem při výběru byly periférie pro komunikaci, dostatek paměti a DMA (*Direct Memory Access*), díky kterému se sníží výpočetní výkon potřebný pro vykonávání přerušení např. v komunikaci.

Jako výkonnější MCU byla zvolena série *ATSAME70*, která má jádro *ARM® Cortex®-M7* s taktovací frekvencí 300 MHz a periférie jsou taktovány frekvencí maximálně 150 MHz. Dále je vybaven jednotkou pro výpočet s plovoucí desetinou čárkou (FPU - Floating Point Unit), která umí pracovat se *Single-precision* a *Double–precision*. Instrukční list je *Thumb®-2* rozšířený o DSP (*Digital Signal Procesor*) instrukce, které jsou určeny pro práci se signály. *ATSAME70* se vyrábí v několika provedeních, která jsou rozdílná v počtu periférií, počtu vývodů pouzdra a velikosti paměti. Pro procesorovou jednotku a HMI (displej) byl vybrán *ATSAME70Q21* kvůli dostatečnému počtu periférií (při návrhu s menším typem docházelo k překrývání výstupů periférií na pouzdře a muselo být vybráno větší pouzdro). Pro analyzátor sítě byl pak vybrán *ATSAME70N19*, jelikož potřebných periférií pro analyzátor sítě by bylo méně.

Méně výkonné MCU pochází ze série *ATSAMC2x* a konkrétně *ATSAMC20J17*, který nabízel dostatečný počet potřebných periferií i výpočetní výkon. *ATSAMC2x*

má jádro *ARM® Cortex®-M0+*, který může být taktován na frekvenci 48 MHz. Na rozdíl od *ATSAME70* má *ATSAMC2x* novější architekturu, která obsahuje systém eventů (*Event System*), který dovoluje omezit počet přerušení procesoru na minimum.

Jako příklad výhod periférie *Event System* bude uvedena zjednodušená část komunikace. V kapitole 12.2 je rozepsán návrh vlastního komunikačního protokolu. Část jeho problematiky programování bude popsána zde. Hlavním bodem pro tento příklad je časová mezera mezi jednotlivými přijatými pakety, tedy doba, po kterou nepřijde žádný byte, bude znamenat konec rámce a bude vyžadovat zpracování procesorem. Data jsou tedy přijímána pomocí periférie, ze které jej čte DMA a zapisuje do předdefinovaného paměťového místa. Následně je nastaven časovač, který generuje přerušení při vypršení doby časovače, čímž je detekován konec rámce a zavolána obslužná rutina pro jeho zpracování. Po každém příchozím bytu je nutné tento časovač vynulovat, aby konec rámce byl detekován správně, což tedy znamená, že při každém přijatém bytu je vyvoláno přerušení, které musí obsloužit procesor. Nová architektura toto řeší pomocí systému eventů, kde je generován takzvaný event při každém přijatém bytu, který je přesměrován touto periférií do časovače a resetuje jej bez nutnosti zásahu procesoru. Výsledkem je pak generování přerušení indikované procesorem až po přijetí celého komunikačního rámce dat. Procesory z řady *ATSAME70* jsou vybaveny funkcí *timeout* v periferii *USART*, takže popsaná problematika s časovou mezerou ve vysílání je pokrytá touto funkcí. Toto řešení nese určitou výhodu, jelikož má jednodušší nastavení a nepotřebuje další systémové prostředky pro svoji funkci, ale nejsou jim vybaveny všechny procesory.

Pokud by výpočetní výkon *ATSAMC20J17* nestačil, je možné jej nahradit MCU z řady `ATSAMD5x`, který má jádro *ARM® Cortex®-M4F* a jeho taktovací frekvence může být až 120 MHz. Zmiňovaná záměna by byla možná, jelikož rozmístění vývodů *ATSAMC2x* a `ATSAMD5x` je obdobné. Nejlepší by bylo použít MCU z řady `ATSAMD5x` a `ATSAME5x` (je rozšířen o komunikační rozhraní Ethernet či CAN) ve všech modulech, aby se sjednotila programová výbava pro moduly.

5 Návrh obecných obvodů

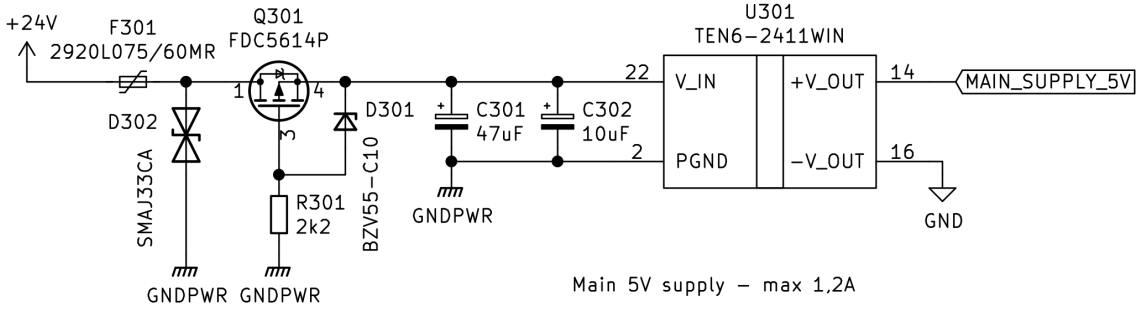
Tato kapitola se zabývá popisem elektronických obvodů, které se vyskytují ve více modulech. V kapitolách popisujících jednotlivé moduly budou odkazy na tato obecná zapojení.

5.1 Napájecí obvody

Některé desky jsou velmi rozmanité z hlediska napájecích úrovní i potřebným množstvím ideí jednotlivých zdrojů. Hlavní ideou návrhu bylo vytvoření galvanického oddělení mezi hlavní procesorovou jednotkou a vstupně–výstupními periferiemi (vše mimo vnitřní logiku, kromě USB).

5.1.1 Hlavní zdroj napájení

Důležitým faktorem pro napájecí obvody bylo napájecí napětí. V průmyslu se používá především napájecí napětí $24\,V$. Aby řídicí systém nebyl omezen pouze na jedno napětí s určitou tolerancí, byl přidán požadavek na možnost napájení i pomocí $12\,V$. Pro zlepšení použitelnosti systému, byl stanoven rozsah napájecího napětí na $10\,V$-$30\,V$. Zapojení napájecího obvodu je zobrazeno na obrázku 5.1.



Obr. 5.1: Zdroj hlavního napájecího napětí $5\,V$.

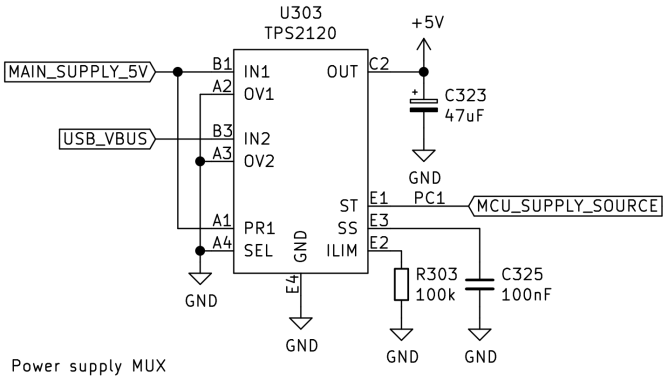
Nedílnou součástí napájecích obvodů je také ochrana proti přepólování, přepětí a zkratu. Ochrana proti zkratu je zajištěna pomocí *PPTC* pojistky, která je vratná a není nutné ji měnit při jejím vybavení. Pojistka spolu s *TVS* diodou tvoří ochranu proti přepětí, kde *TVS* dioda se stane vodivou a vytvoří tak proud, který vybaví pojistku. Ochrana proti přepólování je zajištěna pomocí *MOSFET* tranzistoru. Toto zapojení s *MOSFET* tranzistorem má výhodu oproti klasickému zapojení s antiparalelní diodou v rychlosti a v nulovém napětí. Pokud je totiž použito zapojení s antiparalelní diodou, je na výstupu tohoto ochranného obvodu úbytek napětí na diodě (napětí v propustném směru, nejčastěji $-0,6\,V$), což některým obvodům může

vadit. Zapojení s *MOSFET* tranzistorem musí být rozšířeno ještě o rezistor a Zenerovu diodu kvůli vysokému vstupnímu napětí, které je vyšší než napětí U_{GS}.

Pro snížení napětí je použit galvanicky oddělený *DC-DC* měnič `TEN6-2411WIN`.

5.1.2 Napájecí multiplexer

Pro lepší uživatelské pohodlí byla přidána možnost napájet řídicí systém pomocí USB. Je tedy nutné přepínat napájecí napětí z hlavního zdroje (uvedeného v kapitole 5.1.1). Pro tuto funkci je ideální zapojení s unipolárními tranzistory *Drain-To-Drain*. Pro zjednodušení i úsporu součástek existují specializované obvody, které jsou vhodné přímo pro tuto aplikaci. Tyto obvody jsou již přímo vybaveny řídicí logikou, zpětnou vazbou, ale také i ochrannými obvody, jako je hlídání napájecích úrovní (podpětí a přepětí), teplota obvodu a maximální proud. Byl vybrán obvod *TPS2120* a jeho zapojení je zobrazeno na obrázku 5.2.



Obr. 5.2: Multiplexer napájecího napětí.

Možnost napájet zařízení pomocí USB je ovšem omezené, protože některé moduly mohou vyžadovat pro svoji funkcionalitu hlavní napájecí napětí, které je vyšší než disponuje USB, nebo procesor modulu je galvanicky oddělen od komunikační sběrnice *MCSBUS*. Není tak tedy možné poskytnout veškerou funkcionalitu, ale především je zamýšleno pro naprogramování zařízení, jeho konfiguraci nebo stažení dat.

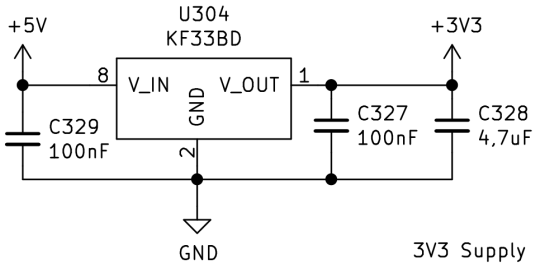
5.1.3 Napájení vnitřních obvodů

Pro napájení vnitřních obvodů se nejčastěji používá napájení $3,3\,V$ pro MCU a jeho přidružené periférie. Pro periférie vyžadující napětí $5\,V$ se používá zdroj napětí z hlavního zdroje popsaného v kapitole 5.1.1. Některé periferie vyžadující napájení skrz galvanické oddělení, je využit galvanicky oddělený DC-DC, který je

popsán v kapitole 5.1.3. V případě externího napájení (kvůli vyšší spotřebě) je použit normální DC-DC měnič s ochranou vstupu, který je popsán v kapitole 5.1.3.

Zdroj napětí 3V3

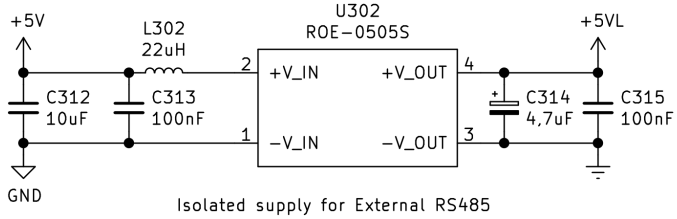
Pro napájecí napětí $3,3\,V$ byl zvolen stabilizátor napětí (*LDO*) z napájecího napětí $5\,V$, které je z jiného zdroje napětí (DC-DC měnič, USB, atd.). Použití *LDO* má výhodu potlačení šumu z napájecího napětí, který se u *LDO* udává jako parametr *PSRR*. Toto přináší především výhodu pro analogové obvody, které jsou velmi citlivé na šum napájecího napětí. Příklad zapojení *LDO* je zobrazen na obrázku 5.3. Velikost blokačních kondenzátorů u *LDO* bývá zpravidla určena doporučením od výrobce.



Obr. 5.3: Stabilizátor napětí LDO.

Galvanicky oddělené napájení 5V

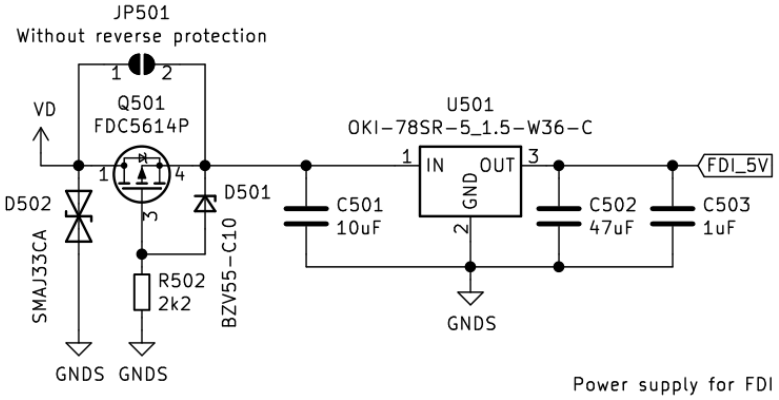
Pro obvody, které jsou galvanicky oddělené od hlavního napájení a jejich spotřeba není velká, je použit galvanicky oddělený DC-DC měnič. Jeho zapojení je na obrázku 5.4.



Obr. 5.4: Galvanicky izolovaný zdroj napětí $5\,V$.

Zdroj napájení pro galvanicky oddělené periférie 5V

Pro periferie, které potřebují napájení, jsou galvanicky oddělené od MCU a mají vyšší spotřebu, byl použit standardní DC-DC měnič. Byl vybrán modul *OKI-78SR-5/1.5-W36-C* od společnosti *MURATA POWER SOLUTIONS*, který pro svoji funkcionalitu potřebuje pouze blokační a filtrační kondenzátory. Zapojení měniče i s ochranou (popsáno v kapitole 5.1.1 na straně 61) zobrazuje obrázek 5.5.



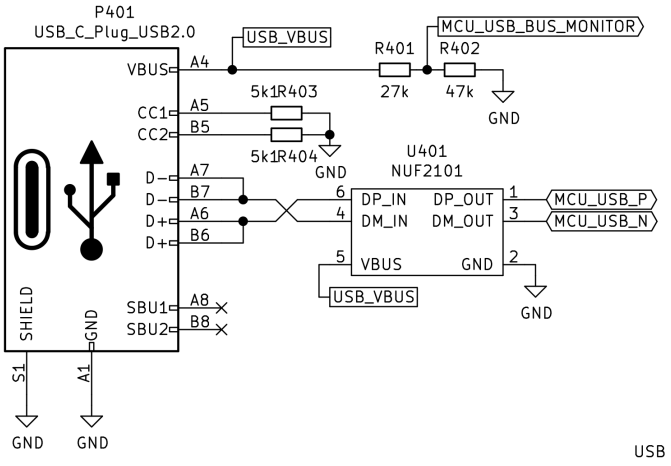
Obr. 5.5: Zdroj napětí $5\,V$ pro periférie.

5.2 Komunikační rozhraní

Tato kapitola popisuje řešení implementovaných komunikačních sběrnic, které slouží pro komunikaci mezi jednotlivými moduly modulárního řídicího systému a také pro vnější komunikaci s externími moduly se standardizovanou komunikací.

5.2.1 USB

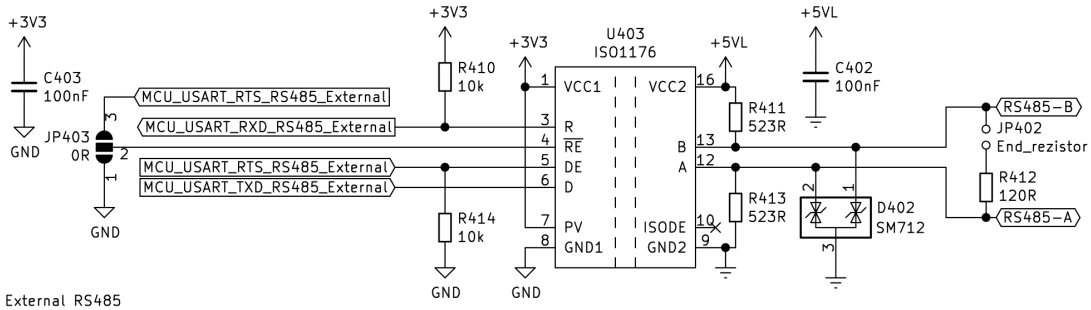
V dnešní době je standardem implementace USB do zařízení, jelikož se jedná o komunikaci, kterou je vybaven každý počítač i mobil a je tak tedy snadné propojit tato zařízení. V průmyslu se však nejedná o velmi spolehlivou sběrnici pro dlouhodobé použití, ale pro krátkodobé připojení je použitelná. Zapojení USB konektoru je zobrazeno na obrázku 5.6. Jako konektor USB byla vybrána varianta *USB-C* pro svoje rozšíření. K USB je připojen ochranný obvod *NUF2101*, který v jednom pouzdře obsahuje všechny potřebné ochrany. Dále je připojeno napájení USB do vstupu MCU přes odporový dělič napětí, aby bylo zjistitelné připojení USB.



Obr. 5.6: Zapojení USB.

5.2.2 RS485

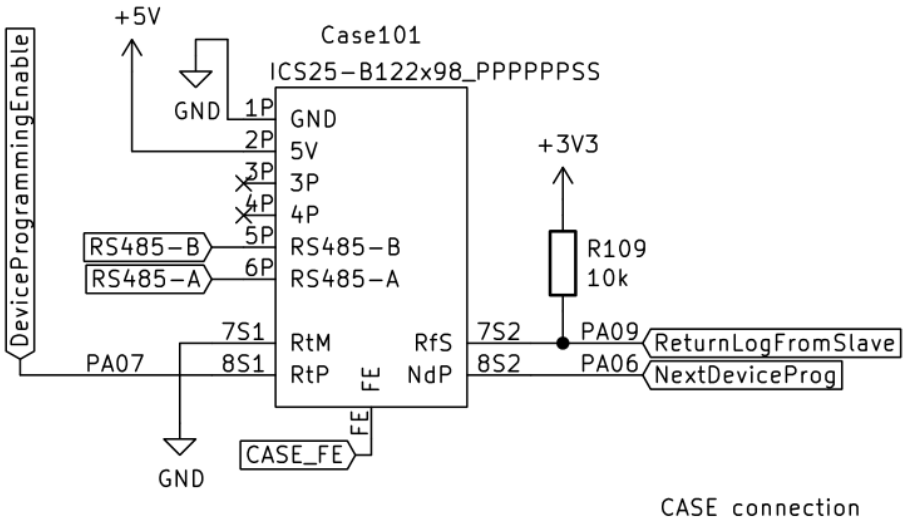
Sběrnice *RS485* je popsána v kapitole 2.1.2 na straně 31. Zapojení driveru pro sběrnici je znázorněno na obrázku 5.7. Pro externí komunikaci je sběrnice galvanicky oddělena pomocí galvanicky odděleného driveru *ISO1176*, který potřebuje napájecí napětí na obou stranách. Pro napájení externí strany driveru je použit DC-DC měnič popsaný v kapitole 5.4 na straně 63. Odpory *R411* a *R413* se využijí pouze pokud se jedná o jediné master (z hlediska vyšších vrstev komunikačního protokolu) zařízení na sběrnici, jelikož definují klidové napěťové úrovně stavu sběrnice a nejsou povinné. Odpor *R412* je zakončovací odpor a bude připojen pomocí jumperu *JP402* pouze, pokud je zařízení na konci sběrnici. Bližší popis sběrnice, funkce driveru a výpočty jsou uvedeny v kapitole 2.1.2.



Obr. 5.7: Galvanicky oddělená sběrnice RS485.

5.2.3 MCSBUS

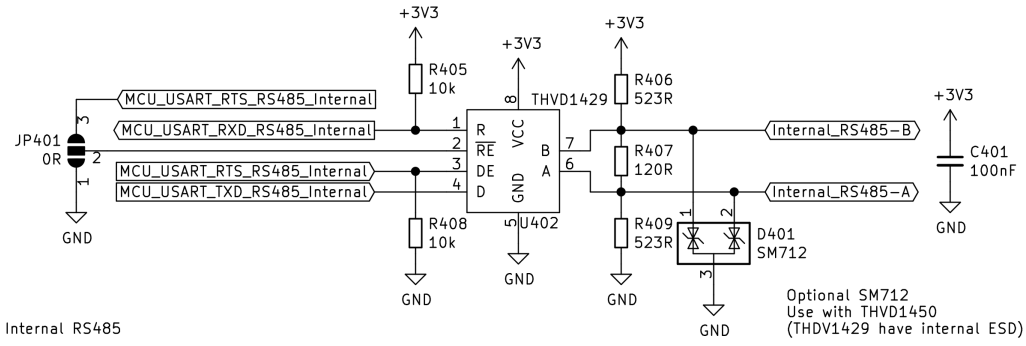
Komunikační protokol *MCSBUS* a jeho sběrnice s řídicími signály je popsán v kapitole 12 na straně 99. Na obrázku 5.8 je znázorněno zapojení jednotlivých signálů na sběrnici ke konektoru.



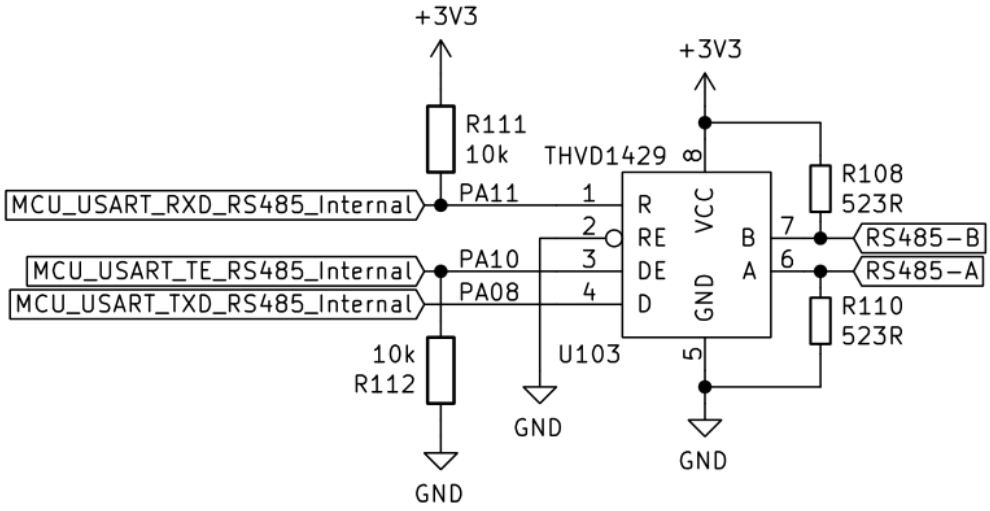
Obr. 5.8: Zapojení konektoru MCSBUS modulárního řídicího systému.

Standardní provedení

Na obrázcích 5.9 a 5.10 je ukázka zapojení driveru sběrnice *RS485*, kterou využívá *MCSBUS*. Sběrnice *RS485* je blíže popsána v kapitole 2.1.2 na straně 31.



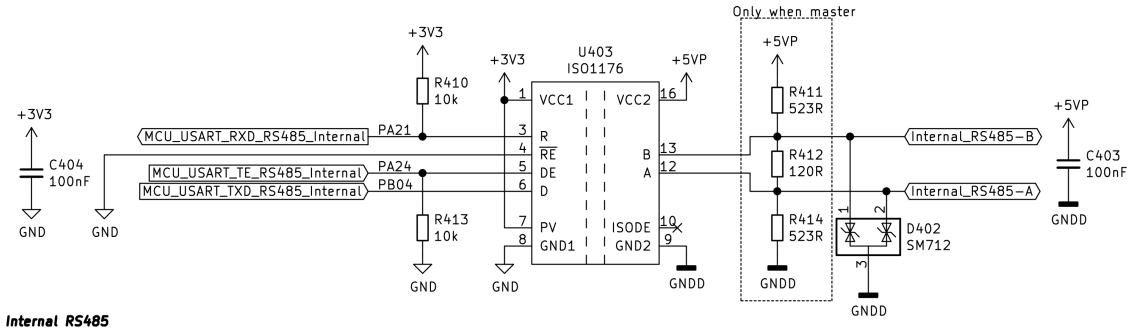
Obr. 5.9: Zapojení driveru RS485 MCSBUS pro master modul.



Obr. 5.10: Zapojení driveru RS485 MCSBUS pro slave modul.

Galvanicky oddělené provedení

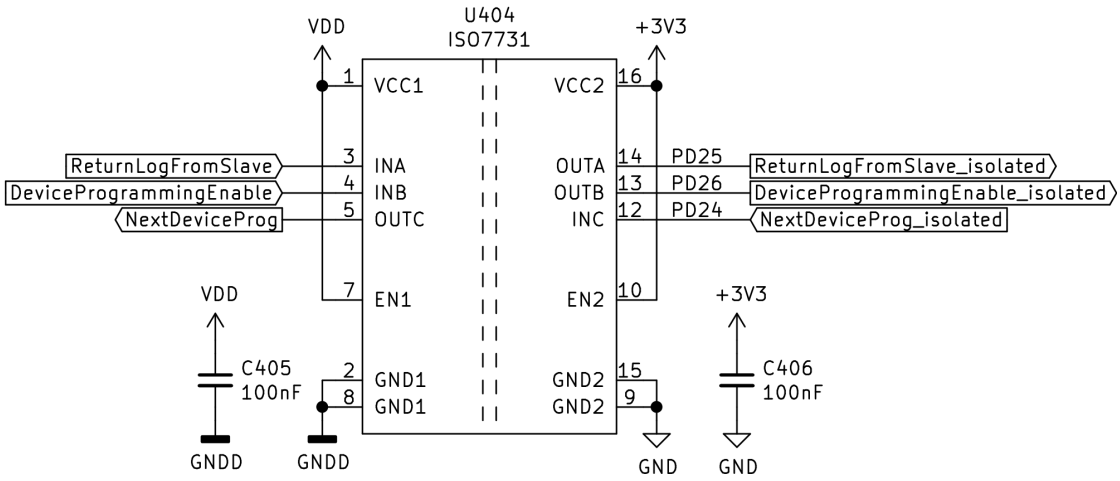
Při galvanickém oddělení sběrnice *MCSBUS* je nutné galvanicky oddělit sběrnici *RS485* a řídicí signály. Pro galvanické oddělení sběrnice *RS485* byl použit obvod *ISO1176*, jehož zapojení je zobrazeno na obrázku 5.11. Pro řídicí signály je použit obvod *ISO7731*, jehož zapojení je zobrazeno na obrázku 5.12. Galvanicky oddělený zdroj zde není potřeba, jelikož strana *MCU* je napájena stejným napájením jako samotné *MCU* a strana na sběrnici využívá napájení sběrnice.



Obr. 5.11: Galvanicky oddělená sběrnice RS485 pro MCSBUS.

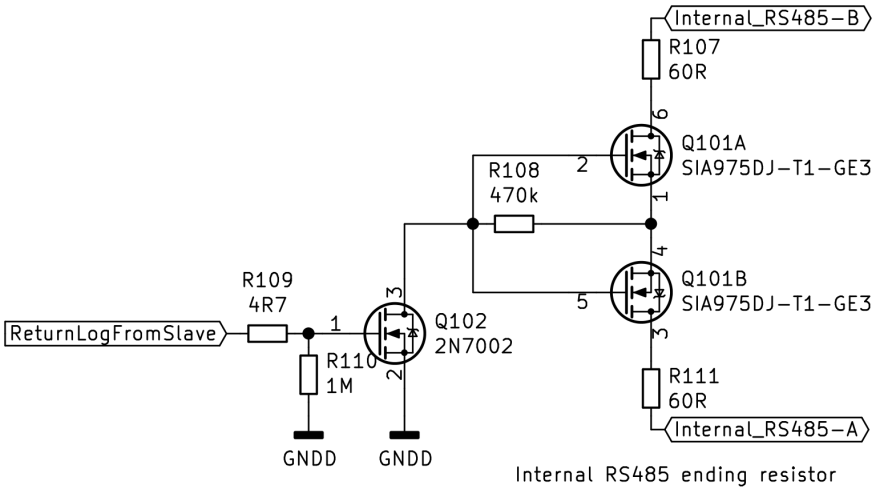
Zakončovací odpory

Sběrnice *RS485* vyžaduje pro svoji správnou funkcionalitu zakončovací odpory, jejichž význam je popsán v kapitole 2.1.2 na straně 31. Aby se předešlo nutnosti



Obr. 5.12: Galvanické oddělení řídicích signálů MCSBUS.

vkládat další modul, který by obsahoval zakončovací odpory (některé řídicí systémy používají bočnice, ve kterých jsou umístěny tyto zakončovací odpory) bylo vymyšleno elektronické spínání zakončovacích odporů posledního modulu. Zapojení elektronického spínání zakončovacího odporu je znázorněno na obrázku 5.13.

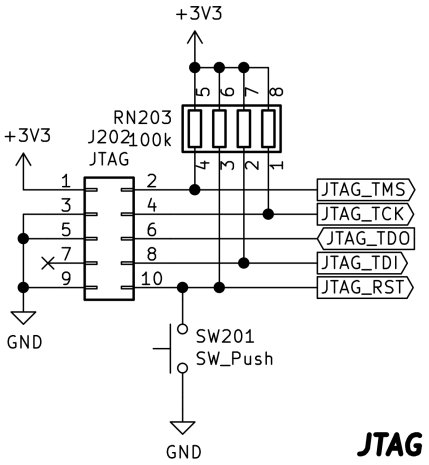


Obr. 5.13: Elektronicky spínaný zakončovací odpor.

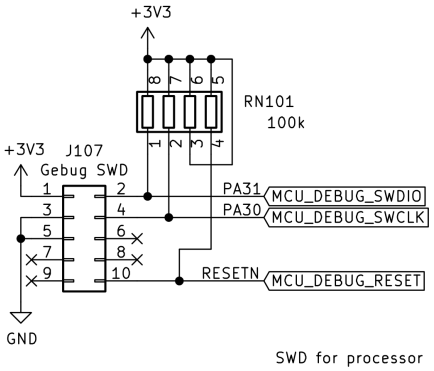
5.2.4 Programovací konektor

Pro programování modulů je používán programátor *Atmel ICE*, který využívá pro programování *pinhead* $2 \cdot 5$ s roztečí $1,27\,mm$. Tento programátor je univerzální a nabízí různé možnosti programovacích rozhraní a je nutné správně zapojit tento

konektor. Vybrané procesory se programují pomocí *SWD* a zapojení konektoru pro něj je zobrazeno na obrázku 5.15. Procesor z rodiny *ATSAME70* má však piny popsané jako pro *JTAG*, jelikož programovací rozhraní *JTAG* slouží především výrobci při kontrole procesoru. Ukázka zapojení *JTAG* je na obrázku 5.14.



Obr. 5.14: Zapojení programovacího konektoru pro JTAG.



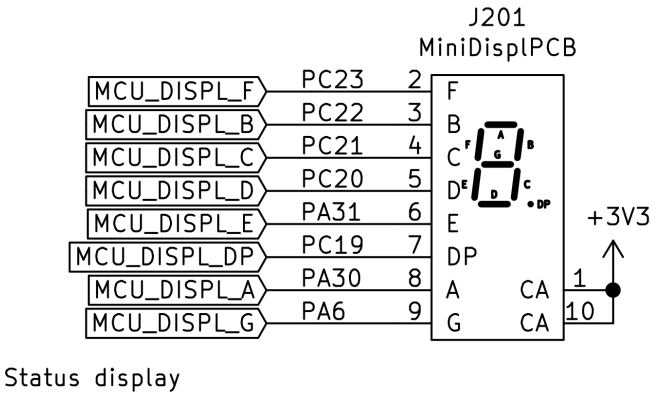
Obr. 5.15: Zapojení programovacího konektoru pro SWD.

5.3 Periferní obvody

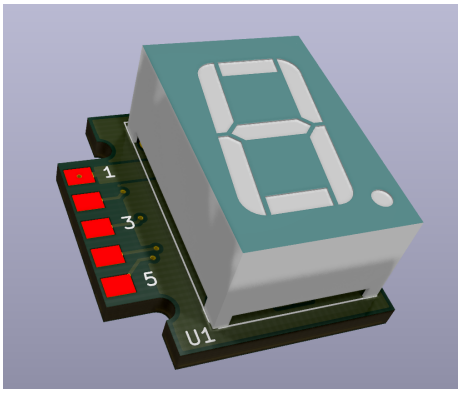
Pro sjednocení obecných vstupně-výstupních periférií, byly použity obvody popsané v této kapitole.

5.3.1 Segmentový displej

Složitější moduly, či moduly, které mohou fungovat jako master moduly, mohou být vybaveny 7-segmentovým displejem. Jelikož bylo nutné z mechanického hlediska mít displej kolmo k DPS, kde bude osazen, byl displej osazen na samostatnou DPS. Na DPS byly také přidány předřadné odpory pro displej nastavené pro napětí $3,3\,V$ pro úsporu místa. Výsledný modul je zobrazen na obrázku 5.17 a jeho připojení je zobrazeno na obrázku 5.16.



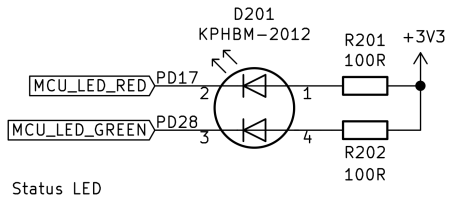
Obr. 5.16: Zapojení segmentového displeje.



Obr. 5.17: Modul segmentového displeje.

5.3.2 Led dioda

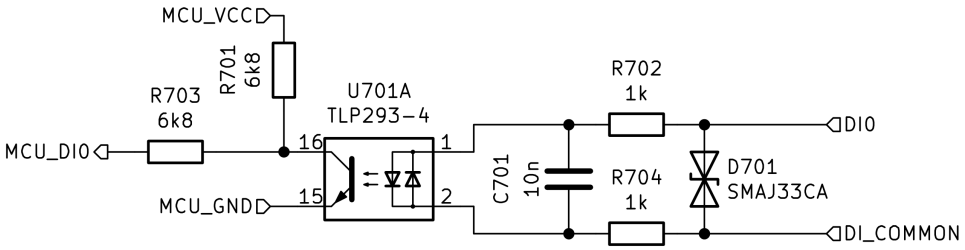
Pro jednoduchou signalizaci byla použita dvoubarevná LED dioda (červená a zelená), pomocí které může modul signalizovat svůj stav (běh či poruchu). Zapojení LED diody je na obrázku 5.18.



Obr. 5.18: Zapojení signalizační LED diody.

5.3.3 Digitální vstupy

Zapojení jednoho digitálního vstupu je zobrazeno na obrázku 5.19. Pro snížení počtu součástek, byl použit čtyřnásobný optočlen. Optočlen má 2 antiparalelní diody, takže je možné použít vstup jako *Sink* nebo *Source* v závislosti na připojení společného vstupu k zemi nebo napájecímu napětí. Vstup je navržen na napěťový rozsah $10\,V$ až $30\,V$.

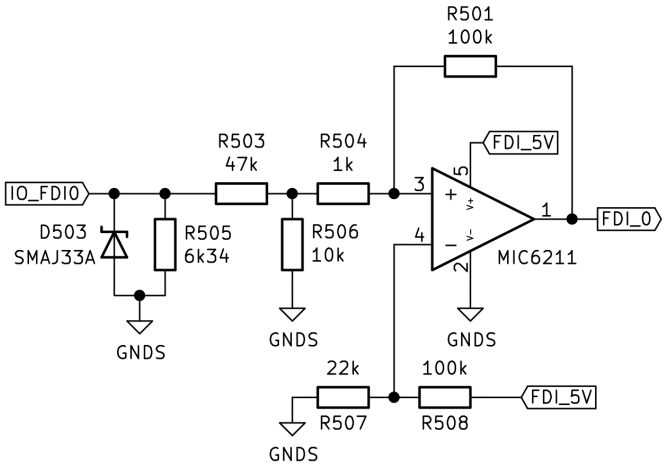


Obr. 5.19: Zapojení jednoho digitálního vstupu.

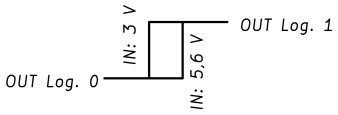
5.3.4 Rychlé digitální vstupy

Jelikož standardní vstupy jsou optimalizovány na pomalejší děje (tlačítka, snímače, atd.), je nutné mít pro snímače s vysokou výstupní frekvencí, jako jsou enkodéry, nebo snímače s frekvenčním výstupem speciální vstupy, které jsou připojeny k příslušným periferiím (čítače a časovače). Pro zrychlení digitálních vstupů musel být použit rychlejší galvanický izolátor, který není optický, ale s kapacitní vazbou a to *ISO7720*, který je zobrazen na obrázku 5.22. Tento galvanický izolátor se také nazývá jako digitální izolátor a na vstupu podporuje napěťové úrovně. Pro zlepšení rychlosti hran byl implementován komparátor s hysterezí realizovaný pomocí operačního zesilovače, jehož zapojení je znázorněno na obrázku 5.20. Hystereze vstupu je znázorněna na obrázku 5.21. Tento obvod s komparátorem zajišťuje nejen zrychlení hran, ale také unifikaci pro vstupní napětí. Komparátor s digitálním obvodem

vyžaduje vlastní napájecí napětí, kterým je $5\,V$. Byla zvolena možnost externího napájení, které je zobrazeno na obrázku 5.5 na straně 64.



Obr. 5.20: Zapojení logiky rychlých digitálních vstupů.



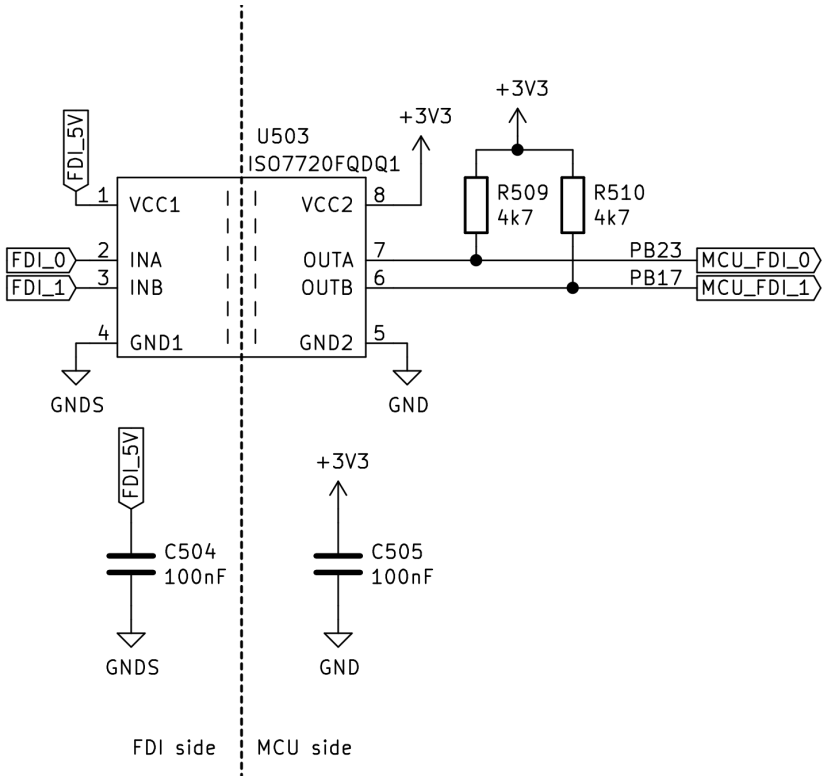
Obr. 5.21: Vstupní napěťová hystereze.

5.3.5 Digitální výstupy

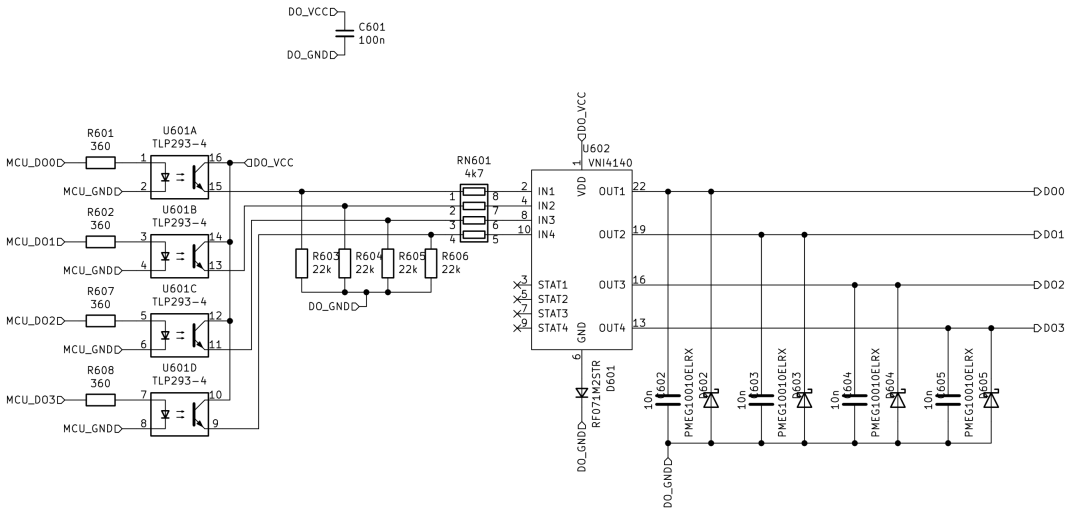
Digitální výstupy jsou realizovány pomocí obvodu *VNI4140*, což je čtyřnásobný spínací prvek určený pro spínání i induktivních zátěží. Jeho výstupní topologie je s otevřeným kolektorem a obsahuje také ochrany jednotlivých kanálů. Zapojení čtveřice digitálních výstupů s galvanickým oddělením je zobrazeno na obrázku 5.23.

5.3.6 Rychlé digitální výstupy

Digitální výstupy popsané v kapitole 5.3.5 mají nevýhodu, že jejich maximální spínací frekvence může být do $10\,kHz$, což pro některé úlohy nemusí být dostačující. Pro rychlé digitální výstupy byl použit obvod *DRV884*, který je primárně určen pro řízení motorů. *DRV884* obsahuje 4 poloviční H-Můstky. Pro galvanické oddělení je použit digitální izolátor *ISO7740*. Zapojení rychlých digitálních výstupů je zobrazeno na obrázku 5.24.



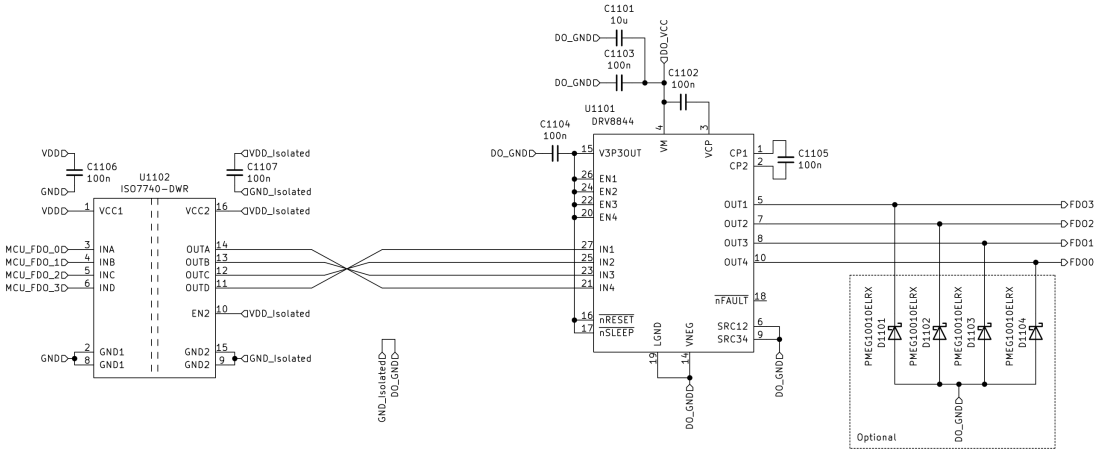
Obr. 5.22: Galvanické oddělení rychlých vstupů.



Obr. 5.23: Zapojení čtveřice digitálních výstupů.

5.4 Specializované obvody

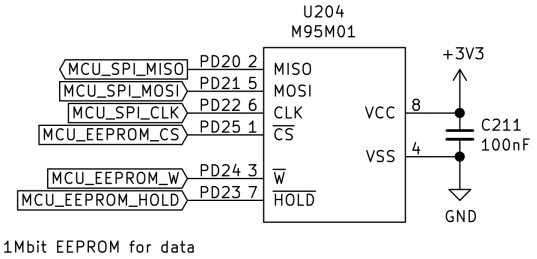
Tato kapitola popisuje používané obvody se speciální funkcí.



Obr. 5.24: Zapojení čtveřice rychlých digitálních výstupů.

5.4.1 EEPROM paměť

Pro ukládání nastavení byla zvolena *EEPROM* paměť *M95M01*, která je o velikosti 1 *Mbit*. Paměť je se sběrnicí SPI a její zapojení je na obrázku 5.25.



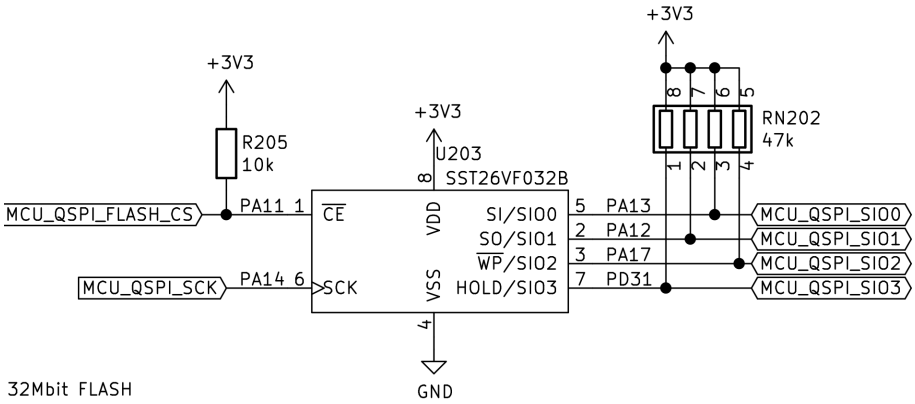
Obr. 5.25: Zapojení EEPROM paměti (M95M01).

5.4.2 FLASH paměť

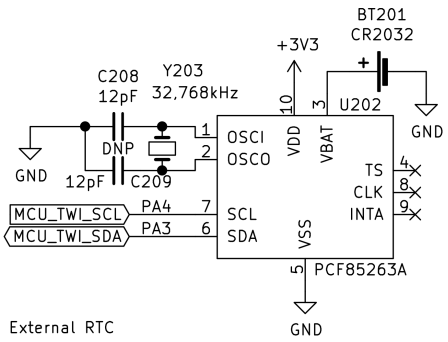
Pro ukládání objemnějších dat jsou některé moduly vybaveny pamětí *SST26VF032B* typu *FLASH* o velikosti 32 *Mbit*. Tato paměť je vybavena sběrnicí QSPI.

5.4.3 RTC – hodiny reálného času

Jelikož MCU nemají zálohování hodin reálného času, je nutné použít externí obvod. Jako obvod RTC byl použit *PCF85263A*, který má sběrnici I^2C. Jeho zapojení je zobrazeno na obrázku 5.27.



Obr. 5.26: Zapojení FLASH paměti (SST26VF032B).



Obr. 5.27: Zapojení obvodu RTC (PCF85263A).

6 Návrh procesorové jednotky

Nejdůležitější částí modulárního řídicího systému je procesorová jednotka (procesorový modul, řídicí modul, atd.). Jeho hlavní činností je vykonávání uživatelského programu a komunikace se slave moduly, které jsou vstupně výstupními jednotkami. Procesorová jednotka tedy nemusí obsahovat žádné vstupně/výstupní periférie. Nejvhodnější však je, pokud procesorová jednotka obsahuje i externí komunikace, jelikož v sobě nese všechna potřebná data a nemusejí se tato data předávat jinému modulu, který by zprostředkovával komunikaci těchto dat.

V závislosti na různých požadavcích byly navrženy dvě procesorové jednotky. Jedna procesorová jednotka je zaměřena na výkon a komunikace a je popsána v kapitole 6.1. Druhá procesorová jednotka (kapitola 6.2, strana 77) je odlehčenější, volné místo zaplňují vstupně/výstupní periferie a je označena jako *LITE*.

6.1 Procesorová jednotka

Nejdůležitějšími prvky procesorové jednotky jsou:

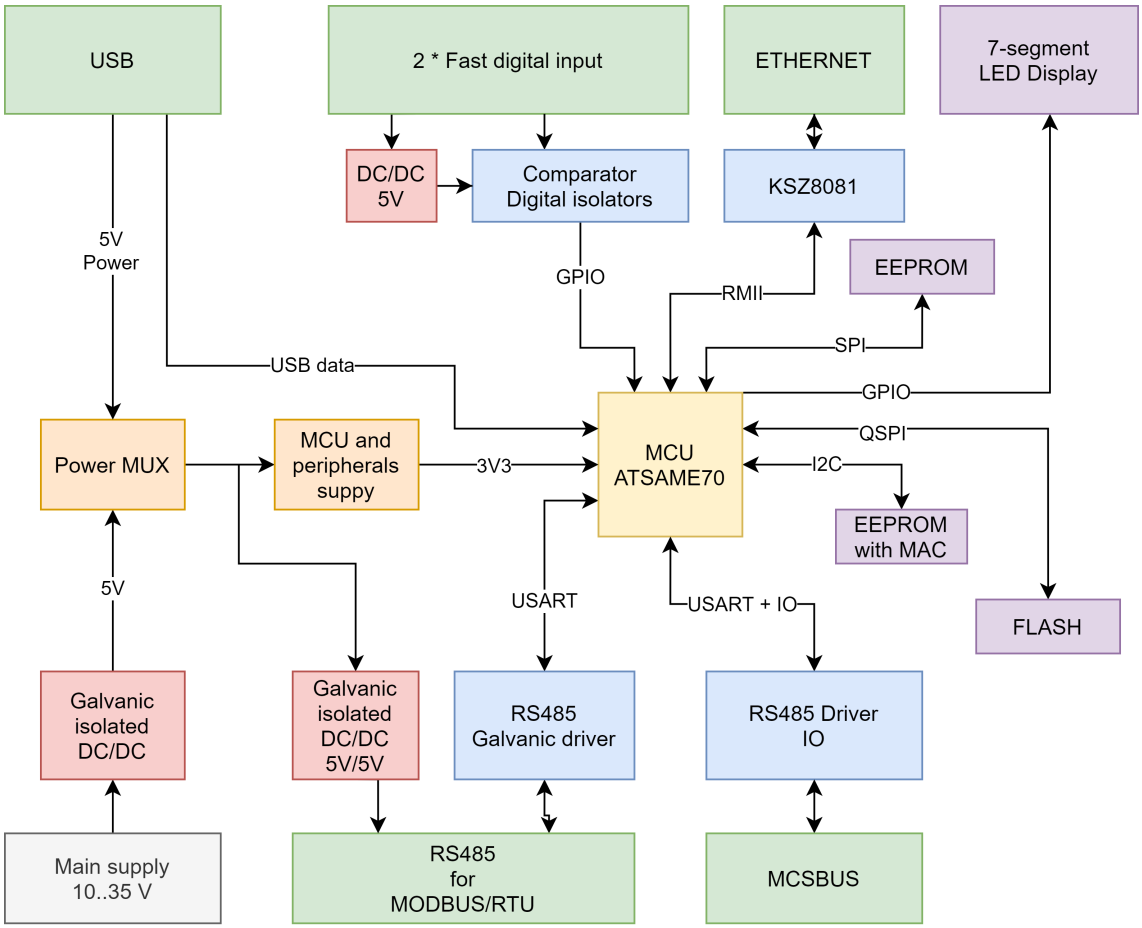
- **Ethernet** – kapitola 6.1.2 na straně 76.
- **MODBUS/RTU** (*RS485*) – kapitola 5.2.2 na straně 65.
- **USB** – kapitola 5.2.1 na straně 64.
- **EEPROM paměť** – kapitola 5.4.1 na straně 74.
- **FLASH paměť** – kapitola 5.4.2 na straně 74.
- **7-segmentový displej** – kapitola 5.3.1 na straně 70.
- **2 rychlé digitální vstupy** – kapitola 5.3.4 na straně 71.

