VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Radim Vítek

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

AUTOMATIZOVANÉ TESTOVACÍ ZAŘÍZENÍ PRO VÝSTUPNÍ KONTROLU VÝROBKŮ
AUTOMATIC TESTER FOR QUALITY CONTROL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Radim Vítek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Zdeněk Bradáč, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**
Ústav automatizace a měřicí techniky

Student: Bc. Radim Vítek ***ID:*** 173776
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Automatizované testovací zařízení pro výstupní kontrolu výrobků

POKYNY PRO VYPRACOVÁNÍ:

1. Navrhněte a zrealizujte digitálně řízený automatický tester pro výstupní kontrolu kvality výrobků. Proveďte literární rešerši problematiky.
2. Navrhněte celkovou koncepci testeru jako externího periferního zařízení k PC. Tester navrhněte jako universální tester s možností rozšíření pro širší spektrum testovaných výrobků.
3. Navrhněte a zrealizujte elektronické obvody testeru, oživte a testujte funkčnost HW.
4. Navrhněte koncepci programového vybavení a zrealizujte je. Otestujte a demonstrujte funkčnost SW.
5. Zhodnoťte zvolené řešení, vyhodnoťte výsledky a popište nasazení.

DOPORUČENÁ LITERATURA:

Pavel Herout: Učebnice jazyka C, KOPP, 2004, IV. přepracované vydání, ISBN 80-7232-220-6

Dle pokynů vedoucího práce.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 13.5.2019

Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.
Konzultant:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Práce se zabývá návrhem a konstrukcí zařízení pro automatickou kontrolu výrobků. Popisuje různé přístupy k testování elektronických výrobků ve výrobě. Popisuje jejich výhody a nevýhody. Hlavní část práce je věnována samotnému návrhu. Je zde popsáno jakým způsobem je výsledné zařízení navrženo jak po hardwarové stránce tak i po stránce softwarové.

KLÍČOVÁ SLOVA

Automatická kontrola, sběrnice I2C, flying-probe tester, AOI, bed-of-nails, Python, testovací předpis, rack

ABSTRACT

The thesis deals with design and construction of device for automatic product inspection. Thesis describes methods for testing products during manufacture. Thesis describes their main advantages and disadvantages. The main part of this thesis is design of testing device. There is described method of designing this device. It describes design of software and hardware.

KEYWORDS

Automatic testing, I2C bus, flying-probe tester, AOI, bed-of-nails, Python, testing recipe, rack

VÍTEK, Radim. *Automatizované testovací zařízení pro výstupní kontrolu výrobků*. Brno, Rok, 83 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: doc. Ing. Zdeněk Bradáč, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Automatizované testovací zařízení pro výstupní kontrolu výrobků“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Zdeňkovi Bradáčovi, Ph.D. za odborné vedení a trpělivost. Dále bych rád poděkoval své rodině a všem z firmy EGMedical, s.r.o., díky nimž mohla tato práce vzniknout.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Každá firma, která na zakázku vyvíjí elektroniku, musí ke svým zařízením také vytvářet testovací přípravky či celá pracoviště. Je to způsobeno potřebou při výrobě ověřit zda je zařízení funkční či nikoliv. Protože firma zabývající se výrobou ručí za správnou funkčnost, je vyžadován spolehlivý způsob kontroly funkcionality výrobku. O tuto kontrolu se jako tvůrce stará vývojářská firma.

Pro takovou firmu existuje několik možností jak zajistit kontrolu funkcionality. Jednou z nich je zajištění testovacího postupu, podle nějž pracovník zařízení zkontroluje. Vezmeme-li v potaz spolehlivost lidí, není tento způsob zcela ideální. Další možností je tvorba zcela specifických testovacích zařízení. Ta jsou ovšem problematická na údržbu a nejsou dostatečně flexibilní při úpravách průběhu testu. Další možností je použít již existující testovací platformu.

Cílem této práce je navrhnout vlastní testovací platformu, kterou by bylo možné používat univerzálně pro různá zařízení. Platforma si však musí při svojí univerzálnosti zachovat jednoduchost, jakou obvykle mají jednoúčelové testery.

Tato práce se v teoretické části zabývá metodami testování elektroniky ve výrobě. Popisuje běžně používané techniky i techniky, které jsou v oboru testování novinkami.

Praktická část se zabývá návrhem HW a SW. Jsou zde popsány požadavky, které jsou na vytvářený tester kladeny. Dále je zde popsán návrh struktury testeru s popisem okolností, které k daným způsobům řešení vedly. Je zde popsán celý návrh HW rozdělený na jednotlivé části. Popis HW je následován popisem SW včetně postupů pro vytváření testů.