6.1.1 Blokové schéma zapojení

Na obrázku 6.1 je zobrazeno zjednodušené blokové schéma zapojení procesorové jednotky. Celé schématické zapojení procesorové jednotky je uvedeno v příloze A.1 na straně 133.

6.1.2 Ethernet

Čip realizující fyzickou vrstvu je *KSZ8081*, jehož zapojení je na obrázku 6.2, který komunikuje s procesorem pomocí *RMII*. Fyzická vrstva *KSZ8081* podporuje *10Base-T/100Base-TX* podle normy *IEEE 802.3*. Pro svoji správnou funkci potřebuje *KSZ8081* přiřadit *MAC adresu*, což je jedinečná adresa síťového zařízení. *MAC adresa* se musí koupit od dodavatele a nejjednodušším způsobem při malé sérii výroby je použití paměťového obvodu, který ji má v sobě již přidělenou. Byla vybrána



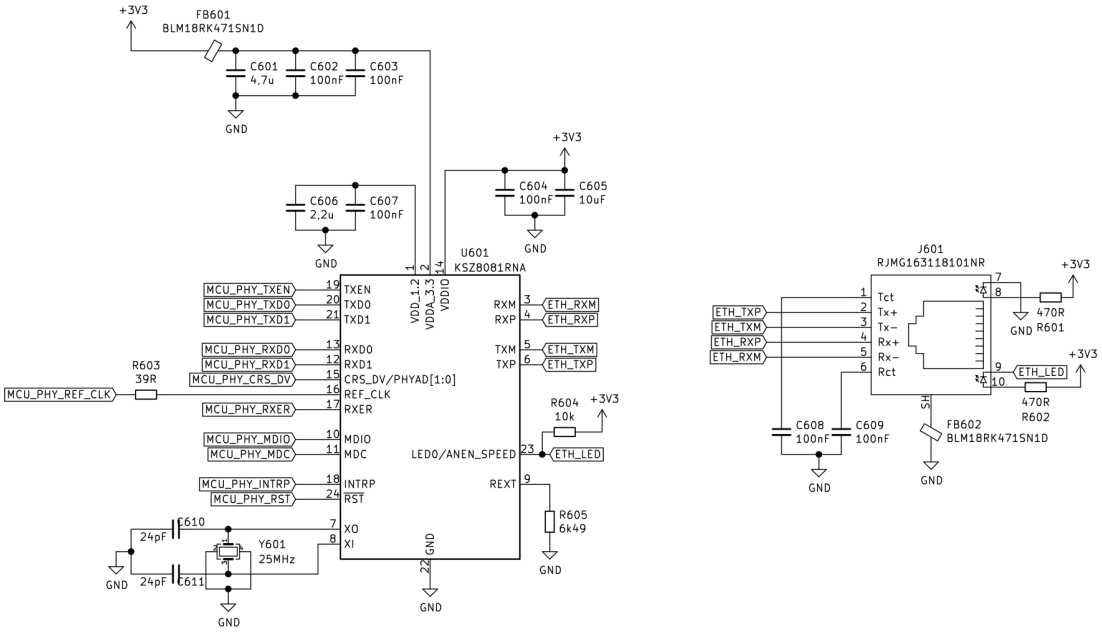
Obr. 6.1: Blokové schéma zapojení procesorové jednotky.

paměť typu *EEPROM* s označením *AT24CS02-MAHM*, jejíž zapojení je zobrazeno na obrázku 6.3.

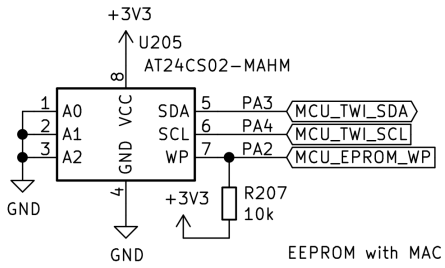
6.2 Procesorová jednotka LITE

Nejdůležitějšími prvky procesorové jednotky LITE jsou:

- **MODBUS/RTU** (*RS485*) – kapitola 5.2.2 na straně 65.
- **USB** – kapitola 5.2.1 na straně 64.
- **EEPROM paměť** – kapitola 5.4.1 na straně 74.
- **FLASH paměť** – kapitola 5.4.2 na straně 74.
- **7-segmentový displej** – kapitola 5.3.1 na straně 70.
- **4 digitální výstupy** – kapitola 5.3.5 na straně 72.
- **8 digitálních vstupů** – kapitola 5.3.3 na straně 71.
- **2 rychlé digitální vstupy** – kapitola 5.3.4 na straně 71.



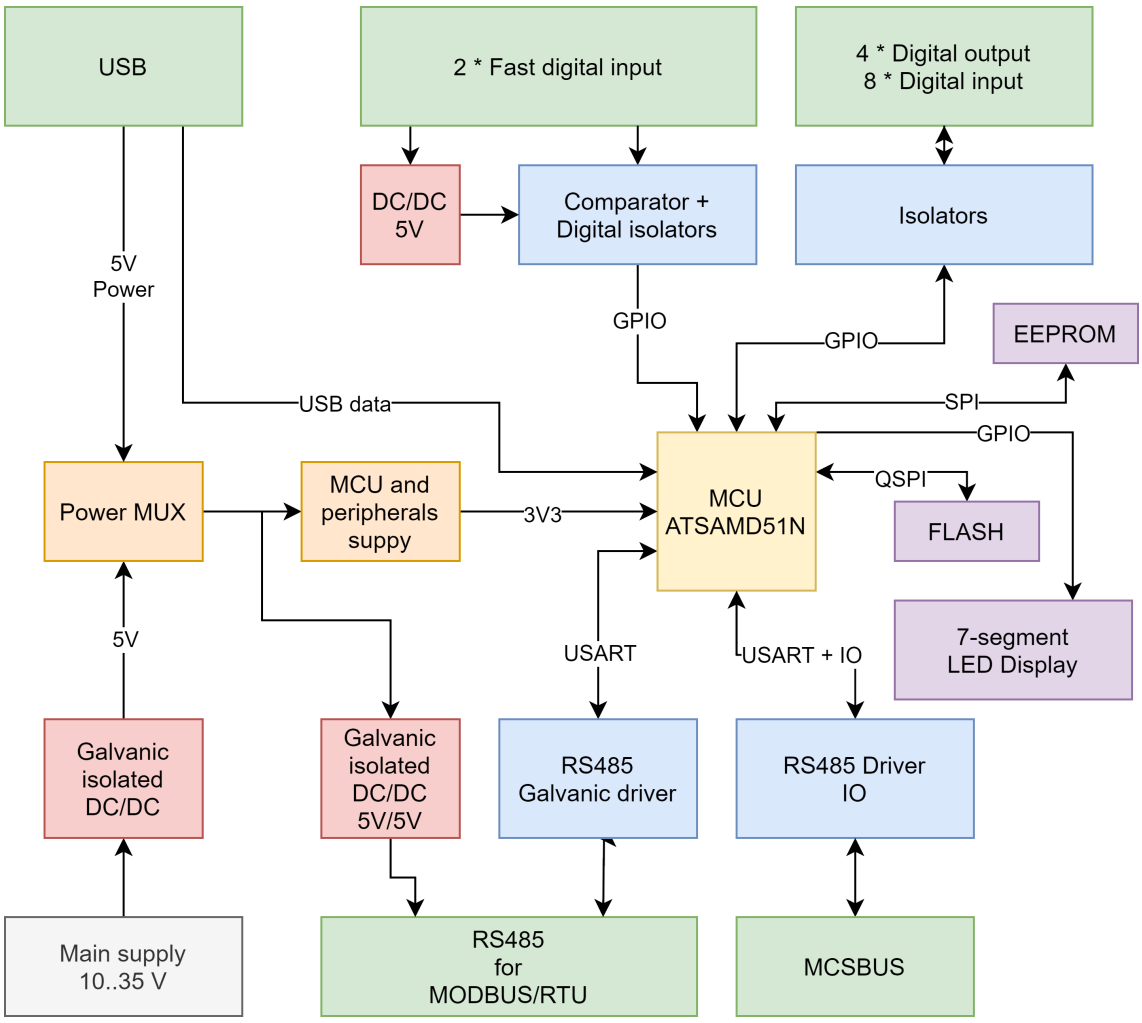
Obr. 6.2: Zapojení Ethernetového driveru.



Obr. 6.3: Zapojení paměti EEPROM s MAC adresou.

6.2.1 Blokové schéma zapojení

Na obrázku 6.4 je zobrazeno zjednodušené blokové schéma zapojení procesorové jednotky LITE. Celé schématické zapojení procesorové jednotky je uvedeno v příloze A.2 na straně 140.



Obr. 6.4: Blokové schéma zapojení procesorové jednotky LITE.

7 Návrh analyzátoru sítě

Požadavky na analyzátor sítě, jakožto modul modulárního řídicího systému, se zaměřovaly spíše na možnost měření síťových parametrů, jako je napětí, proud, výkon a účiník. Kromě těchto základních parametrů byl také požadavek na možnost měření frekvenčního spektra signálů pomocí *FFT*. Hlavní využití tohoto modulu je typicky přehledové a určené pro řízení, kde oproti konvenčním analyzátorům sítě není určen pro hodnocení kvality sítě. Dalším důležitým parametrem byla možnost měření stejnosměrného napětí. Snahou pro širší aplikaci tohoto modulu bylo udělat ho nezávislým a samostatně použitelným, nebo jej použít jako master modul (procesorovou jednotku) modulárního řídicího systému.

Nejdůležitějšími prvky analyzátoru sítě jsou:

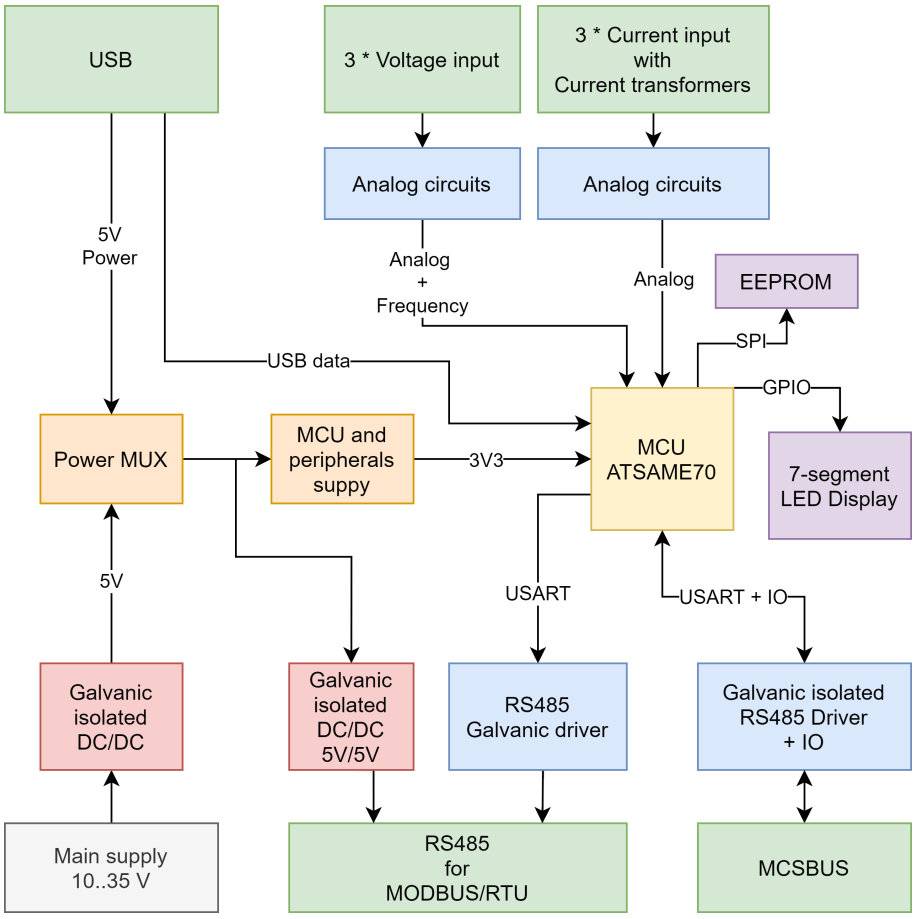
- **MODBUS/RTU** (*RS485*) – kapitola 5.2.2 na straně 65.
- **USB** – kapitola 5.2.1 na straně 64.
- **EEPROM paměť** – kapitola 5.4.1 na straně 74.
- **7-segmentový displej** – kapitola 5.3.1 na straně 70.
- **Měření napětí** – kapitola 7.2.2 na straně 82.
- **Měření frekvence** – kapitola 7.2.3 na straně 83.
- **Měření proudu** – kapitola 7.2.4 na straně 84.

Na obrázku 7.1 je zobrazeno zjednodušené blokové schéma zapojení procesorové jednotky. Celé schématické zapojení modulu analyzátoru sítě je uvedeno v příloze A.3 na straně 149.

7.1 Parametry analyzátoru sítě

Parametr	Hodnota
Počet měřených kanálů	3
Rozsah měření napětí	$0..500\,VDC$
Rozlišení měření napětí	$250\,mVDC$
Rozsah měření proudu	$0..5\,AAC$
Rozlišení měření proudu	$1\,mAAC$
Vzorkovací frekvence	$40\,kHz$
Rozsah měření frekvence	$0..100\,Hz$
Šířka pásma	$7\,kHz$
Nejvyšší měřená harmonická	64
Rozlišení AD převodníku	$12\,bit$

Tab. 7.1: Parametry měření analyzátoru sítě.



Obr. 7.1: Blokové schéma zapojení analyzátoru sítě.

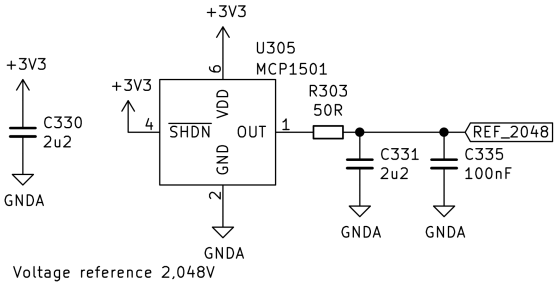
7.2 Měřicí obvody

Pro napájení analogových obvodů je použito napětí $3,3\,V$, které je tvořeno lineárním stabilizátorem z hlavního napájecího obvodu $5\,V$. Napěťová reference pro měření je popsána v kapitole 7.2.1, kterou využívají obvody pro měření. Pro úsporu byl použit ADC procesoru a analogové obvody musely být přizpůsobeny jeho parametrům. Použitím ADC procesoru se může dosáhnout jednoduše vyšší vzorkovací frekvence, malý fázový posun mezi jednotlivými kanály (vyžaduje využití DMA) a nízké ceny.

7.2.1 Analogové napájecí obvody

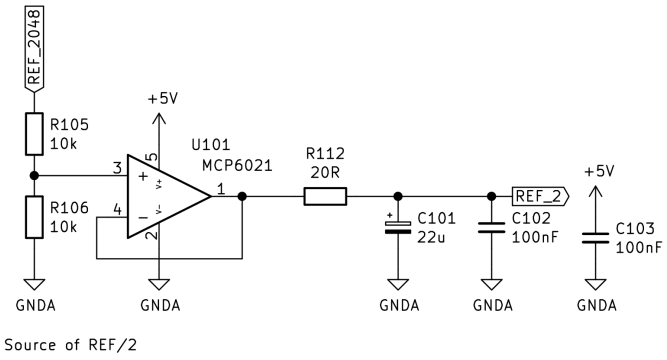
Zapojení napěťové reference je zobrazeno na obrázku 7.2. Napětí napěťové reference bylo zvoleno $2,048\,V$.

Jelikož procesor nepodporuje měření záporného napětí, ale pouze rozdílové měření napětí, bylo nutné modulovat hodnoty okolo střední hodnoty napětí reference.



Obr. 7.2: Zapojení zdroje referenčního napětí.

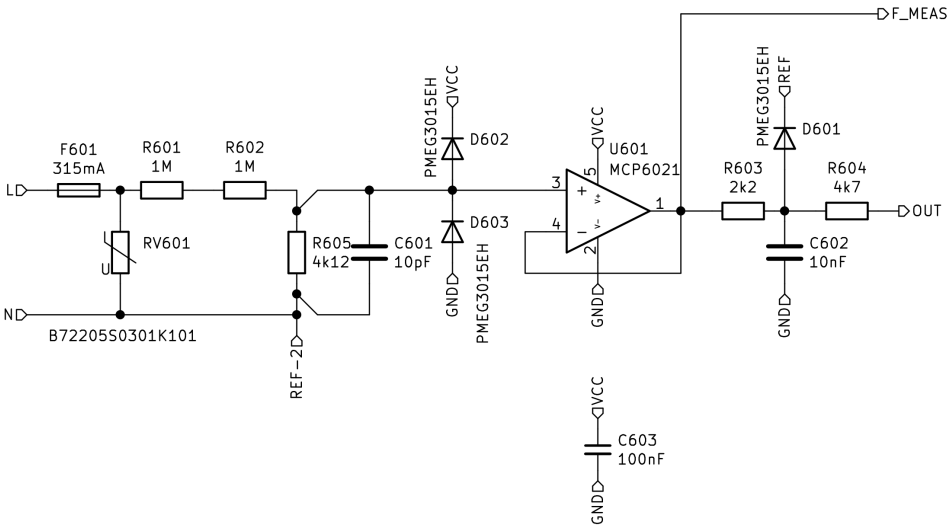
Shématické zapojení zdroje polovičního napětí referenčního napětí je znázorněno na obrázku 7.3.



Obr. 7.3: Zapojení děliče napětí referenčního napětí.

7.2.2 Měření napětí

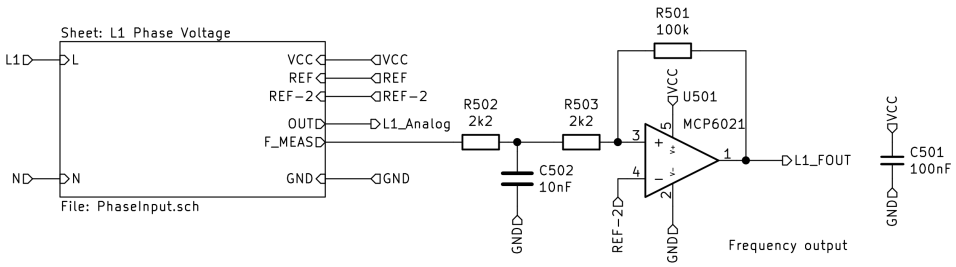
Zapojení analogového zpracování signálů pro měření napětí je zobrazeno na obrázku 7.4. Na vstupu je ochranný obvod, který se skládá z pojistky (ochrana zkratu) a varistoru, který při vysokém napětí způsobí vyšší proud do vstupu, což vybaví pojistku. Následuje napěťový dělič, kde je jeho neutrální pól připojen k polovičnímu referenčnímu napětí, čímž je zajištěna modulace okolo této hodnoty. Následně je použit operační zesilovač v zapojení sledovače pro přizpůsobení vstupní impedance. Na vstupu operačního zesilovače jsou zapojeny ochranné diody, které omezují případné napětí mimo povolený rozsah na jeho vstupu. Na výstupu se nachází ochranný obvod, zajišťující ochranu analogového vstupu, aby výstupní napětí nebylo vyšší, než referenční (navýšené o hodnotu napětí diody *D601* v propustném směru) a dolnopropustní *RC* článek.



Obr. 7.4: Zapojení zpracování analogového signálu síťového napětí.

7.2.3 Měření frekvence

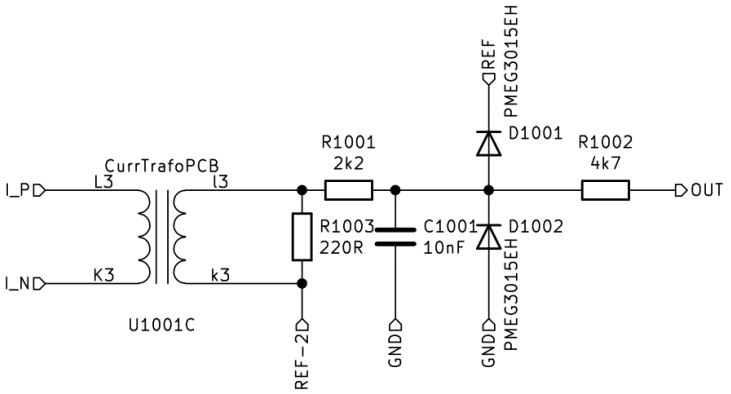
Měření frekvence je kvůli přesnosti realizováno čítačem procesoru. Čítač procesoru podporuje externí vstup nebo vstup od vnitřní periférie *ACC* (analogový komparátor). Byl zvolen přístup pomocí externího vstupu, který vyžaduje upravení signálu a generování signálu obdélníkového tvaru (případný přechod na vnitřní periférii nevyžaduje zásah do vnějšího zapojení). Zapojení generátoru obdélníkového signálu je zobrazeno na obrázku 7.5, který využívá napěťový průběh upraveného vstupního signálu, jehož zpracování je popsánop v kapitole 7.2.2. Operační zesilovač je v zapojení komparátoru bez hystereze. Měření frekvence probíhá pouze v jedné fázi.



Obr. 7.5: Zapojení zpracování signálu síťové frekvence.

7.2.4 Měření proudu

Zapojení měření proudu je zobrazeno na obrázku 7.6. Pro měření proudu byl zvolen princip měření pomocí proudového transformátoru. Na výstupu proudového transformátoru je připojen odpor *R1003*, který převádí výstupní proud na napětí a udává tak citlivost. Jelikož proudový transformátor má nízkou výstupní impedanci, nebylo nutné přizpůsobovat vstupní impedanci. Na výstupu proudového transformátoru je dále dolnopropustní *RC* článek a ochrana vstupu procesoru.

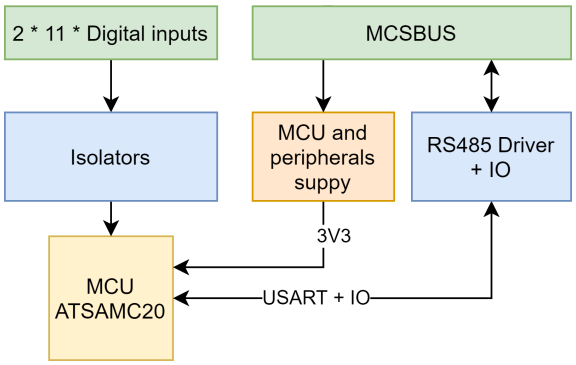


Obr. 7.6: Zapojení zpracování analogového signálu síťového proudu.

8 Návrh digitálních vstupů

Modul digitálních vstupů je příkladem jednoduchého modulu vstupně/výstupních periférií, kde procesor slouží především pro zpracování komunikace (přepsání vstupů a výstupů v rámci komunikace). Digitální vstupy jsou popsány v kapitole 5.3.3 na straně 71.

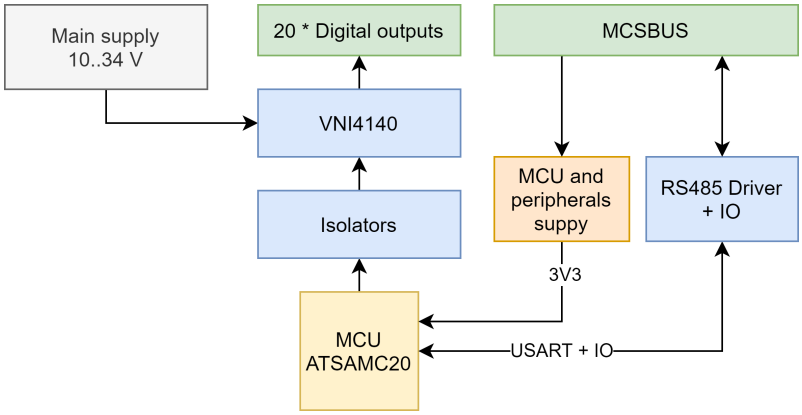
Na obrázku 8.1 je zobrazeno zjednodušené blokové schéma zapojení modulu digitálních vstupů. Celé schématické zapojení modulu digitálních vstupů je uvedeno v příloze A.4 na straně 162.



Obr. 8.1: Blokové schéma zapojení modulu digitálních vstupů.

9 Návrh digitálních výstupů

Na obrázku 9.1 je zobrazeno zjednodušené blokové schéma zapojení modulu digitálních vstupů. Celé schématické zapojení modulu digitálních výstupů je uvedeno v příloze A.5 na straně 170.



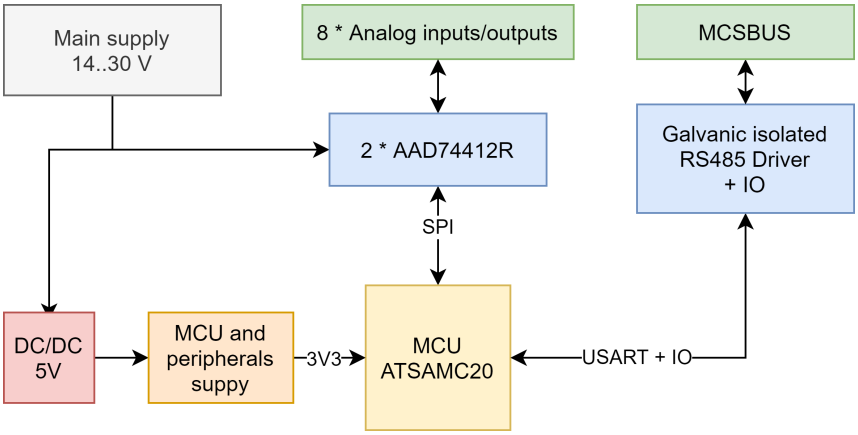
Obr. 9.1: Blokové schéma zapojení modulu digitálních výstupů.

10 Návrh analogových vstupů a výstupů

Standardní řešení periférií je řešeno pomocí oddělených periférií, kde každá svorka je vyhrazena pro analogové vstupy, analogové výstupy, digitální vstupy a digitální výstupy. Některá řešení mají tyto svorkovnice rozšířené, jestli se jedná o proudový nebo napěťový typ. Toto se v poslední době mění a výrobci začínají jít cestou *SWIO* (softwarově konfigurovatelných vstupně–výstupních periférií). Technolog pak nemusí řešit rozdělení periférií, ale pouze celkový počet potřebných periférií, protože periférie se nastaví programově. Další výhodou může být lepší výměna technologie (výměna pomocí přepojení konektorů), kde například jedna technologie využívá více analogových vstupů a druhá má více analogových výstupů a dojde pouze ke změně konfigurace modulu. Další nespornou výhodou může být zaměnitelnost modulu, kde zákazník bude mít na skladě pouze jeden typ modulu a nemusí řešit skladování několika druhů vstupně–výstupních periferních modulů. Nevýhodou je však vyšší cena, složitost, nároky na programování a také menší hustota periférií na modul. Pokud tedy výsledná aplikace má pouze přesně definované periférie, které jsou neměnné po celou dobu existence, je pak asi výhodnější mít pouze moduly se specifickou funkcí.

10.1 Blokové schéma zapojení

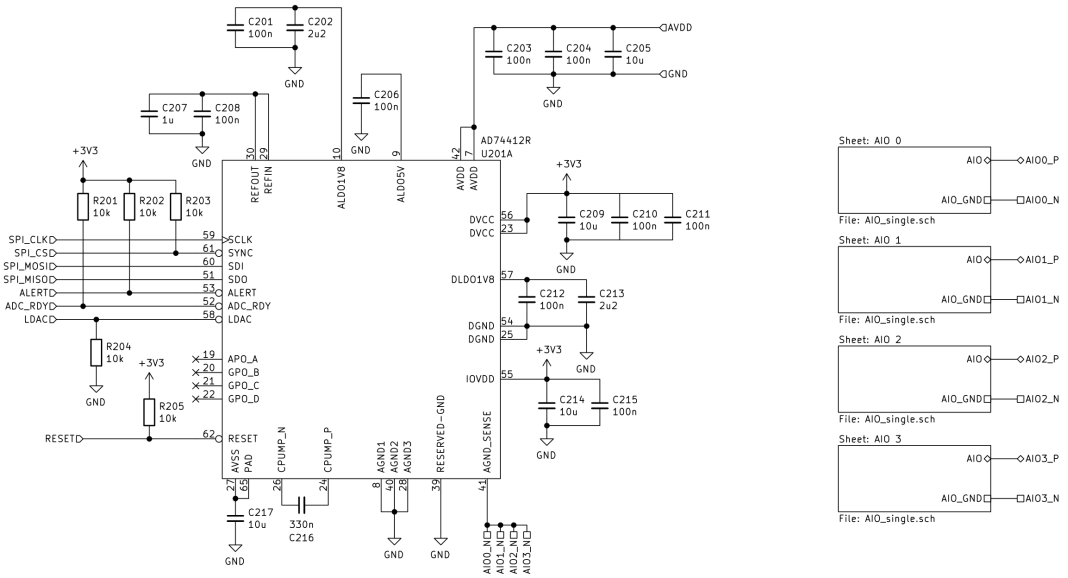
Na obrázku 10.1 je zobrazeno zjednodušené blokové schéma zapojení modulu digitálních vstupů. Celé schématické zapojení modulu analogových vstupů a výstupů je uvedeno v příloze A.6 na straně 177.



Obr. 10.1: Blokové schéma zapojení modulu analogových periférií.

10.2 Návrh SWIO

Požadavky na možnost analogové periferie byly dle průmyslových standardů. Jedná se především o napěťové (rozsah 0..10 V) a proudové (rozsah 0..20 mA nebo 4..20 mA) vstupně–výstupní periferie a měření odporu (odporové snímače - např. *RTD*). Z těchto požadavků byl vybrán obvod *AD74412R*, jehož zapojení je znázorněno na obrázku 10.2 a 10.3. Z hlediska přehlednosti bylo zapojení rozděleno do 5 stránek, kde na jedné straně je zapojení komunikace a napájení (obrázek 10.2) a výstupní úroveň je na 4 stranách (jeden *AD74412R* obsahuje 4 IO), kde každá strana představuje jednu výstupní úroveň, jelikož je opakující se (obrázek 10.3).

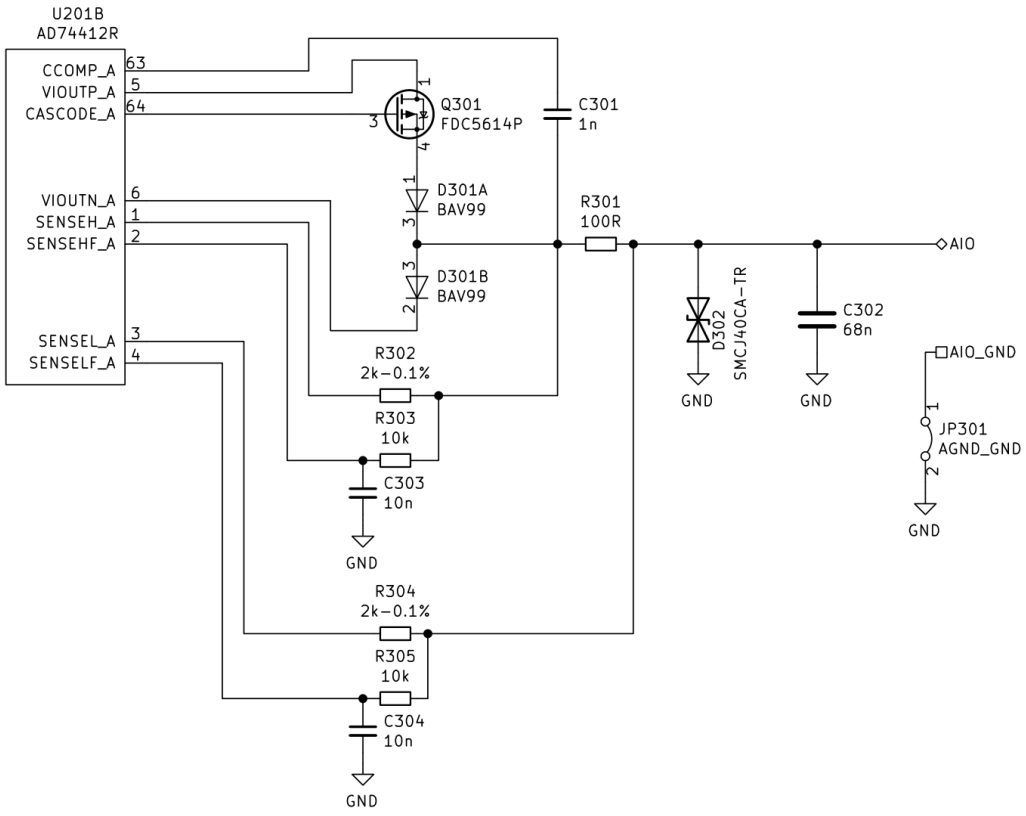


Obr. 10.2: Zapojení napájení a komunikace obvodu AD74412R.

Aby byl modul zaplněn, byly použity 2 čipy *AD74412R* a modul má tedy 8 konfigurovatelných periférií. Obvod *AD74412R* komunikuje s *MCU* pomocí sběrnice *SPI*, pomocí které se obvod nakonfiguruje a následně čtou data. Pro zrychlení komunikace jsou oba obvody na samostatné sběrnici *SPI*.

10.3 Možnosti zapojení

Každý měřicí kanál má vyvedeny 2 připojovací body a to `AIOx_P` a `AIOx_N`. Vstup `AIOx_N` je společný a má stejný potenciál pro všechny kanály (*GND*). Každý kanál je softwarově konfigurovatelný, což zamezilo použití přepínačů (konfigurátorů) nebo



Obr. 10.3: Zapojení výstupního obvodu obvodu AD74412R.

více vstupních svorek na kanál. Tato kapitola popisuje jednotlivé možnosti zapojení měřicích obvodů podporovaných tímto modulem.

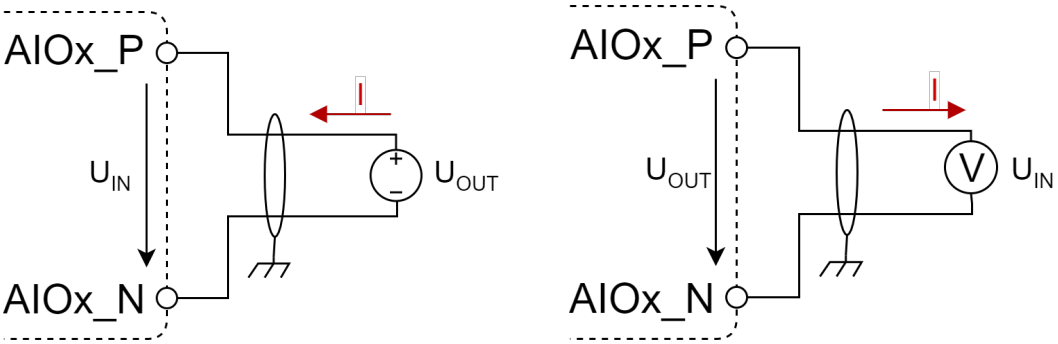
10.3.1 Napěťový vstup a výstup

Zapojení napěťového vstupu je znázorněno na obrázku 10.4a a zapojení výstupu je znázorněno na obrázku 10.4b.

10.3.2 Digitální vstup

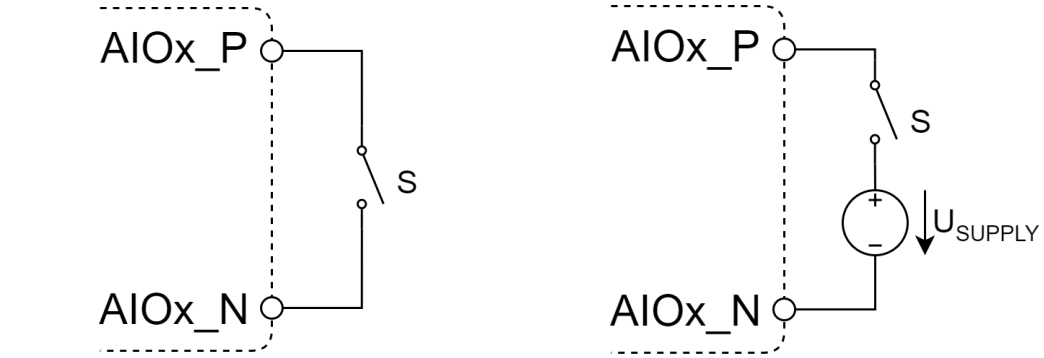
Analogový kanál také podporuje funkci digitálního vstupu ve dvou režimech:

- **Vlastní napájení**, kde není nutné napájet spínač externím napájením, ale zapojí se mezi svorky kanálu. Vyhodnocování probíhá na základě proudu spínačem. Zapojení je zobrazeno na obrázku 10.5a.
- **Externí napájení**, kdy je spínač napájen pomocí externího zdroje napětí. Vyhodnocení je na základě měření napětí. Zapojení je znázorněno na obrázku 10.5b.



(a) Napěťový vstup. (b) Napěťový výstup.

Obr. 10.4: Připojení napěťového vstupu a výstupu ke svorkám modulu.



(a) Digitální vstup s vlastním napájením. (b) Digitální vstup s externím napájením.

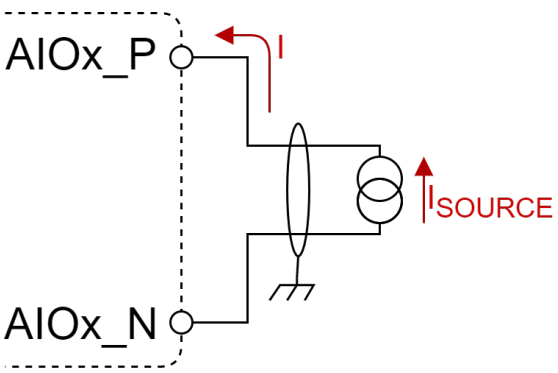
Obr. 10.5: Připojení digitálního vstupu ke svorkám modulu.

10.3.3 Proudový vstup a výstup

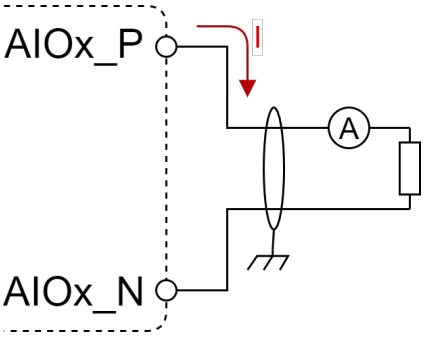
Zapojení proudového vstupu s externím napájením je znázorněno na obrázku 10.6a. Je možné také zapojit analogový vstup tak, aby byl snímač napájen přímo z kanálu, což je znázorněno na obrázku 10.6b. Svorka `AIOx_P` se pak připojí k napájecímu napětí analogového obvodu a měří se proud protékající do snímače. Stejné zapojení se používá pro proudový výstup, jen vnitřní logiku tvoří zdroj proudu.

10.3.4 Měření odporu

Připojení snímače s proměnlivým odporem pro měření jeho odporu je znázorněno na obrázku 10.7a. Náhradní schéma je znázorněno na obrázku 10.7b, podle kterého lze vypočítat proud snímačem pro stanovení jeho chyby. *Pull-up* odpor v tomto zapojení má hodnotu 2100 Ω.

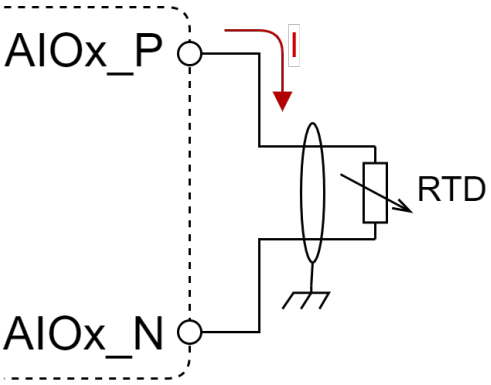


(a) Proudový vstup s externím napájením.

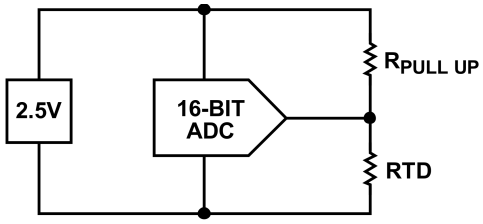


(b) Proudový výstup nebo vstup s vlastním napájením.

Obr. 10.6: Připojení proudového vstupu a výstupu ke svorkám modulu.



(a) Připojení měřeného odporu ke svorkám modulu.



(b) Náhradní schématické zapojení měřicího obvodu.[2]

Obr. 10.7: Zapojení měřeného odporu.

11 Návrh HMI (displeje)

Modul displeje je velice specifický, jelikož v době návrhu bylo zapotřebí mít řídicí systém, který je vhodné použít pro řízení vstřikovacího lisu na plasty[25]. Modul displeje tak tedy bylo nutné rozšířit o několik základních periférií, aby bylo možné *HMI* použít jako samostatný kompaktní řídicí systém.

Požadavky pro periférie displeje byly tedy následující:

- **12 digitálních vstupů** – kapitola 5.3.3 na straně 71.
- **12 digitálních výstupů** – kapitola 5.3.5 na straně 72.
- **4 rychlé digitální výstupy** – kapitola 5.3.6 na straně 72.
- **4 analogové vstupy** – kapitola 11.2 na straně 92.
- **4 vstupy pro termočlánky** – společné s analogovými vstupy.
- **Displej** – kapitola 11.3 na straně 96.
- **MODBUS/RTU** (*RS485*) – kapitola 5.2.2 na straně 65.
- **USB** – kapitola 5.2.1 na straně 64.
- **EEPROM paměť** – kapitola 5.4.1 na straně 74.
- **FLASH paměť** – kapitola 5.4.2 na straně 74.
- **Uživatelské rozhraní** – kapitola 11.4 na straně 96.

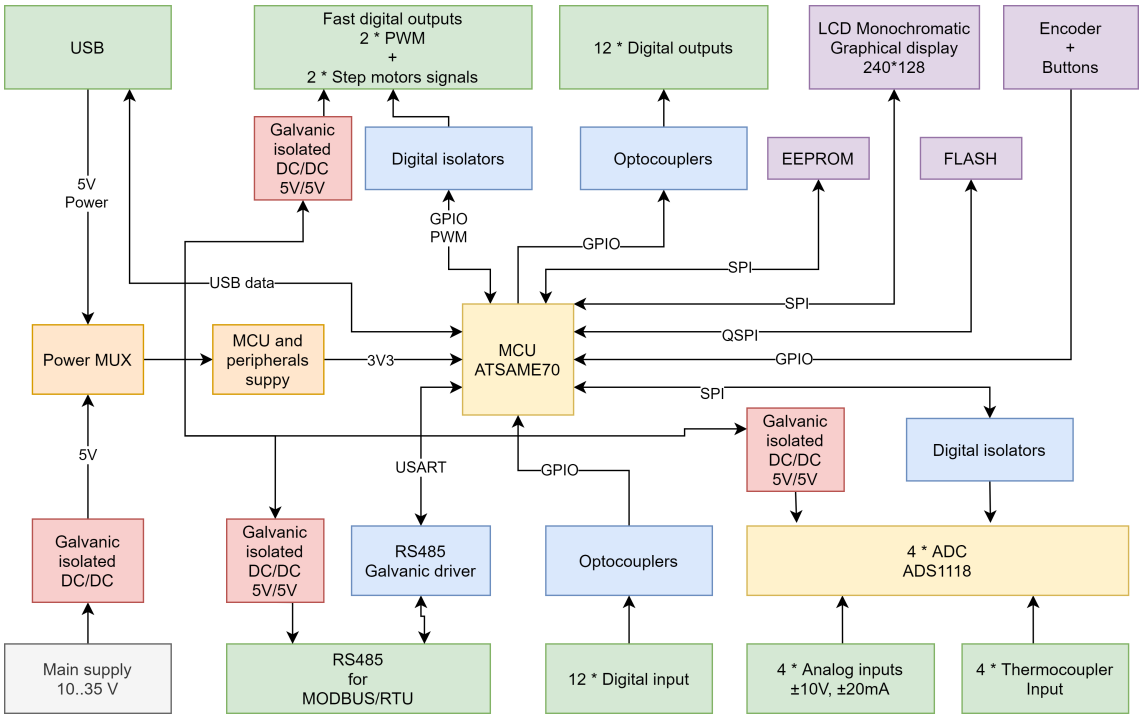
11.1 Blokové schéma zapojení

Na obrázku 11.1 je zobrazeno zjednodušené blokové schéma zapojení modulu digitálních vstupů. Celé schématické zapojení displeje (HMI) je uvedeno v příloze A.7 na straně 189.

11.2 Analogové vstupy

Důležitým požadavkem pro analogové vstupy byla možnost měření teploty pomocí termočlánků typu K. Dalším požadavkem byla možnost analogových vstupů. Jelikož přednostní byly vstupy pro termočlánky, byl zvolen obvod *MAX6675* (popsán v kapitole 11.2.1), který je určen přímo pro převod termoelektrického napětí na teplotu (převod provádí čip samotný). Z hlediska ceny vychází stejně i možnost s *ADC* (popsáno v kapitole 11.2.2), který měří termoelektrické napětí a teplotu samostatně a následně je nutné výslednou teplotu vypočítat v *MCU*. Vždy je možné vybrat pouze jednu variantu analogového vstupu pro jeden kanál (*MAX6675* nebo zapojení s *ADC*).

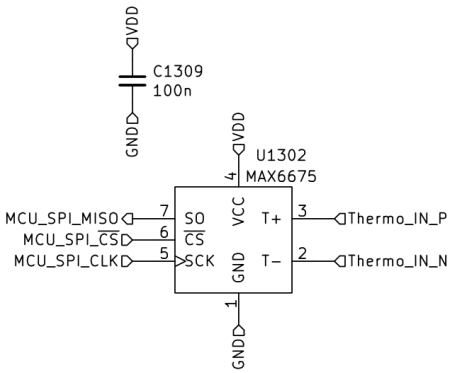
Galvanické oddělení analogových vstupů je popsáno v kapitole 11.2.3.



Obr. 11.1: Blokové schéma zapojení displeje.

11.2.1 Analogové vstupy pomocí MAX6675

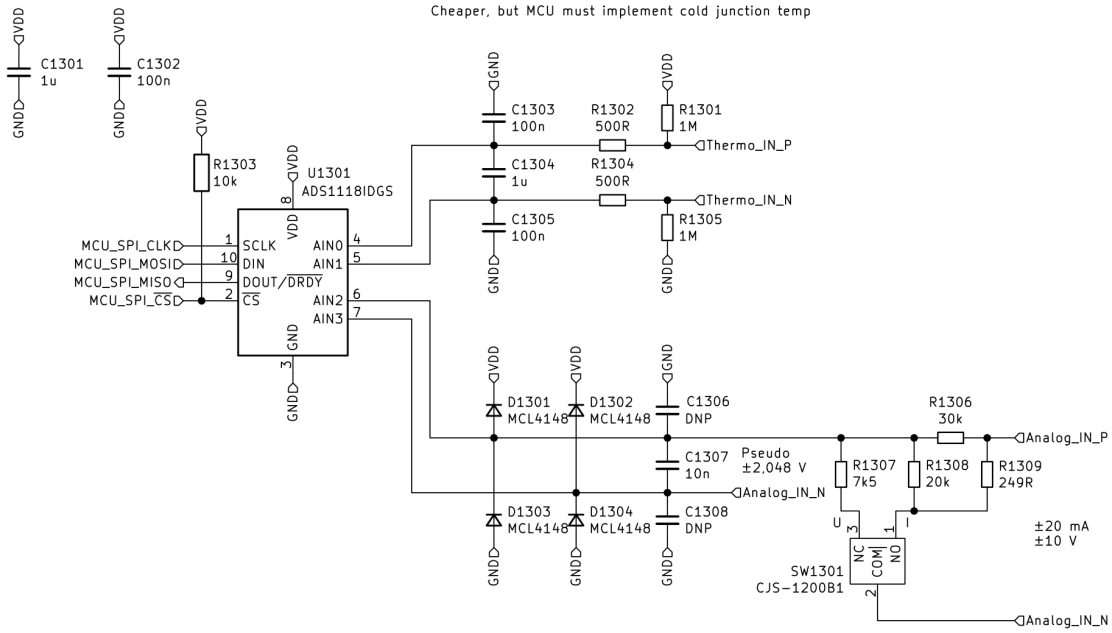
Zapojení obvodu *MAX6675* je na obrázku 11.2. Zapojení tohoto obvodu je velice jednoduché a nepotřebuje pro měření žádné další součástky. Komunikace je v tomto případě zjednodušena, jelikož neumožňuje nastavovat žádná data a proto zde není signál *MOSI*.



Obr. 11.2: Zapojení analogových vstupů s MAX6675.