1 Testování elektroniky

Aby bylo možné zajistit dostatečnou kvalitu výrobků, je třeba při výrobě a vývoji výrobky testovat. Při vývoji je testování součástí činnosti, kdy se postupně kontroluje požadovaná funkčnost zařízení a postupně se chyby korigují. Při výrobě je testování dodatečnou činností, která je ovšem nezbytná. Pro samotnou výrobu testování nutné není, ale v takovém případě budou vznikat vadné výrobky vlivem různých příčin.

Aby se tyto vadné výrobky do prodeje nedostaly, je třeba každý výrobek řádně otestovat. K tomu se používá řada metod, přípravků a zařízení, které testování nejen umožní, ale i zefektivní a usnadní.

1.1 In-Circuit testy

In-Circuit (ICT) testy jsou jednou z nejrozšířenějších metod výstupní kontroly při výrobě elektronických zařízení. Testování pomocí ICT spočívá v měření signálů na DPS, kdy se pro systém generují budicí signály a kontroluje se zda je reakce obvodu správná. Existuje řada postupů jak ICT realizovat.

1.1.1 Fixtura s jehlovým polem (Bed of nails testing)

V tomto případě tester obsahuje měřicí jehly, na které dosedá DPS svými testovacími ploškami. Tím je zajištěno připojení vyhodnocovací jednotky k DPS. V závislosti na velkosti DPS a její složitosti může fixtura obsahovat i několik set jehel. Jednotlivé jehly mohou sloužit k různým účelům, například: napájení DPS, programovací rozhraní, generování a měření signálů apod. Kromě jehel je možné na fixturu umisťovat i prvky, které zajistí testování mechanických částí, dávají zpětnou vazbu o správné činnosti signalizace apod.

Obr. 1.1: Bed of nails tester [7]

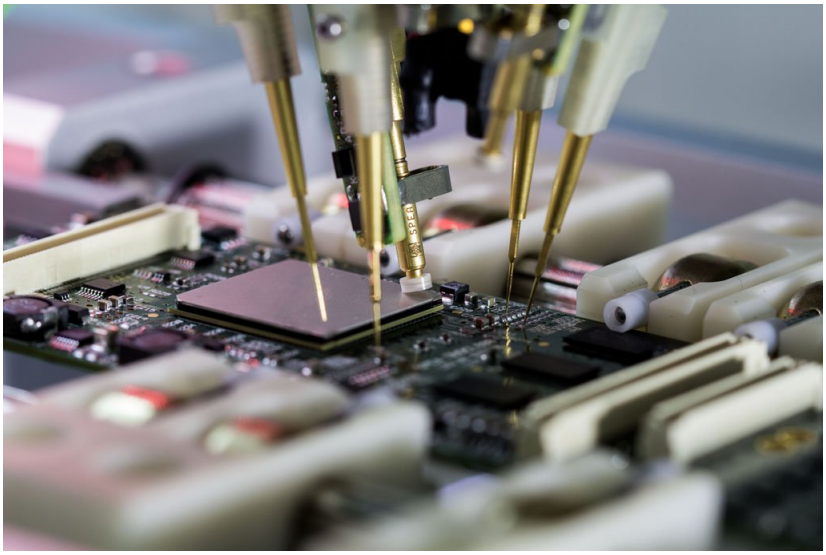
Bed of nails se nepoužívá pro testování malého počtu kusů nebo pro testování zařízení, kde se předpokládá častá změna rozložení DPS. Důvodem je nemožnost

změnit rozložení jehel, vždy je nutné pro každou změnu rozložení testovacích bodů vyrobit nové jehlové pole. Výroba nového pole může být drahá a zdlouhavá záležitost.

Naproti tomu je Bed of nails velmi výkonné pro testování větších sérií. Protože se DPS vyrábí a osazují v panelu, lze konstruovat tester pro celý panel. Testy mohou probíhat na každé DPS současně a tím lze snížit dobu potřebnou pro testování. V případě testování v panelu musí být možné otestovat i samostatné DPS kvůli opravám, reklamacím apod.