11.2.2 Analogové vstupy pomocí ADS1118

Zapojení analogových vstupů s *ADC* (*ADS1118*) je zobrazeno na obrázku 11.3. *ADS1118* je *16-bit* $\Sigma\Delta$ *ADC* s *PGA*, snímačem teploty, napěťovou referencí a multiplexorem pro vhodnou volbu vstupního kanálu. Převodník je diferenciální a neměří tedy napětí jen proti zemi, ale rozdílové napětí mezi dvěma vstupními kanály. Pro zjednodušení návrhu je každý čip ve stejném zapojení a obsahuje vstup pro termočlánek, tak pro standardní průmyslový analogový vstup s rozsahem $\pm 10\,V$ a $\pm 20\,mA$.



Obr. 11.3: Zapojení analogových vstupů s ADS1118.

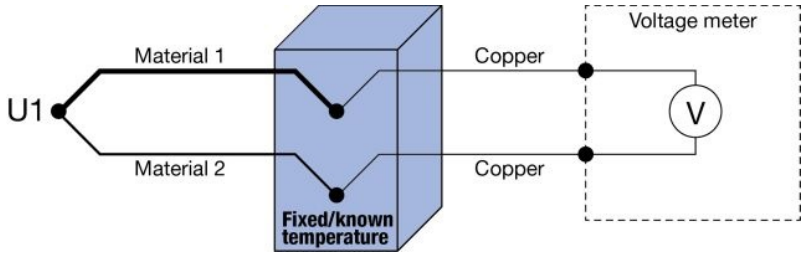
Přepočet termoelektrického napětí na teplotu

Pro převod termoelektrického napětí se používá metoda kompenzace studeného spoje (*cold junction compensation*). Na obrázku 11.4 je znázorněno zapojení měření termoelektrického napětí. Jelikož pro vznik termoelektrického napětí je potřeba dvou měřících bodů s rozdílnou teplotou, tak jeden konec termočlánku je umístěn na měřeném bodu a druhý měřicí bod je umístěn na místě se známou teplotou. Pro jednoduchost se v historii využívala káď s ledovou tříští, která měla teplotu $0\,°C$ a výsledné napětí bylo přímým výsledkem teploty v měřeném místě. Toto řešení však není použitelné v malém kompaktním řešení a řeší se měřením teploty studeného spoje. Tato metoda však vyžaduje výpočetní výkon a měření teploty studeného spoje. Softwarový blokový diagram je znázorněn na obrázku 11.5, ze kterého je

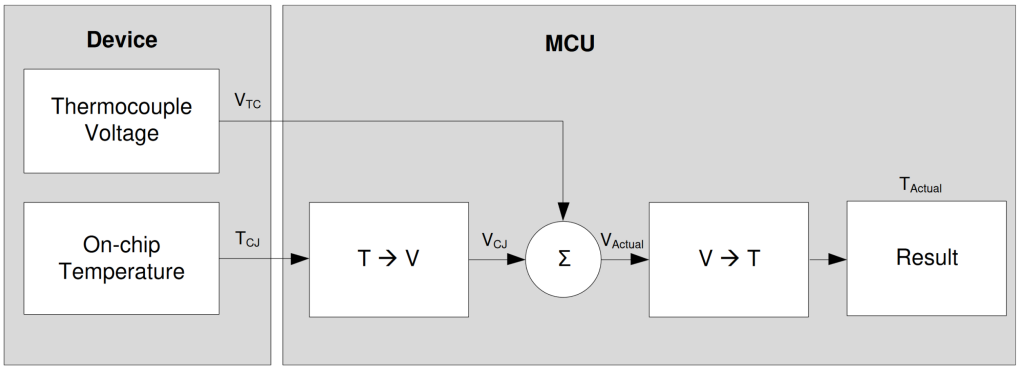
možné určit, že výsledná teplota není pouze rozdílem teplot, ale musí se provádět přepočet přes termoelektrická napětí.

Postup výpočtu teploty s kompenzací studeného spoje:

1. Změřená teplota studeného spoje se musí převést na termoelektrické napětí odpovídající danému termočlánkovému typu.
2. Termoelektrická napětí se sečtou.
3. Výsledné termoelektrické napětí se převede na teplotu dle odpovídající převodní charakteristiky zvoleného termočlánku.



Obr. 11.4: Obecné zapojení termočlánku[21].

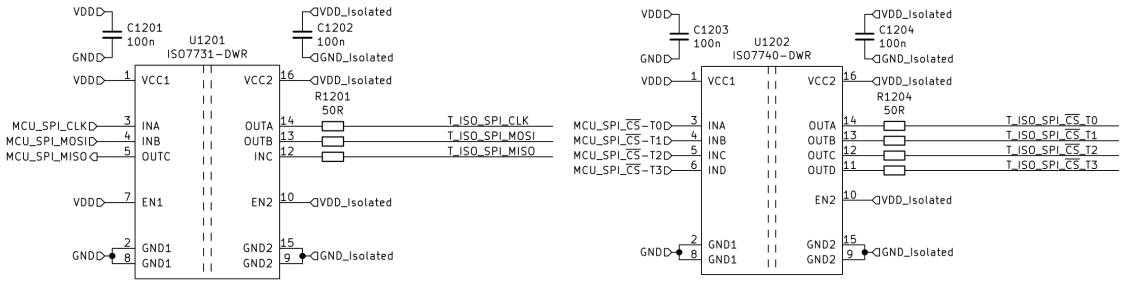


Obr. 11.5: Diagram výpočtu kompenzace studeného spoje termočlánku[35].

11.2.3 Galvanické oddělení analogových vstupů

Stejnosměrné analogové signály je problém galvanicky oddělit (pro střídavé signály je možné v omezené míře použít transformátory). Nejčastějším řešením galvanického oddělení analogových signálů je pomocí digitální komunikace, kde analogový signál je vzorkován pomocí *ADC* s digitální komunikací, digitální komunikaci je snadné galvanicky oddělit a na druhé straně je *DAC*, který převede digitální hodnotu na analogovou.

Zvolené *ADC* komunikují s *MCU* pomocí sběrnice *SPI*, která využívá signály *MOSI*, *MISO* a *CLK* pro vlastní komunikaci a následně signál *SS* (někdy označován jako CS), který slouží pro výběr slave čipu. Sběrnice je většinou definována (upravována) výrobcem slave čipu, ale základ sběrnice zůstává stejný. Galvanické oddělení komunikace je zobrazeno na obrázku 11.6. Pro větší počet signálů komunikace byly komunikační signály rozděleny na 2 samostatné obvody. Obvod *ISO7731* (obrázek 11.6a) byl zvolen pro oddělení signálů *MOSI*, *MISO* a *CLK*, které jsou pro všechny obvody stejné. Obvod *ISO7740* (obrázek 11.6b) byl zvolen pro oddělení signálů *SS*, kde jeden signál *SS* je pouze pro jeden čip a celkem jsou tedy 4.



(a) Signály komunikace SPI.

(b) Signály SS SPI.

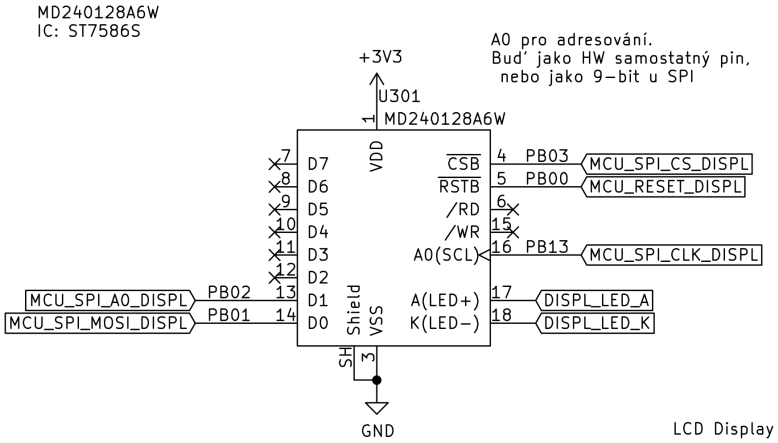
Obr. 11.6: Zapojení galvanického oddělení analogových vstupů.

11.3 Displej

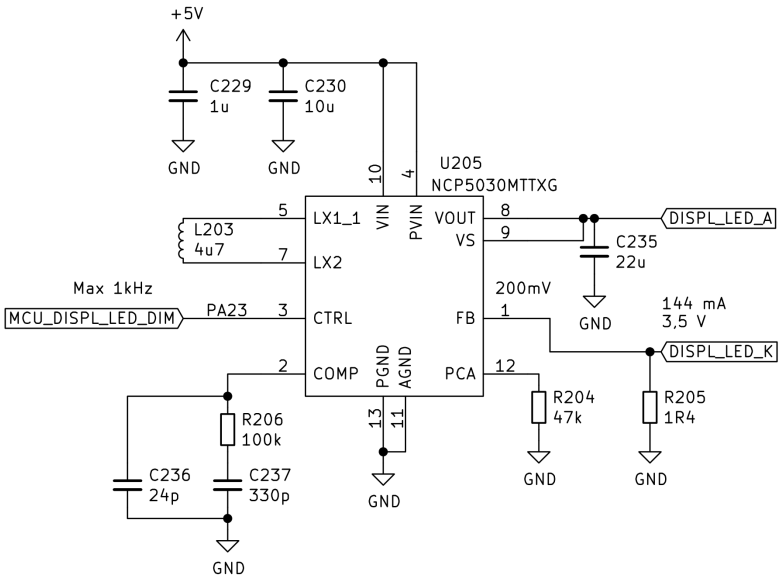
Byl vybrán monochromatický grafický displej *MD240128A6W*, který je dostačující pro danou aplikaci. Displej má rozlišení $240 \cdot 128$ pixelů s komunikačním rozhraním *SPI*. Připojení displeje je znázorněno na obrázku 11.7. Pro podsvícení displeje byl zvolen obvod *NCP5030MTTXG* (obrázek 11.8), který se řídí střídou z *PWM* signálu o maximální frekvenci $1\,kHz$.

11.4 Uživatelské periférie

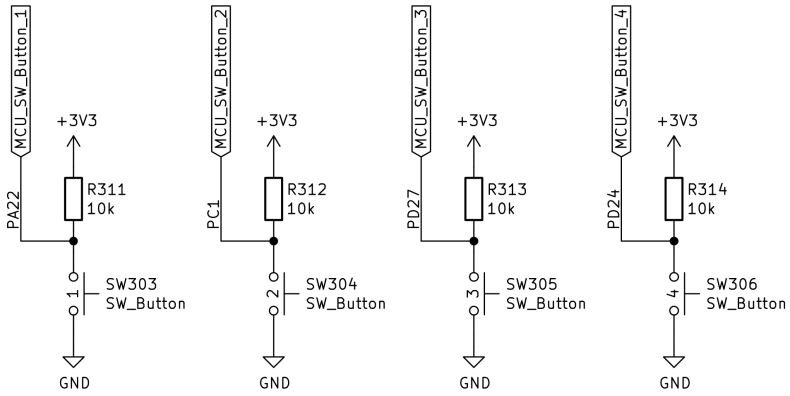
Aby uživatel mohl interagovat s displejem, byla pro uživatele přidána 4 tlačítka (obrázek 11.9) a enkodér s tlačítkem (obrázek 11.10).



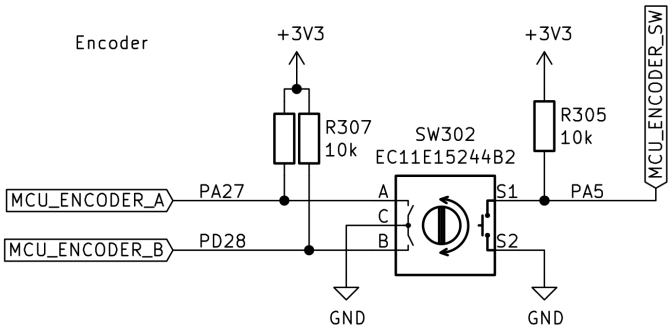
Obr. 11.7: Zapojení konektoru displeje.



Obr. 11.8: Zapojení driveru pro podsvícení displeje.



Obr. 11.9: Zapojení uživatelských tlačítek.



Obr. 11.10: Zapojení uživatelského enkodéru.

12 Komunikační protokol pro komunikaci mezi moduly

Průmyslové komunikační sběrnice jsou popsány v kapitole 2.1 a komunikační protokoly jsou popsány v kapitole 2.2. Popsané komunikační protokoly jsou ovšem optimalizovány pro komunikaci mezi řídicím systémem a externími zařízeními, která bývají od různých výrobců a spojují je tyto komunikační protokoly, mají svoji vlastní specifikaci, kterou musejí výrobci dodržovat. Toto nese omezení v podobě nedostatků příkazů, které vyžaduje komunikace mezi jednotlivými moduly modulárního řídicího systému. Jelikož se jedná o interní komunikaci, je možné tyto protokoly upravit pro požadavky modulárního řídicího systému nebo navrhnout vlastní.

12.1 Výběr interní komunikace

Jako komunikační sběrnice byla zvolena sběrnice `RS485`, která je velmi jednoduchá oproti ostatním popsaným v předchozích kapitolách. Sběrnice `RS485` je typu multimaster, tedy každé zařízení je typu master na sběrnici. Samotný standart sběrnice ale nedefinuje její přístup na sběrnici a je nechána na vyšších protokolech. Pro řídicí systém bude řešen přístup na sběrnici pomocí *Master-Slave*, kde bude jediný modul *master* a ostatní budou *slave*. Master zařízení tedy bude odesílat požadavky a slave zařízení budou odpovídat na tyto požadavky.

Další výhodou sběrnice `RS485` je její rozšířenost a také cena jednotlivých driverů potřebných pro její realizaci. Pro realizaci sběrnice `RS485` stačí *MCU* vybavený komunikací `UART`, driver sběrnice (nejčastěji čip v pouzdře `SOIC-8`) a pár pasivních součástek.

Jako komunikační protokol byl zvolen vlastní návrh komunikačního protokolu, který bude odpovídat přesným požadavkům pro modulární systém.

12.2 Návrh vlastního komunikačního protokolu

V dnešní době existuje spousta komunikačních protokolů, které jsou funkční a spolehlivé, proto nebude vymýšlen úplně nový komunikační protokol, ale bude postaven na některých principech velmi rozšířeného *MODBUS/RTU*. Vlastní komunikační protokol byl nazván **MCSBUS**.

12.2.1 HW požadavky komunikace

Nastavení asynchronní komunikace je následující:

- **Přenosová rychlost**: 1 $Mbit/s$.
- **Formát přenášených dat**:
  - `1 start bit`
  - `8 datových bitů`
  - `1 bit parity` – Sudá parita
  - `1 stop bit`
- **Konec přenosu**: mezera větší jak 3,5 bytu.

Sběrnice byla dále rozšířena o další řídicí signály, jako jsou **Next device programming** a **Return from slave**, aby mohla proběhnout inicializace komunikace (modulů). Sběrnice má komunikační rychlost stanovenou na 1 $Mbit/s$, která je omezená použitými procesory v modulech. Pokud by se vyměnili za jiné, které jsou taktovány na vyšší frekvenci, bylo by možné dosáhnout komunikační rychlosti až 10 $Mbit/s$.

Řídicí signál – Next device programming

Signál *Next device programming* je výstupem směrem od *master* ke *slave* modulu. Slouží pro určování pořadí přidělování adres *slave* modulů. Na straně *slave* modulu je signál pojmenovaný jako *Device programming enable*.

Řídicí signál – Return from slave

Pro rozpoznání posledního modulu slouží signál *Return from slave*, který se přenáší směrem od *slave* modulu do *master* modulu. Při vyjmutí modulu za chodu je pak možné zjistit absenci modulu místo jeho nefunkčnosti. Na straně *master* modulu je nutné zapojení s *pull–up* rezistorem a na straně *slave* modulu je připojen signál na zem, čímž by měla být zajištěna možnost zjištění přítomnosti porouchaného modulu.

12.2.2 Datový rámec

Na obrázku 12.1 je zobrazen datový rámec MCSBUS protokolu.

Začátek	Adresa	Řízení	ID	Délka	Objekt	CRC	Konec
$> 3.5\,B$	1 B	1 B	1-2 B	1 B	X B	2 B	$> 3.5\,B$

Obr. 12.1: Rámec protokolu MCSBUS.

Adresa

Adresa určuje adresu modulu ve výsledném řídicím systému. Rozdělení adres je následující:

- **Adresa 0** je vyhrazena pro *broadcast* komunikaci, tedy takovou komunikaci, kdy jedno zařízení vysílá pro všechna zařízení, ale žádné neodpovídá. V MCSBUS protokolu je *broadcast* komunikace využita pro přidělování adres *slave* zařízením. Dále může být využito pro synchronizaci všech zařízení nebo přenos společné informace (například nastavení času).
- **Master** modul nemá přidělenou adresu, jelikož při komunikaci není požadována - posílá požadavky slave modulům a tyto moduly odpovídají zprávou se svojí adresou.
- **Adresy 1..255** jsou vyhrazeny pro **slave** moduly a tyto adresy jim přiřazuje **master** modul. Celkově může být tedy použito až 255 modulů s ohledem pouze na komunikační protokol.

Adresy *slave* modulům přiřazuje tedy *master* modul, čímž je dosaženo to, že V/V moduly nenesou žádný specializovaný kód pro danou aplikaci a je možné je vyměnit za stejné bez nutnosti jejich konfigurace. Postup přiřazování adres je popsán v kapitole 12.2.7.

Řízení

Byte řízení se skládá z několika řídicích bitů, které udávají charakter přenášené zprávy. Jednotlivé bity mají následující význam:

- **bit 7**: $R/\overline{W}$ – při čtení je 1 a pro zápis 0.
- **bit 6**: 1 pokud se jedná o zápis s požadavkem na zpětnou odpověď. Použije se, pokud jsou objekty, které mají část pro zápis a část pouze pro čtení (např. modul analogových IO může využívat jednu výměnu dat pro zápis a přečtení všech kanálů, jelikož jsou moduly konfigurovatelné jako vstupní, nebo výstupní).
- **bit 5**: 1 pokud se jedná o přidělování adresy (bit 7 musí být nastaven na 0, jinak nebude adresa přidělena).
- **bit 4**: rezervován – vždy se čte jako 0.
- **bit 3**: značí chybu – vrací *slave* modul, jelikož *master* odesílá pouze požadavky a případné přijetí chybového kódu odešle požadavek znovu.
- **bit 2**: potvrzení zápisu – 1 při odpovědi potvrzení zápisu, jinak je ignorován.
- **bit 1**: typ přenosu – 0 pro požadavek (*Request*) a 1 pro odpověď (*Response*).
- **bit 0**: používá se pro rozšířené ID. 0 pro 8-bit ID a 1 pro 16-bit ID.

ID

Identifikátor (**ID**) označuje adresuje objektu (jeho ID) v rámci zařízení. ID 0 až 15 je vyhrazeno pro systémové funkce a ID 16 až 255 (16 až 65535 v případě rozšířeného identifikátoru) je vyhrazeno pro specifické funkce daného modulu.

Délka

Délka obsahuje délku objektu v bytech, jelikož objekty mohou být různě dlouhé. Jedná se také o ověřovací prvek, protože pokud neodpovídá předpokládané délce objektu, zpráva je zamítnuta.

Objekt

Oproti `MODBUS`, který využívá 16-bit registry, tak `MCSBUS` pracuje s objekty, které jsou specifické pro každé zařízení. Objekty se indexují pomocí ID a může jich být až 65536, kde 16 jich je definováno komunikačním protokolem a zbytek pro uživatelskou alokaci (daný pro jednotlivé moduly).

Objekty jsou univerzální a jsou omezeny pouze maximální velikostí, která byla stanovena na $200\,B$. Každý objekt má nastavitelná práva přístupu, která mohou být:

- **RW**: Přístupné pro čtení a zápis.
- **RO**: Přístupné pouze pro čtení. Při pokusu o zápis je vrácena odpověď o neprovedení akce (popsáno v kapitole 12.2.4).
- **WO**: Přístupné pouze pro zápis. Při pokusu o čtení je vrácena odpověď o neprovedení akce (popsáno v kapitole 12.2.4).

Příklad systémových objektů je uveden v tabulce 12.1–12.2 na straně 102.

CRC

CRC (*Cyclic Redundancy Check* – Cyklický redundantní součet) slouží pro kontrolu správnosti přijetí zprávy (detekce chyb). Používá se 16-bitový CRC s generujícím polynomem $x^{16} + x^{15} + x^2 + 1$. Vypočítá se z celého datového rámce kromě sebe samotného.

12.2.3 Systémové objekty

Systémové objekty jsou povinné pro všechny slave moduly a zajišťují správné fungování.

Stav modulu

Jméno	Velikost	Interpretace	Popis
Status	$1B$	UINT8	Stav modulu

Tab. 12.1: Systémový objekt: Modul status($0x00$:RO)

Kde *Status* je bitově orientovaná proměnná a její jednotlivé bity mají tento význam:

- **bit 7**: Stav vstupu *Return from slave* pro zjištění, zda modul je poslední, či ne. Používá se také pro detekci chybějícího modulu.
- **bit 6**: Rezervován.
- **bit 5**: Rezervován.
- **bit 4**: Rezervován.
- **bit 3**: Rezervován.
- **bit 2**: Rezervován.
- **bit 1**: Rezervován.
- **bit 0**: Aktivita modulu *Stop* a *Run*. Slouží pro volbu přepisování hodnot. Např. modul digitálních výstupů nebude aplikovat na své výstupy přijaté hodnoty, pokud je modul ve *Stop* stavu.

Informace o modulu

Jméno	Velikost	Interpretace	Popis
ID number	$2B$	UINT16	Identifikační číslo modulu
HW ver.	$1B$	UINT8 / 10	Verze HW modulu–0.0 až 25.5
SW ver.	$1B$	UINT8 / 10	Verze SW modulu–0.0 až 25.5
Name	$16B$	STRING[16]	Název modulu

Tab. 12.2: Systémový objekt: Modul info ($0x01$:RO)

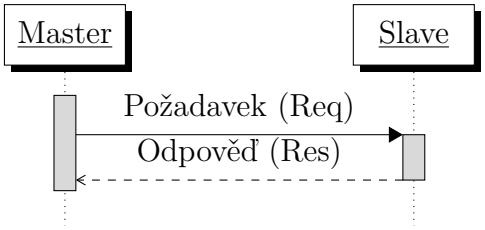
WDT - Watchdog Timer

Jméno	Velikost	Interpretace	Popis
Time set	$2B$	UINT16	Nastavený čas $[ms]$
Time act.	$2B$	UINT16	Aktuální čas $[ms]$
Hits	$16B$	STRING[16]	Počet vypršení WDT

Tab. 12.3: Systémový objekt: Modul info ($0x02$:RO)

12.2.4 Popis transakce

Každé provádění instrukce by mělo mít svoje potvrzení o úspěšnosti kromě *broadcast* zpráv. Tato výměna informací je pojmenována jako transakce a její znázornění je na obrázku 12.2. Zprávy jsou označovány jako požadavky (*Request*) a odpovědi (*Response*). Rozdíl mezi požadavkem a odpovědí určuje bit 1 v bytu řízení. Adresa při požadavku značí, od jakého modulu je požadována odpověď (adresované zařízení) a při odpovědi označuje zařízení, které odesílá odpověď.



Obr. 12.2: Transakce komunikace MCSBUS.

Transakce čtení

Pro rozlišení požadavku čtení a zápisu je použit bit 7 v bytu řízení (pro čtení musí být 1). ID značí požadovaný objekt pro čtení a délka je nulová. Pokud délka objektu bude větší, budou data ignorována.

Transakce zápisu

Pro zápis musí být bit 7 v bytu řízení 0. Objekt obsahuje data zapisovaná. Vždy se musí zapisovat celý objekt, jinak nebude brán jako platný zápis a bude vrácena chybová zpráva. Pro potvrzení *slave* zařízení odešle zprávu bez objektu a nastaveným bitem 2 v bytu řízení.

Transakce zápisu se zpětným čtením

S ohledem na rychlejší výměnu dat byla přidána speciální transakce, kde je možné v jednom cyklu zapsat data a zpětně je přečíst. Pro statické objekty tato transakce nemá význam, jelikož data zapsaná se tak rychle nezmění. Je tedy specifikem pro objekty, které mají určitou část přístupnou pouze pro čtení a zbylá část je určená pouze pro zápis. Příkladem je modul analogových vstupů a výstupů, které jsou konfigurovatelné, a aby bylo možné dodržet stálou strukturu, je nutné mít transakci tohoto typu.

Tento typ transakce je definován bitem 6 v bytu řízení. Pokud je tento bit nastaven na 1, tak požadavek vypadá stejně jako zápis a bude vrácena odpověď jako při

čtení. Pokud je požadavek typu čtení, tak by měla být vrácena odpověď definující chybu s kódem *5*.

Transakce s chybou

Pokud je přijatá zpráva (požadavek) přijata s chybou, tak je odeslána odpověď s informací o chybě. Zpráva o chybě je definovaná bitem 3 v bytu řízení. Objekt zprávy je definován jako UINT8 obsahující kód chyby. Možné detekovatelné chyby a jejich řešení jsou popsány v kapitole 12.2.6 a kódy chyb jsou obsaženy v tabulce 12.5.

12.2.5 Broadcast zprávy

Pro zrychlení komunikace, kde je zapotřebí informovat nebo zapsat do všech modulů bez nutnosti zpětné zprávy, se použije *broadcast* zpráva a omezí se tím adresace modulu po modulu. Všechny *broadcast* zprávy nejsou povinné a lze je směřovat pro všechny moduly stejné, či obdobné funkce zastoupeny ve *slave* modulech (např. vykonání specifických úloh, jako je trigger). Přehled možných *broadcast* zpráv je v tabulce 12.4.

Jméno	ID	Povinnost	Určení
Reset	0	Ano	Všechny
Přidělování adres	1	Ano	Všechny
Řízení stavu	2	Ano	Všechny
Trigger	3	Ne	Dobrovolné

Tab. 12.4: Přehled ID *broadcast* zpráv

Reset

Slouží pro obnovení výchozího stavu *slave* modulů v ohledu chování na sběrnici. *Slave* modul poté bude mít nastavenou svoji adresu jako 0 a bude přijímat pouze *broadcast* zprávy a nastaví výstup *Next device programming* do stavu log. 0. Následně uvede svoje výstupy a nastavení do předdefinovaného stavu. Např. výstupní modul bude mít na všech výstupech log. 0, vstupně–výstupní modul bude mít všechny **IO** nakonfigurované jako vstupní, pokud není definováno jinak.

Zpráva **Reset** nemá žádná data v objektu.

Přidělování adres

Tato zpráva pro přidělení adresy daného modulu. Popis přidělování adres je popsán v kapitole 12.2.7. Tato zpráva má délku objektu $1\,B$, který má formát *UINT8*

a nese informaci o přidělované adrese.

Řízení stavu

Zpráva o řízení stavů slouží ke změně aktivity modulu *Stop* a *Run*. Objekt obsahuje data o velikosti 1 B, kde pro stav *Stop* je nutné poslat data s hodnotou $0x5A$ a stavu *Run* odpovídá hodnota $0xA5$. Stav každého modulu lze zjistit ze systémového objektu 0

Trigger

Tato zpráva může mít objekt o velikosti 1 B s formátem *UINT8*, který slouží pro specifikaci činnosti, která má být provedena. Objekt je dobrovolný a jeho funkcionalita není zatím popsána. Moduly, které nevyžadují specifikaci triggeru, provedou svoji funkci normálním způsobem.

12.2.6 Chyby

Aby byl komunikační protokol spolehlivý, musí počítat také s možností výskytu chyb, jejich detekcí a řešení těchto stavů. Jako detekce chyb je použita parita sériové komunikace, která je schopná pokrýt lichý počet chyb. Pro detekci chyb v celém rámci je použitý cyklický redundantní součet. Možnosti detekce chyb je vysvětlena v kapitole 2.3.

Chyba kontrolního součtu

Kontrolní součet je počítán z celého datového rámce a je tedy možné zjistit také chybu adresy. Toto zjištění by mohlo mít dva různé způsoby k přístupu k této chybě:

1. **Odeslat chybovou zprávu** o špatném CRC. V případě chyby v adresovém poli by mohlo vést k tomu, že jedno zařízení dostane adresu správně a druhé špatně, pak bude kolize na sběrnici. Jelikož je použita sběrnice `RS485`, která neřeší přístup k těmto kolizím a jediným řešením je pak použití driverů pro sběrnici `RS485`, které snesou kolize a nedojde k jejich zničení. Následně je také možné tyto kolize kontrolovat pomocí MCU (některá MCU mají zabudovanou tuto funkci), pokud i při vysílání zůstane přijímač aktivní a vytvoří se tím zpětná vazba (*loopback*).
2. **Neprovádět nic** a čekat na další požadavek. Následně je na *master* modulu, aby toto vyhodnotil a aktivoval se *timeout* (vypršení časovače, který sleduje maximální dobu, po kterou neodpovídá tázané zařízení). Tento přístup je šetrnější ke kolizím na sběrnici, ale zpomaluje sběrnici, protože se musí čekat na *timeout*.

Tyto 2 metody jsou pro `MCSBUS` přijatelné a záleží na individuálním modulu, jak je implementována tato problematika. Doporučeným řešením je implementace v závislosti na přijaté adrese modulu. Pokud je přijatá adresa 0, tedy zpráva vysílaná jako *broadcast* zpráva, nebo s adresou jinou, tak zařízení nebude reagovat. Pokud je adresa zařízení shodná s přijatou adresou, pak dojde k odeslání chybové zprávy. Tento způsob reagování může snížit počet kolizí na sběrnici a také ji zrychlit v případě chyb na sběrnici, jelikož zařízení nemusí čekat na *timeout* a reagují téměř okamžitě. Dále lze statisticky vyjádřit možnost chyby v adresním bytu, jelikož adresa má velikost $1\,B$ a minimální délka zprávy je $6\,B$, pak je maximální pravděpodobnost $16,7\,\%$.

Popsané metody se implementují pouze do *slave* modulů, jelikož *master* modul řeší přijaté chyby odesláním nového požadavku. Počet těchto opakování by neměl být nekonečný a *master* modul by měl obsahovat čítač těchto chyb a pokud by byl překročen, tak *master* jednotka by měla ukončit komunikaci s tímto modulem na nějakou dobu a pokud by ani po třech pokusech nebyla komunikace úspěšná, měla by *master* jednotka přejít do poruchového stavu.

Ostatní chyby

Jako ostatní chyby můžeme brát v potaz:

- Neodpovídající délka objektu.
- Nesprávné ID – mimo rozsah modulu, nebo nepodporování rozšířeného ID.
- Pokus o zápis dat, která jsou pouze pro čtení.
- Nesprávná hodnota dat – data jsou mimo povolený rozsah.

Kódy chyb

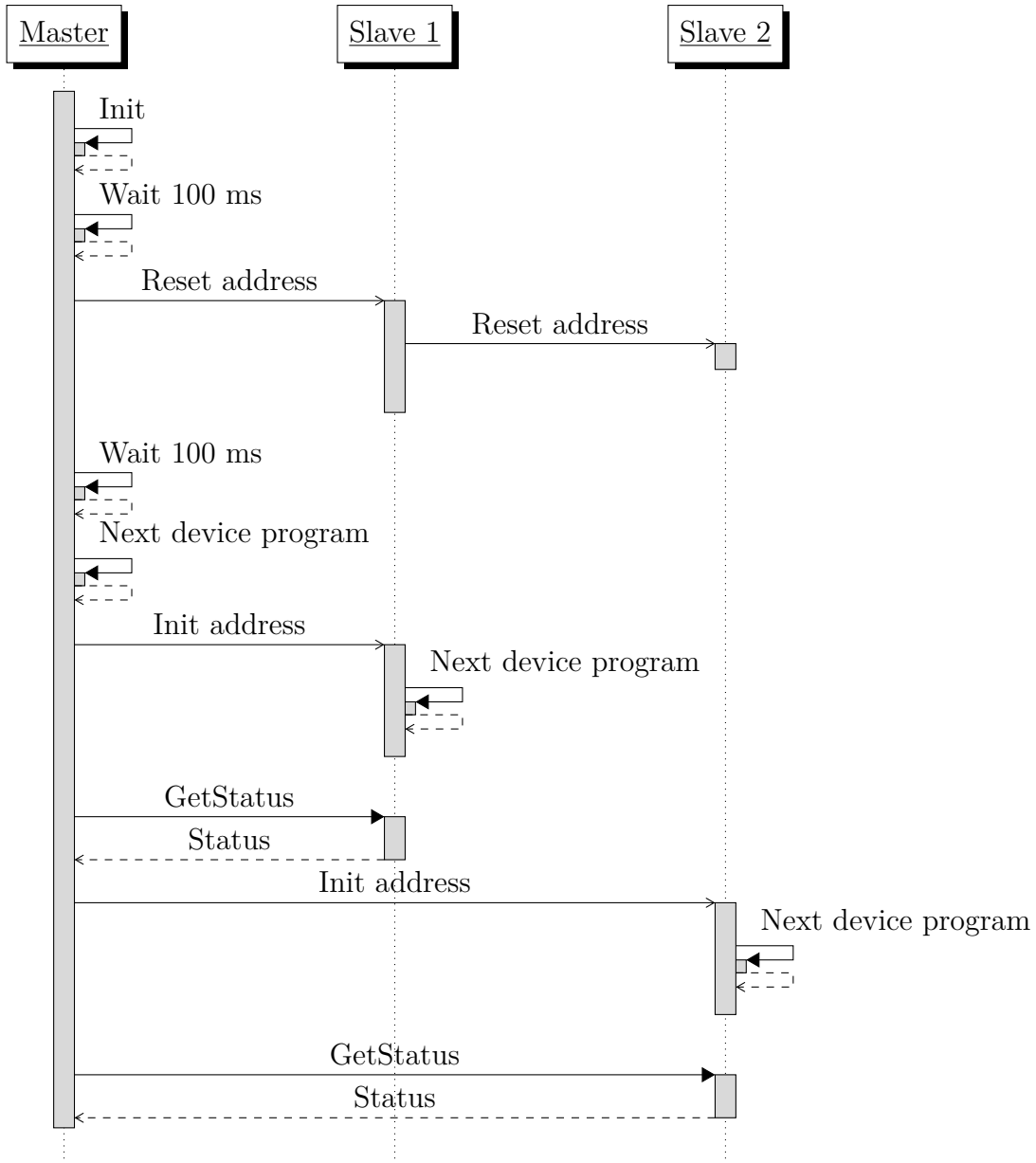
Pro lepší rozpoznání chyb vzniklé na sběrnici a v zařízeních má každá chyba vlastní kód, díky čemuž je lépe trasovatelná příčina. Seznam kódů chyb je v tabulce 12.5.

Jméno chyby	Kód
Přesně nespecifikovaná chyba (obecná)	0
Nesprávné ID	1
Špatná délka objektu	2
Objekt není přístupný pro čtení	3
Objekt není přístupný pro zápis	4
Špatný požadavek na zpětné čtení	5
Chyba CRC	255

Tab. 12.5: Přehled kódů chyb.

12.2.7 Inicializace komunikace modulů

Aby byla dosažena kýžená modularita, kde pouze procesorová jednotka nese informace o celém systému, tak jednotlivé moduly musejí být konfigurovány procesorovou jednotkou. Tato konfigurace se provádí po startu systému případně na vyžádání proběhne konfigurace jednoho modulu, který ztratil svoji paměť (výměna modulu za chodu).



Obr. 12.3: Sekvenční graf znázorňující přidělování adres slave modulům.

12.2.8 Detekce chybějícího modulu

Jak již bylo popsáno v kapitole 12.2.1, je možné detekovat absenci modulu a případně také jeho nefunkčnost. U modulárního řídicího systému je možné vyměnit modul za chodu (tehdy, kdy je celý řídicí systém pod napětím). Detekce chybějícího modulu a jeho opětovné nakonfigurování je zajištěno *master* modulem, který provádí správu modulů, a pokud je zjištěna přítomnost nenakonfigurovaného modulu, spustí sekvenci přiřazení adresy modulu. Je nutné však zajistit, aby moduly reagovali na změnu přítomnosti modulů pomocí signálů *Next device programming* a *Device*

programming enable. Přítomnost daného modulu je detekována pomocí vstupu *Return from slave*(udává informaci o přítomnosti následujícího modulu na sběrnici), jehož stav je obsažen v systémovém objektu 0, jež je popsán v kapitole 12.2.3 na straně 103.

Je nutné si uvědomit, že konfigurace modulů je prováděna postupně a je nutné uspořádat moduly řídicího systému tak, aby nevznikala prázdná místa na sběrnici, jinak nedojde ke konfiguraci. Pokud bude modul odebrán po konfiguraci modulů a bude odebrán tak, že vznikne mezi moduly mezera, je stále umožněna komunikace *master* modulu s ostatními *slave* moduly. Pokud bude přidán na konec sběrnice další modul, nebude nakonfigurován dříve, než bude doplněn chybějící modul pro vytvoření nepřerušené sběrnice.

Cyklus správy modulů byl stanoven na $1\,s$, aby zbytečně nezatěžoval komunikační sběrnici. Pokud bychom počítali s touto funkcí pouze pro výměnu modulu uživatelem, bylo by možné také prodloužit tuto délku cyklu, jelikož výměna modulu zabere mnohonásobně více času. S ohledem na možnost vniku nepříznivé situace, kdy může dojít k restartování modulu (např. v rámci *WDT* nebo podpětí), je nutné zajistit funkčnost modulu co nejdříve.

12.2.9 Řízení stavu modulů

MCSBUS definuje 3 stavy modulu a to:

- **Non-initiate**, tedy kdy modul nemá přidělenou adresu.
- **Stop**, kdy lze provádět konfiguraci, ale funkce modulu nemusí být funkční.
- **Run**, kdy jsou funkce modulu aktivní a nemusí být modul konfigurovatelný (záleží na specifikaci modulu - lze měnit konfiguraci za chodu).

Stav *Non-initiate* má zařízení po zapnutí (HW restartu, nebo restartu po komunikace pomocí broadcast zprávy) a nelze jej nijak jinak dosáhnout. Ve stavu *Non-initiate* nemá modul přidělenou adresu a nelze tak provádět standardní operace skrze komunikační protokol MCSBUS. Adresa modulu je přidělena pomocí broadcast zprávy (popsáno v kapitole 12.2.5 a 12.2.7) a po přidělení adresy modul přejde automaticky do stavu *Stop*. Stavy *Stop* a *Run* jsou plně ovladatelné pomocí broadcast zpráv (popsáno v kapitole 12.2.5 na straně 105).

Jelikož moduly obsahují také výstupní kanály, tak je implementován *SW watchdog timer*, který sleduje aktivní komunikaci na sběrnici a je tak možné zajistit bezpečný stav těchto výstupů. Pokud dojde k vypršení *WDT*, modul přejde do *Stop* stavu, který opět přejde do *Run* stavu po broadcast zprávě řídicí stav modulu (pokud modul nespecifikuje jinak). *WDT* se nastavuje pomocí systémového objektu č. 2 (viz. kapitola 12.2.3), který obsahuje nastavený čas, aktuální čas a čítač aktivity. *WDT* je aktivní, pokud nastavený čas je nenulový a jeho stav je možné zkontrolovat

pomocí aktuálního času, kdy musí být také nenulový, pokud časovač běží. Každý modul může implementovat restartování *WDT* dle sebe - např. zda se bude restartovat při jakékoli zprávě na sběrnici, nebo při interakci se specifickým objektem, který je důležitý pro daný modul.

12.3 Popis datových objektů modulů

Jednotlivé uživatelské objekty využívané pro komunikaci s moduly jsou zmíněny v příloze B. Tato kapitola se zabývá popisem obsahu těchto dat.

12.3.1 Modul digitálních vstupů

Uživatelské objekty pro modul digitálních vstupů jsou znázorněny v kapitole B.1. Uživatelský objekt č. 0 (příloha B.1.1) obsahuje data digitálních vstupů reprezentující data z posledního cyklu čtení digitálních vstupů. Perioda cyklu čtení je konfigurovatelná v uživatelském objektu č. 16 (příloha B.1.6).

Jelikož perioda cyklu čtení digitálních vstupů může být mnohem rychlejší, než perioda čtení dat z modulu, jsou přidány objekty zachycující hrany na digitálních vstupech, pro zachycení rychlejších dějů, než je komunikace s jednotlivými moduly. Byly zvoleny 3 objekty (přílohy B.1.2-B.1.4) popisující výskyt náběžné, seběžné a nebo obou hran. Tyto objekty drží informaci o těchto hranách, dokud není přečtena jejich hodnota. Mají také přístup pro zápis a lze jej tak vynulovat i zápisem.

Pro optimalizaci cyklické komunikace je vytvořen uživatelský objekt č. 4 (příloha B.1.5), který obsahuje jak data posledního čtení digitálních vstupů, tak data o náběžné a seběžné hraně mezi cyklickým čtením. Cyklická data pak tedy nevyžadují vícenásobnou inicializaci přenosu dat a je tak možné rychleji komunikovat s modulem.

12.3.2 Modul digitálních výstupů

Uživatelské objekty pro modul digitálních výstupů jsou znázorněny v kapitole B.2. Uživatelský objekt č. 0 (příloha B.2.1) obsahuje data aktuálních binárních výstupů. Některé technologie vyžadují z hlediska bezpečnosti definování klidového stavu, nebo alespoň stavu, kdy je ovládané zařízení v nečinnosti. Uživatelský objekt č. 17 (příloha B.2.5) tedy obsahuje možnost nastavit klidový stav výstupu, pokud není modul řízen (stop stav). Podrobnější vysvětlení možnosti stop stavu je vysvětleno v kapitole 12.2.9 na straně 110.

Pro rozšíření funkcionality modulu digitálních výstupů byla přidána možnost generování *PWM*. PWM se generuje SW a má tedy svá omezení v maximální frekvenci

(dalším omezením je použitý HW, který limituje maximální frekvenci do jednotek *kHz*). Tato funkcionalita je například nejpřínosnější pro systémy s pomalejší dynamikou, jako je například topení ovládané *SSR* se spínáním v nule. Nastavení generování PWM se provádí pomocí uživatelského objektu č. 16 (příloha B.2.4), který definuje periodu v násobcích $100\,\mu s$. Pokud je hodnota periody nastavena jako 0, tak se digitální výstup použije jako klasický binární a ovládá se pomocí uživatelského objektu č. 0. Aktuální hodnota střídy se nastavuje pomocí uživatelského objektu č. 1 (příloha B.2.2), pomocí kterého jsme schopni nastavovat všechny PWM výstupy. Pomocí uživatelského objektu č. 2 (příloha B.2.3) lze nastavit pouze jeden specifický výstup, který je optimalizován pro přenos dat do modulu obsahujícího pouze malý počet PWM výstupů.

12.3.3 Modul analogových vstupů a výstupů

Modul analogových IO se skládá z 8 identických kanálů, kde každý kanál je možné nezávisle nakonfigurovat. Pro konfiguraci se využívá uživatelského objektu č. 16 (příloha B.4.10) pro konfiguraci všech kanálů najednou. Je možné použít také uživatelský objekt č. 17 (příloha B.4.11) pomocí kterého se nakonfiguruje pouze jeden kanál. Konfigurace obsahuje nastavení fyzického rozhraní kanálu (popsáno v kapitole 10.3), jeho rozsah a případně rozsah inženýrských hodnot, který se aplikuje přímo v modulu podle lineární aproximace.

Pro nastavení či čtení hodnoty je možné použít několik přístupů k zadávání, pokud není sjednoceno v procesorové jednotce. První možností je použití uživatelských objektů č. 0 až 2 (příloha B.4.1 až B.4.3) s binární hodnotou převodníku. Uživatelské objekty č. 3 až 12 (příloha B.4.4 až B.4.6) slouží k přístupu k fyzikální veličině kanálu (např. $10\,mA$). Uživatelské objekty č. 13 až 15 (příloha B.4.7 až B.4.9) se používají pro přístup k inženýrským hodnotám. Zadávání hodnot do kanálu je na sobě závislé pro výstupy. Pokud se tedy provede zápis inženýrské hodnoty, tak se projeví při zpětném čtení ve fyzické hodnotě a hodnotě převodníku. Při zápisu se tedy na převodníku projeví hodnota, která byla zapsána jako poslední. Pro vstupní kanál je možné číst jakoukoli reprezentaci dat.

Pro sjednocení interpretace dat jsou binární hodnoty převodníku převedeny na maximální rozsah datového objektu. Datový objekt binárního přenosu dat má velikost $16\,bit$ů a modul analogových IO obsahuje $16\,bit$ *ADC* a $13\,bit$ *DAC*, což pro dosažení stejného výsledku má jiné hodnoty. V tomto případě se na hodnoty z *DAC* aplikuje binární posun o 3 bity vlevo, čímž dosáhneme stejné interpretace výsledků (pro dekódování se použije inverzní operace, tedy posun o 3 bity vpravo).

Modul je také výstupního charakteru a proto také obsahuje pro konfiguraci uživatelský objekt č. 18 (příloha B.4.12), který definuje výstupní hodnotu v případech

definovaných v kapitole 12.2.9 na straně 110.

12.3.4 Modul analyzátoru sítě

Uživatelský objekt č. 0 (příloha B.3.1) obsahuje naměřená data ze všech fází a bude tedy nejlépe vyhovovat pro cyklické čtení. Uživatelské objekty č. 1 až 3 (příloha B.3.2) zahrnují data pouze z jedné fáze a je vhodné, pokud se používá pouze jedna fáze (modul může mít variantu pouze s jednou měřenou fází). Pro napětí a proud je vypočítáno FFT spektrum, které je dostupné přes uživatelské objekty č.4 až 9 (příloha B.3.3). Uživatelský objekt č. 16 (příloha B.3.4) slouží pro konfiguraci a obsahuje možnost nastavení napěťového a proudového transformátoru a výchozí frekvenci sítě ($50\,Hz$ nebo $60\,Hz$).

13 Realizace a ověření funkčnosti

Z popsaných modulů byla realizována procesorová jednotka (kapitola 6.1), digitální vstupy (kapitola 8), digitální výstupy (kapitola 9), analyzátor sítě (kapitola 7) a analogové vstupy a výstupy (kapitola 10). Byl také realizován modul displeje (kapitola 11), který také slouží pro řízení malého vstřikovacího lisu. Modul displeje však není připojen pomocí vnitřní sběrnice MCSBUS, ale komunikuje pomocí externí komunikační sběrnice (*RS485*), čímž je zajištěna kompatibilita s jinými zařízeními a nezávislost na modulárním řídicím systému.

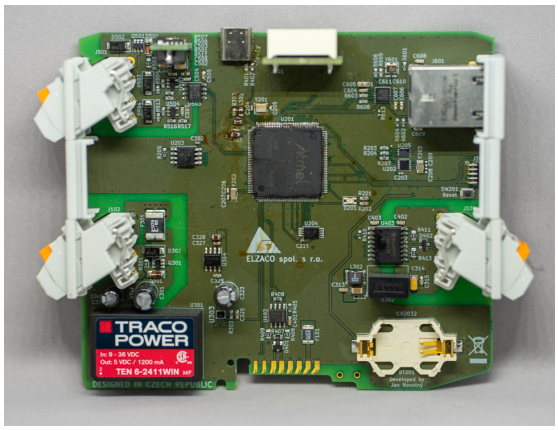
13.1 Realizace

Realizované moduly jsou zobrazeny na obrázku 13.1 a na obrázku 13.2 je zobrazen modul displeje. Obrázek 13.3 zobrazuje systém modulárního systému v krabičkách nasazenými na DIN liště a spojené konektorem sběrnice v DIN liště.

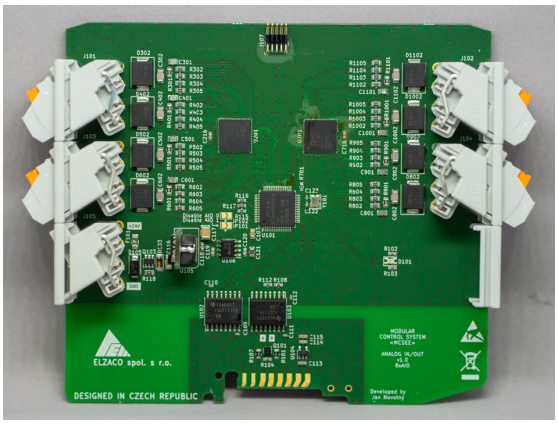
Obr. 13.3: Modulární řídicí systém umístěný v krabičkách na DIN liště.

13.2 Cenová kalkulace

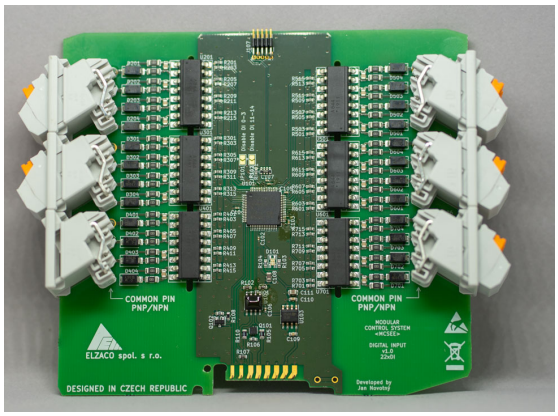
Uvedené ceny jsou vypočítaný k datu *20.4.2022* a bez daně (*netto* cena), byť návrh a nákup komponentů byl prováděn dříve a některé komponenty jsou například dvakrát dražší, než při původním návrhu.



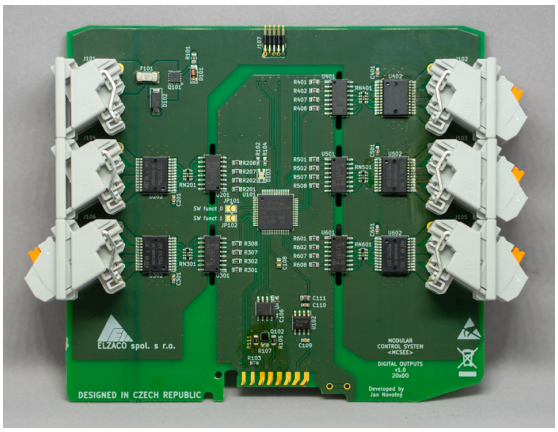
(a) Osazený modul procesorové jednotky.



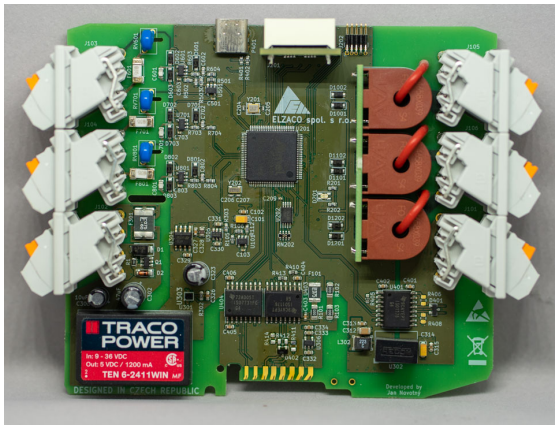
(b) Osazený modul analogových vstupů a výstupů.



(c) Osazený modul digitálních vstupů.

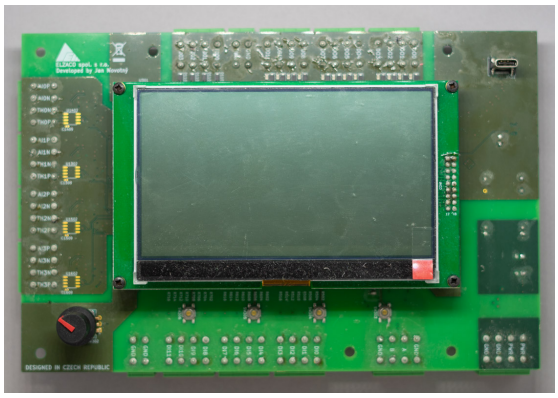


(d) Osazený modul digitálních výstupů.

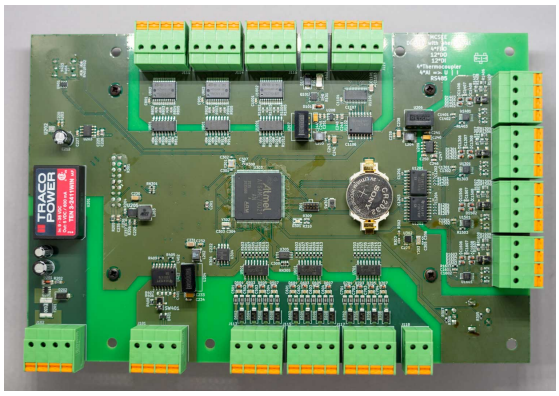


(e) Osazený modul analyzátoru sítě.

Obr. 13.1: Realizované moduly modulárního řídicího systému.



(a) Osazený displej z pohledu displeje.



(b) Osazený displej z pohledu osazených součástek.

Obr. 13.2: Realizovaný modul displeje.

13.2.1 Cena krabičkového systému

Modulární řídicí systém se skládá z několika modulů, které se skládají převážně z obdobných komponentů kromě elektronických součástek. Tabulka 13.1 udává přibližné cenové údaje pro systém `ICS` od výrobce *Phoenix Contact*[26] jednotlivých komponentů bez dalších poplatků. Je bráno v potaz pouze jedna velikost krabičky (šířka modulu 25 mm, výška 122, 5 mm a hloubka 108, 35 mm), ale systém `ICS` umožňuje použití různých velikostí a je také možné použít také krabičky jiných podporovaných systémů, jako je např. `ME-IO`.

Popis	Cena za kus [Kč]
Spodní díl krabičky	136 Kč
Horní neprůhledný díl krabičky	57 Kč
Horní průhledný díl krabičky	91 Kč
Konektor sběrnice do DIN lišty	362 Kč
Zásuvka konektoru 4-pólová	56 Kč
Zástrčka konektoru 4-pólová	81 Kč
Záslepka (připojovací clona) plná	16 Kč
Záslepka (připojovací clona) s otvory	27 Kč

Tab. 13.1: Přehled cen krabičkového systému.[26]

Cena krabičky na jeden modul je přibližně 1377 Kč, který se skládá ze spodního dílu, horního neprůhledného dílu, konektoru sběrnice a šesti konektorů skládajících se ze zásuvky a zástrčky.

13.2.2 Cena DPS

Na základě jedné vybrané velikosti krabičky je velikost *DPS* (*Deska Plošných Spojů*) stejná. Malý rozdíl nastane, pokud se jedná o krabičku s průhledným víkem, nebo neprůhledným, ale tento rozdíl bude pro cenovou kalkulaci zanedbán. Velikost jedné DPS činí zhruba $119 \cdot 107\,mm$. Modul displeje je samostatný modul, který má rozměry DPS $215 \cdot 141\,mm$. Pro výrobu DPS byl vybrán výrobce *Aisler*[1] na základě dlouhodobé zkušenosti. DPS však nesplňují všechny náležitosti, které by měla mít výsledná DPS a to především požadavek na tzv. "Gold fingers", které zvyšují životnost konektoru na hraně desky. Pro navržené moduly se používá DPS se 2 a 4 měděnými vrstvami a jejich cena je zobrazena v tabulce 13.2.

Popis	Cena za kus [Kč]
2-vrstvá DPS	309 Kč
4-vrstvá DPS	411 Kč
4-vrstvá DPS pro displej	915 Kč

Tab. 13.2: Cena použitých DPS do vybraných modulů.

13.2.3 Cena elektronických komponentů

Tabulka 13.3 zobrazuje ceny komponentů pro jednotlivé elektronické komponenty na modul.

Popis	Cena za kus [Kč]
Procesorová jednotka	2598 Kč
Analyzátor sítě	2864 Kč
Digitální vstupy	1049 Kč
Digitální výstupy	1598 Kč
Analogové IO	2283 Kč
Displej	7342 Kč

Tab. 13.3: Cena elektronických součástek na modul.

13.2.4 Odhadovaná cena modulů

V tabulce 13.4 jsou uvedeny odhadované ceny jednotlivých modulů.

Popis	Cena za kus [Kč]
Procesorová jednotka	4090 Kč
Analyzátor sítě	4686 Kč
Digitální vstupy	2735 Kč
Digitální výstupy	3284 Kč
Analogové IO	3961 Kč
Displej	8257 Kč

Tab. 13.4: Cena elektronických součástek na modul.

13.2.5 Porovnání cen s dostupnými produkty na trhu

Porovnání cen je spíše přehledové, jelikož se jedná o porovnání hotového produktu s produktem ve vývoji. Ceny uvedené u vyvíjeného modulárního řídicího systému jsou pouze pro náklady na jeden kus a je tedy skoro nemožné srovnávat sériovou výrobu s výrobou jednoho kusu. Očekává se, že náklady na nákup materiálu jednoho kusu mohou být zlomkové oproti těmto odhadovaným cenám, ale přibudou poplatky spojené s výrobou.

Procesorová jednotka

Porovnání cen procesorové jednotky je velmi složité, jelikož průmyslové PLC většinou nabízí mnohem více funkcí, než navržený řídicí systém, avšak jako procesorovou jednotku lze použít také analyzátor sítě. Cena navržené procesorové jednotky je 4090 Kč. PLC *UniStream* (`U-USC-B5-B1`) od výrobce *Unitronics* je nejvíce podobné (pokud neuvažujeme programovatelná relé), jehož cena je 8453 Kč[34].

Analyzátor sítě

Jelikož řídicí systémy nemají analyzátor sítě obsažený jako modul modulárního systému, používají se externí moduly a je tedy nutné se spoléhat na komunikační protokol, kterým disponuje jak řídicí systém, tak i samotný analyzátor. Modul navrženého analyzátoru sítě stojí 4686 Kč a ceny externích analyzátorů sítí s komunikací se pohybují např. od 4000 do 10000 Kč, ale mívají certifikaci o přesnosti měření (certifikace přesnosti není pro ostrovní síť důležitá).

Digitální periférie

Pro porovnání ceny digitálních periférií je nejvýstižnější použití přepočtu ceny modulu na jeden kanál, jelikož prodávané moduly se prodávají v různých konfigu-

racích. Cena jednoho kanálu digitálního vstupu je 124 Kč a 164 Kč pro digitální výstup. Pro porovnání bude použit modul digitálních vstupů `U-UID-1600`, který obsahuje 16 digitálních vstupů a stojí 2897 Kč[34], tedy 181 Kč na kanál. Modul `U-UID-0016T` obsahující 16 tranzistorových výstupů stojí 3731 Kč[34], tedy 233 Kč na kanál.

Analogové periférie

Navržený modul analogových vstupů a výstupů je těžko srovnatelný, jelikož průmyslové řídicí systémy mají analogové periférie rozděleny na vstupy nebo výstupy a jejich funkcionalita je tedy předdefinovaná pomocí HW. *Unitronics* nabízí modul 8 analogových vstupů `U-UIA-0800N`, který stojí 5106 Kč[34] a modul 6 analogových výstupů `U-UIA-0006` za 5940 Kč[34]. Další variantou je modul pro teplotní snímače RTD `U-UIS-04PTN` s cenou 5450 Kč. Cena navrženého modulu s 8 analogovými kanály, které jsou konfigurovatelné programově je 3961 Kč. Největší výhodou navrženého modulu je konfigurovatelnost, kde jeden modul pokryje funkcionalitu dalších modulů oproti standardnímu PLC (existují moduly s různou HW konfigurací - např. 4 analogové vstupy a 2 analogové výstupy).

13.3 Ověření funkcionality řídicího systému

Tato kapitola se zabývá popisem ověření realizovaných funkcí pro navržený modulární řídicí systém.

13.3.1 Ověření RTOS

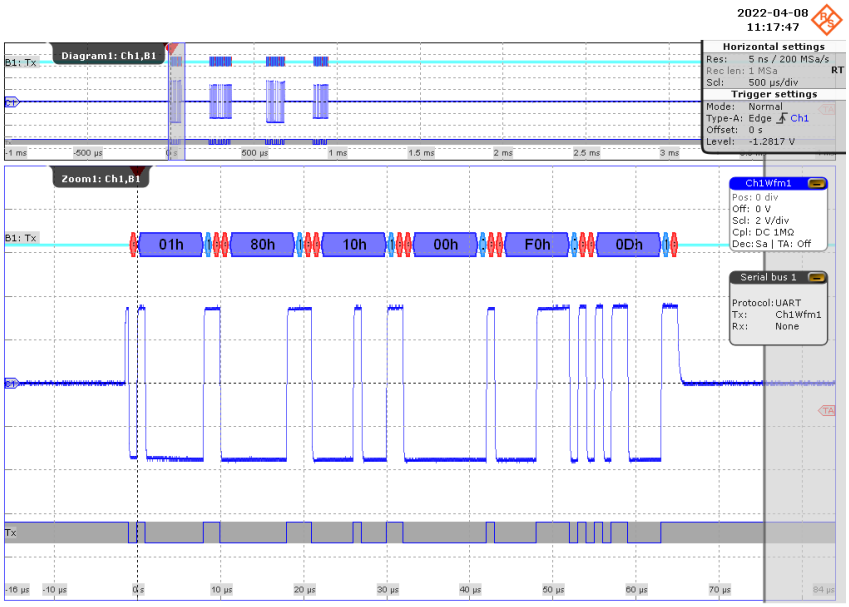
FreeRTOS nabízí několik diagnostických funkcí, kterou může být zjištění času vykonávání jednotlivých úloh. Na obrázku 13.4 je ukázka tohoto diagnostického výpisu v terminálu z modulu displeje s programem pro řízení vstřikovacího lisu. Nejjednodušší diagnostikou je tak zjištění času, po který je procesor v "nečinném"stavu, tedy kdy nevykonává žádný užitečný kód. V tomto případě je to úloha *Idle*, která je spouštěna, pokud neběží žádná jiná úloha. Tato úloha má podíl vykonávaného času 97 %, kdy byl program v manuálním stavu a zpracovával pouze požadavky od přerušení. Výsledná hodnota nemusí odpovídat reálné hodnotě, jelikož diagnostika je spouštěna od časovače s předdefinovaným časem a rozlišení je diskrétní (perioda spouštění této diagnostiky by neměla být příliš malá, aby neovlivňovala výpočetní výkon zařízení).

```
Task            Abs time        Time
****************************************
CDC_CLI_R       110             <1%
IDLE            516064          97%
AutoMode        391             <1%
Tmr Svc         7854            1%
UI              4880            <1%
TempRead        377             <1%
DIRead          904             <1%
AIRead          0               <1%
CDC_Sende       1               <1%
CDC_Send_       157             <1%
MoldMove        0               <1%
ScrewMove       2               <1%
```

Obr. 13.4: Časová analýza běhu RTOS.

13.3.2 Ověření MCSBUS

Na obrázku 13.5 je zobrazen výsledek měření pomocí osciloskopu sběrnice MCSBUS na části *RS485*. Časový diagram zobrazuje cyklickou komunikaci mezi master modulem a 2 slave moduly. Z této komunikace je pořízen detail na první požadavek, který odpovídá požadavku čtení ze slave modulu s adresou 1 uživatelského objektu č. 0 (modul digitálních vstupů - binární data).



Obr. 13.5: Měřený průběh komunikace - požadavek čtení.

Obrázek 13.6 zobrazuje výpis z terminálu na příkaz `GAMI`, který vrací informaci o všech připojených modulech. Výpis odpovídá předpokladu a zobrazuje všechny aktuálně připojené moduly.

```
GAMI

Module 0 info:
        ID: 100
        SW ver: 1.0
        HW ver: 1.0
        Name: PowerAnalyser


Module 1 info:
        ID: 400
        SW ver: 1.0
        HW ver: 1.0
        Name: MCS-DI-22


Module 2 info:
        ID: 500
        SW ver: 1.0
        HW ver: 1.0
        Name: MCS-DO-20


Module 3 info:
        ID: 600
        SW ver: 1.0
        HW ver: 1.0
        Name: MCS-AIO-8
```

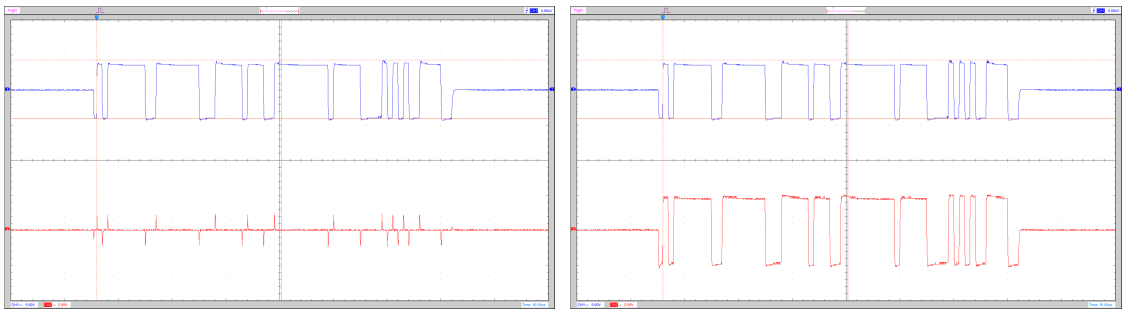
Obr. 13.6: Výpis informace o modulech.

13.3.3 Zakončovací odpory

Obrázek 13.7 zachycuje průběhy měření na sběrnici pro zakončovací odpory. Modrý průběh představuje napětí na sběrnici a červený průběh představuje úbytek napětí na odporu (nulové napětí značí nulový proud a tedy tranzistory v obvodu mají vysoký odpor).

Na obrázku 13.7b je znázorněn průběh, pokud je modul poslední a zakončovací odpor by měl být připojen. Výsledek zobrazuje očekávané hodnoty. Na obrázku 13.7a znázorňuje průběch předposledního modulu, tedy kdy se očekává, že tranzistory nebudou sepnuty a měřené napětí bude nulové. Měřený průběh však obsahuje napěťové špičky, které jsou nejspíše způsobené pomalejšími tranzistory. Obvod využívá vodivosti vnitřních diod, při změně logické úrovně na sběrnici je nutné brát v úvahu dobu zotavení těchto diod.

Při měření byl použit jako master modul analyzátor sítě pouze s jedním zakončovacím odporem, proto je klidový stav sběrnice měřen jako nulové napětí (není definován klidový stav sběrnice.



(a) Předposlední modul. (b) Poslední modul.

Obr. 13.7: Změřený průběh komunikační sběrnice MCSBUS pro verifikaci zakončovacích odporů.

Závěr

V této práci byly popsány průmyslové řídicí systémy a to především systémy PLC, u kterých byli vypsáni významní i zajímaví výrobci. Dále byly rozebrány vybrané průmyslové sběrnice *RS485* a *CAN bus* a byl popsán komunikační protokol *MODBUS*. Byly popsány operační systémy reálného času, které se využívají pro embedded systémy. Jako RTOS byl vybrán *FreeRTOS*, který je využit v praktické části této práce.

V praktické části se podařilo navrhnout, realizovat, oživit a naprogramovat prototyp vlastního modulárního řídicího systému pro ověření funkčnosti modulárního řešení. Pro mezimodulovou komunikaci byl navržen a popsán vlastní komunikační protokol s definicí vlastní sběrnice *MCSBUS* s ohledem na optimalizaci výsledného použití. V návrhu komunikační sběrnice a protokolu je prostor na zdokonalení, kde by bylo možné implementovat ještě do sběrnice *MCSBUS* sběrnice *CANBus* pro přenos cyklických dat, díky níž by se omezil počet požadavků, což by vedlo ke zvýšení průchodnosti celé sběrnice. Vnitřní sběrnice *RS485* by pak sloužila pro konfiguraci, složitá data a především pro asynchronní komunikaci.

S ohledem na rozmanitost nabídky trhu je v dnešní době nevýhodné vytvářet vlastní řídicí systém, který bude konkurovat řídicím systémům PLC, jelikož se většinou jedná o léta zaběhlé systémy, na kterých usilovně pracují velké týmy z různých odvětví. Použití navrženého modulárního řídicího systému by tedy bylo velmi náročné v průmyslových aplikacích a musela by se provést ještě spousta vývoje na jeho dokončení. Jeho přínos je ale pozitivní pro zadávající firmu, jelikož má systém, na kterém může provádět testy různých nových obvodů, které PLC nemají v základu, jako je například analyzátor sítě zabudovaný přímo do řídicího systému. Tento systém by bylo vhodné použít pro sériové výrobky, kde řídicí systémy PLC jsou nákladné a potřebují spoustu zařízení (externích modulů) k dosažení kýžené funkcionality. Z požadavků na výrobu nebude uvažován tento návrh jako návrh konečného produktu pro vodní mikrozdroje, ale bude spíše směřován na jednodušší kompaktní řídicí systémy u kterých by se vývoj rozdělil do několika jednodušších návrhů elektroniky (samostatné DPS propojené sběrnicí bez složité komunikace), což by mohlo přinést finanční úspory. Aktuální modulární řídicí systém by mohl najít uplatnění v jiných aplikacích.

Dalším krokem by mohlo být zveřejnění tohoto projektu jako *Open-Source*, aby se na vývoji mohli podílet další lidé a vytvořit tak komunitu vyvíjející tuto platformu. Vytvoření vlastního programovacího prostředí by pomohlo zaplnit mezeru v trhu *Open-Source* řešení a nabízet tak kompletní řešení SW a HW implementující především standard `EN 61131-3`. Pro rozšíření komunity by byla zvažována varianta, která by byla schopná vytvořit také programový kód založený na `EN 61131-3`

pro *Arduino*, *Raspberry Pi* a další platformy s cílením na jednoduchost. Takto vytvořený produkt by mohl být zajímavý i pro výrobce speciální elektroniky, kde je nabízeno standardní řešení jednoduše rozšířitelné o další moduly a může k tomuto řídicímu systému přispět navržením modulu (elektroniky), který zveřejní, bude sám prodávat a nebo jej využije pouze pro své vlastní účely.

Aktuální moduly by mohly být více optimalizovány a rozšířeny o další vstupy a výstupy (zvýšit hustotu periférií na modul), případně navrhnuty další. Pro větší obsažení trhu by mohla být zvažována verze s tvarem pro osazení do domovních rozvaděčů, kompaktní řešení s nízkou cenou a další varianty. Výrazným vylepšením by bylo použití MPU s operačním systémem založeném na *Linux*, což by zjednodušilo práci s periférierni.

Použití navrženého řídicího systému nemusí být omezeno pouze jen jako řídicí systém, ale může být použit jako vzdálené IO jednotky, což by také umožnilo funkci decentralizovaného řídicího systému.

Literatura

[1] AISLER Germany GmbH, aisler.net [online]. *Webové stránky výrobce DPS Aisler*. [cit. 1. 2. 2022] Dostupné z URL: <https://aisler.net/>

[2] Analog Devices: *Datasheet součástky AD74412R* [online]. [cit. 1. 4. 2022] Dostupné z URL: <https://www.analog.com/media/en/technical-documentation/data-sheets/AD74412R.pdf>

[3] ARM, Jakub. *Detekce anomálií běhu RTOS aplikace*. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2020.

[4] Arm. os.mbed.com [online]. *Webové stránky pro RTOS MbedOS*. [cit. 1. 2. 2022] Dostupné z URL: <https://os.mbed.com/mbed-os/>

[5] BECKHOFF. Beckhoff Information System. infosys.beckhoff.com [online]. *Ladder diagram (LD)* [cit. 1. 2. 2022] Dostupné z URL: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20ld.htm&id>

[6] BECKHOFF. Beckhoff Information System. infosys.beckhoff.com [online]. *The Continuous Function Chart Editor (CFC)* [cit. 1. 2. 2022] Dostupné z URL: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20cfc.htm&id>

[7] BECKHOFF. Beckhoff Information System. infosys.beckhoff.com [online]. *Function Block Diagram (FBD)* [cit. 1. 2. 2022] Dostupné z URL: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20fbd.htm&id>

[8] BECKHOFF. Beckhoff Information System. infosys.beckhoff.com [online]. *Sequential Function Chart (SFC)* [cit. 1. 2. 2022] Dostupné z URL: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20sfc.htm&id>

[9] BECKHOFF. Beckhoff Information System. infosys.beckhoff.com [online]. *Structured Text (ST)* [cit. 1. 2. 2022] Dostupné z URL: <https://infosys.beckhoff.com/english.php?content=../content/1033/tcplccontrol/html/tcplcctrl_languages%20st.htm&id>

[10] Controllino. CONTROLLINO GmbH [online]. *Výrobce HW založeného na arduino* [cit. 12. 4. 2022] Dostupné z URL: <https://www.controllino.com/>

[11] CORRIGAN, S.: *Introduction to the Controller Area Network (CAN)* [online]. [cit. 1. 2. 2022] Dostupné z URL: <https://www.ti.com/lit/pdf/sloa101b>

[12] CiA. can-cia.org [online]. *Interní dokumenty organizace CiA* [cit. 1. 2. 2022] Dostupné z URL: <https://www.can-cia.org/can-knowledge/>

[13] FreeRTOS. freertos.org [online]. *FreeRTOS Kernel Secondary Docs.* [cit. 1. 2. 2022] Dostupné z URL: <https://www.freertos.org/kernel/secondarydocs.html>

[14] Gate Vidyalay. gatevidyalay.com [online]. *Checksum in Networking* [cit. 1. 2. 2022] Dostupné z URL: <https://www.gatevidyalay.com/checksum-checksum-example-error-detection/>

[15] GeeksforGeeks. geeksforgeeks.org [online]. *Undefined Behavior in C and C++* [cit. 1. 2. 2022] Dostupné z URL: <https://www.geeksforgeeks.org/undefined-behavior-c-cpp/>

[16] GeeksforGeeks. geeksforgeeks.org [online]. *Vertical Redundancy Check (VRC) or Parity Check* [cit. 1. 2. 2022] Dostupné z URL: <https://www.geeksforgeeks.org/vertical-redundancy-check-vrc-or-parity-check/>

[17] GeeksforGeeks. geeksforgeeks.org [online]. *Longitudinal Redundancy Check (LRC)/2-D Parity Check* [cit. 1. 2. 2022] Dostupné z URL: <https://www.geeksforgeeks.org/longitudinal-redundancy-check-lrc-2-d-parity-check/>

[18] HYMEL, S.: *Introduction to RTOS - Solution to Part 3 (Task Scheduling).* digikey.cz [online]. [cit. 1. 2. 2022] Dostupné z URL: <https://www.digikey.cz/en/maker/projects/introduction-to-rtos-solution-to-part-3-task-scheduling/8fbb9e0b0eed4279a2dd698f02ce125f>

[19] KUGELSTADT, T.: *The RS-485 Design Guide* [online]. [cit. 1. 2. 2022] Dostupné z URL: <https://www.ti.com/lit/pdf/slla272>

[20] LAPLANTE, P. A.: *Real-time Systems Design and Analysis.* John Wiley & Sons, Inc., 2004, ISBN 0-471-22855-9.

[21] LAURILA, H.: *Thermocouple Cold (Reference) Junction Compensation.* blog.beamex.com [online]. [cit. 1. 2. 2022] Dostupné z URL: <https://blog.beamex.com/thermocouple-cold-junction-compensation>

[22] Mervis. ENERGOCENTRUM PLUS, s.r.o. [online]. *SW Mervis pro řídicí systémy* [cit. 12. 4. 2022] Dostupné z URL: <https://mervis.info>

[23] Modbus Organization. modbus.org [online]. *Specifikace protokolu MODBUS* [cit. 1. 2. 2022] Dostupné z URL: <https://modbus.org/specs.php>

[24] NOVOTNÝ, Jan. *Řídicí systém pro zařízení využívající obnovitelné zdroje energie*. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií, 2020.

[25] NOVOTNÝ, Jan. In: Proceedings I of the 27st Conference STUDENT EEICT 2021: General papers [online]. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, s. 213-217 [cit. 2022-01-02]. ISBN 978-80-214-5942-7.

[26] PHOENIX CONTACT GmbH & Co. KG:. phoenixcontact.com [online]. Německo. *Webové stránky PHOENIX CONTACT*. [cit. 1. 2. 2022] Dostupné z URL: <https://www.phoenixcontact.com>

[27] Thiago Alves. OpenPLC [online]. *Projekt OpenPLC - SW platforma pro Raspberry Pi* [cit. 12. 4. 2022] Dostupné z URL: <https://www.openplcproject.com/>

[28] PLC automatizace. plc-automatizace.cz [online]. *Cyklus PLC* [cit. 1. 2. 2022] Dostupné z URL: <http://plc-automatizace.cz/knihovna/plc/plc-cyklus.htm>

[29] Rexgen. REX Controls s.r.o. [online]. *SW Rexygen pro řídicí systémy* [cit. 12. 4. 2022] Dostupné z URL: <https://www.rexygen.com>

[30] RONEŠOVÁ, A.: *Přehled protokolu MODBUS* [online]. [cit. 1. 2. 2022] Dostupné z URL: <http://home.zcu.cz/~ronesova/bastl/files/modbus.pdf>

[31] SANTOS, R.: *Controller Area Network: A Brief Summary*[online]. linkedin.com [cit. 1. 2. 2022] Dostupné z URL: <https://www.linkedin.com/pulse/controlled-area-network-brief-summary-rafael-pimentel-dos-santos>

[32] SEGGER. segger.com [online]. *Webové stránky pro RTOS embOS*. [cit. 1. 2. 2022] Dostupné z URL: <https://www.segger.com/products/rtos/embos/>

[33] Stephen St. Michael:*Introduction To Real-Time Embedded Systems*. allaboutcircuits.com [online]. [cit. 1. 2. 2022] Dostupné z URL:

<https://www.allaboutcircuits.com/technical-articles/introduction-to-real-time-embedded-systems/>

[34] Schmachtl.cz, SCHMACHTL CZ spol. s r.o. [online]. *Internetový obchod Schmachtl* [cit. 12. 4. 2022] Dostupné z URL: <https://www.schmachtl.cz/>

[35] Texas Instruments: *Datasheet součástky ADS1118* [online]. [cit. 1. 2. 2022] Dostupné z URL: <https://www.ti.com/lit/gpn/ads1118>

[36] Unipi technology. Faster CZ spol. s r.o. [online]. *Český výrobce řídicích systémů* [cit. 12. 4. 2022] Dostupné z URL: <https://www.unipi.technology/cs/>

[37] Weston Embedded Solutions. weston-embedded.com [online]. *Webové stránky pro RTOS microC/OS.* [cit. 1. 2. 2022] Dostupné z URL: <https://weston-embedded.com/micrium/overview>

[38] Wind River. windriver.com [online]. *Webové stránky pro RTOS VxWorks.* [cit. 1. 2. 2022] Dostupné z URL: <https://www.windriver.com/products/vxworks>

[39] *RS-232*. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, poslední úprava 30. 9. 2021 [cit. 1. 2. 2022]. Dostupné z URL: <https://cs.wikipedia.org/wiki/RS-232>

[40] *CAN bus*. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, poslední úprava 30. 12. 2021 [cit. 1. 2. 2022]. Dostupné z URL: <https://en.wikipedia.org/wiki/CAN_bus>

[41] *Paritní bit*. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, poslední úprava 25. 10. 2021 [cit. 1. 2. 2022]. Dostupné z URL: <https://cs.wikipedia.org/wiki/Paritn%C3%AD_bit>

[42] *Cyclic redundancy check*. Wikipedia: the free encyclopedia [online]. St. Petersburg (Florida): Wikipedia Foundation, poslední úprava 31. 12. 2021 [cit. 1. 2. 2022]. Dostupné z URL: <https://en.wikipedia.org/wiki/Cyclic_redundancy_check>

Seznam symbolů a zkratek

ADC	Analog to Digital Convertor – Analogově-digitální převodník
bps	bits per second – jednotka rychlosti přenosu dat
CLI	Command Line Interface – příkazový řádek
CPU	Central Processing Unit – centrální procesorová jednotka
DAC	Digital to Analog Convertor – aigitálně-analogový převodník
DSP	Digital Signal Procesor – číslicový signálový procesor
FPU	Floating Point Unit – výpočetní jednotka pro práci s čísly s plovoucí desetinou čárkou
MCU	Microcontroller Unit – mikrokontrolér
HMI	Human-Machine Interface – uživatelské rozhraní
HW	Hardware
I^2C	Inter-Integrated Circuit – synchronní sběrnice mikrokontroléru někdy označována jako I2C nebo IIC
IPC	Industrial PC – průmyslový počítač
LDO	Low-Dropout regulator – lineární stabilizátor napětí
MAC	Medium Access Control Layer – označení linkové vrstvy Ethernetu
OS	Operating System – operační systém
PAC	Programmable Automation Controller – jiný výraz pro PLC
PHY	Physical Layer – označení fyzické vrstvy Ethernetu
PGA	Programmable Gain Amplifier – programovatelný zesilovač (proměnlivé zesílení)
PLC	Programmable Logic Controller – programovatelný logický automat (průmyslový řídicí systém)
PSRR	Power Supply Rejection Ratio – vlastnost obvodu potlačení rušení na napájecích vstupech
PWM	Pulse Width Modulation – pulsně šířková modulace

QSPI Quad Serial Peripheral Interface – rozšířená sběrnice SPI s vyšším počtem vodičů

RMII Reduced Media-Indipended Interface – sběrnice pro propojení MAC a PHY Ethernetu

RTC Real Time Clock – hodiny reálného času

RTD Resistance Temperature Detector - odporový snímač teploty

RTOS Real-Time Operating System – operační systém reálného času

SPI Serial Peripheral Interface – sériová komunikace mikrokontroléru

SW Software

SWIO softwarově konfigurovatelných vstupně–výstupních periférií

TWI Two Wire Interface – sběrnice od Atmel shodná s i^2C

UART Universal Asynchronous Receiver-Transmitter – univerzální asynchronní sériová komunikační sběrnice mikrokontroléru

USART Universal Synchronous Asynchronous Receiver-Transmitter – rozšířená sběrnice UART se synchronizační vlastností

Seznam příloh

A Schématická zapojení

A.1 Procesorová jednotka

Sheet: Supply

File: Supply.sch

Sheet: MCU

File: Procesor.sch

Sheet: Communication

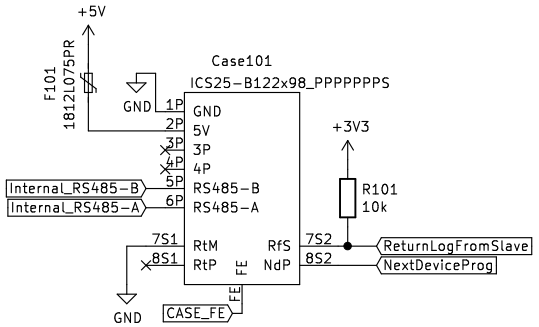
File: Communication.sch

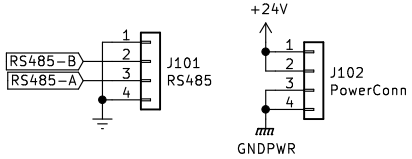
Sheet: Fast digital input

File: FDI2.sch

Sheet: Ethernet

File: Eth.sch





Mechanical Case102 ICS-fill

Mechanical Case103 ICS-fill

Mechanical Case104 ICS-fill

G102
GraphSymbol
WEEEE

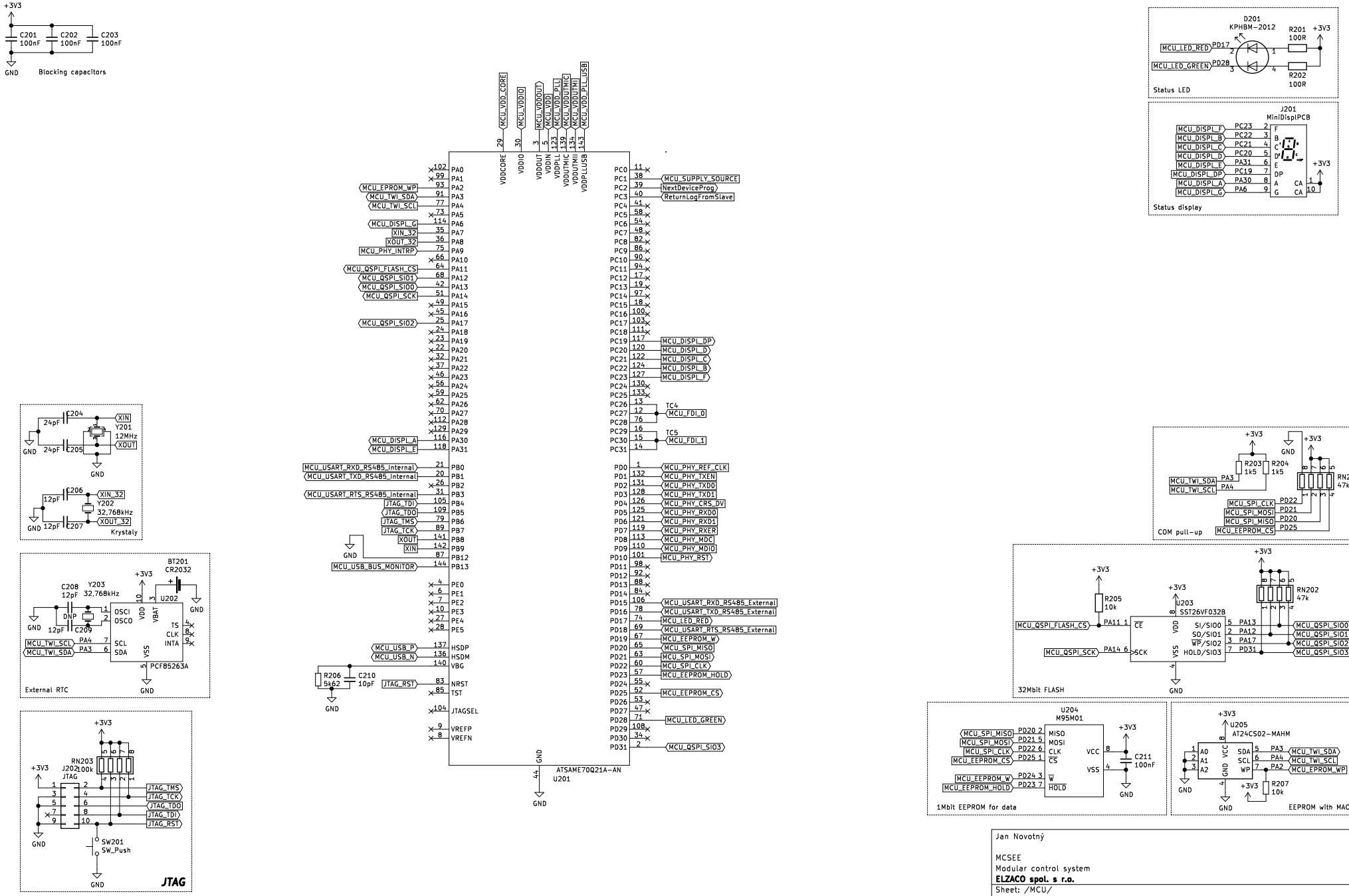
Jan Novotný

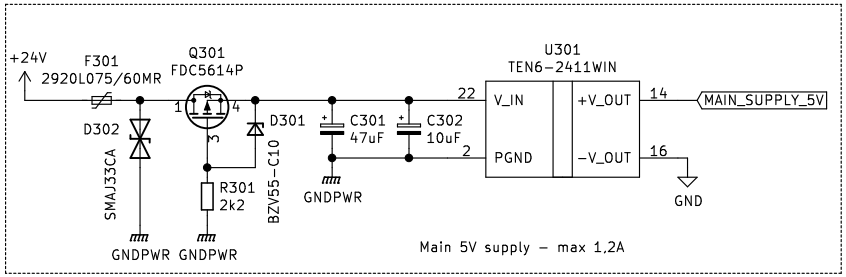
MCSEE
Modular control system
ELZACO spol. s r.o.

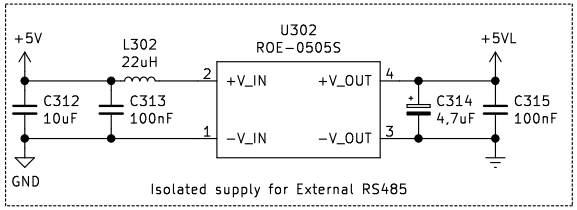
Sheet: /
File: RidiciSystem.sch

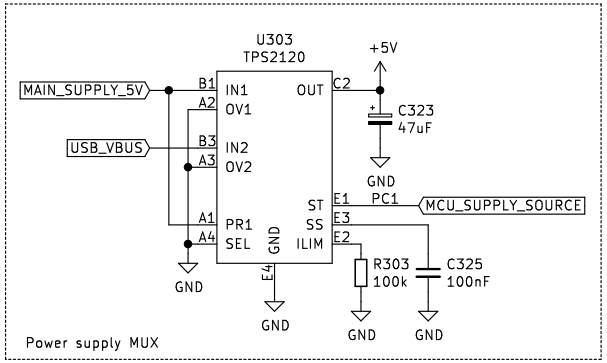
Title: Control unit for Modular control system

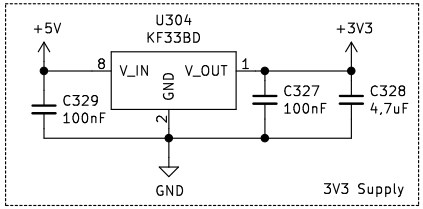
Size: A4	Date: 2021-09-27	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 1/6

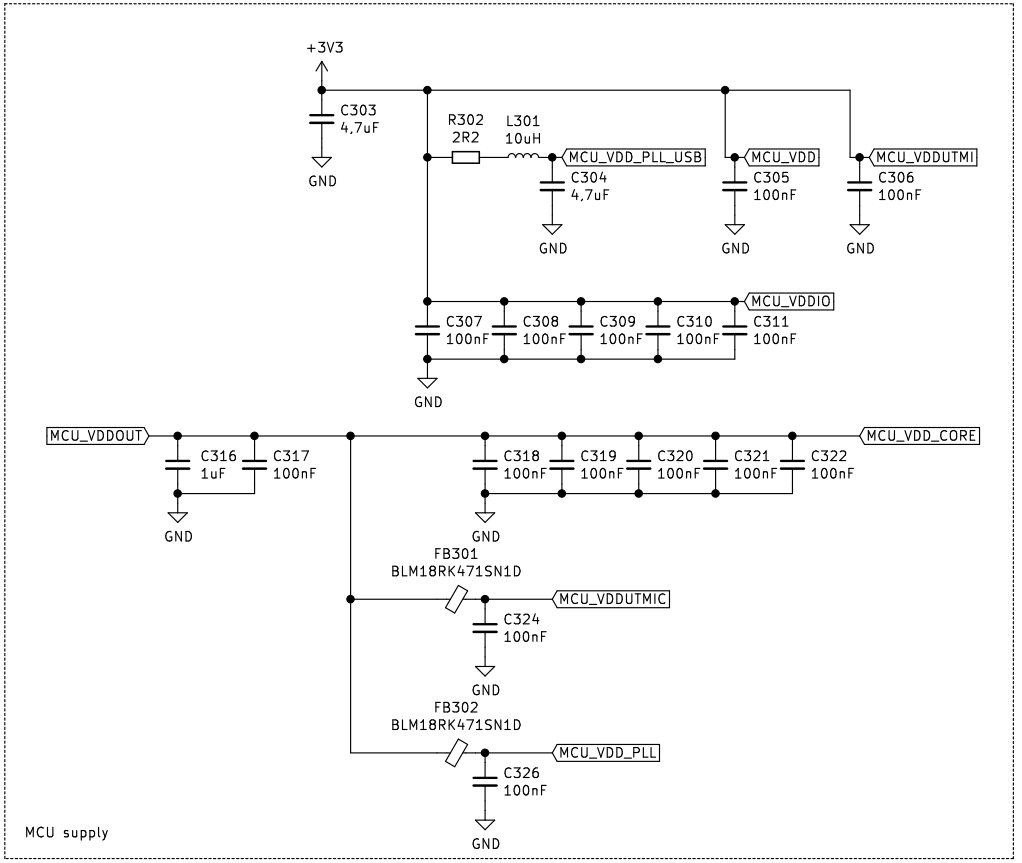

+3V3
C201 100nF
C202 100nF
C203 100nF
GND
Blocking capacitors
MCU_VDD_CORE
MCU_VDDIO
MCU_VDDOUT
MCU_VDD
MCU_VDD_PLL
MCU_VDDUTMC
MCU_VDDUTMI
MCU_VDD_PLL_USB
VDDCORE
VDDIO
VDDOUT
VDDIN
VDDPLL
VDDUTMC
VDDUTMII
VDDPLLUSB
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 PA8 PA9 PA10 PA11 PA12 PA13 PA14 PA15 PA16 PA17 PA18 PA19 PA20 PA21 PA22 PA23 PA24 PA25 PA26 PA27 PA28 PA29 PA30 PA31
PB0 PB1 PB2 PB3 PB4 PB5 PB6 PB7 PB8 PB9 PB12 PB13
PE0 PE1 PE2 PE3 PE4 PE5
HSDP HSDM VBG NRST TST JTAGSEL VREFP VREFN
PC0 PC1 PC2 PC3 PC4 PC5 PC6 PC7 PC8 PC9 PC10 PC11 PC12 PC13 PC14 PC15 PC16 PC17 PC18 PC19 PC20 PC21 PC22 PC23 PC24 PC25 PC26 PC27 PC28 PC29 PC30 PC31
PD0 PD1 PD2 PD3 PD4 PD5 PD6 PD7 PD8 PD9 PD10 PD11 PD12 PD13 PD14 PD15 PD16 PD17 PD18 PD19 PD20 PD21 PD22 PD23 PD24 PD25 PD26 PD27 PD28 PD29 PD30 PD31
MCU_SUPPLY_SOURCE
NextDeviceProg
ReturnLogFromSlave
MCU_DISPL_DP
MCU_DISPL_D
MCU_DISPL_C
MCU_DISPL_B
MCU_DISPL_F
MCU_FDI_0
MCU_FDI_1
MCU_PHY_REF_CLK
MCU_PHY_TXEN
MCU_PHY_TXD0
MCU_PHY_TXD1
MCU_PHY_CRS_DV
MCU_PHY_RXD0
MCU_PHY_RXD1
MCU_PHY_RXER
MCU_PHY_MDC
MCU_PHY_MDIO
MCU_PHY_RST
MCU_USART_RXD_RS485_External
MCU_USART_TXD_RS485_External
MCU_LED_RED
MCU_USART_RTS_RS485_External
MCU_EEPROM_W
MCU_SPI_MISO
MCU_SPI_MOSI
MCU_SPI_CLK
MCU_EEPROM_HOLD
MCU_EEPROM_CS
MCU_LED_GREEN
MCU_QSPI_SIO3
ATSAME70Q21A-AN
U201
MCU_EPROM_WP
MCU_TWI_SDA
MCU_TWI_SCL
MCU_QSPI_FLASH_CS
MCU_QSPI_SIO1
MCU_QSPI_SIO0
MCU_QSPI_SCK
MCU_QSPI_SIO2
MCU_DISPL_G
XIN_32
XOUT_32
MCU_PHY_INTRP
MCU_DISPL_A
MCU_DISPL_E
MCU_USART_RXD_RS485_Internal
MCU_USART_TXD_RS485_Internal
MCU_USART_RTS_RS485_Internal
JTAG_TDI
JTAG_TDO
JTAG_TMS
JTAG_TCK
XOUT
XIN
MCU_USB_BUS_MONITOR
MCU_USB_P
MCU_USB_N
JTAG_RST
R206 5k62
C210 10pF
C204 24pF
C205 24pF
Y201 12MHz
C206 12pF
C207 12pF
Y202 32,768kHz
Krystaly
C208 12pF
DNP
C209 12pF
Y203 32,768kHz
BT201 CR2032
U202 PCF85263A
External RTC
RN203 100k
J202 JTAG
SW201 SW_Push
JTAG
D201 KPHBM-2012
R201 100R
R202 100R
MCU_LED_RED
MCU_LED_GREEN
Status LED
J201 MiniDisplPCB
MCU_DISPL_F
MCU_DISPL_B
MCU_DISPL_C
MCU_DISPL_D
MCU_DISPL_E
MCU_DISPL_DP
MCU_DISPL_A
MCU_DISPL_G
Status display
R203 1k5
R204 1k5
RN201 47k
COM pull-up
R205 10k
U203 SST26VF032B
RN202 47k
32Mbit FLASH
U204 M95M01
C211 100nF
1Mbit EEPROM for data
U205 AT24CS02-MAHM
R207 10k
EEPROM with MAC
Jan Novotný
MCSEE
Modular control system
ELZACO spol. s r.o.
Sheet: /MCU/
File: Procesor.sch
Title: Control unit for Modular control system
Size: A3
Date: 2021-09-27
Rev: 1.0.2
KiCad E.D.A. kicad (5.1.10)-1
Id: 2/6



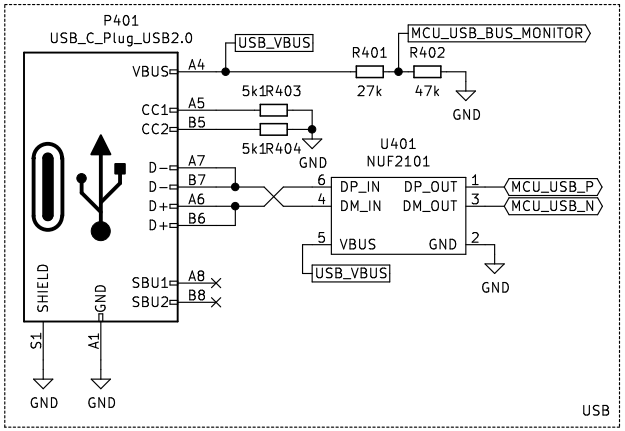


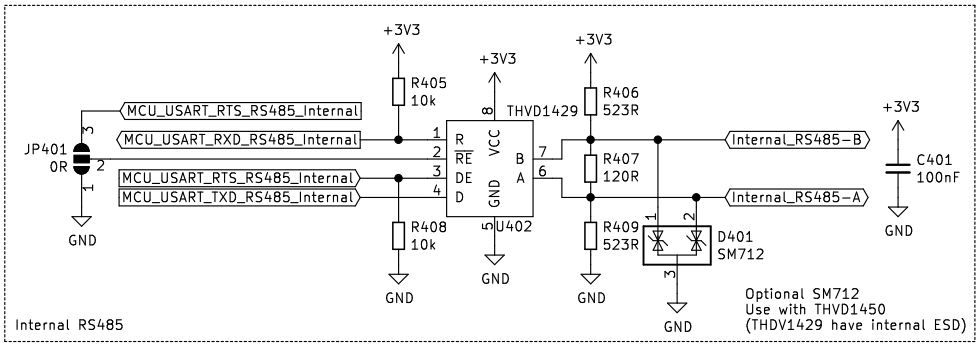


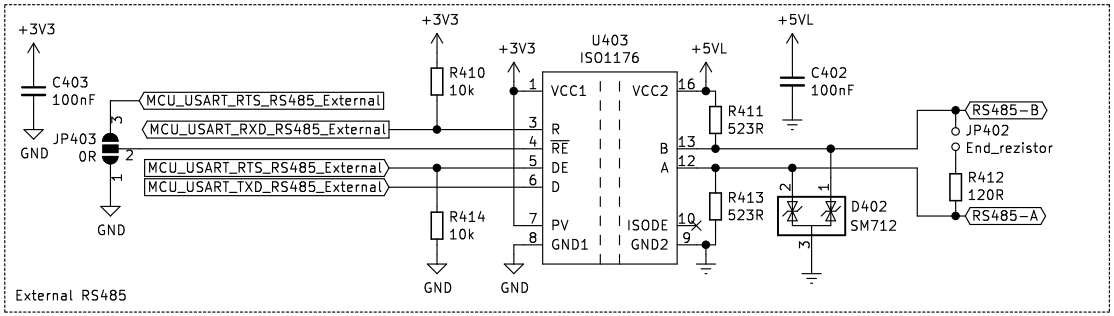




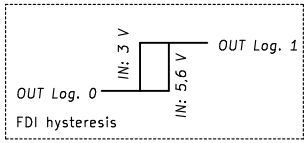
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Supply/ File: Supply.sch		
Title: Control unit for Modular control system		
Size: A4	Date: 2021-09-27	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 3/6