1.1.2 Flying probe tester

Princip je podobný testování pomocí Bed of nails. Jehly jsou umístěny v hlavách, které se pohybují nad DPS a dotknou se vždy jen několika testovacích bodů. Tester je konstruován jako CNC stroj. Je tedy možné dráhu jehel a pozice testovacích bodů velmi snadno a rychle měnit. Moderní Flying probe testery mohou testovat pomocí 24 sond s možností zásahu bodu o velikosti 100 µm. [4] Namísto jehel je možné do některých testerů umístit např. detektor světla (často s možností měření spektra), antény pro vysílání a příjem RF signálu, čtečky čárových kódů, případně jiné senzory a snímače.



Obr. 1.2: Flying probe tester při testování [3]

Flying probe testery jsou vhodné pro menší série. Oproti Bed of nails testerům jsou výrazně pomalejší. Dalším omezujícím parametrem použití je výška součástek, kdy tester musí součástku objet. V extrémních případech, kdy je součástka příliš vysoká, mohou na DPS vzniknout místa, kam nelze jehlou dosáhnout vůbec. Výhoda testeru spočívá zejména v univerzalitě. Výhodou také je, že na DPS nemusí být

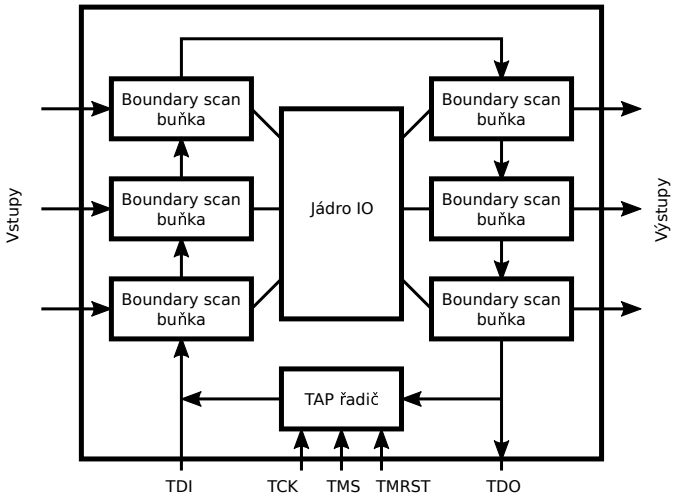
umístěny speciální testovací body. Namísto toho lze využít nezamaskované prokovy, piny součástek, odmaskované části cesty apod. [2]

1.1.3 JTAG / Boundary scan testing

Jedná se o způsob čtení / zapisování stavu jednotlivých pinů integrovaných obvodů. Princip je velmi podobný testování pomocí bed of nails, jen se nepoužívají sondy připojované z vnějšku, ale sondy vytvořené přímo na čipu. Proto je někdy tento způsob testování označován jako „silicon nails“. Systém je postaven na rozhraní JTAG. Díky čemuž je možné jednu sběrnici využít při ladění, programování i testování.

Jednotlivé komponenty podporující boundary scan jsou zapojeny za sebou v topologii daisy–chain, viz obr. 1.3. Teoreticky je možné do řetězce zapojit neomezené množství zařízení. Při vyšších frekvencích a větším množství zařízení ovšem může komunikace selhávat.

Výhodou této technologie je, že se testy vytváří pouze na PC, bez nutnosti používat testovací přípravky. Metoda selhává v případech, kdy se chyba vyskytuje mimo testovací řetězec (např. špatně připájený konektor). Tento problém je možné řešit kombinováním více metod testování.



Obr. 1.3: Princip metody boundary scan

1.2 Funkcionální testování

Zařízení se testuje jako celek bez kontroly vnitřních signálů. Tester na vstupy přivádí budicí signály a kontroluje odezvu na výstupech. Pokud zařízení obsahuje programovatelné prvky, obvykle se do nich nahrává tzv. testovací FW, který umožňuje snazší

detekci chyb. Testování tímto způsobem obvykle vyžaduje interakci obsluhy, která zajišťuje testování mechanických prvků, případně signalizačních komponent (LED diod, displejů, akustické signalizace, ...).

Testovací FW obvykle komunikuje s testerem pomocí sběrnice a na požadavky testeru vystavuje na výstupy dané signály, případně vyhodnocuje vlastní vstupy. Některé komplexnější zařízení mohou test provádět i samy. V tomto případě se k zařízení připojí přípravek, který propojuje vstupy na výstupy. Na základě programu se poté vyhodnocuje, jestli se zařízení chová správně.

Funkcionální testování je vhodné zejména pro kusovou výrobu a malé série. Velkou výhodou je nízká cena testeru. V případě interakce obsluhy během testování je možné vnesení lidské chyby, což u jiných postupů testování není problém.