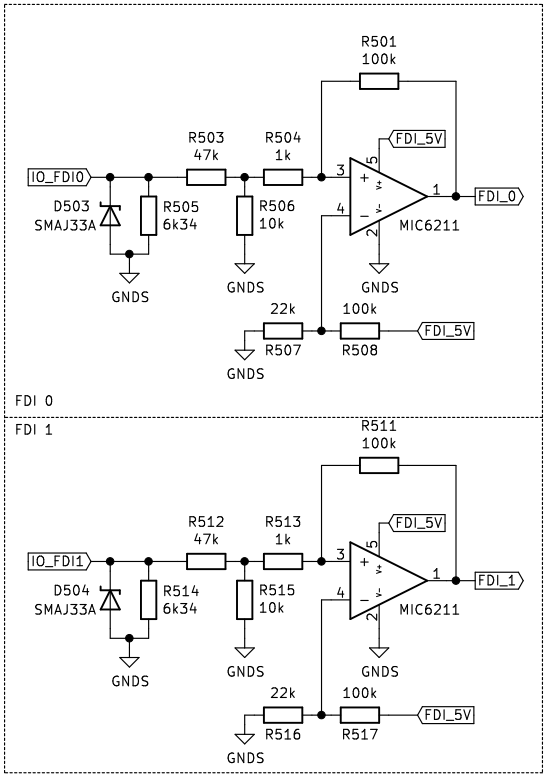


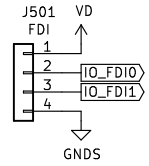


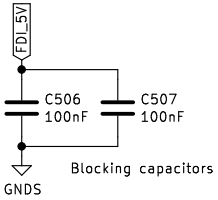


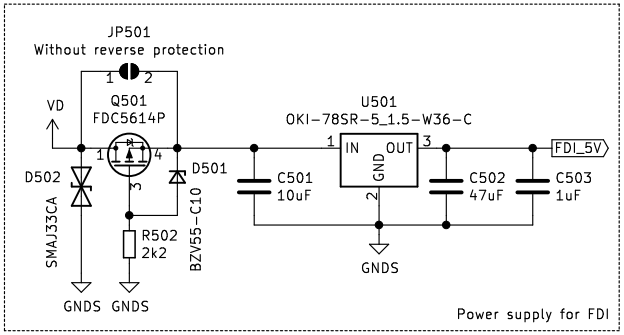
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Communication/ File: Communication.sch		
Title: Control unit for Modular control system		
Size: A4	Date: 2021-09-27	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 4/6

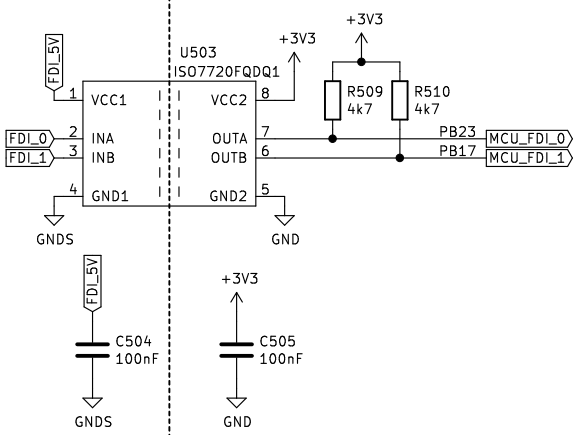












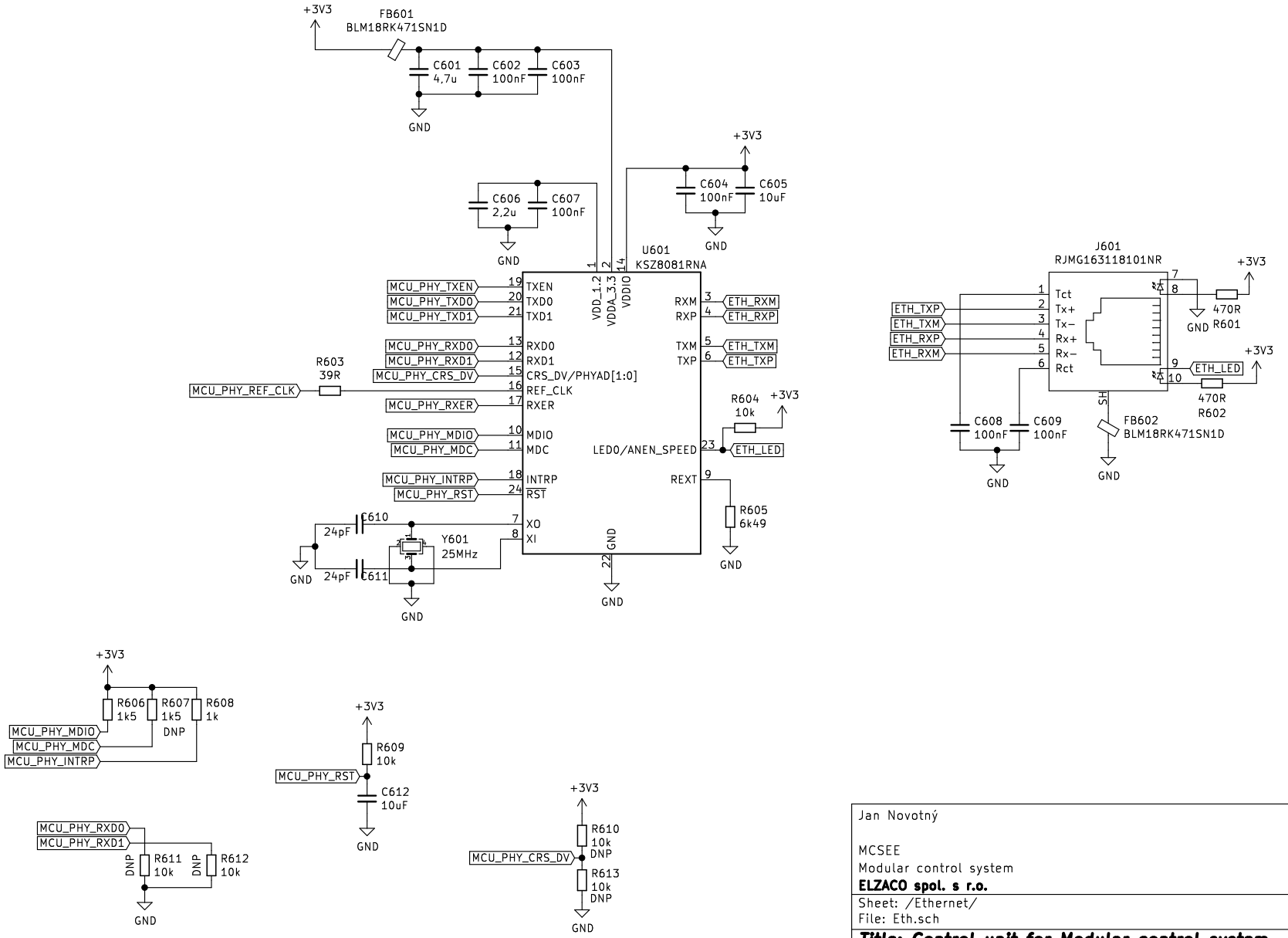
Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Fast digital input/
File: FDI2.sch

Title: Control unit for Modular control system

Size: A4	Date: 2021-09-27	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 5/6



Jan Novotný	
MCSEE	
Modular control system	
ELZACO spol. s r.o.	
Sheet: /Ethernet/	
File: Eth.sch	
Title: Control unit for Modular control system	
Size: A4	Date: 2021-09-27 **Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1	Id: 6/6

A.2 Procesorová jednotka LITE

Sheet: Napajeni

File: Supply.sch

Sheet: MCU

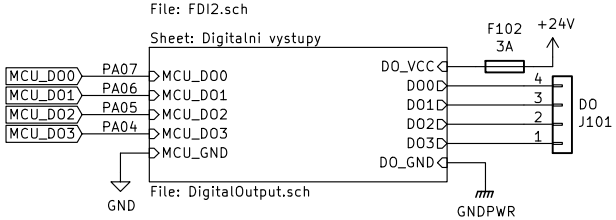
File: Procesor.sch

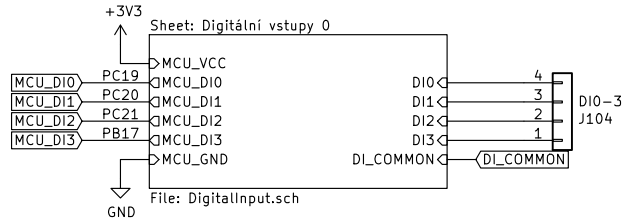
Sheet: Komunikace

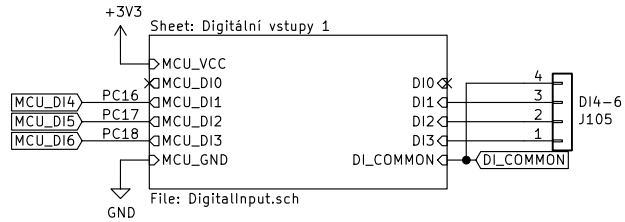
File: Communication.sch

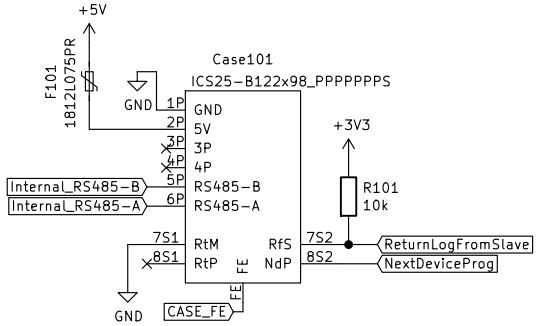
Sheet: RychleDI

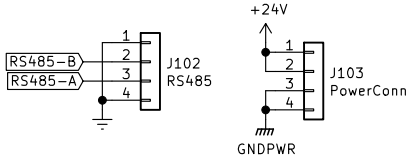
File: FDI2.sch











Jan Novotný

Řídící systém

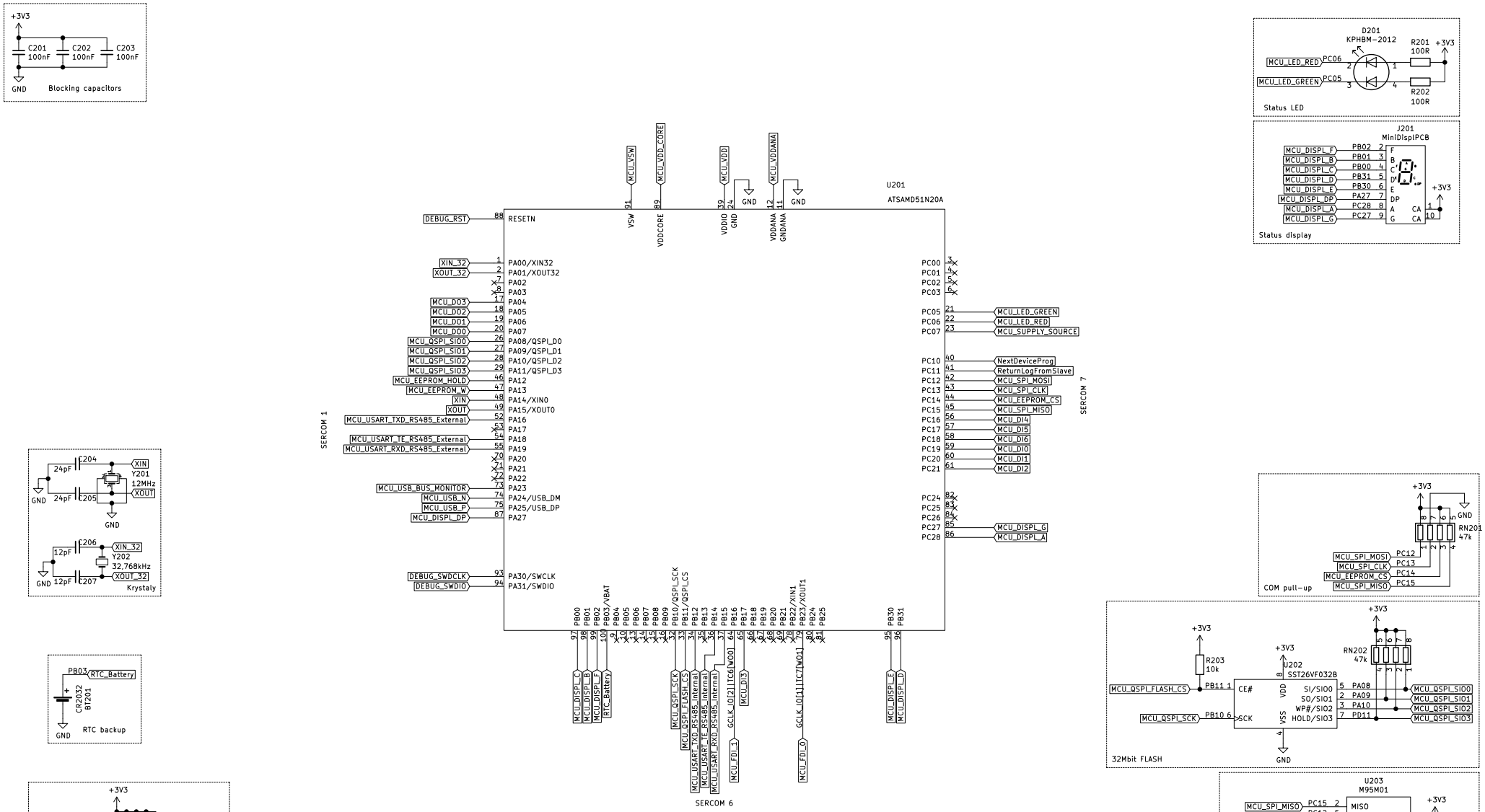
ELZACO spol. s r.o.

Sheet: /

File: RidiciSystemLite.sch

Title: Řídící jednotka pro modulární řídící systém LITE

Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 1/8



TC má hodiny přes Generic clock, tak při měření impulsů jeden rychlý vstup zabije 2 TC, jelikož vždy 2 timery využívají jednoho hodinového signálu, ale pouze pokud čítám vnější impulsy. SAME70 je na tom lépe, jelikož hodinový signál má jako periférii
Každý signál z FDI vést do GCLK_IO a do TCx/WO[n],
aby to bylo použitelné jako čítač impulsů a capture mode

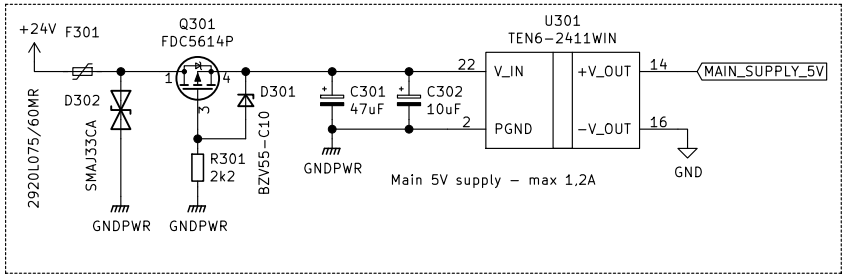
Jan Novotný

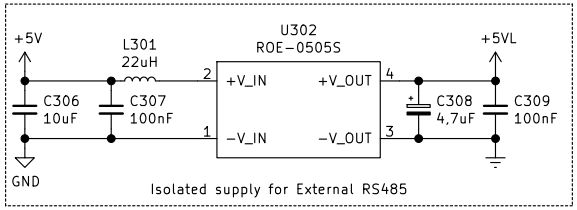
Řídící systém
ELZACO spol. s r.o.

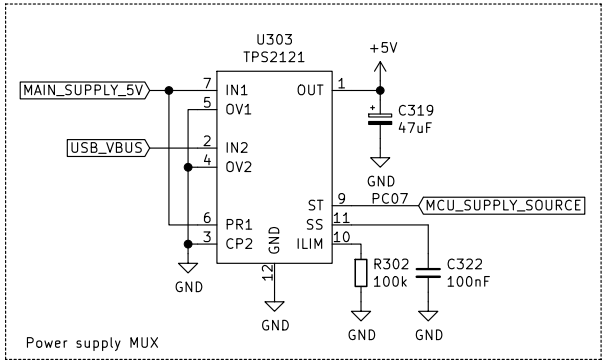
Sheet: /MCU/
File: Procesor.sch

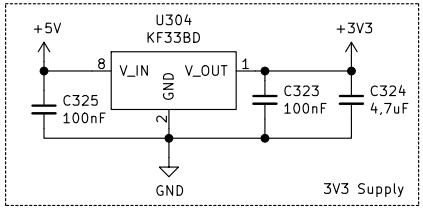
Title: Řídící jednotka pro modulární řídící systém LITE

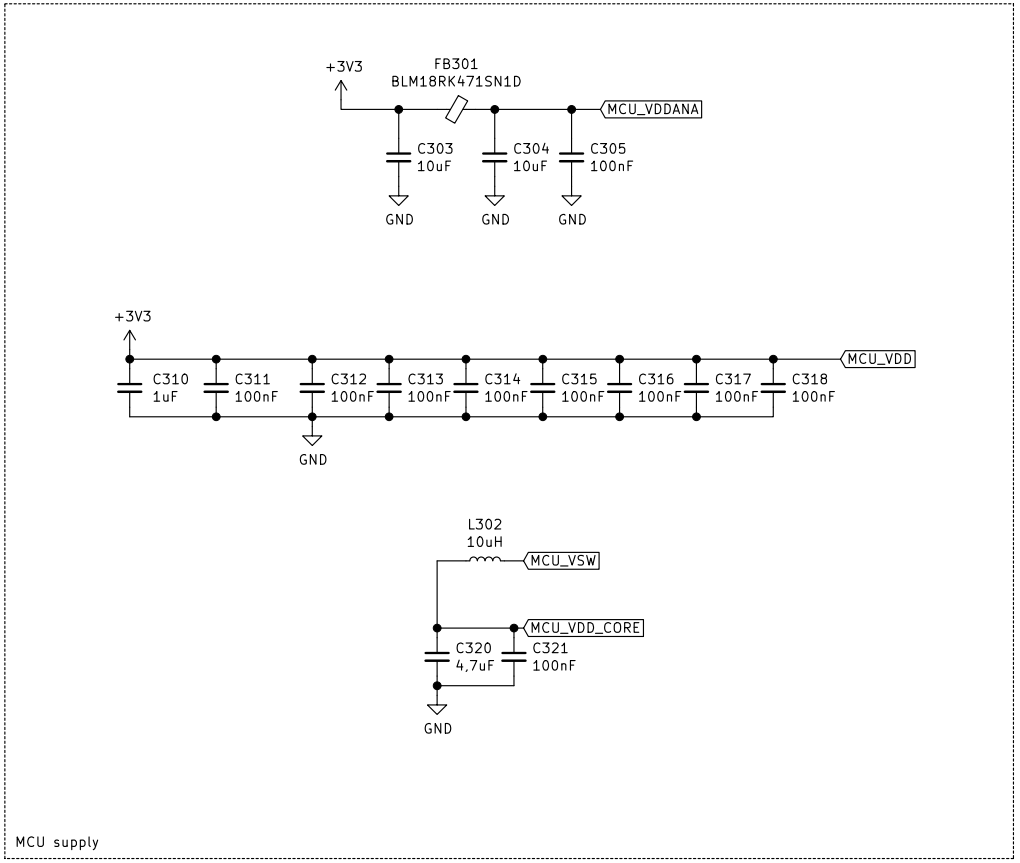
Size: A3	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 2/8



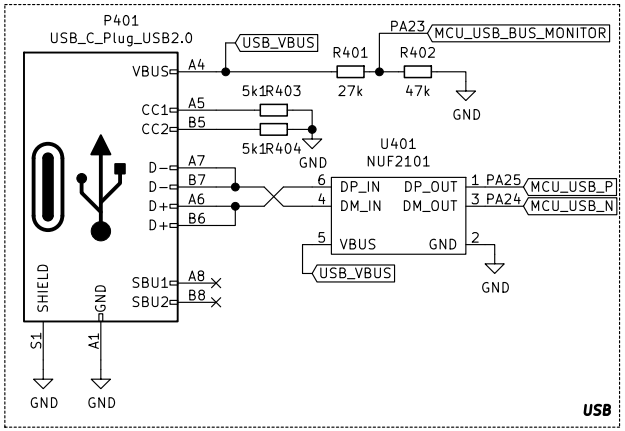


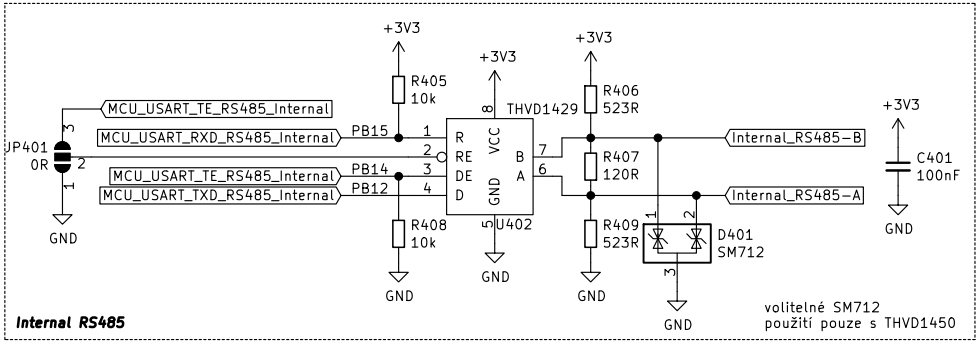


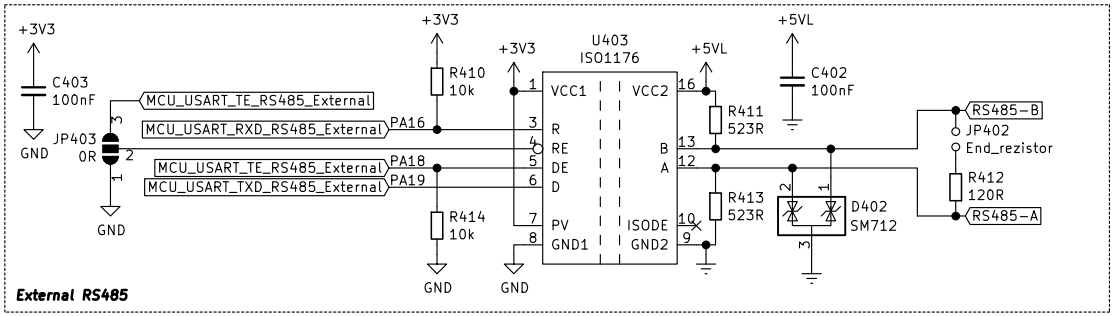




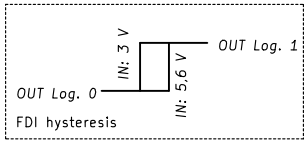
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Napajeni/ File: Supply.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 3/8

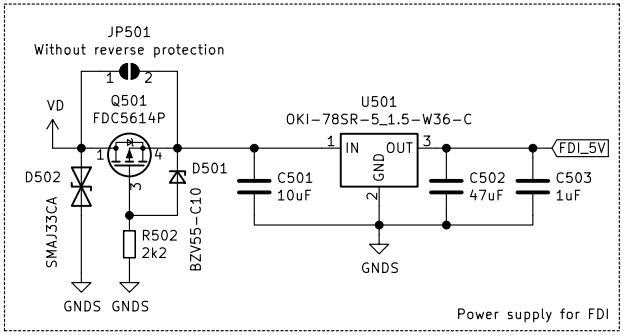


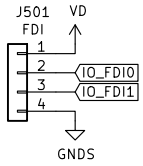


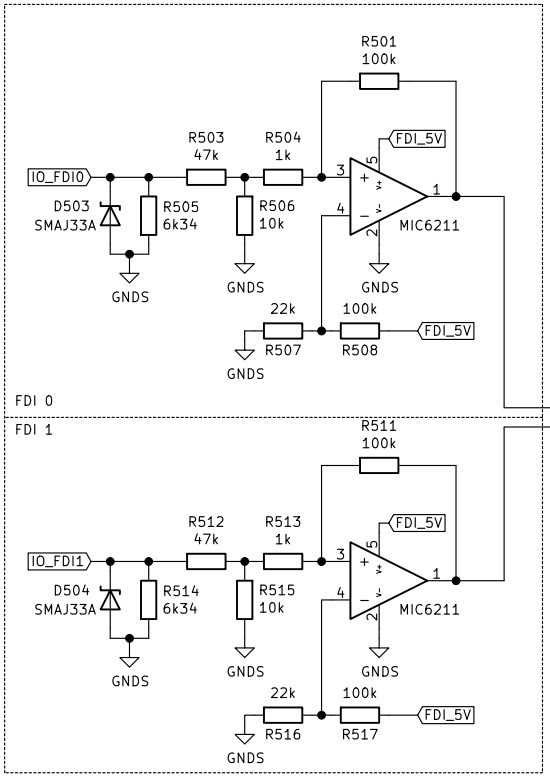


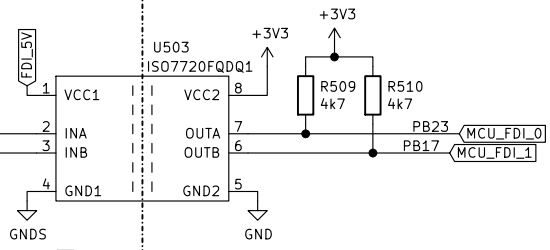
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Komunikace/ File: Communication.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020−08−27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)−1		Id: 4/8

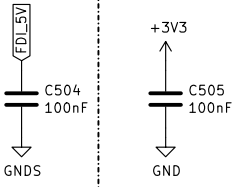




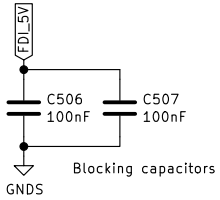




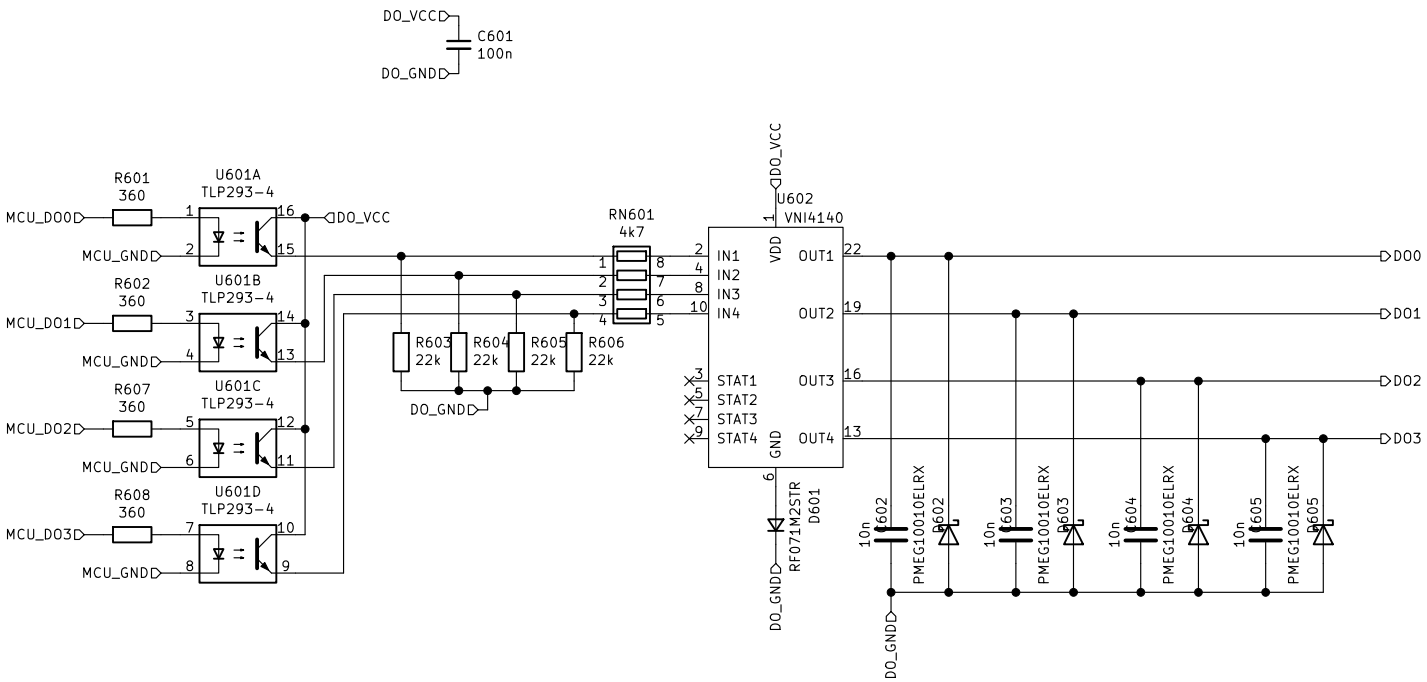




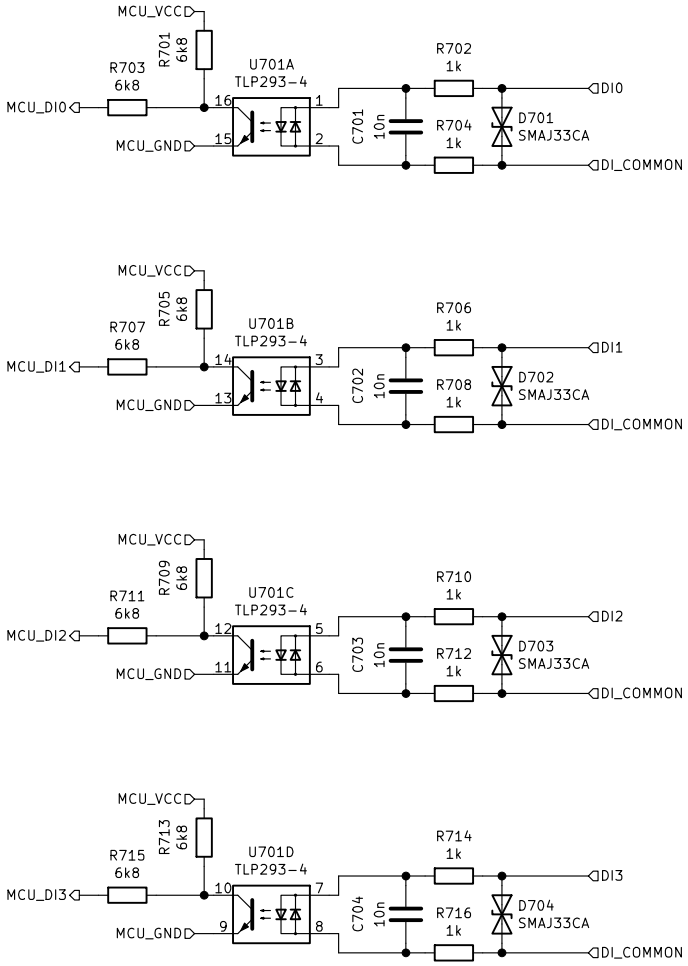
FDI side MCU side



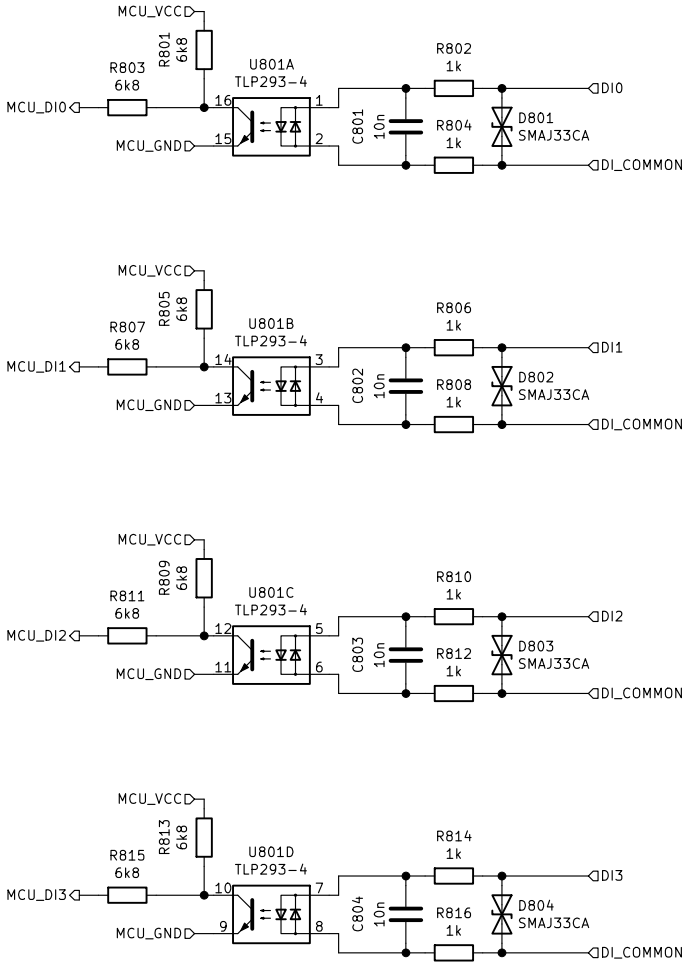
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /RychleDI/ File: FDI2.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 5/8



Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy/ File: DigitalOutput.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 6/8



Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitální vstupy 0/ File: DigitalInput.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 7/8



Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitální vstupy 1/ File: DigitalInput.sch		
Title: Řídící jednotka pro modulární řídící systém LITE		
Size: A4	Date: 2020-08-27	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 8/8

A.3 Analyzátor sítě

Sheet: Supply

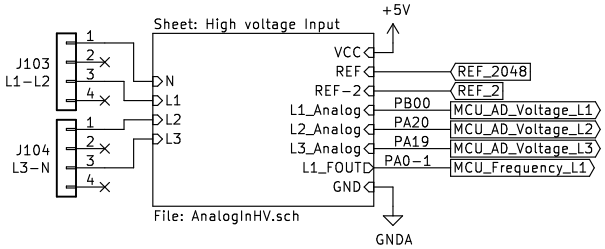
File: Supply.sch

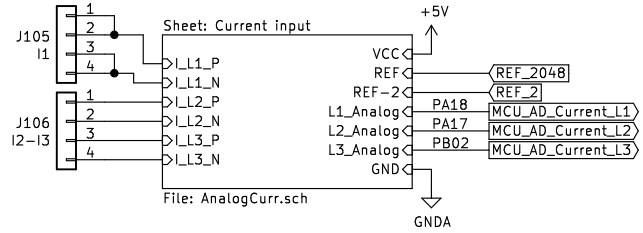
Sheet: MCU

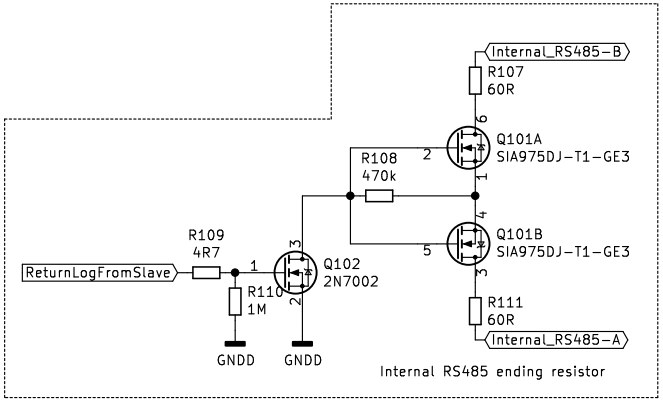
File: Procesor.sch

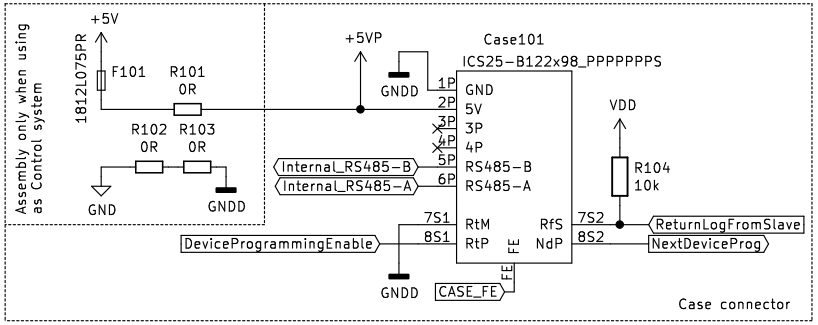
Sheet: Communication

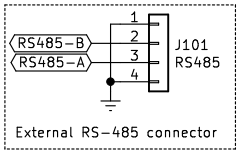
File: Communication.sch

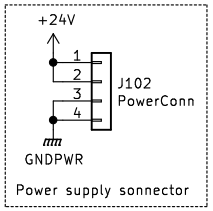


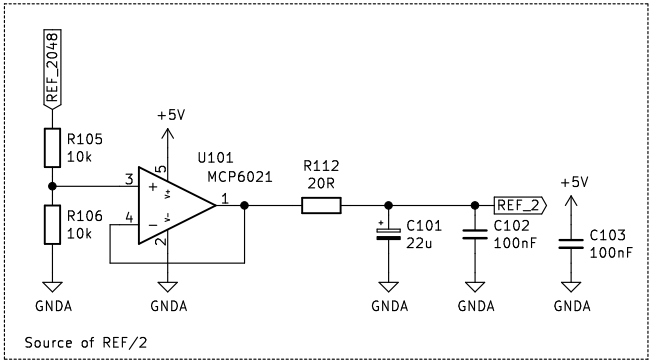












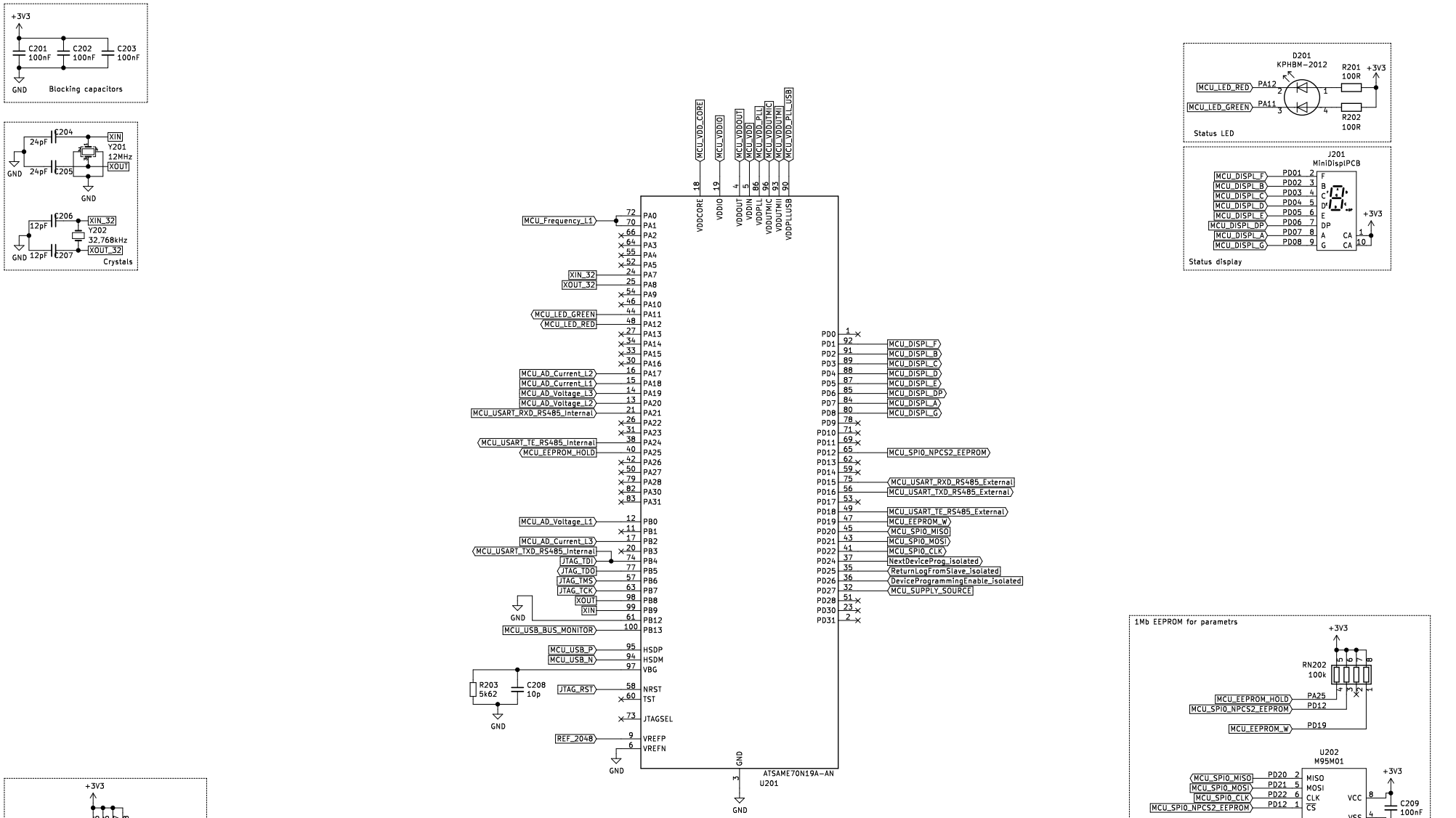
Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /
File: AnalyzatorSite.sch

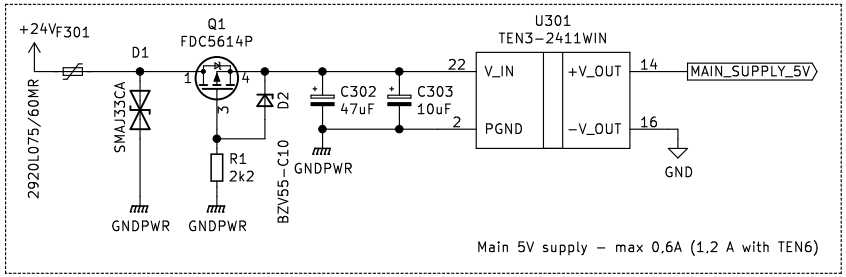
Title: Power analyzer

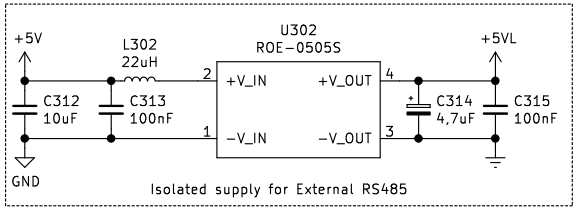
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 1/12

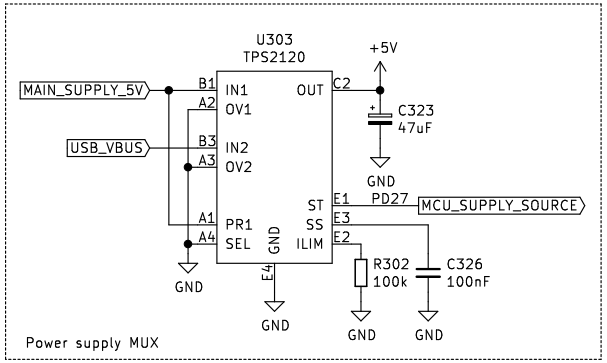


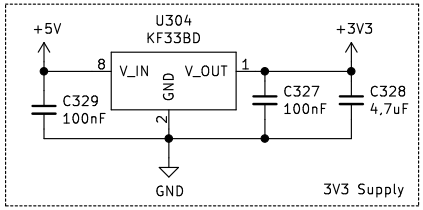
Jan Novotný	
MCSEE	
Modular control system	
ELZACO spol. s r.o.	
Sheet: /MCU/	
File: Procesor.sch	
Title: Power analyzer	
Size: A3	Date: 2021-09-28
KiCad E.D.A. kicad (5.1.10)-1	Id: 2/12

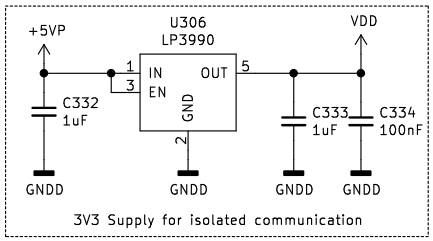
Rev: 1.0.2

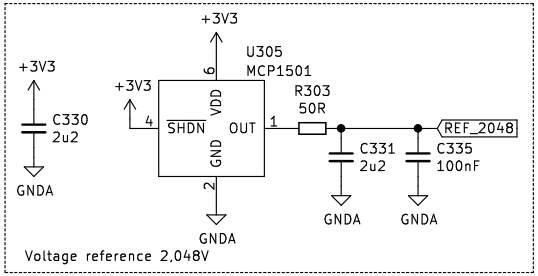


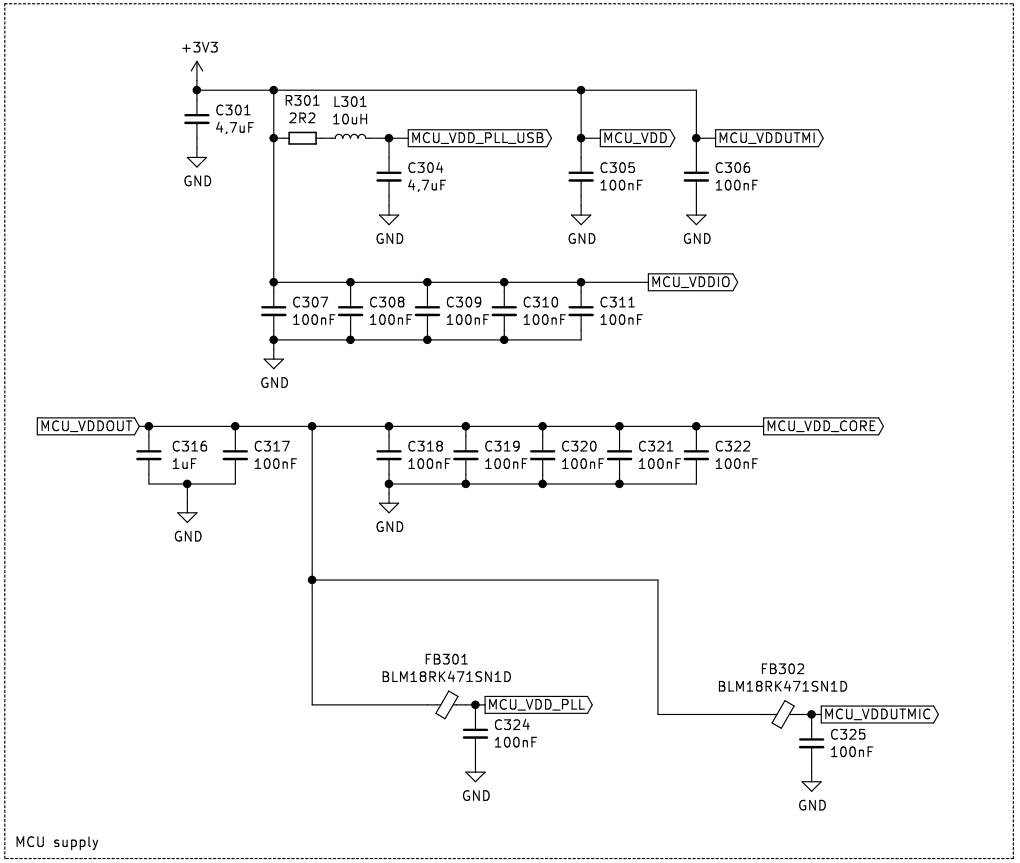




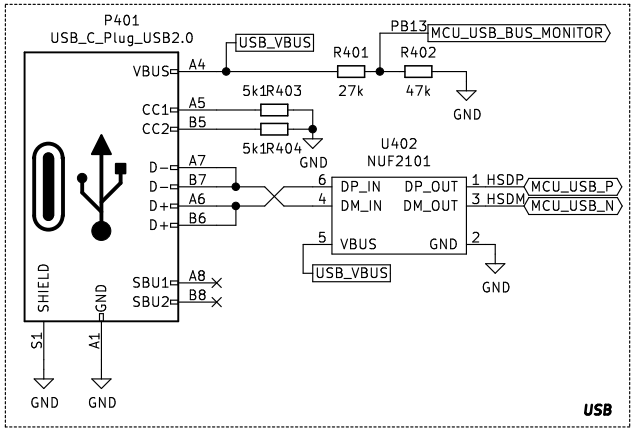


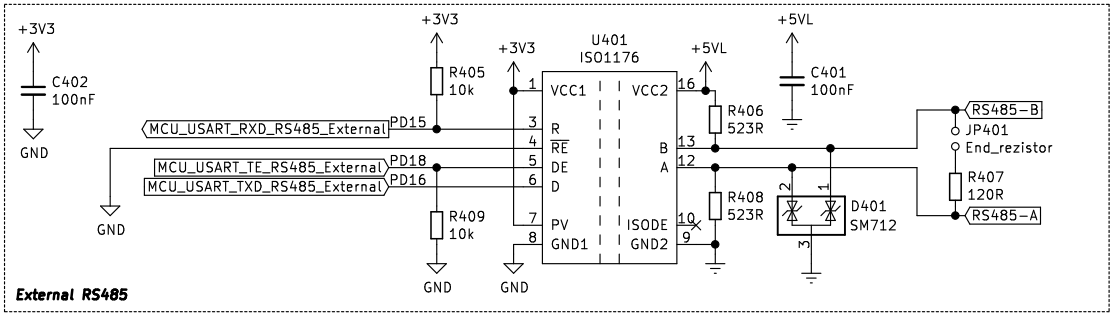


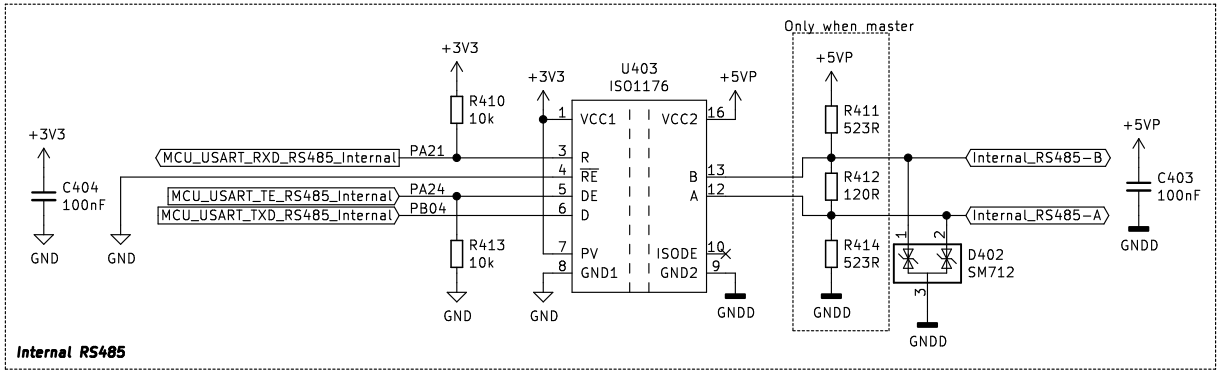


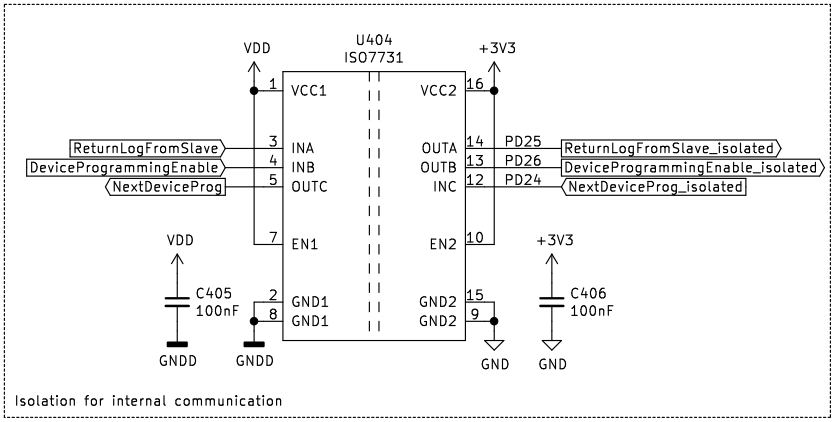


Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Supply/ File: Supply.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 3/12

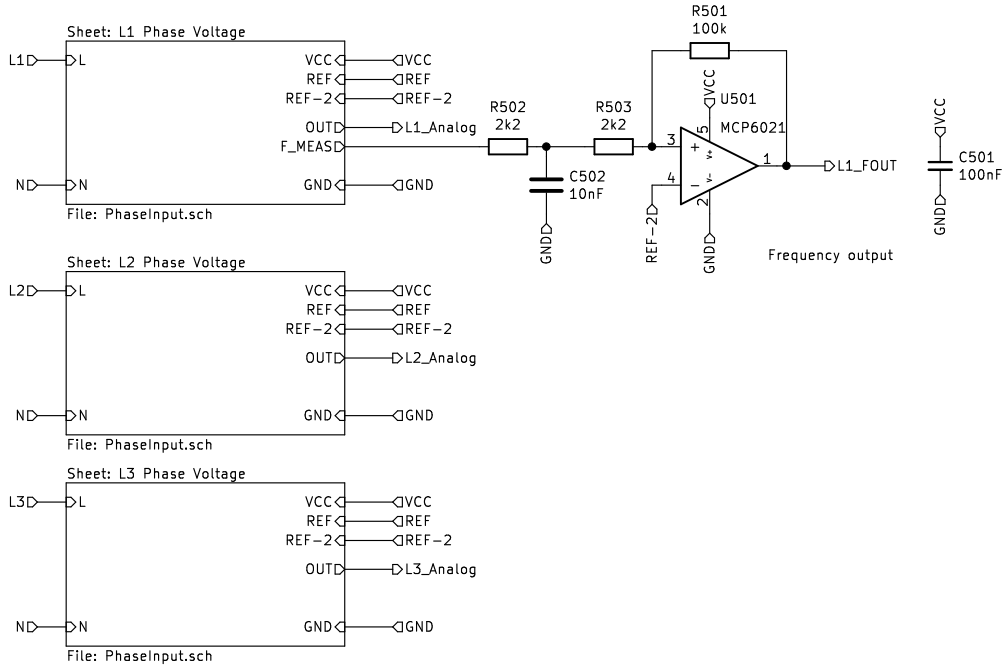




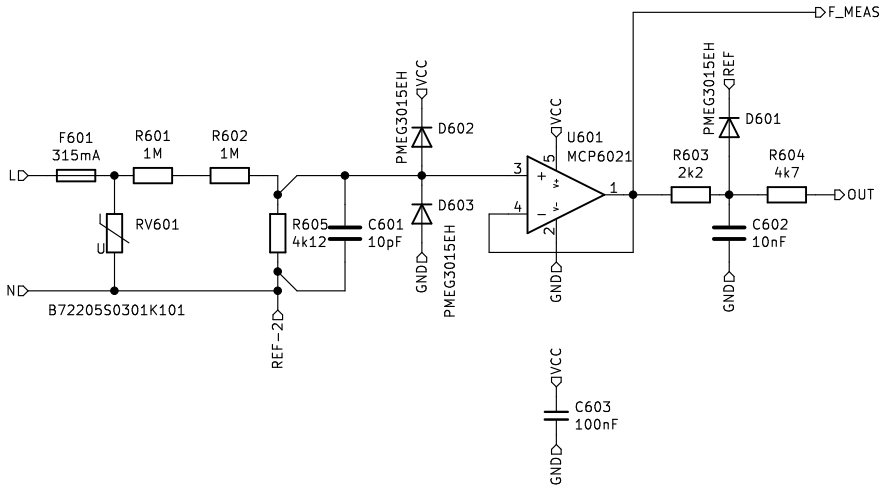




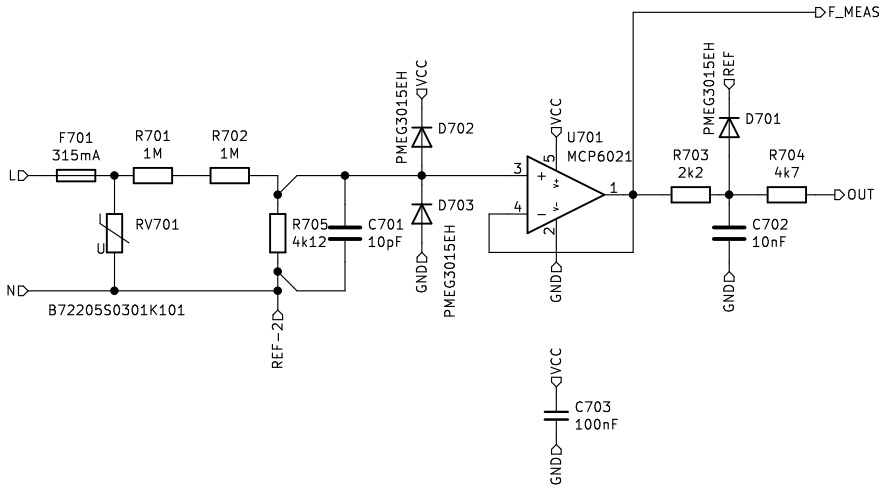
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Communication/ File: Communication.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 4/12



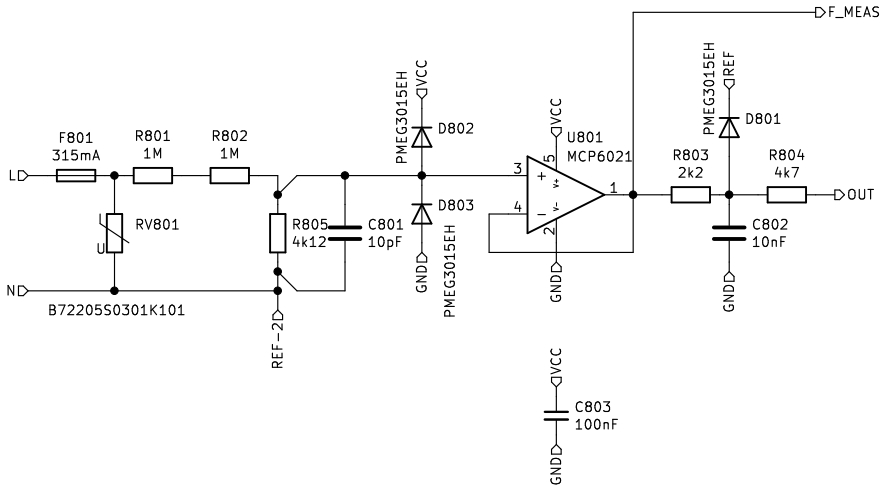
Jan Novotný MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /High voltage Input/ File: AnalogInHV.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 5/12



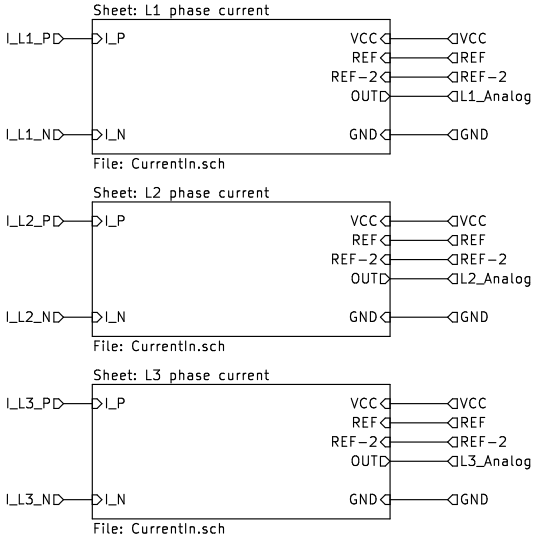
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /High voltage Input/L1 Phase Voltage/ File: PhaseInput.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 6/12



Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /High voltage Input/L2 Phase Voltage/ File: PhaseInput.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 7/12



Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /High voltage Input/L3 Phase Voltage/ File: PhaseInput.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 8/12



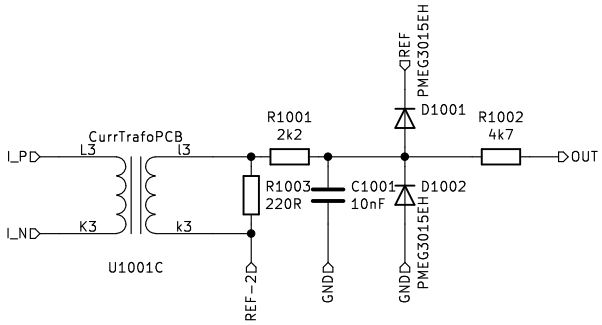
Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

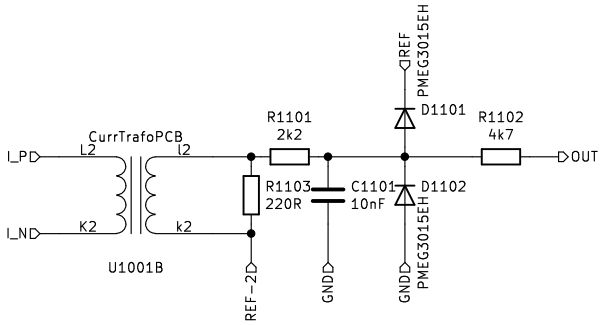
Sheet: /Current input/
File: AnalogCurr.sch

Title: Power analyzer

Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 9/12



Jan Novotný MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Current input/L1 phase current/ File: CurrentIn.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 10/12



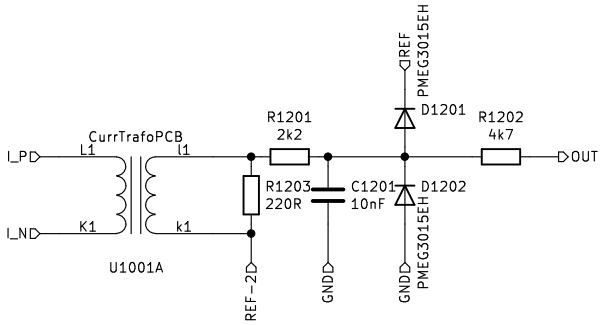
Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Current input/L2 phase current/
File: CurrentIn.sch

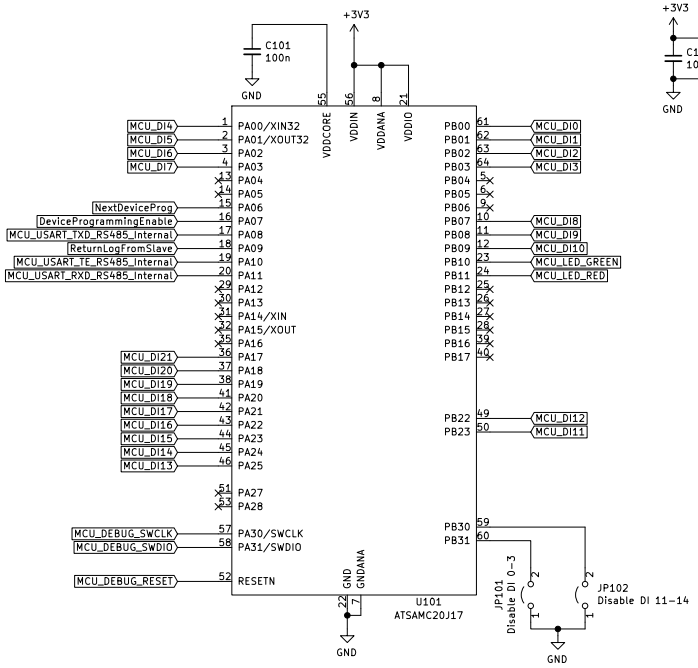
Title: Power analyzer

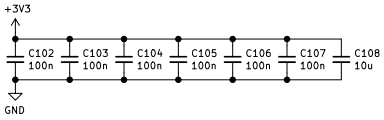
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 11/12

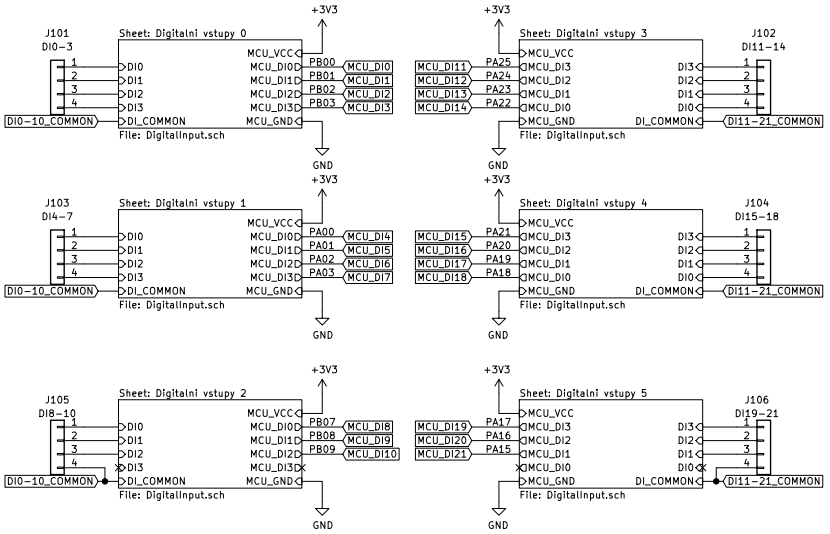


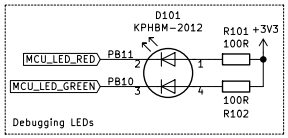
Jan Novotný MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Current input/L3 phase current/ File: CurrentIn.sch		
Title: Power analyzer		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (5.1.10)-1		Id: 12/12

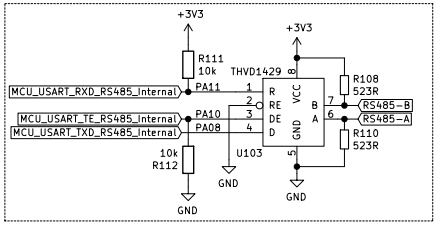
A.4 Digitální vstupy

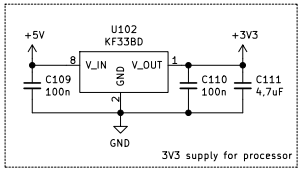


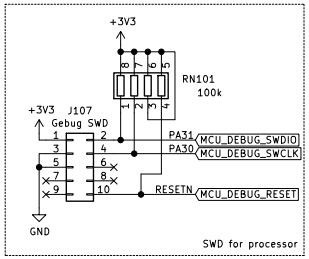


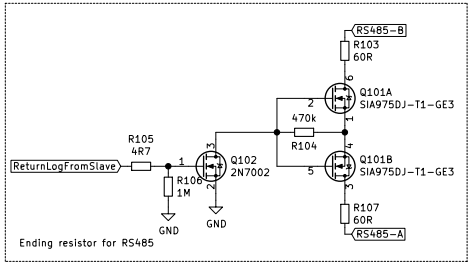


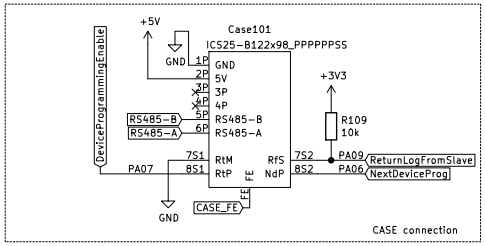












Jan Novotný

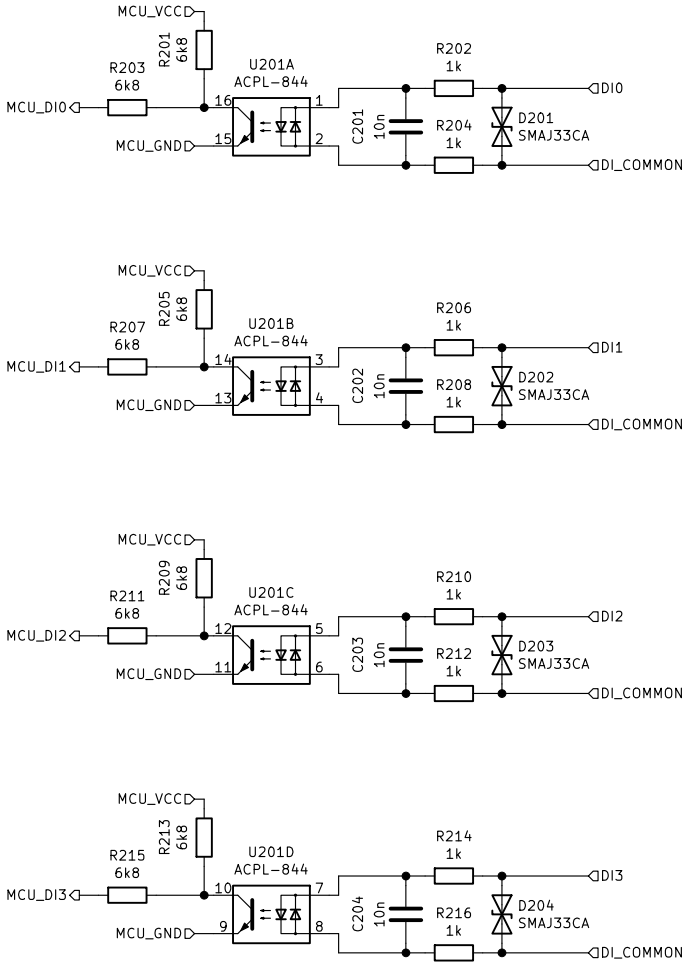
Řídící systém

ELZACO spol. s r.o.

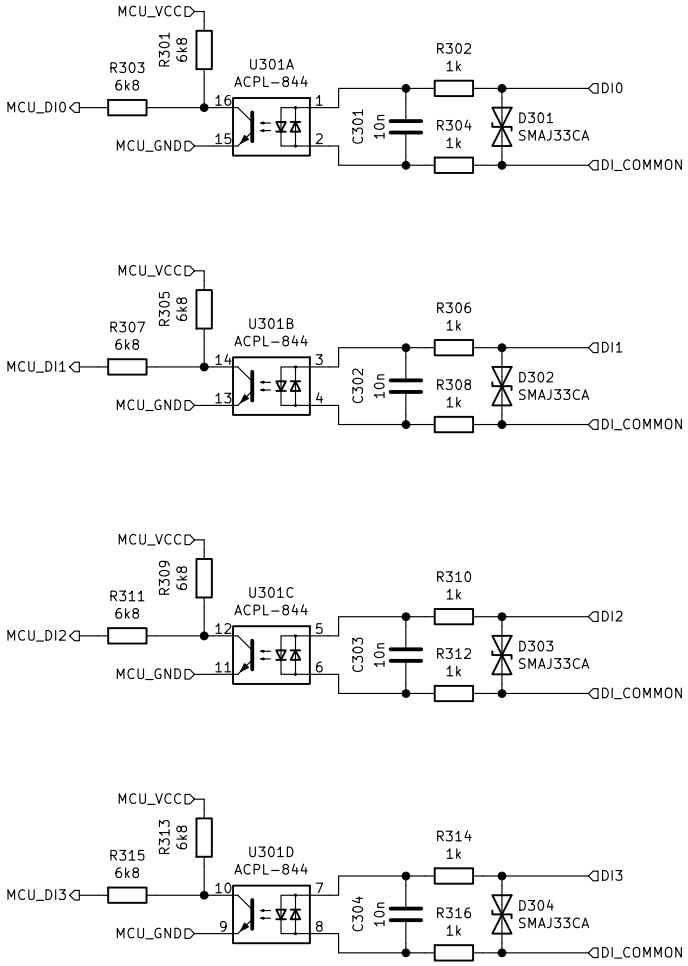
Sheet: /
File: DigitalniVstupy.sch

Title: Digitální vstupy

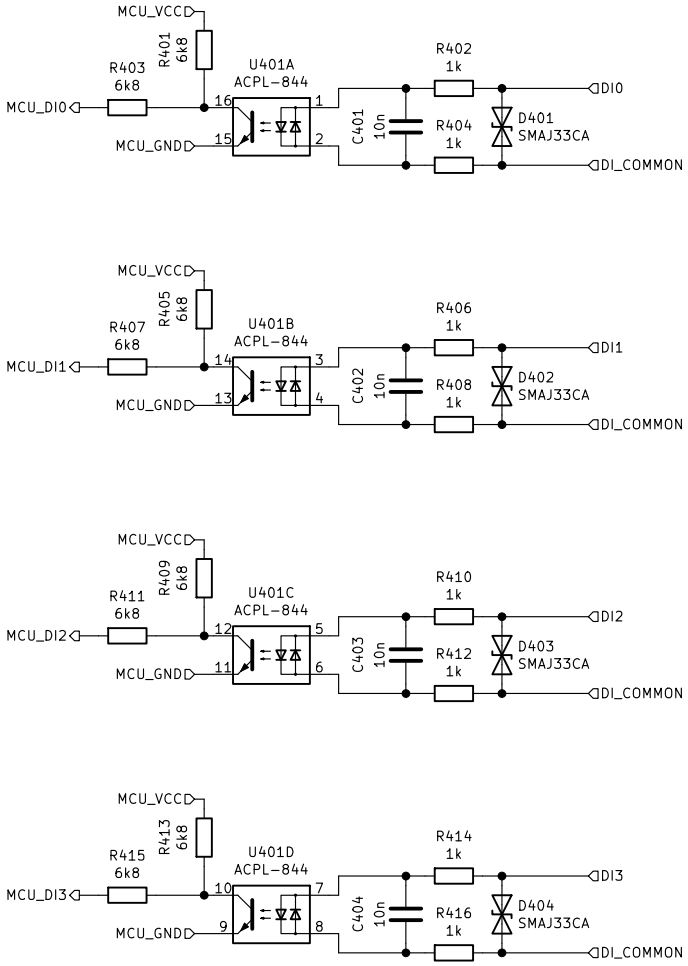
Size: A3	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 1/7



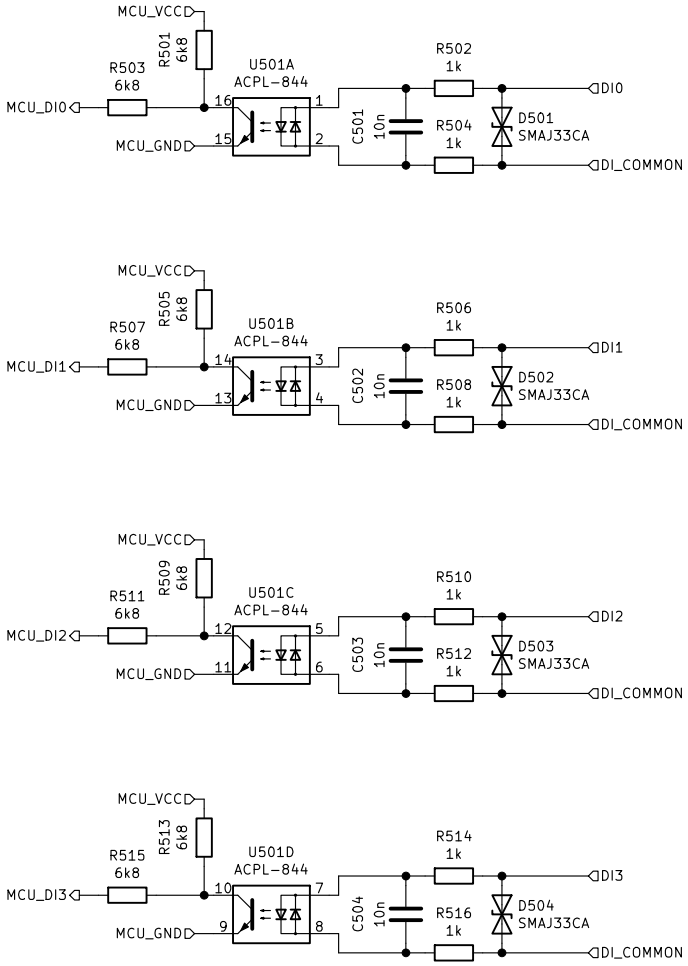
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 0/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 2/7



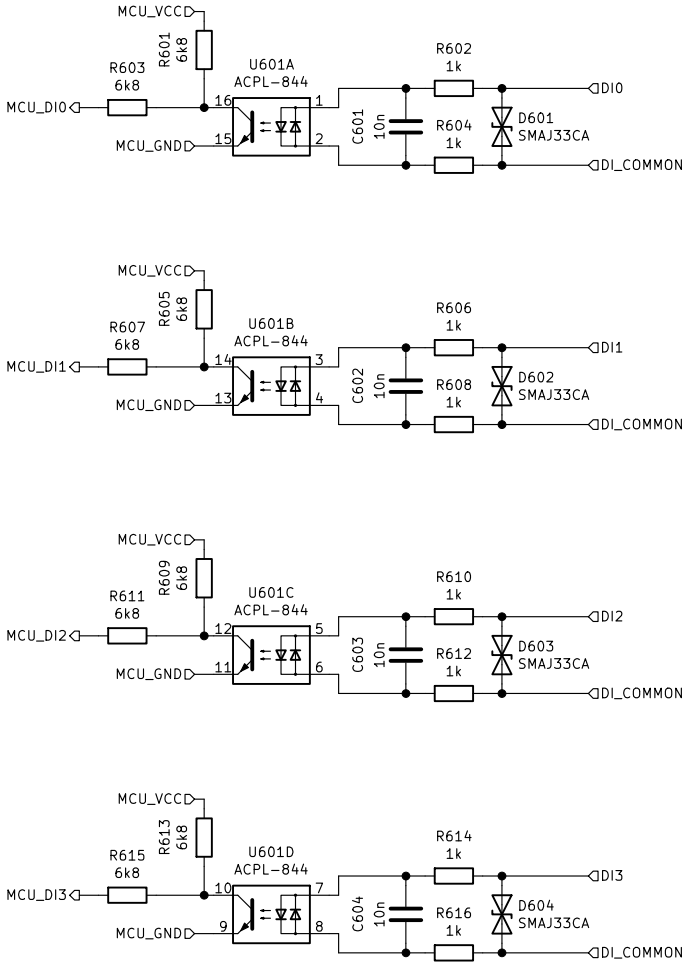
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 1/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 3/7



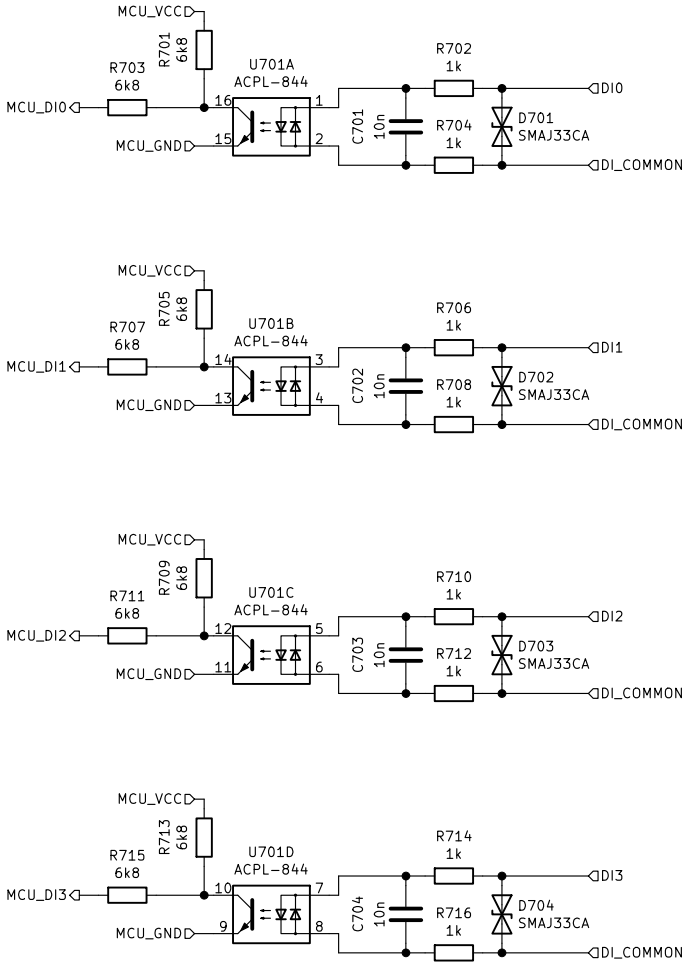
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 2/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 4/7



Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 3/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 5/7

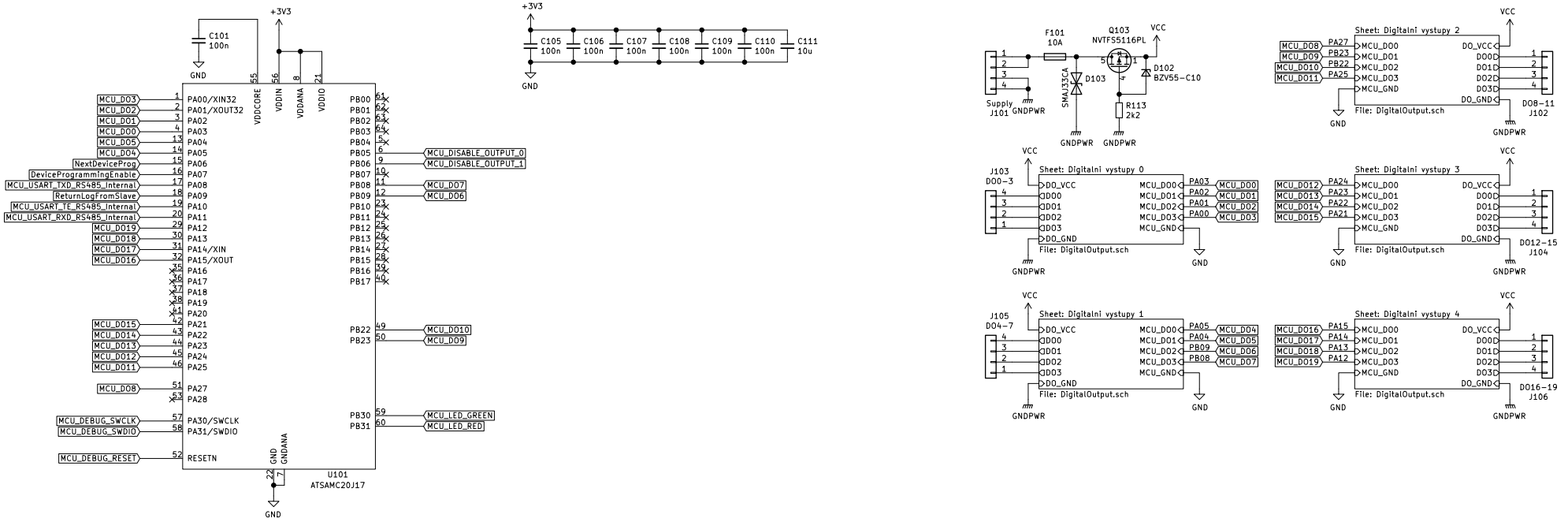


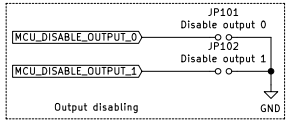
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 4/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 6/7

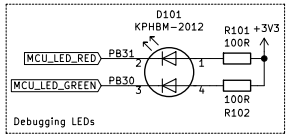


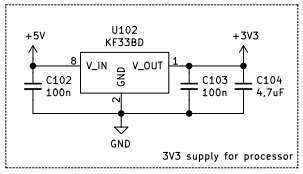
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vstupy 5/ File: DigitalInput.sch		
Title: Digitální vstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 7/7

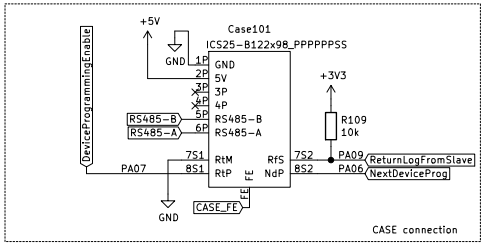
A.5 Digitální výstupy

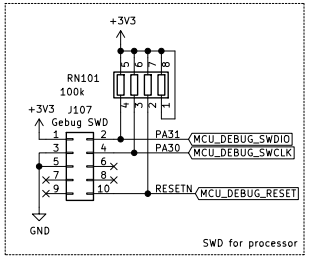


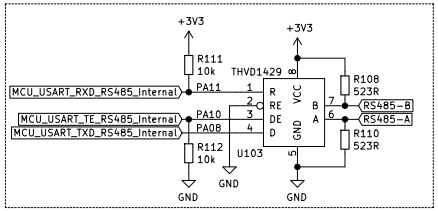


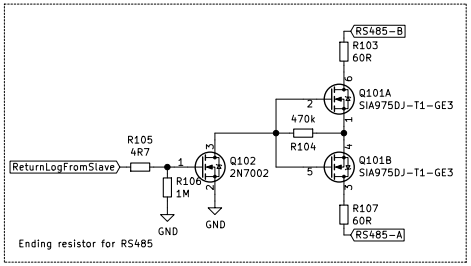


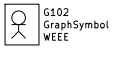












Jan Novotný

Řídící systém

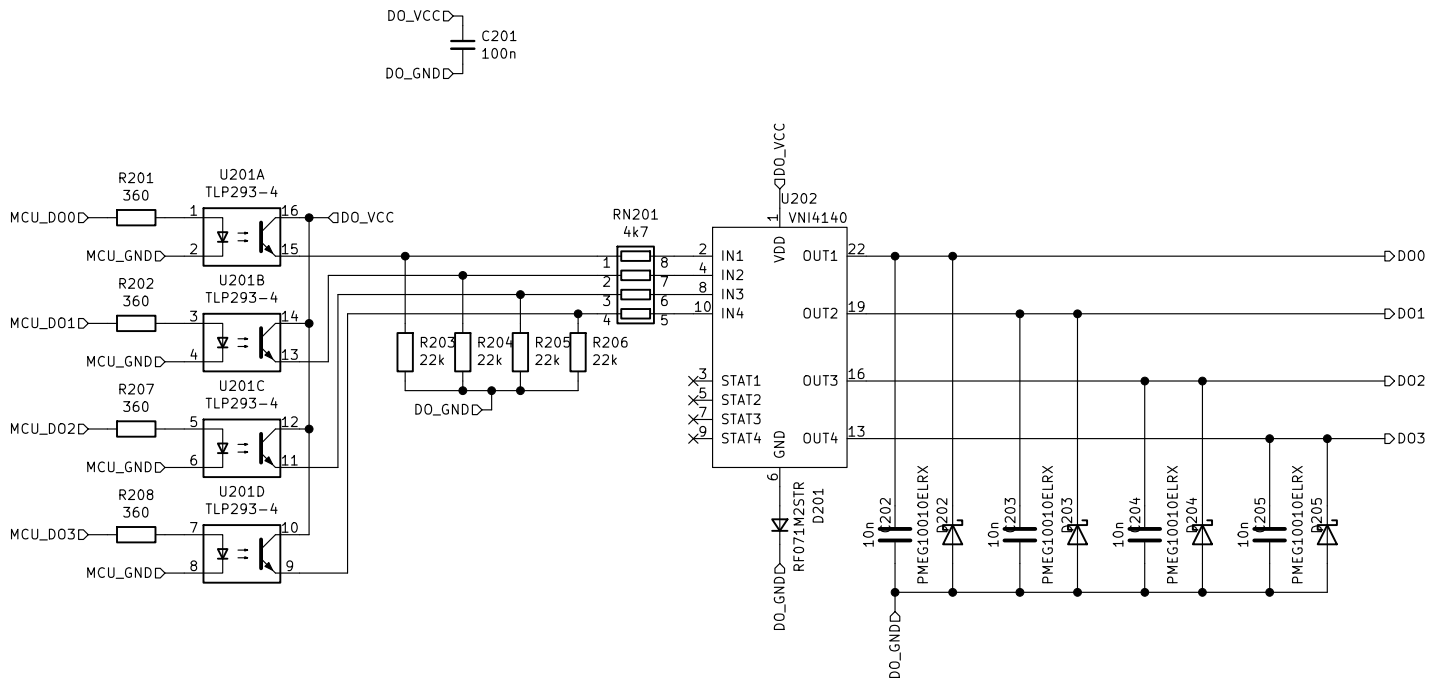
ELZACO spol. s r.o.

Sheet: /

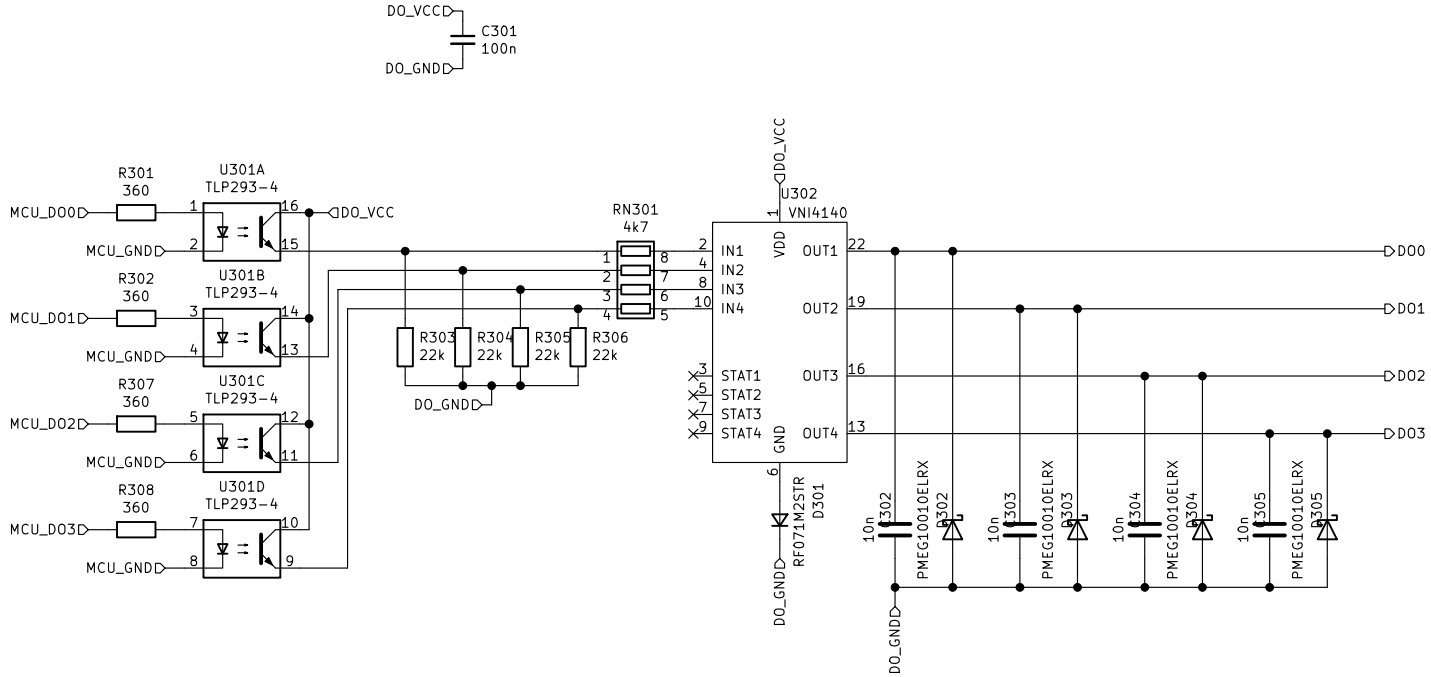
File: DigitalniVystupy.sch

Title: Digitální výstupy

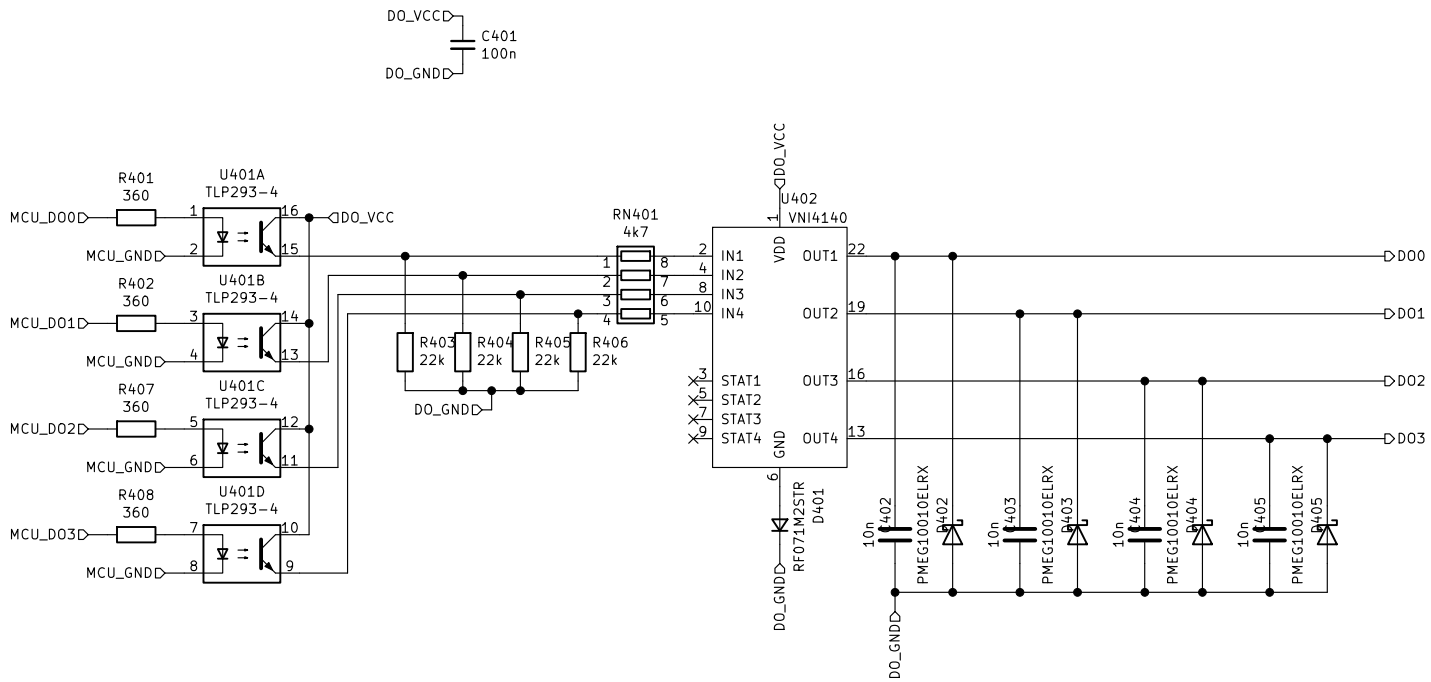
Size: A3	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 1/6



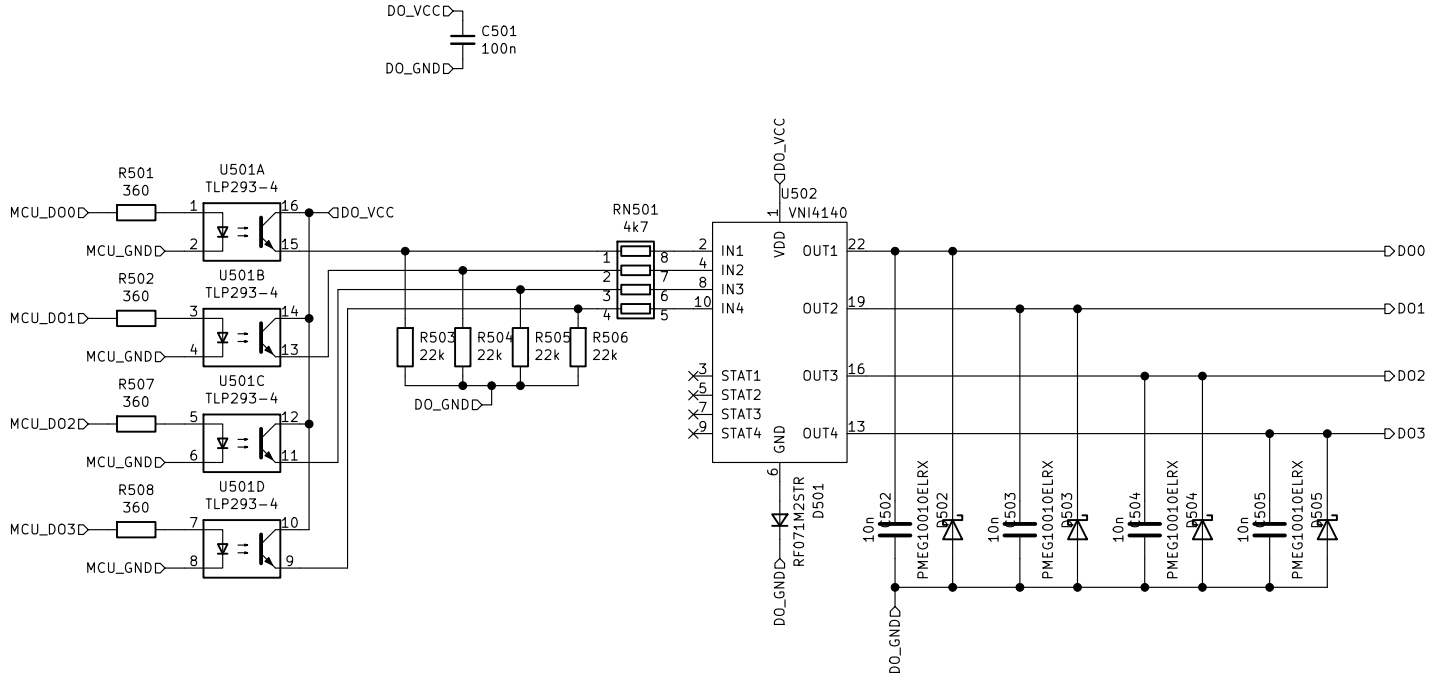
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy 0/ File: DigitalOutput.sch		
Title: Digitální výstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 2/6



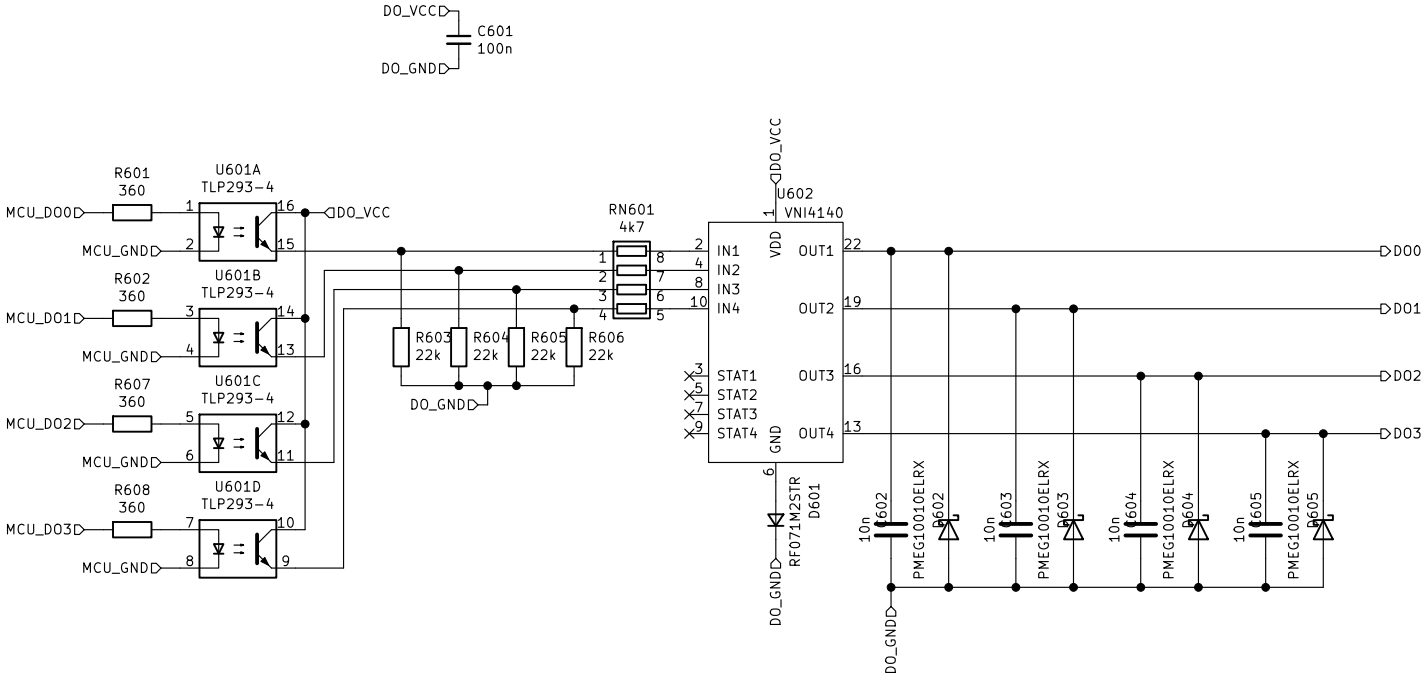
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy 1/ File: DigitalOutput.sch		
Title: Digitální výstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 3/6



Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy 2/ File: DigitalOutput.sch		
Title: Digitální výstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 4/6