1.3 Optická inspekce

Optická inspekce je proces, kdy se kontroluje kvalita pájených spojů, absence součástek a jejich správná poloha. Optickou inspekci lze provádět pouhým okem (prototypy a kusová výroba), nebo pomocí automatických strojů (AOI - automatická optická inspekce).

Pro automatickou optickou inspekci jsou vyráběny speciální stroje, které obsahují pohyblivé kamery, jimiž je snímána DPS. Pro testování se vytváří program, kde je specifikován vzhled a poloha součástky. Systém je schopen rozpoznat vady pájeného spoje, chybějící součástky, můstky mezi piny IO apod. AOI může kontrolovat DPS ve 2D nebo i ve 3D. 3D technologie se snadněji programují a mají vyšší úspěšnost v detekci chyb a nižší množství falešných detekcí. [1]

1.4 Kouřové testování

V oblasti testování elektroniky se jako kouřové testování označuje postup, kdy se hledají závažné chyby v hardwaru. Obvykle jde o prvotní test, kdy se na DPS připojí napájení a zjišťuje se, jestli se některá součástka nepoškodí. Vstupem tohoto testu obvykle bývá DPS, která prošla optickou inspekcí.

Tento způsob testování se také někdy používá pro odhalování vadných součástek. Vada součástky se často projevuje nadměrnou produkcí tepla. Toto je možné kontrolovat např. pomocí termokamery.

2 Požadavky na tester

Pro úspěšný návrh testeru je třeba definovat požadavky na způsob testování, HW testeru, postup vytváření testů a na rozhraní pro uživatele. Základními požadavky jsou:

- testování metodou funkčních testů,
- modularita a univerzálnost HW,
- robustní a spolehlivá konstrukce,
- jednotné a jednoduché vytváření testů,
- jednoduché ovládání testeru,
- informování obsluhy o potřebných zásazích do testu.

Testování má probíhat metodou funkčních testů. Metoda nepotřebuje zásahy do HW testovaného zařízení (DUT), viz kapitola 1.2. Takto je možné testovat libovolné zařízení. Metoda funkčních testů předpokládá, že zařízení prošla alespoň základní kontrolou osazení, případně dalšími testy.

Požadavek na modularitu a univerzálnost vznikl z potřeby efektivně spravovat testery. Jelikož je celý tester poměrně drahá záležitost a často i rozměrná, není možné mít od každého jednoúčelového testeru kopii pro vývoj. Proto se testery často převáží z výroby na vývoj, přičemž v mezičase není možné vyrábět. Řešením je konstruovat tester modulárně tak, aby veškeré nákladné části byly základem testeru a specifika jednotlivých DUT byla řešena pomocí aplikačně specifických modulů. Díky modulárnímu systému je možné mít na vývoji jediný tester s kompletní sadou specifických modulů.

Specifické moduly jsou nevyhnutelnou součástí zajišťující přizpůsobení testeru na každé testované zařízení. Zároveň by specifické moduly měly být co nejjednodušší a také by jejich cena by měla být nízká. V ideálním případě by měly zajišťovat jen rozhraní mezi testerem a DUT.

Požadavek na robustní a spolehlivou konstrukci vychází z kritického postavení testeru ve výrobě. Pokud by se s testerem cokoli stalo, není možné daný výrobek vyrábět. V lepším případě má výrobní firma k dispozici náhradní tester pro případy poruchy. V každém případě musí tester vydržet nešetrné zacházení. Je také třeba brát v úvahu, že množství testovaných zařízení může být obrovské. Proto by měly být pro konstrukci použity průmyslové komponenty, které zaručí spolehlivost a dlouhou životnost testeru.

Způsob vytváření testů musí být jednoduchý a maximálně univerzální. Postup pro testování je pro každé zařízení jiný. Je tedy nutné zajistit možnost definovat testy dostatečně komplexně. Navíc průběh testu musí být z jeho konfigurace snadno čitelný a upravitelný.

Ovládání testeru musí být intuitivní. Správným návrhem ovládání je možné zvý-

šit produktivitu testeru. Například je vhodné, aby obsluha nemusela pro každé potvrzení úkonu myší kliknout na tlačítko v ovládacím programu. Rychlejší a méně náročné pro ovládání je mít na testeru mechanické tlačítko jehož stiskem se provede daná akce.