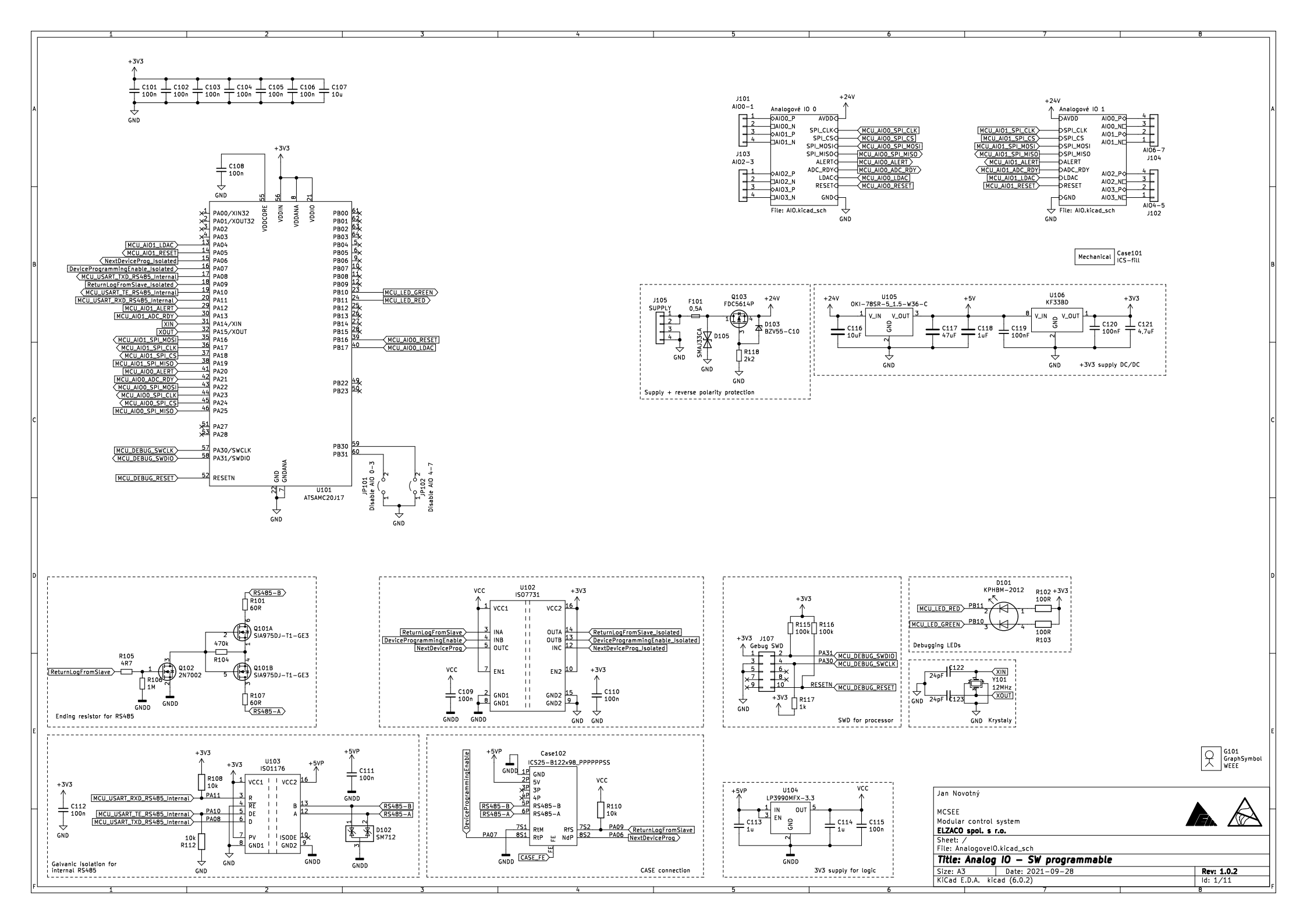


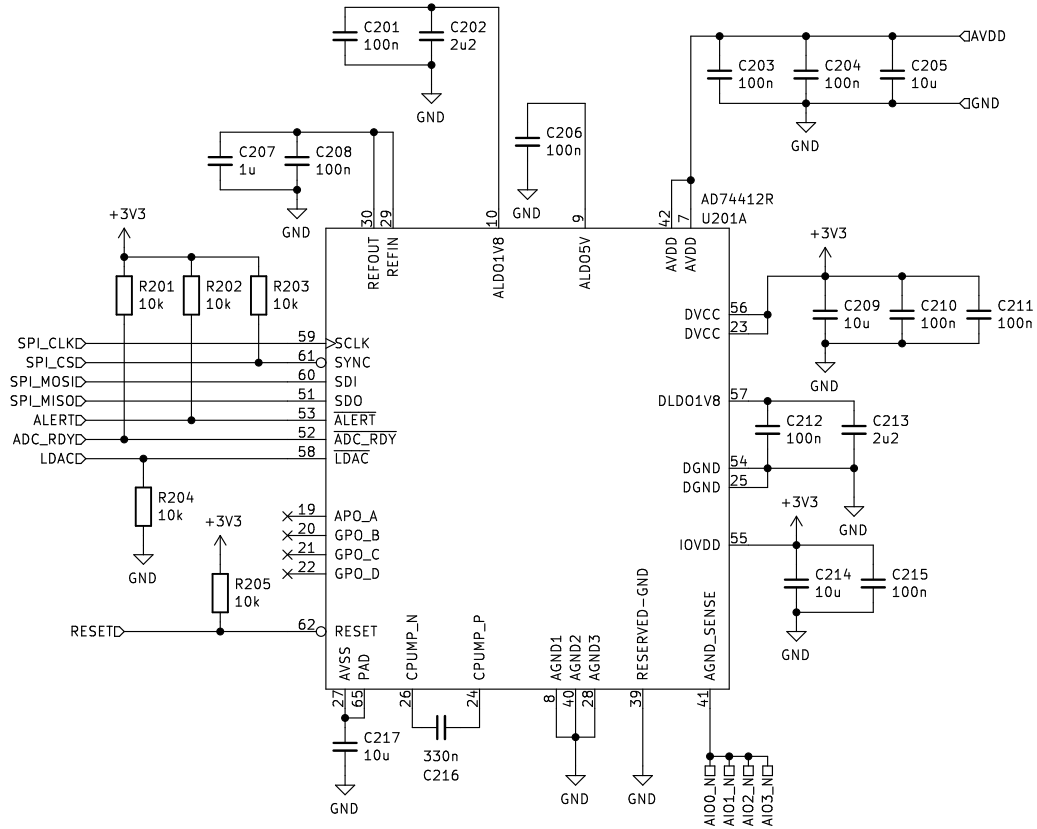
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy 3/ File: DigitalOutput.sch		
Title: Digitální výstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 5/6



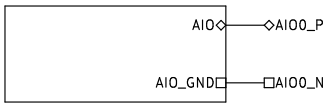
Jan Novotný		
Řídící systém **ELZACO spol. s r.o.**		
Sheet: /Digitalni vystupy 4/ File: DigitalOutput.sch		
Title: Digitální výstupy		
Size: A4	Date: 2020-08-18	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 6/6

A.6 Analogové vstupy a výstupy


+3V3
C101 100n
C102 100n
C103 100n
C104 100n
C105 100n
C106 100n
C107 10u
GND
C108 100n
U101 ATSAMC20J17
PA00/XIN32
PA01/XOUT32
PA02
PA03
PA04
PA05
PA06
PA07
PA08
PA09
PA10
PA11
PA12
PA13
PA14/XIN
PA15/XOUT
PA16
PA17
PA18
PA19
PA20
PA21
PA22
PA23
PA24
PA25
PA27
PA28
PA30/SWCLK
PA31/SWDIO
RESETN
PB00
PB01
PB02
PB03
PB04
PB05
PB06
PB07
PB08
PB09
PB10
PB11
PB12
PB13
PB14
PB15
PB16
PB17
PB22
PB23
PB30
PB31
VDDCORE
VDDIN
VDDANA
VDDIO
GND
GNDANA
MCU_AIO1_LDAC
MCU_AIO1_RESET
NextDeviceProg_isolated
DeviceProgrammingEnable_Isolated
MCU_USART_TXD_RS485_Internal
ReturnLogFromSlave_Isolated
MCU_USART_TE_RS485_Internal
MCU_USART_RXD_RS485_Internal
MCU_AIO1_ALERT
MCU_AIO1_ADC_RDY
XIN
XOUT
MCU_AIO1_SPI_MOSI
MCU_AIO1_SPI_CLK
MCU_AIO1_SPI_CS
MCU_AIO1_SPI_MISO
MCU_AIO0_ALERT
MCU_AIO0_ADC_RDY
MCU_AIO0_SPI_MOSI
MCU_AIO0_SPI_CLK
MCU_AIO0_SPI_CS
MCU_AIO0_SPI_MISO
MCU_DEBUG_SWCLK
MCU_DEBUG_SWDIO
MCU_DEBUG_RESET
MCU_LED_GREEN
MCU_LED_RED
MCU_AIO0_RESET
MCU_AIO0_LDAC
JP101 Disable AIO 0-3
JP102 Disable AIO 4-7
J101 AIO0-1
Analogové IO 0
AIO0_P
AIO0_N
AIO1_P
AIO1_N
J103 AIO2-3
AIO2_P
AIO2_N
AIO3_P
AIO3_N
AVDD
SPI_CLK
SPI_CS
SPI_MOSI
SPI_MISO
ALERT
ADC_RDY
LDAC
RESET
MCU_AIO0_SPI_CLK
MCU_AIO0_SPI_CS
MCU_AIO0_SPI_MOSI
MCU_AIO0_SPI_MISO
MCU_AIO0_ALERT
MCU_AIO0_ADC_RDY
MCU_AIO0_LDAC
MCU_AIO0_RESET
File: AIO.kicad_sch
Analogové IO 1
MCU_AIO1_SPI_CLK
MCU_AIO1_SPI_CS
MCU_AIO1_SPI_MOSI
MCU_AIO1_SPI_MISO
MCU_AIO1_ALERT
MCU_AIO1_ADC_RDY
MCU_AIO1_LDAC
MCU_AIO1_RESET
AIO0_P
AIO0_N
AIO1_P
AIO1_N
AIO2_P
AIO2_N
AIO3_P
AIO3_N
AIO6-7 J104
AIO4-5 J102
Mechanical
Case101 ICS-fill
J105 SUPPLY
F101 0.5A
Q103 FDC5614P
D103 BZV55-C10
D105 SMAJ33CA
R118 2k2
Supply + reverse polarity protection
U105 OKI-78SR-5_1.5-W36-C
U106 KF33BD
C116 10uF
C117 47uF
C118 1uF
C119 100nF
C120 100nF
C121 4,7uF
+3V3 supply DC/DC
R101 60R
R107 60R
Q101A SIA975DJ-T1-GE3
Q101B SIA975DJ-T1-GE3
Q102 2N7002
R104 470k
R105 4R7
R106 1M
ReturnLogFromSlave
Ending resistor for RS485
U102 ISO7731
DeviceProgrammingEnable
NextDeviceProg
ReturnLogFromSlave_Isolated
DeviceProgrammingEnable_Isolated
NextDeviceProg_isolated
C109 100n
C110 100n
R115 100k
R116 100k
J107 Gebug SWD
R117 1k
SWD for processor
D101 KPHBM-2012
R102 100R
R103 100R
Debugging LEDs
C122 24pF
C123 24pF
Y101 12MHz
XIN
XOUT
Krystaly
U103 ISO1176
R108 10k
R112 10k
C111 100n
C112 100n
Galvanic isolation for internal RS485
Case102 ICS25-B122x98_PPPPPPSS
R110 10k
CASE_FE
CASE connection
U104 LP3990MFX-3.3
C113 1u
C114 1u
C115 100n
3V3 supply for logic
G101 GraphSymbol WEEE
Jan Novotný
MCSEE
Modular control system
ELZACO spol. s r.o.
Sheet: /
File: AnalogovеIO.kicad_sch
Title: Analog IO – SW programmable
Size: A3
Date: 2021-09-28
Rev: 1.0.2
KiCad E.D.A. kicad (6.0.2)
Id: 1/11

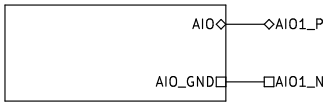


AIO 0



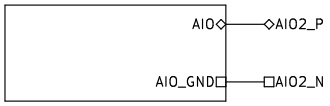
File: AIO_single.kicad_sch

AIO 1



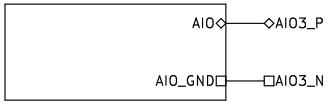
File: AIO_single.kicad_sch

AIO 2



File: AIO_single.kicad_sch

AIO 3



File: AIO_single.kicad_sch

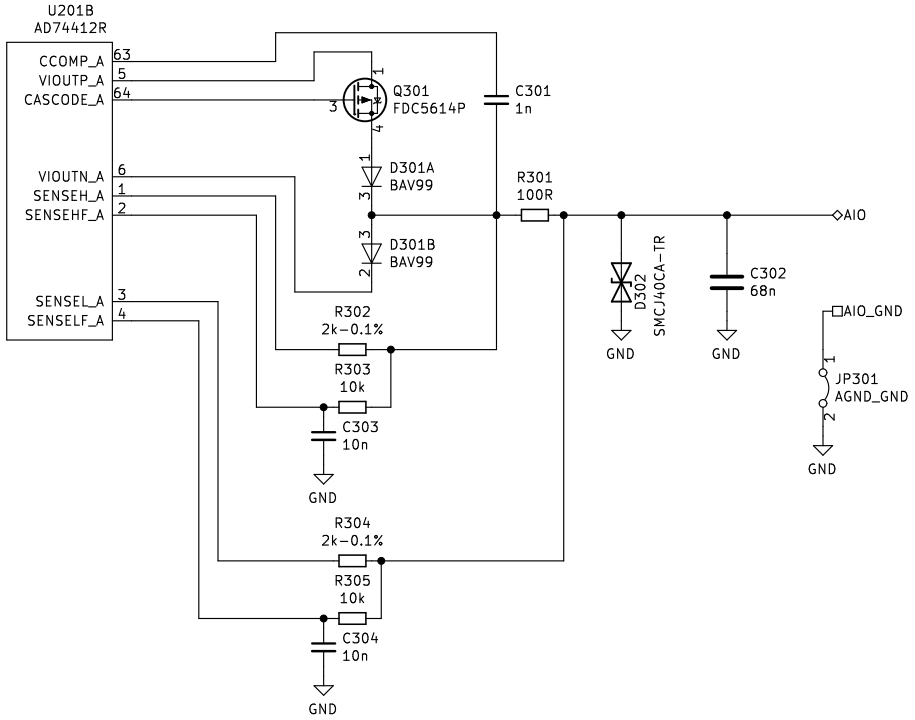
Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

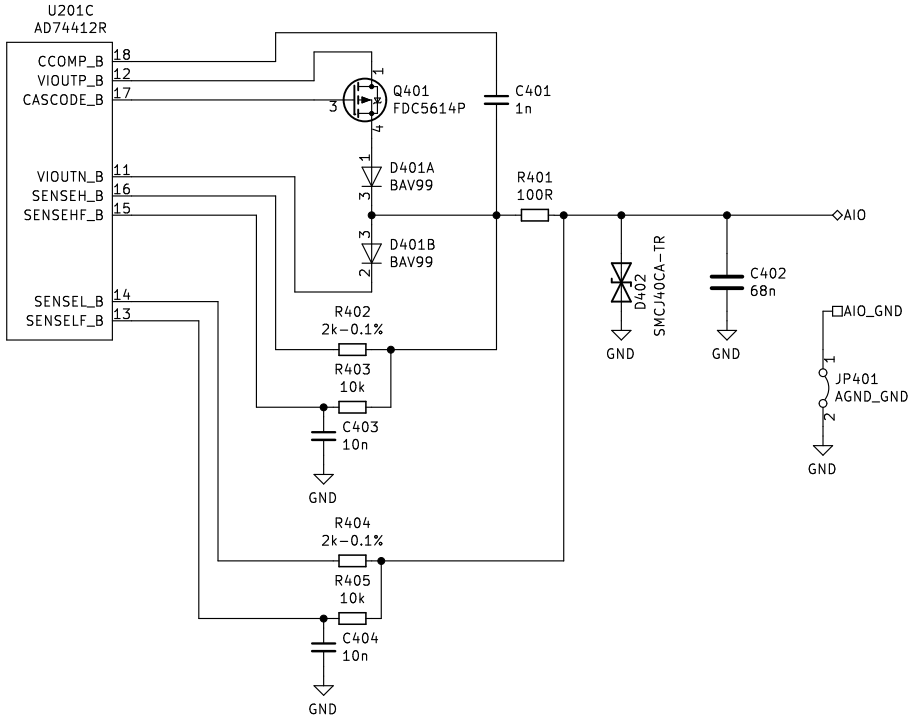
Sheet: /Analogové IO 0/
File: AIO.kicad_sch

Title: Analog IO – SW programmable

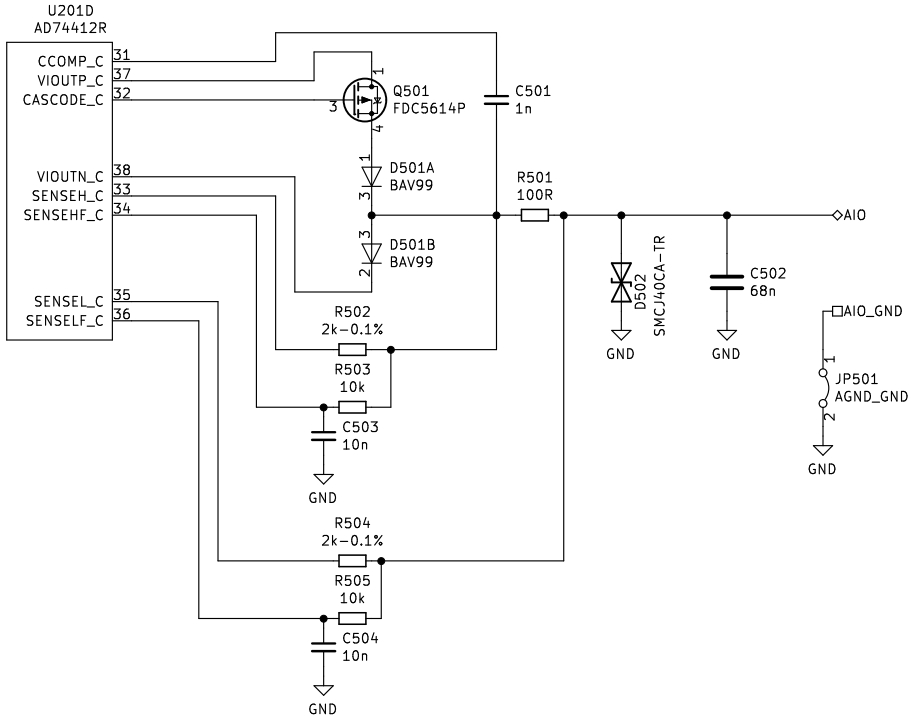
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 2/11



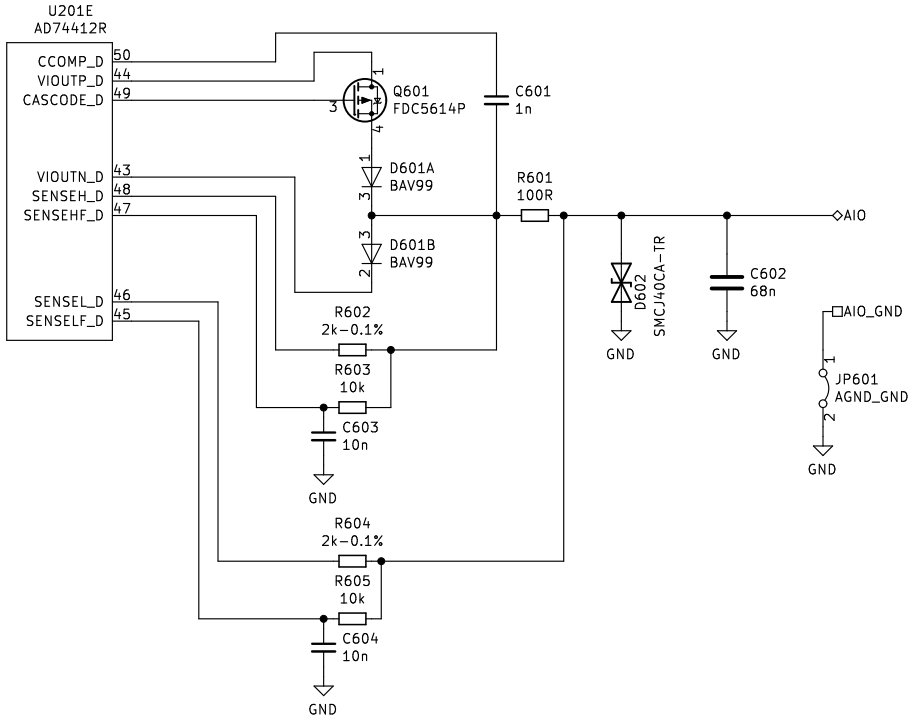
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 0/AIO 0/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 3/11



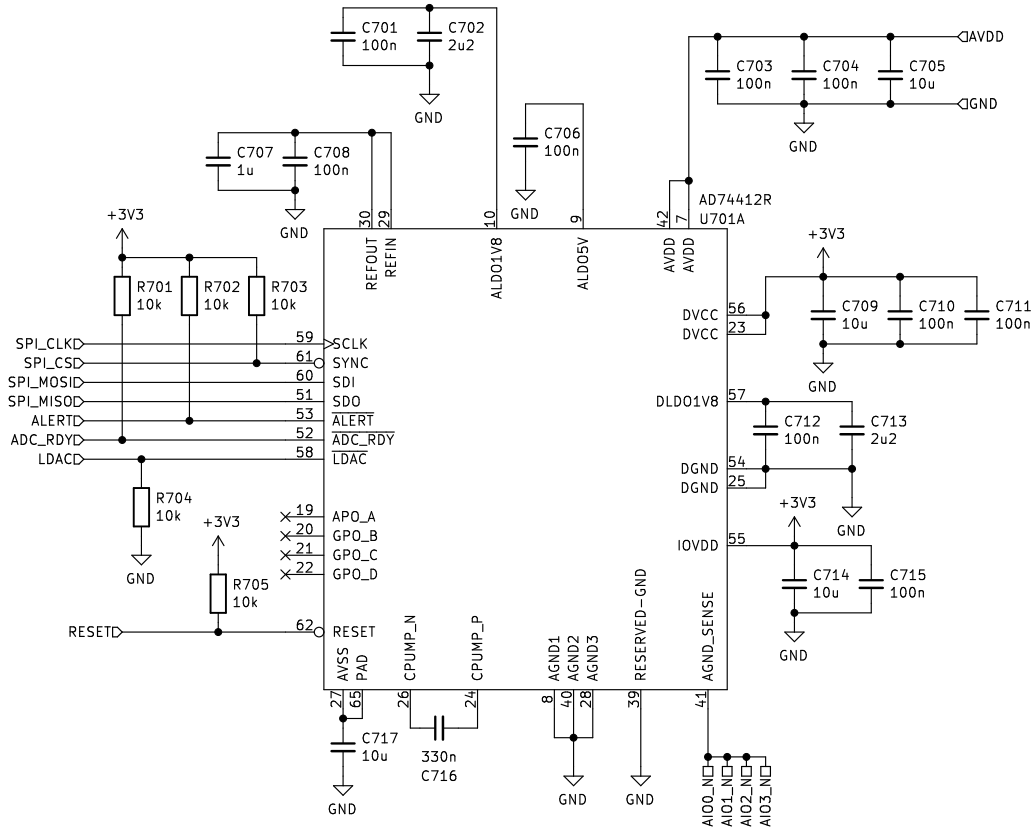
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 0/AIO 1/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 4/11



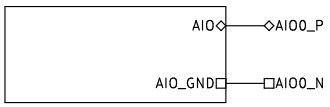
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 0/AIO 2/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 5/11



Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 0/AIO 3/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 6/11

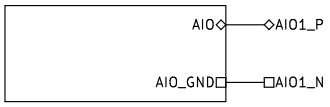


AIO 0



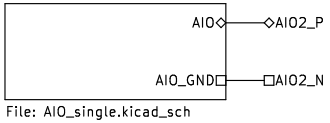
File: AIO_single.kicad_sch

AIO 1



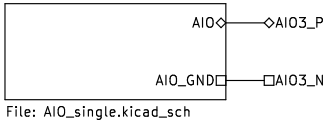
File: AIO_single.kicad_sch

AIO 2



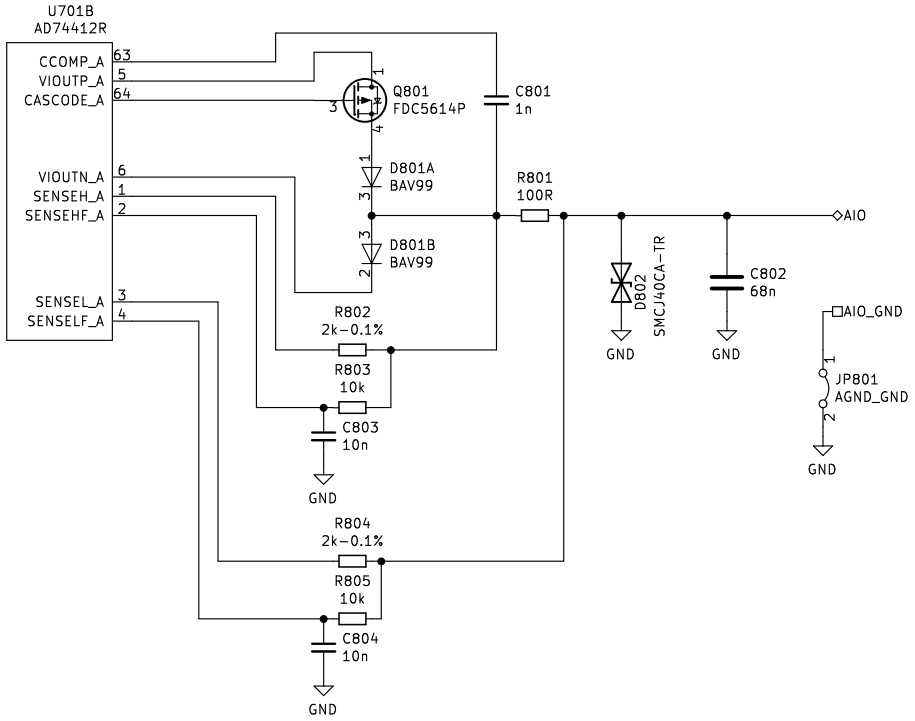
File: AIO_single.kicad_sch

AIO 3

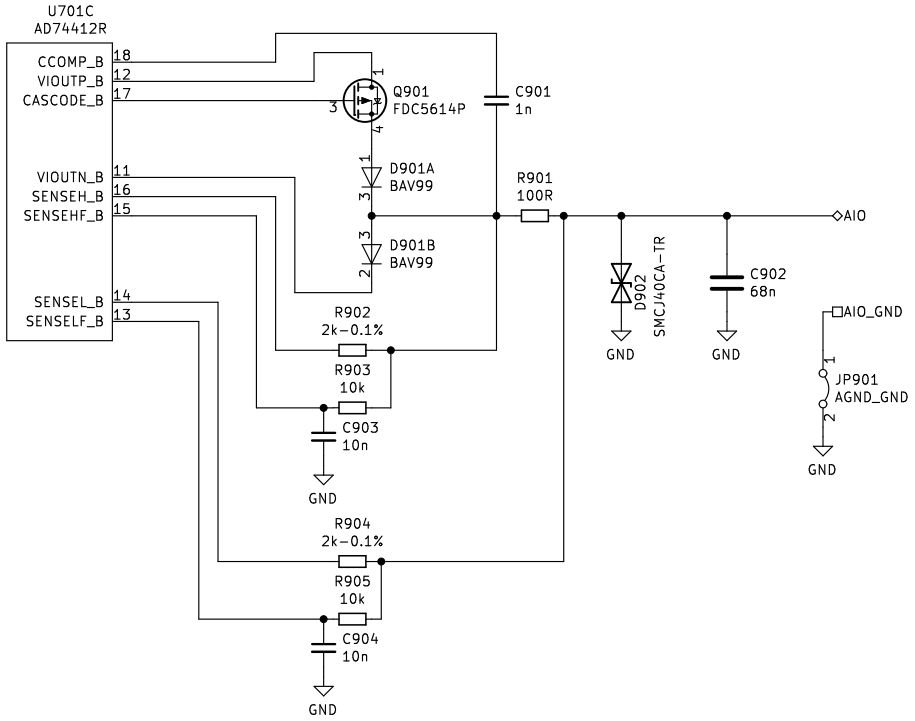


File: AIO_single.kicad_sch

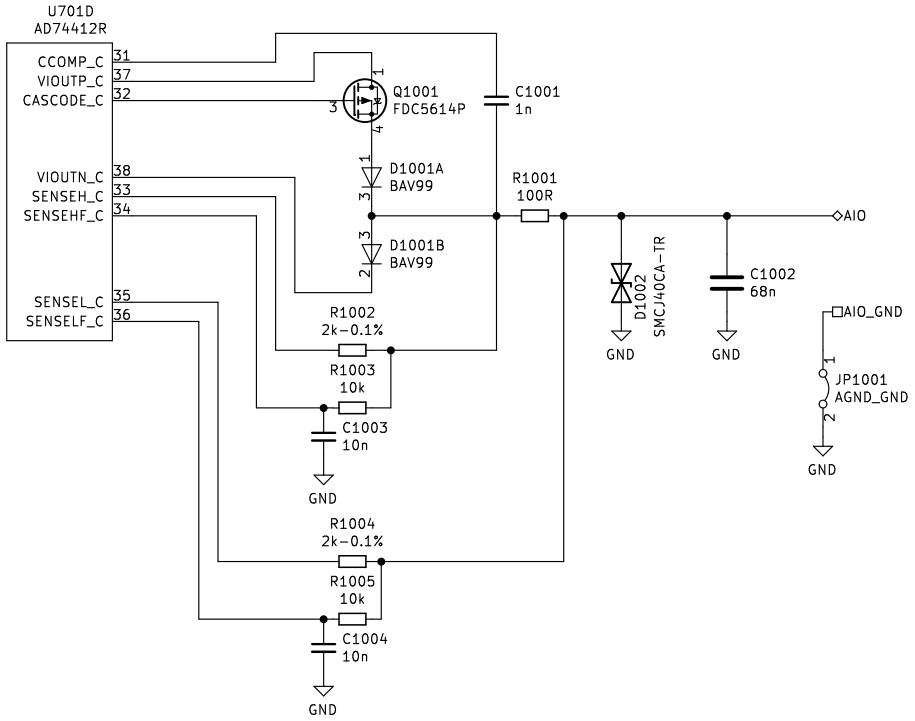
Jan Novotný MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 1/ File: AIO.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 7/11



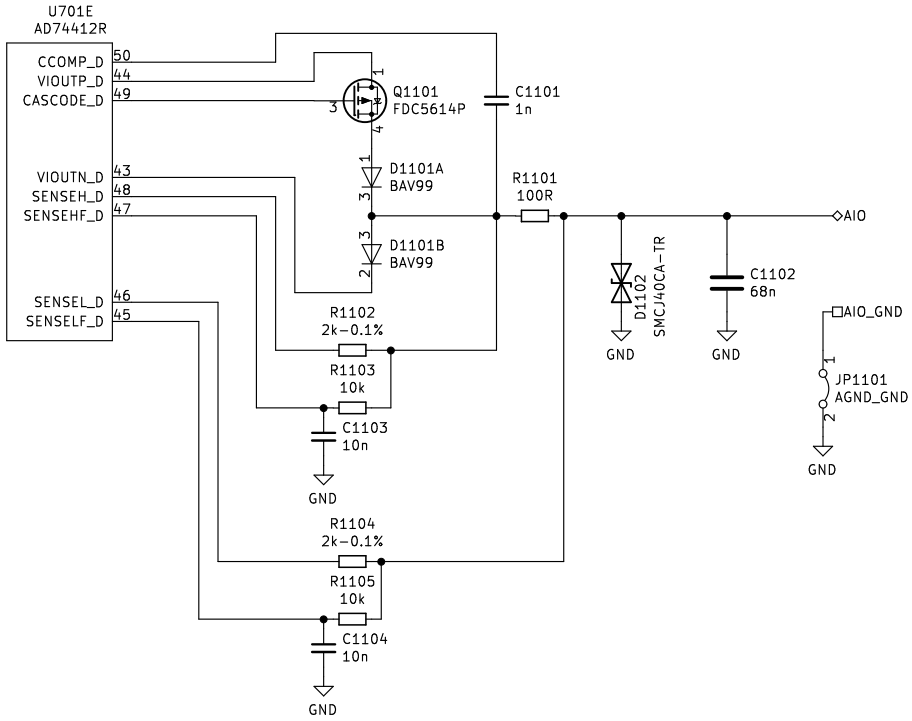
Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 1/AIO 0/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 8/11



Jan Novotný MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 1/AIO 1/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 9/11



Jan Novotný		
MCSEE Modular control system **ELZACO spol. s r.o.**		
Sheet: /Analogové IO 1/AIO 2/ File: AIO_single.kicad_sch		
Title: Analog IO – SW programmable		
Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 10/11



Jan Novotný

MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Analogové IO 1/AIO 3/
File: AIO_single.kicad_sch

Title: Analog IO – SW programmable

Size: A4	Date: 2021-09-28	**Rev: 1.0.2**
KiCad E.D.A. kicad (6.0.2)		Id: 11/11

A.7 Displej (HMI)

Sheet: Supply

File: Supply.sch

Sheet: MCU

File: Procesor.sch

Sheet: Communication

File: Communication.sch

RS485-B
RS485-A
RS485
J101

+24V
J102
PowerConn
GNDPWR

+3V3 +3.3VA
Sheet: Vstupy pro termočlánky
VDD_Isolated
VDD
MCU_SPI_CLK_TH PC24 MCU_SPI_CLK
MCU_SPI_MOSI_TH PC27 MCU_SPI_MOSI
MCU_SPI_MISO_TH PC26 MCU_SPI_MISO
MCU_SPI_CS_TH0 PC25 MCU_SPI_CS-T0
MCU_SPI_CS_TH1 PC28 MCU_SPI_CS-T1
MCU_SPI_CS_TH2 PC29 MCU_SPI_CS-T2
MCU_SPI_CS_TH3 PC30 MCU_SPI_CS-T3
GND
GND_Isolated
Analog_A0_IN_P
Analog_A0_IN_N
Thermo_T0_IN_N
Thermo_T0_IN_P
Thermocoupler & AI 0
J103
Analog_A1_IN_P
Analog_A1_IN_N
Thermo_T1_IN_N
Thermo_T1_IN_P
Thermocoupler & AI 1
J105
Analog_A2_IN_P
Analog_A2_IN_N
Thermo_T2_IN_N
Thermo_T2_IN_P
Thermocoupler & AI 2
J107
Analog_A3_IN_P
Analog_A3_IN_N
Thermo_T3_IN_N
Thermo_T3_IN_P
Thermocoupler & AI 3
J109
GND GNDA
File: Thermocouplers.sch

+3V3
Sheet: Digitální vstupy 0-3
MCU_DI0 PC2
MCU_DI1 PC3
MCU_DI2 PC4
MCU_DI3 PA16
MCU_VCC
MCU_DI0
MCU_DI1
MCU_DI2
MCU_DI3
MCU_GND
DI0
DI1
DI2
DI3
DI_COMMON
J113
DI0-3
GND
File: DigitalInput.sch

+3V3
Sheet: Digitální vstupy 4-7
MCU_DI4 PC7
MCU_DI5 PA15
MCU_DI6 PD26
MCU_DI7 PC6
J115
DI4-7
GND
File: DigitalInput.sch

+3V3
Sheet: Digitální vstupy 8-11
MCU_DI8 PA24
MCU_DI9 PC5
MCU_DI10 PA25
MCU_DI11 PA26
J117
DI8-11
GND
File: DigitalInput.sch

J118
DI GND

Sheet: Digitální výstupy 0-3
MCU_DO0 PA19
MCU_DO1 PA20
MCU_DO2 PC15
MCU_DO3 PC12
MCU_GND
DO_VCC
DO0
DO1
DO2
DO3
DO_GND
DO0-3
J111
File: DigitalOutput.sch

Sheet: Digitální výstupy 4-7
MCU_DO4 PC0
MCU_DO5 PE3
MCU_DO6 PE2
MCU_DO7 PE1
DO4-7
J114
File: DigitalOutput.sch

Sheet: Digitální výstupy 8-11
MCU_DO8 PA31
MCU_DO9 PC19
MCU_DO10 PA30
MCU_DO11 PA6
DO8-11
J116
File: DigitalOutput.sch

GND

Q101
NVTFS5116PL
F101
10A
J112
DO_Supply
BZV55-C10
D101
SMAJ33CA
D102
R101
2k2

+3V3 +3.3VP
Sheet: Rychlé digitální výstupy 0-3
VDD_Isolated
VDD
PWMC1-PWMH0 + TCLK5 MCU_FDO0 PD1
PWMC1-PWMH1 + TCLK2 MCU_FDO1 PD3
PWMC1-PWMH2 MCU_FDO2 PD5
PWMC1-PWMH3 MCU_FDO3 PD7
MCU_FDO_0
MCU_FDO_1
MCU_FDO_2
MCU_FDO_3
GND
GND_Isolated
FDO0
FDO1
FDO2
FDO3
FDO0-3
J119
GND GNDD
File: FastDigitalOutput.sch

H101 M4 Mounting Hole
H102 M4 Mounting Hole
H103 M4 Mounting Hole
H104 M4 Mounting Hole
H105 M4 Mounting Hole
H106 M4 Mounting Hole

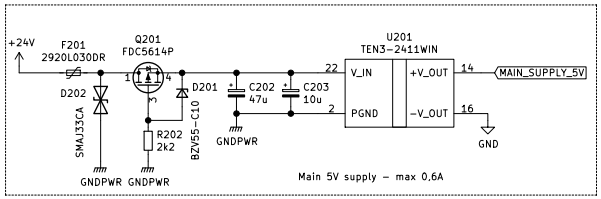
G102
GraphSymbol
WEEEE

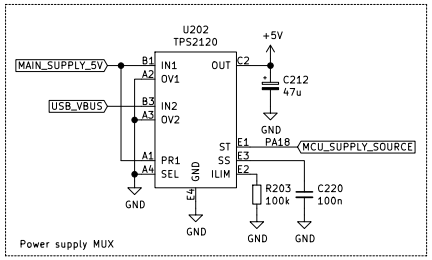
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

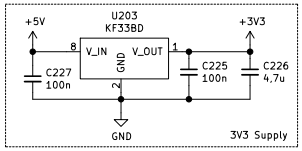
Sheet: /
File: Displej.sch

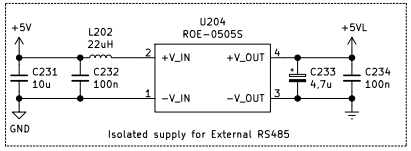
Title: Display with periphery

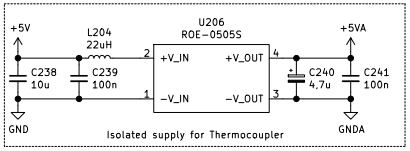
Size: A3	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 1/16

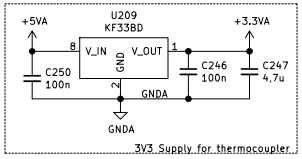


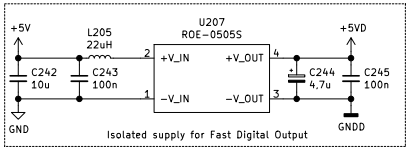


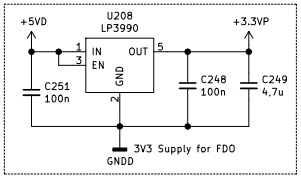


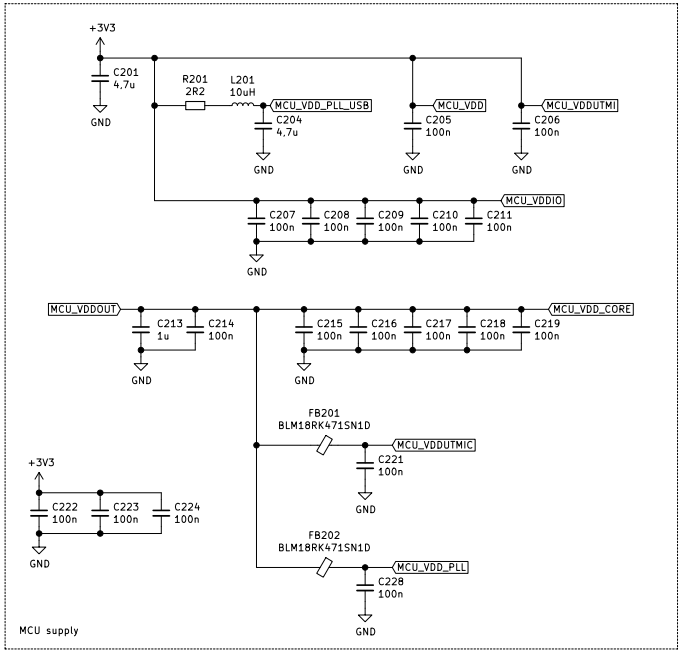


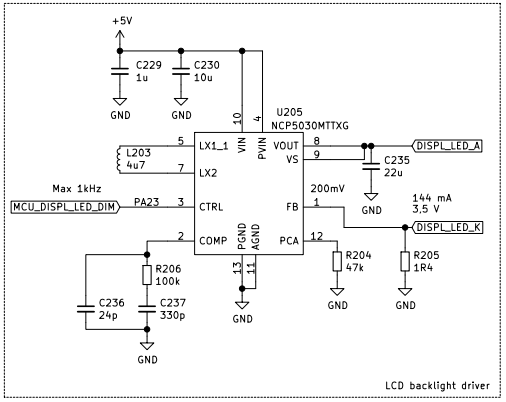




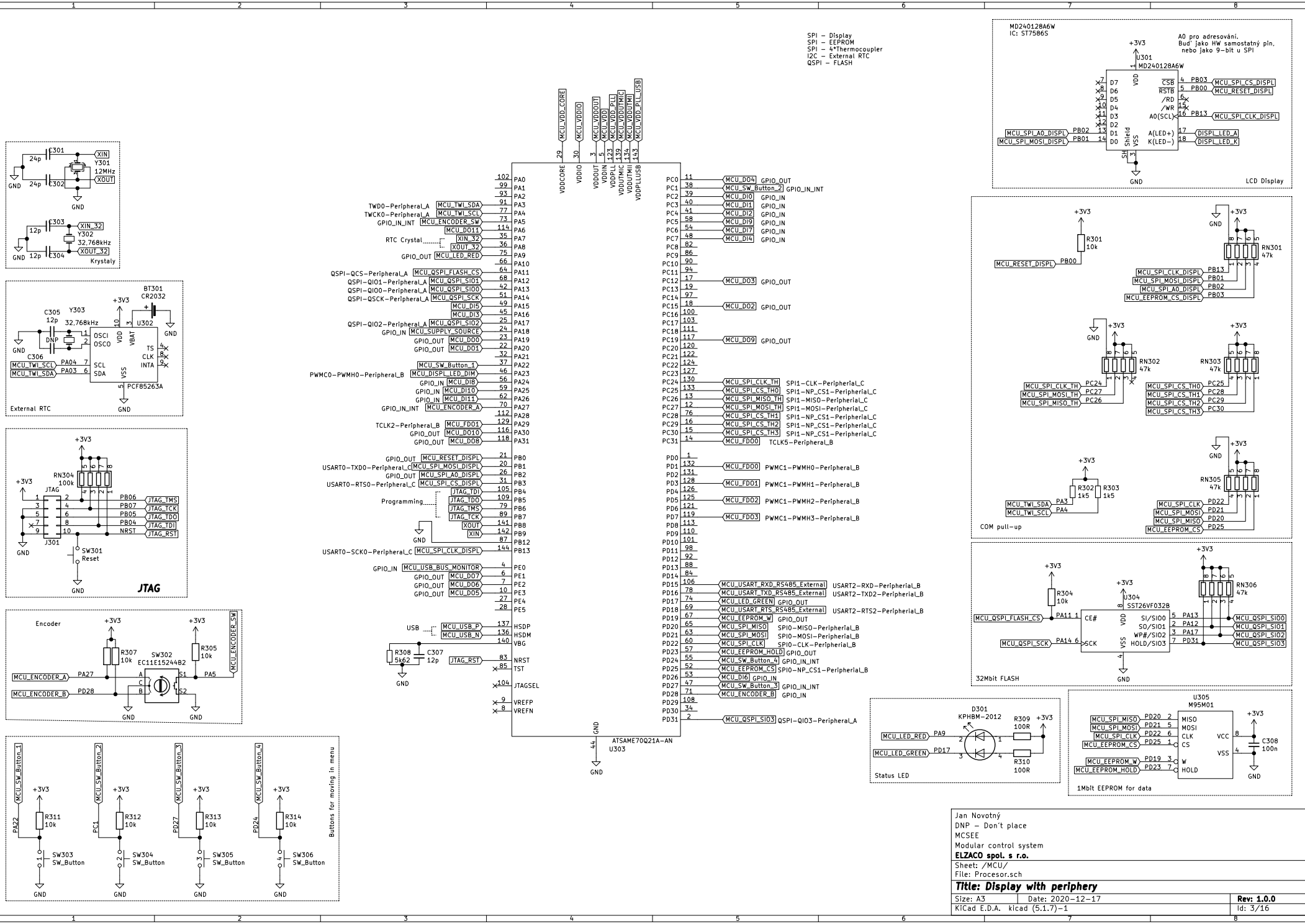








Jan Novotný		
DNP – Don't place		
MCSEE		
Modular control system		
ELZACO spol. s r.o.		
Sheet: /Supply/		
File: Supply.sch		
Title: Display with periphery		
Size: A3	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 2/16

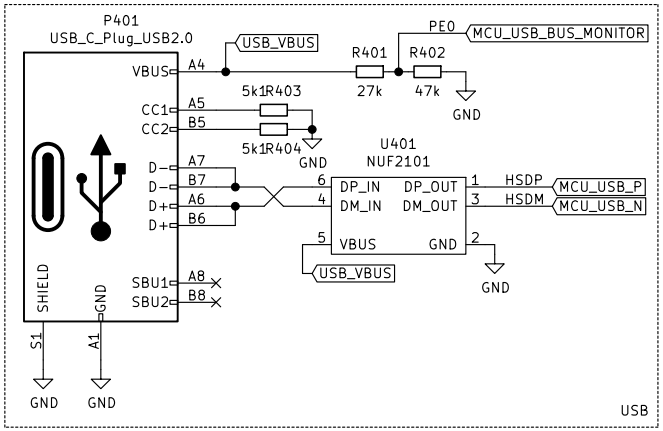


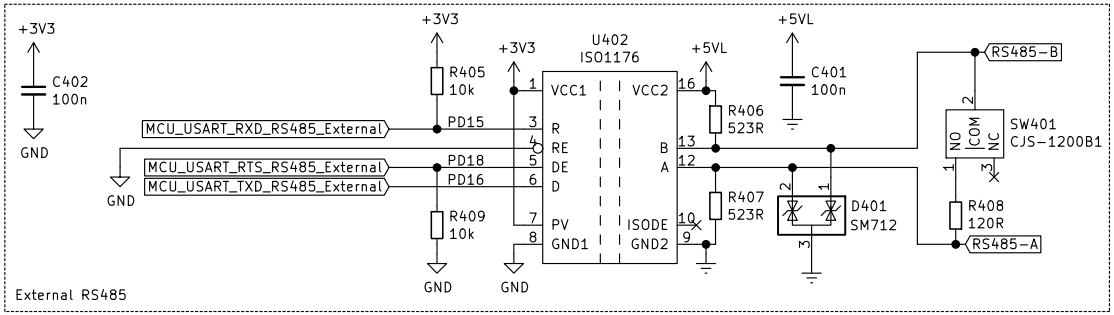
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /MCU/
File: Procesor.sch

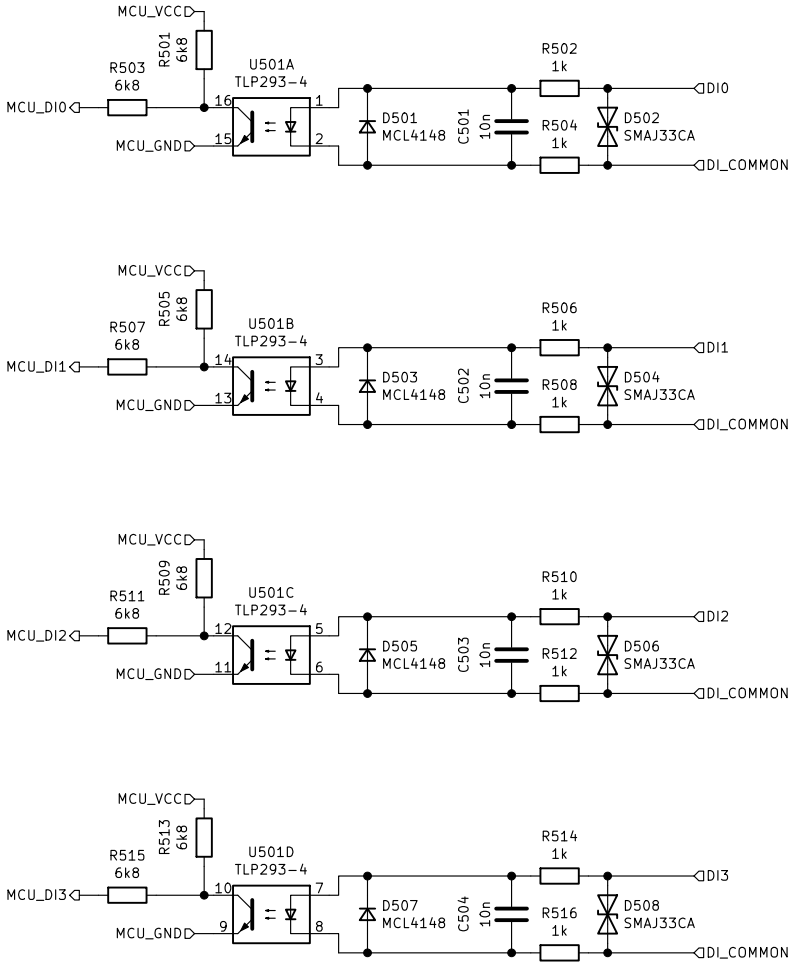
Title: Display with periphery

Size: A3	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 3/16





Jan Novotný	
DNP – Don't place	
MCSEE	
Modular control system	
ELZACO spol. s r.o.	
Sheet: /Communication/	
File: Communication.sch	
Title: Display with periphery	
Size: A4	Date: 2020-12-17 **Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1	Id: 4/16

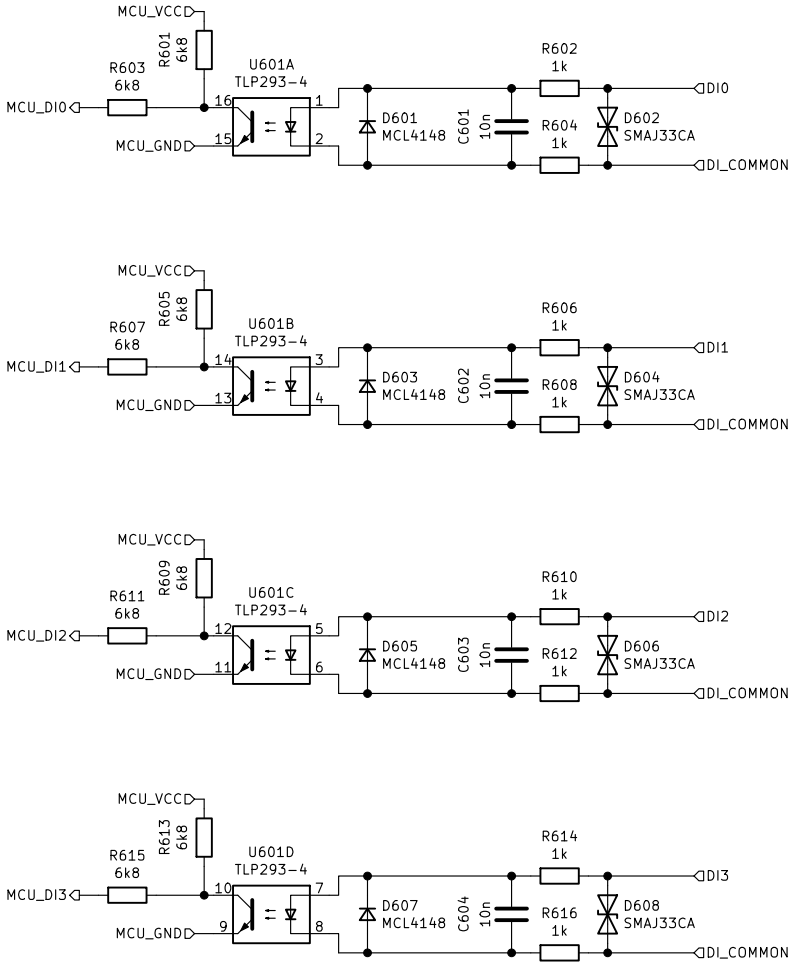


Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Digitální vstupy 0–3/
File: DigitalInput.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 5/16

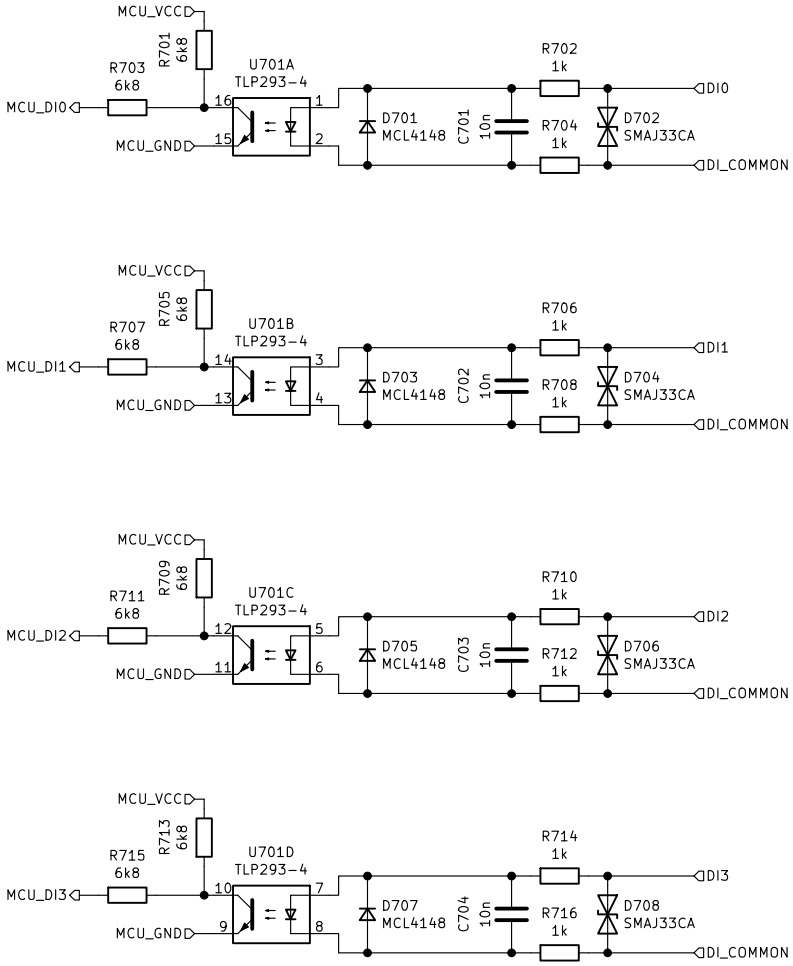


Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

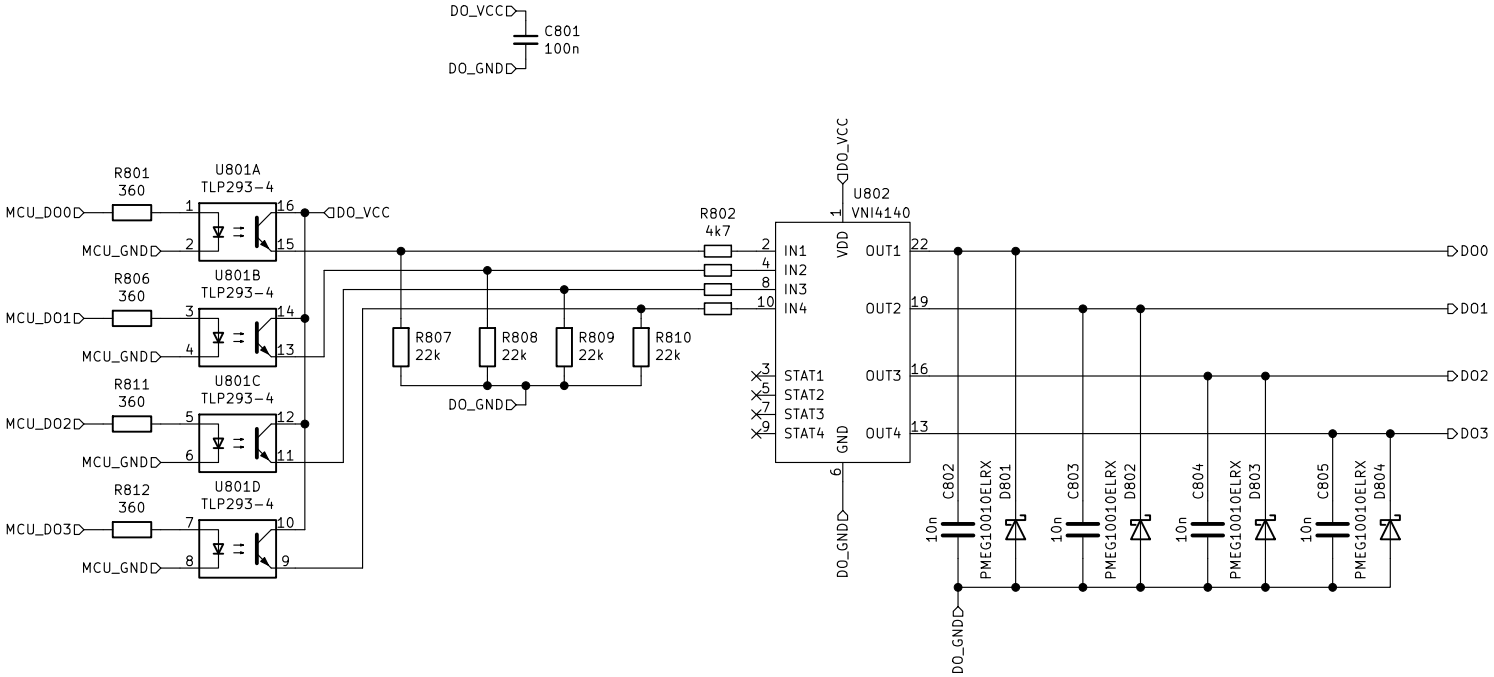
Sheet: /Digitální vstupy 4–7/
File: DigitalInput.sch

Title: Display with periphery

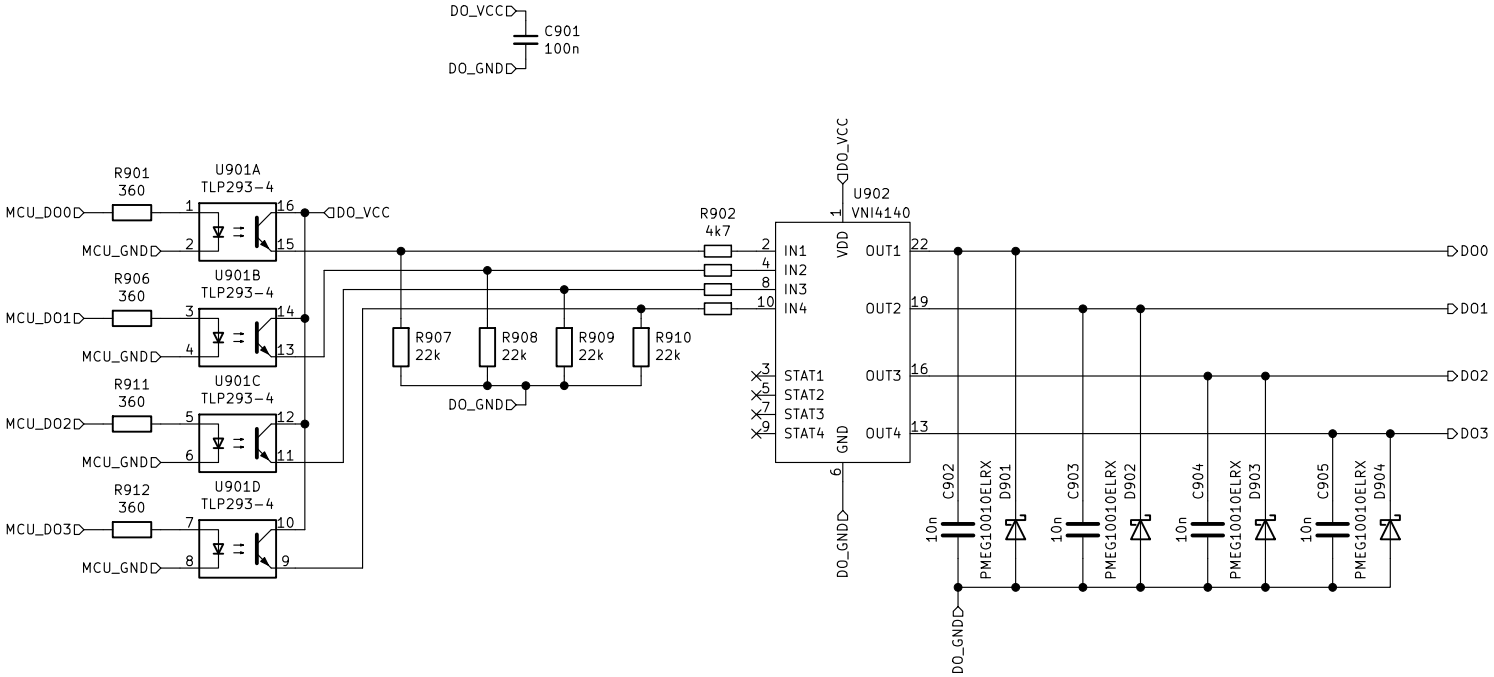
Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 6/16



Jan Novotný		
DNP – Don't place		
MCSEE		
Modular control system		
ELZACO spol. s r.o.		
Sheet: /Digitální vstupy 8–11/		
File: DigitalInput.sch		
Title: Display with periphery		
Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 7/16



Jan Novotný		
DNP – Don't place		
MCSEE		
Modular control system		
ELZACO spol. s r.o.		
Sheet: /Digitální výstupy 0–3/		
File: DigitalOutput.sch		
Title: Display with periphery		
Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 8/16

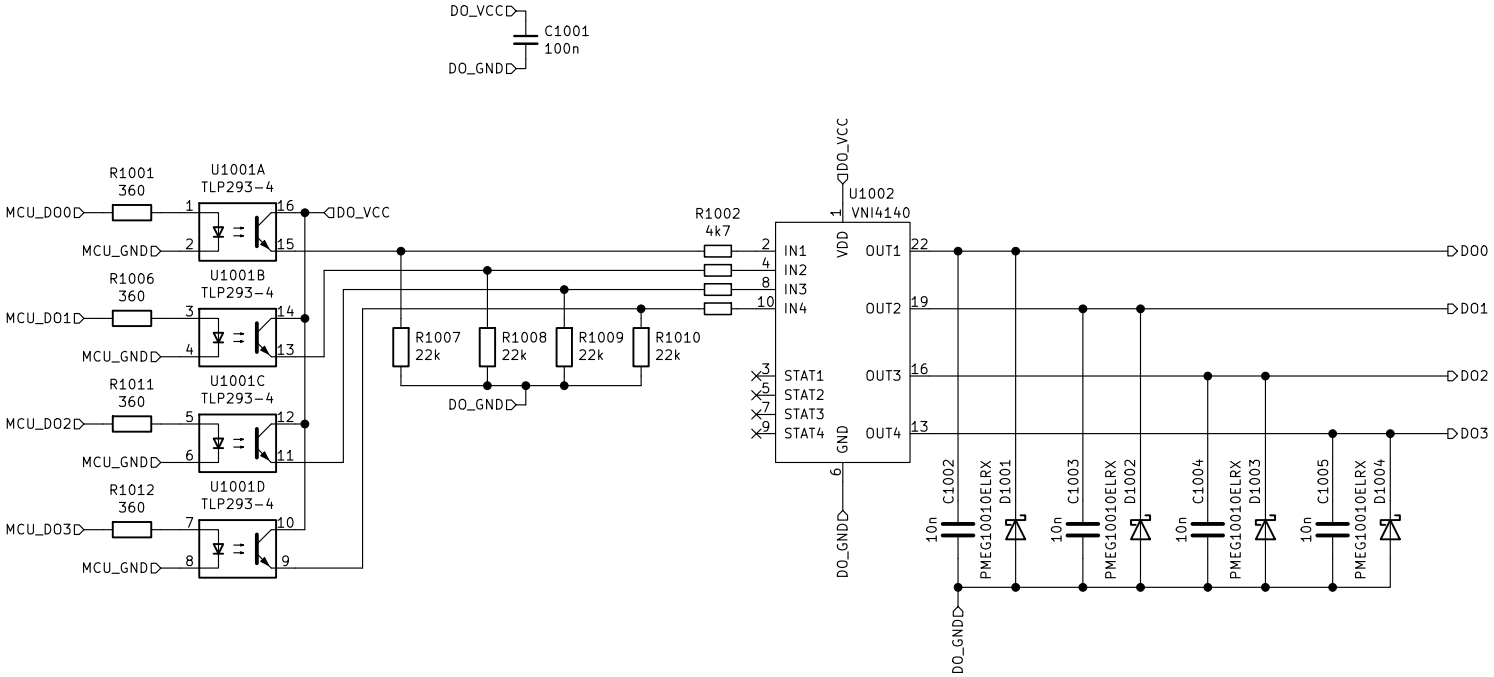


Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

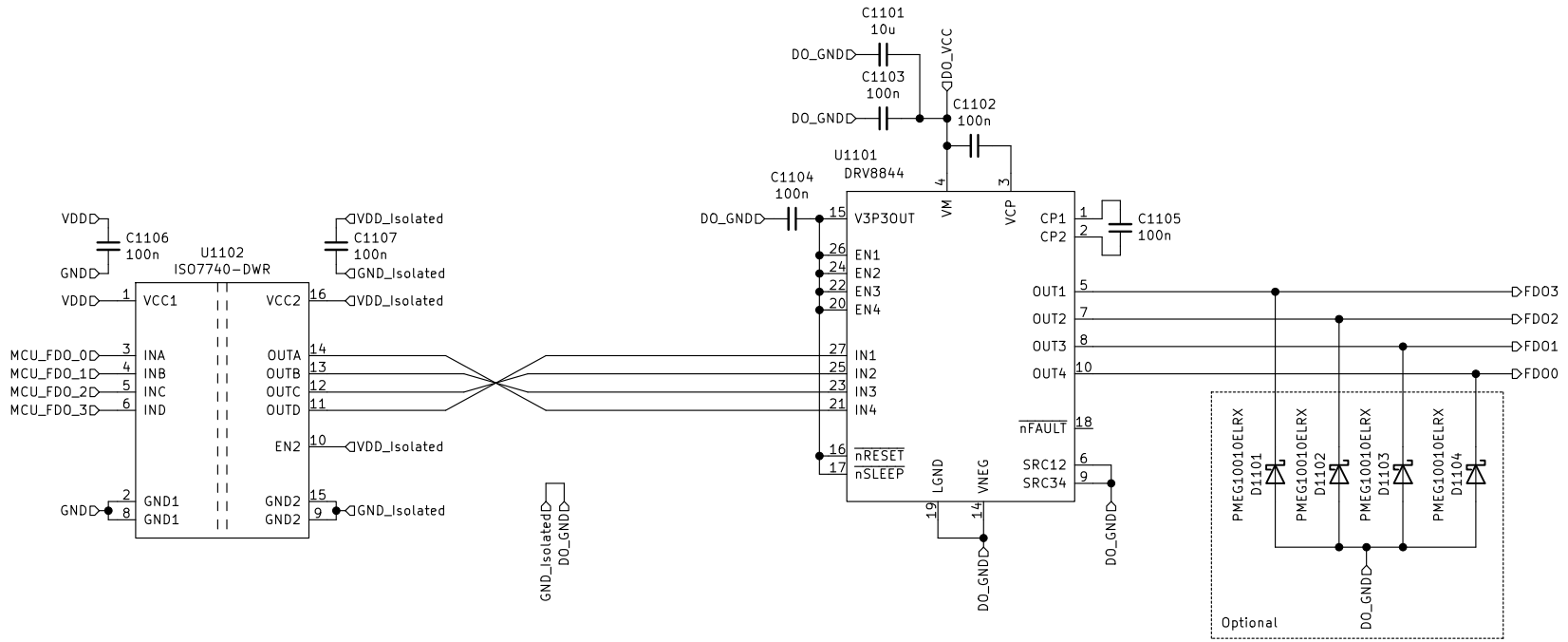
Sheet: /Digitální výstupy 4–7/
File: DigitalOutput.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 9/16

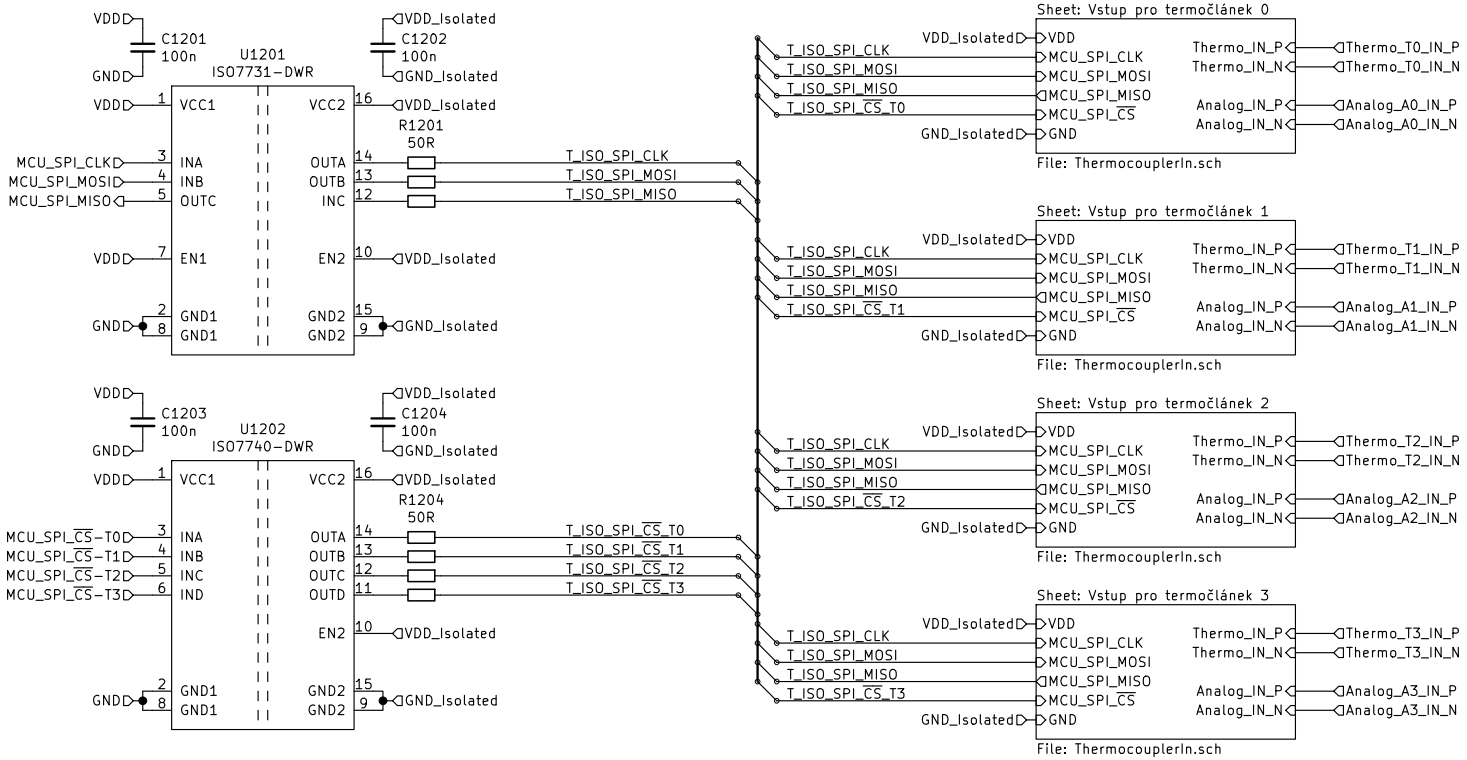


Jan Novotný	
DNP – Don't place	
MCSEE	
Modular control system	
ELZACO spol. s r.o.	
Sheet: /Digitální výstupy 8–11/	
File: DigitalOutput.sch	
Title: Display with periphery	
Size: A4	Date: 2020-12-17
KiCad E.D.A. kicad (5.1.7)-1	Id: 10/16
	Rev: 1.0.0



Jan Novotný	
DNP – Don't place	
MCSEE	
Modular control system	
ELZACO spol. s r.o.	
Sheet: /Rychlé digitální výstupy 0–3/	
File: FastDigitalOutput.sch	
Title: Display with periphery	
Size: A4	Date: 2020-12-17
KiCad E.D.A. kicad (5.1.7)-1	Id: 11/16

Rev: 1.0.0

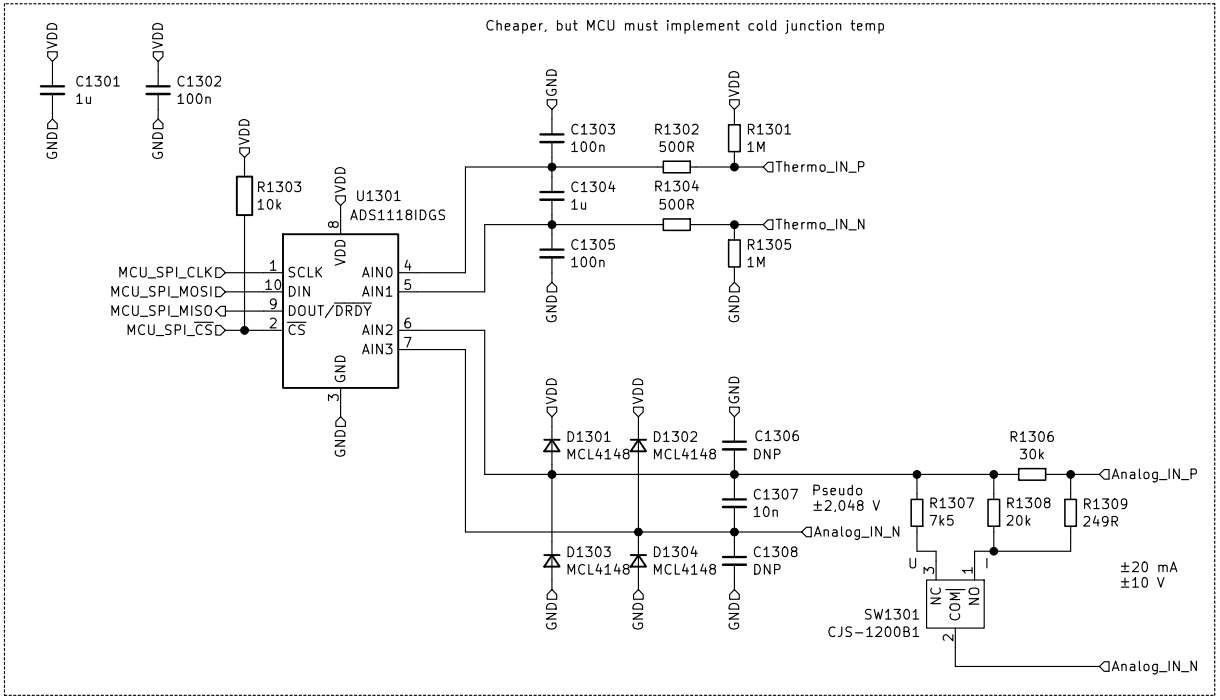


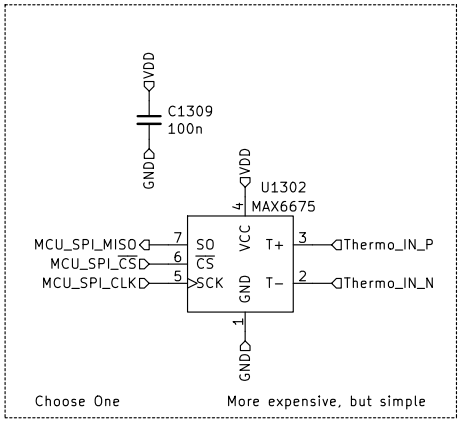
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Vstupy pro termočlánky/
File: Thermocouplers.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 12/16



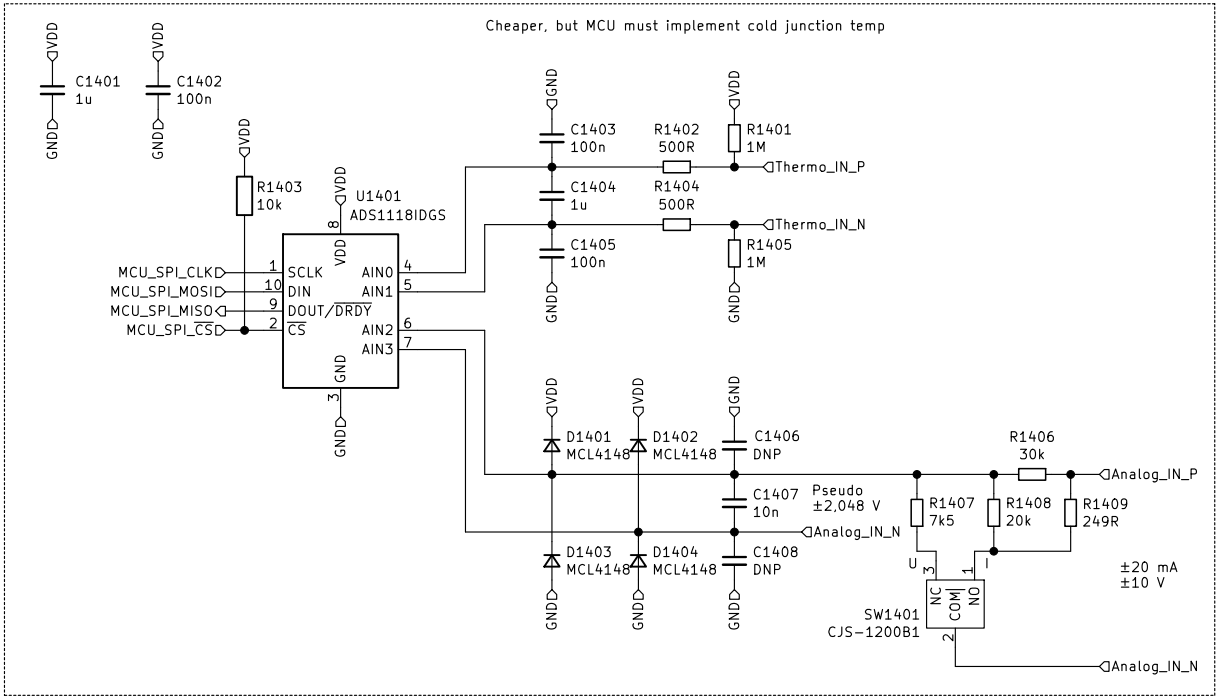


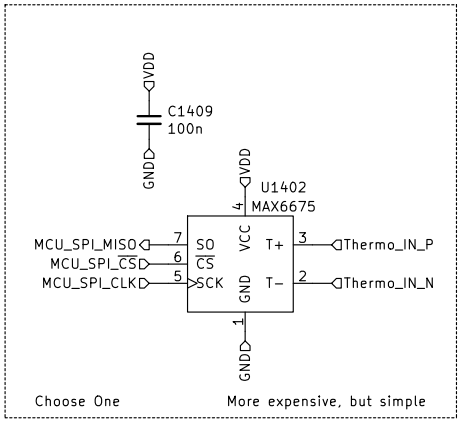
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Vstupy pro termočlánky/Vstup pro termočlánek 1/
File: ThermocouplerIn.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 13/16



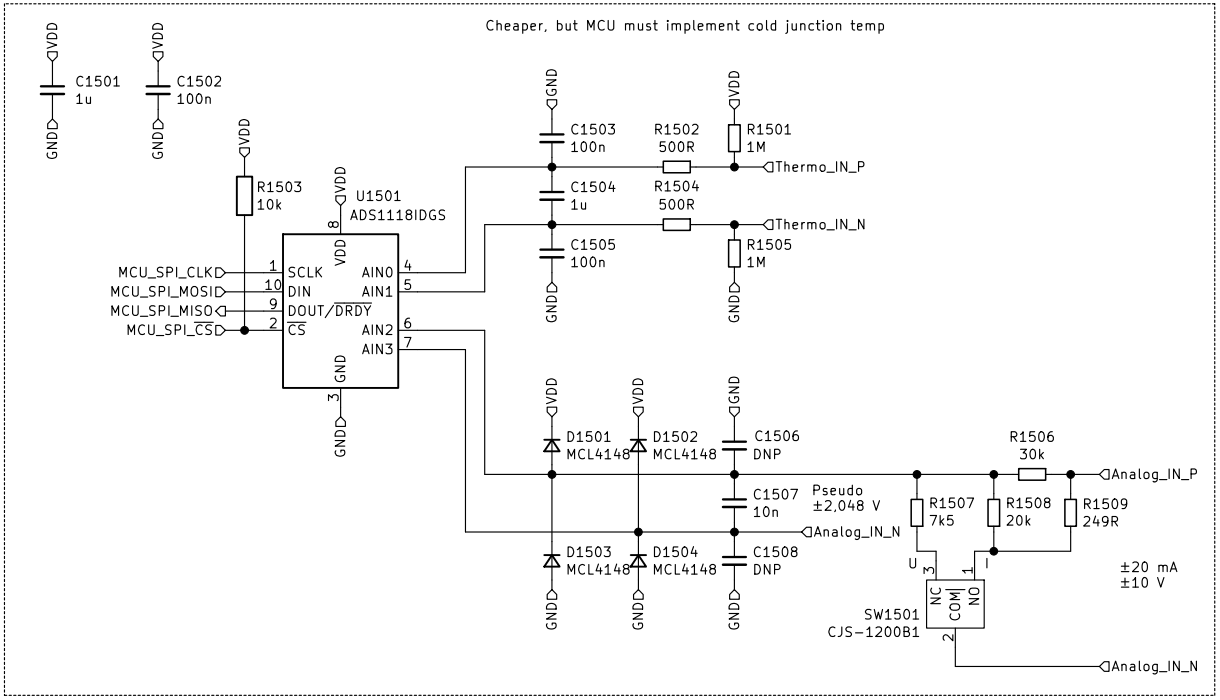


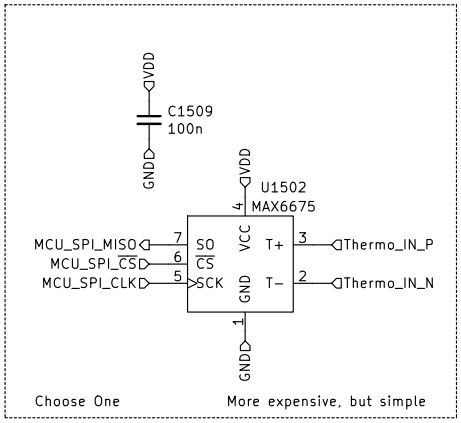
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Vstupy pro termočlánky/Vstup pro termočlánek 0/
File: ThermocouplerIn.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 14/16



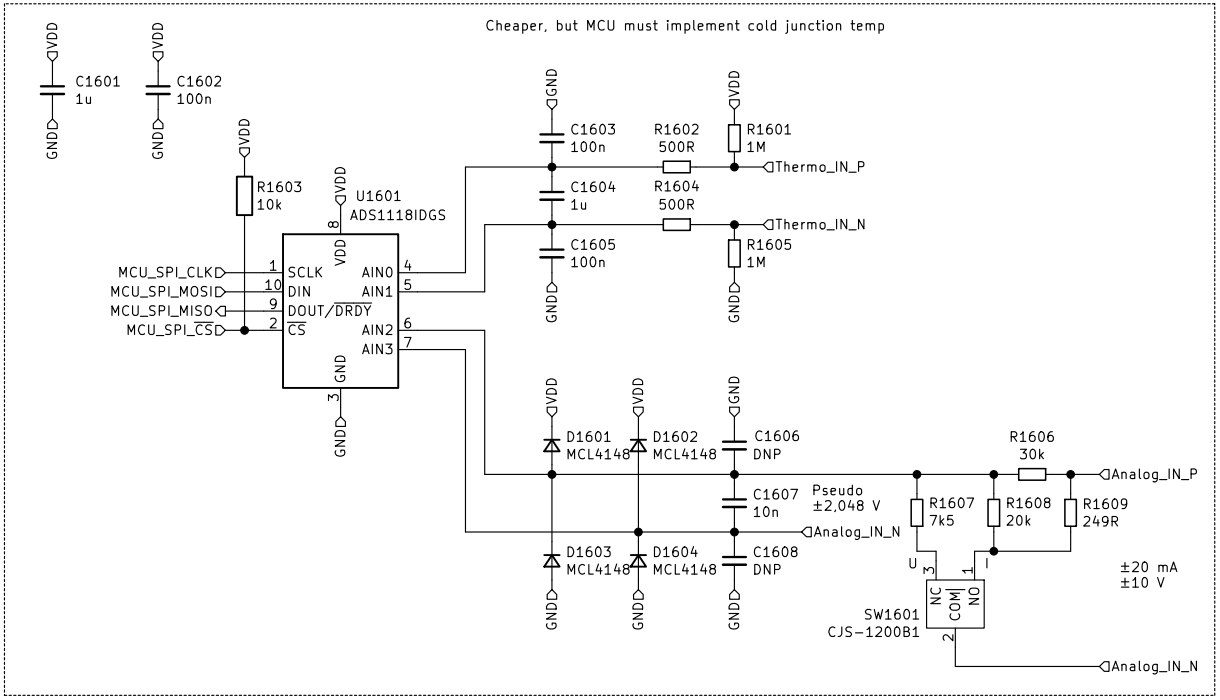


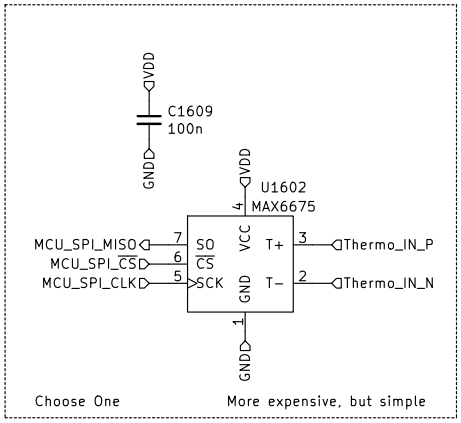
Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Vstupy pro termočlánky/Vstup pro termočlánek 2/
File: ThermocouplerIn.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 15/16





Jan Novotný
DNP – Don't place
MCSEE
Modular control system
ELZACO spol. s r.o.

Sheet: /Vstupy pro termočlánky/Vstup pro termočlánek 3/
File: ThermocouplerIn.sch

Title: Display with periphery

Size: A4	Date: 2020-12-17	**Rev: 1.0.0**
KiCad E.D.A. kicad (5.1.7)-1		Id: 16/16

B Objekty kom. protokolu MCSBUS

Tato příloha obsahuje popis jednotlivých komunikačních objektů uživatelské sekce, které jsou pro každý modul jedinečné, pro navržený komunikační protokol *MCSBUS*. Interpretace proměnných je popsána v příloze C.

B.1 Digitální vstupy

Číslo	Velikost	Název	Popis
0	$4B$	Input data	Data dig. vstupů z posledního čtecího cyklu
1	$4B$	Edge	Informace o přítomnosti hran mezi čtením
2	$4B$	Rising edge	Informace o přítomnosti náběžných hran mezi čtením
3	$4B$	Falling edge	Informace o přítomnosti seběžných hran mezi čtením
4	$12B$	Cyclic data	Optimalizované data pro cyklickou komunikaci
16	$2B$	Config	Konfigurace digitálních vstupů

Tab. B.1: Přehled uživatelských objektů pro modul digitálních vstupů

B.1.1 Objekt č. 0 - Input data

Jméno	Velikost	Interpretace	Popis
Data	$4B$	UINT32	Data posledního čtení digitálních vstupů

Tab. B.2: Uživatelský objekt č. 0 pro modul digitálních vstupů ($0x10$:RO)

B.1.2 Objekt č. 1 - Edge

Jméno	Velikost	Interpretace	Popis
Data	$4B$	UINT32	Data o změně signálu

Tab. B.3: Uživatelský objekt č. 1 pro modul digitálních vstupů ($0x11$:RO)

B.1.3 Objekt č. 2 - Rising edge

Jméno	Velikost	Interpretace	Popis
Data	$4B$	UINT32	Data o přední (stoupající) hraně na vstupu

Tab. B.4: Uživatelský objekt č. 2 pro modul digitálních vstupů ($0x12$:RO)

B.1.4 Objekt č. 3 - Falling edge

Jméno	Velikost	Interpretace	Popis
Data	$4B$	UINT32	Data o zadní (klesající) hraně na vstupu

Tab. B.5: Uživatelský objekt č. 3 pro modul digitálních vstupů ($0x13$:RO)

B.1.5 Objekt č. 4 - Cyclic data

Jméno	Velikost	Interpretace	Popis
Input	$4B$	UINT32	Data posledního čtení digitálních vstupů
R-Edge	$4B$	UINT32	Data o přední (stoupající) hraně na vstupu
F-Edge	$4B$	UINT32	Data o zadní (klesající) hraně na vstupu

Tab. B.6: Uživatelský objekt č. 3 pro modul digitálních vstupů ($0x14$:RO)

B.1.6 Objekt č. 16 - Config

Jméno	Velikost	Interpretace	Popis
Scan time	$2B$	UINT16	Doba cyklu čtení digitálních vstupů $[ms]$
Event gen	$8B$	enum[32]	Generování eventu při výskytu hran (rezervováno pro novější sběrnici s CAN bus)

Tab. B.7: Uživatelský objekt č. 16 pro modul digitálních vstupů ($0x20$:RW)

B.2 Digitální výstupy

Číslo	Velikost	Název	Popis
0	$4B$	Binary data	Binární výstupní data
1	$64B$	Duty data	Data pro PWM výstup (všechny najednou)
2	$3B$	Duty data fast	Data pro PWM výstup (pouze jeden)
16	$64B$	Config	Nastavení digitálních výstupů
17	$4B$	Config stop mode	Nastavení úrovně výstupů ve stop módu

Tab. B.8: Přehled uživatelských objektů pro modul digitálních výstupů

B.2.1 Objekt č. 0 - Binary data

Jméno	Velikost	Interpretace	Popis
Data	$4B$	UINT32	Binární data výstupu

Tab. B.9: Uživatelský objekt č. 0 pro modul digitálních výstupů ($0x10$:RW)

B.2.2 Objekt č. 1 - Duty data

Jméno	Velikost	Interpretace	Popis
Duty	$64B$	UINT16[32]	Data o střídě výstupu

Tab. B.10: Uživatelský objekt č. 1 pro modul digitálních výstupů ($0x11$:RW)

B.2.3 Objekt č. 2 - Duty data (rychlý přenos)

Jméno	Velikost	Interpretace	Popis
Output	$1B$	UINT8	Číslo výstupu
Duty	$2B$	UINT16	Data o střídě výstupu

Tab. B.11: Uživatelský objekt č. 2 pro modul digitálních výstupů ($0x12$:RW)

B.2.4 Objekt č. 16 - Config PWM

Jméno	Velikost	Interpretace	Popis
Period	$64B$	UINT16[32]	Perioda generovaného PWM * 100 [μs]

Tab. B.12: Uživatelský objekt č. 16 pro modul digitálních výstupů ($0x20$:RW)

B.2.5 Objekt č. 17 - Config stop mode

Jméno	Velikost	Interpretace	Popis
Default	$4B$	UINT32	Hodnota digitálního výstupu ve stop stavu

Tab. B.13: Uživatelský objekt č. 17 pro modul digitálních výstupů ($0x21$:RW)

B.3 Analyzátor sítě

Číslo	Velikost	Název	Popis
0	$52B$	Meas data	Naměřená data pro všechny fáze
1..3	$16B$	Phase data	Naměřená data pro jednu fázi
4..9	$200B$	FFT	Data FFT analýzy
16	$25B$	Config	Nastavení všech analogových IO najednou

Tab. B.14: Přehled uživatelských objektů pro modul analyzátoru sítě

B.3.1 Objekt č. 0 - Meas data

Jméno	Velikost	Interpretace	Popis
P-N Voltage	$12B$	REAL[3]	Hodnota změřeného napětí [V]
P-P Voltage	$12B$	REAL[3]	Hodnota změřeného napětí [V]
Current	$12B$	REAL[3]	Hodnota změřeného proudu [A]
Power	$12B$	REAL[3]	Hodnota změřeného napětí [W]
Frequency	$4B$	REAL	Hodnota změřené frekvence [Hz]

Tab. B.15: Uživatelský objekt č. 0 pro modul analyzátoru sítě ($0x10$:RO)

B.3.2 Objekt č. 1..3 - Phase data

Jméno	Velikost	Interpretace	Popis
Voltage	$4B$	REAL	Hodnota změřeného napětí [V]
Current	$4B$	REAL	Hodnota změřeného proudu [A]
Power	$4B$	REAL	Hodnota změřeného napětí [W]
Frequency	$4B$	REAL	Hodnota změřené frekvence [Hz]

Tab. B.16: Uživatelský objekt č. 1..3 pro modul analyzátoru sítě ($0x11 - 13$:RO)

B.3.3 Objekt č. 4..9 - FFT

Jméno	Velikost	Interpretace	Popis
Data	$200B$	REAL[50]	Hodnota změřeného napětí [V]

Tab. B.17: Uživatelský objekt č. 4..9 pro modul analyzátoru sítě ($0x14 - 19$:RO)

B.3.4 Objekt č. 16 - Config

Jméno	Velikost	Interpretace	Popis
Base freq	$1B$	bool	Výběr základní frekvence.
Voltage trafo	$12B$	REAL[3]	Převodní poměr napěťového transformátoru
Current trafo	$12B$	REAL[3]	Převodní poměr proudového transformátoru

Tab. B.18: Uživatelský objekt č. 16 pro modul analyzátoru sítě ($0x20$:RW)

B.4 Analogové vstupy a výstupy

Číslo	Velikost	Název	Popis
0	$16B$	Raw data input	Binární data převodníku pro vstupy
1	$16B$	Raw data write all	Binární data převodníku pro všechny výstupy
2	$3B$	Raw data write single	Binární data převodníku pro jeden výstup
3	$32B$	Physical data read	Fyzická data všech vstupů
4	$32B$	Physical data all	Fyzická data všech výstupů
5..12	$5B$	Physical data single	Fyzická data jednoho výstupu
13	$32B$	Ing data all	Inženýrská data všech výstupů
14	$5B$	Ing data single	Inženýrská data všech výstupů
15	$32B$	Ing data read	Inženýrská data všech výstupů
16	XB	Config all	Nastavení všech analogových IO najednou
17	XB	Config single	Nastavení jednoho analogového IO
18	$16B$	Config stop mode	Nastavení nastavení chování analogového výstupu ve stop módu

Tab. B.19: Přehled uživatelských objektů pro modul analogových IO

B.4.1 Objekt č. 0 - Raw data input

Jméno	Velikost	Interpretace	Popis
Data	$16B$	UINT16[8]	Hodnoty ADC převodníku

Tab. B.20: Uživatelský objekt č. 0 pro modul analogových IO ($0x$10:RO)

B.4.2 Objekt č. 1 - Raw data write all

Jméno	Velikost	Interpretace	Popis
Data	$16B$	UINT16[8]	Hodnoty DAC převodníku

Tab. B.21: Uživatelský objekt č. 1 pro modul analogových IO ($0x11$:RW)

B.4.3 Objekt č. 2 - Raw data write single

Jméno	Velikost	Interpretace	Popis
Index	$1B$	UINT8	Index analogového IO
Data	$2B$	UINT16	Hodnota DAC převodníku

Tab. B.22: Uživatelský objekt č. 2 pro modul analogových IO ($0x12$:RW)

B.4.4 Objekt č. 3 - Physical data read

Jméno	Velikost	Interpretace	Popis
Data	$32B$	REAL[8]	Hodnota fyzikálního významu analogového vstupu

Tab. B.23: Uživatelský objekt č. 3 pro modul analogových IO ($0x13$:RO)

B.4.5 Objekt č. 4 - Physical data all

Jméno	Velikost	Interpretace	Popis
Data	$32B$	REAL[8]	Hodnota fyzikálního významu analogového výstupu

Tab. B.24: Uživatelský objekt č. 4 pro modul analogových IO ($0x14$:RW)

B.4.6 Objekt č. 5..12 - Physical data single

Jméno	Velikost	Interpretace	Popis
Data	$4B$	REAL	Hodnota fyzikálního významu analogového výstupu

Tab. B.25: Uživatelský objekt č. 5 až 12 pro modul analogových IO ($0x15 - 1C$:RW)

B.4.7 Objekt č. 13 - Ing data all

Jméno	Velikost	Interpretace	Popis
Data	$32B$	REAL[8]	Inženýrská data kanálu.

Tab. B.26: Uživatelský objekt č. 13 pro modul analogových IO ($0x1D$:RW)

B.4.8 Objekt č. 14 - Ing data single

Jméno	Velikost	Interpretace	Popis
Index	$1B$	UINT8	Index analogového IO
Data	$4B$	REAL	Inženýrská data kanálu.

Tab. B.27: Uživatelský objekt č. 14 pro modul analogových IO ($0x1E$:RW)

B.4.9 Objekt č. 15 - Ing data read

Jméno	Velikost	Interpretace	Popis
Data	$32B$	REAL[8]	Inženýrská data kanálu.

Tab. B.28: Uživatelský objekt č. 15 pro modul analogových IO ($0x1F$:RO)

B.4.10 Objekt č. 16 - Config all

Jméno	Velikost	Interpretace	Popis
Conf	XB	struct[8] (C.2.1)	Struktura obsahující nastavovaná data

Tab. B.29: Uživatelský objekt č. 16 pro modul analogových IO ($0x20$:RW)

B.4.11 Objekt č. 17 - Config single

Jméno	Velikost	Interpretace	Popis
Index	$1B$	UINT8	Index analogového IO
Conf	XB	struct[8] (C.2.1)	Struktura obsahující nastavovaná data

Tab. B.30: Uživatelský objekt č. 17 pro modul analogových IO ($0x21$:RW)

B.4.12 Objekt č. 18 - Config stop mode

Jméno	Velikost	Interpretace	Popis
Data	$16B$	UINT16[8]	Hodnota DAC převodníku nastavovaná ve stop módu

Tab. B.31: Uživatelský objekt č. 18 pro modul analogových IO ($0x22$:RW)

C Definované datové typy pro MCSBUS

C.1 Jednoduché datové typy

Jméno	Velikost	Ekvivalent C	Hodnota
INT8	$1B$	signed char	-128 až 127
UINT8	$1B$	unsigned char	0 až 255
INT16	$2B$	short	-32768 až 32767
UINT16	$2B$	unsigned short	0 až 65535
INT32	$4B$	int	-2147483648 až 2147483647
UINT32	$4B$	unsigned int	0 až 4294967295
REAL	$4B$	float	$1,2 \cdot 10^{-38}$ až $3,4 \cdot 10^{38}$
PRECISE	$8B$	double	$2,3 \cdot 10^{-308}$ až $1,7 \cdot 10^{308}$

Tab. C.1: Základní datové typy použité pro popis komunikace přes MCSBUS

C.2 Struktury

C.2.1 Konfigurace analogového IO

Jméno	Velikost	Interpretace	Popis
Channel type	$1\,B$	enum (C.3.1)	Typ AIO kanálu.
Phy. range	$1\,B$	enum (C.3.2)	Rozsah hodnot fyzického rozhraní.
Ing. min. val.	$4\,B$	REAL	Minimální hodnota inženýrské hodnoty.
Ing. max. val.	$4\,B$	REAL	Maximální hodnota inženýrské hodnoty.

Tab. C.2: Struktura obsahující definici kanálu analogového vstupu a výstupu.

C.3 Enumerátory

C.3.1 Typ analogového IO

Hodnota	Popis
0	Režim vysoké impedance (nepoužitý kanál).
1	Napěťový výstup.
2	Proudový výstup.
3	Napěťový vstup.
4	Proudový vstup s externím napájením proudové smyčky.
5	Proudový vstup s aktivním napájením proudové smyčky.
6	Měření odporu.
7	Digitální vstup s externím napájením.
8	Digitální vstup s aktivním napájením.

Tab. C.3: Enumerátor typu analogového kanálu.

C.3.2 Rozsah fyzického kanálu

Hodnota	Rozsah napětí	Rozsah proudu
0	$0..10\,V$	$0..20\,mA$
1	$\pm 10\,V$	$\pm 20\,mA$
2	$0..5\,V$	$4..20\,mA$

Tab. C.4: Enumerátor pro znázornění rozsahu fyzického rozhraní.

D Obsah elektronické přílohy

/ ..... kořenový adresář přiloženého archivu
- xnovot1i_text_diplomove_prace.pdf
- Navrhy
  - AnalogoveIO.zip
  - AnalyzatorSite.zip
  - DigitalniVstupy.zip
  - DigitalniVystupy.zip
  - Displej.zip
  - MiniDisplej.zip
  - ProudoveTrafaMiniPCB.zip
  - RidiciSystem.zip
  - RidiciSystemLite.zip
  - Schemata_zapojeni
    - SchemaDisplej.pdf
    - Schema_Analogove_IO.pdf
    - Schema_analyzator_site.pdf
    - Schema_Digitalni_vstupy.pdf
    - Schema_Digitalni_vystupy.pdf
    - Schema_Ridici_System.pdf
    - Schema_Ridici_System_Lite.pdf
- Programy
  - AnalyzatorSite.zip
  - ControlUnit.zip
  - Displej.zip
  - FontCreator8x8.rar
  - MCS_AnalogIO.zip
  - MCS_DigitalInput.zip
  - MCS_DigitalOutput.zip