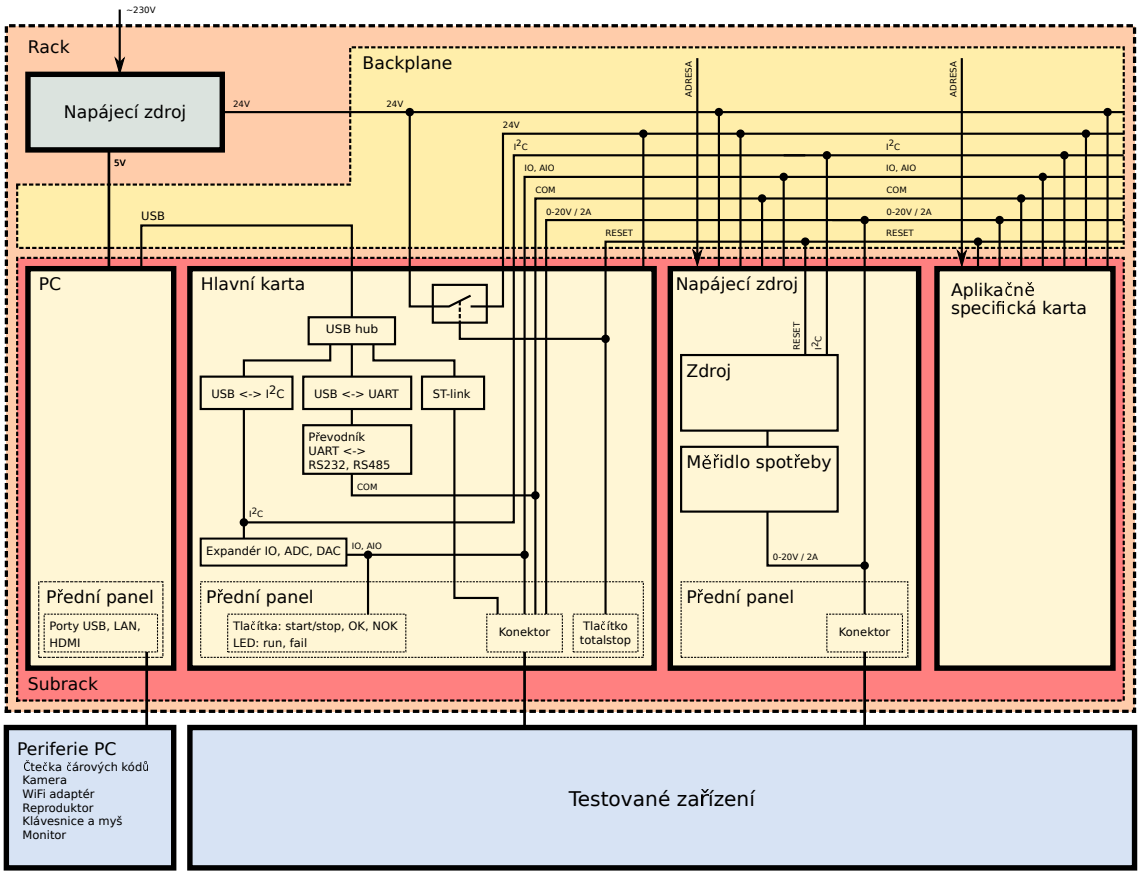
Každé testování musí být popsáno krok po kroku tak, aby člověk pověřený testováním mohl testy přesně opakovat. Existence testovacího postupu je nezbytně nutná i pro testování pomocí automatického testeru. Zde je však nutné mít dva druhy postupů. Jeden pro obsluhu podle kterého testování provádí. A druhý postup s přesným popisem jak celé testování probíhá včetně kroků, které provádí automatický tester. Testovací postup pro obsluhu může být zabudován přímo do programu, který test vykonává. Tester může poté obsluhu sám informovat o činnostech, které je třeba provést.

3 Návrh HW

Tester je koncipován jako systém v rackové skříni o rozměru 19” s výškou 4 U. Jednotlivé moduly jsou zásuvné karty do subracku o rozměru 100 mm × 160 mm.

Na obr. 3.1 je vidět celková struktura testeru. Základem je PC, který bude ovládat celou funkčnost testeru. Poběží na něm program pro testování, bude zajišťovat ukládání reportů apod. K PC může být připojena čtečka čárových kódů, tiskárna štítků, kamera pro dokumentaci testování a další periferie, které budou pro testování potřeba.

PC je spojen s hlavní kartou pomocí USB. Na základní kartě se nachází USB hub z nějž jsou rozvedeny po kartě 3 USB sběrnice. Sběrnice jsou použity pro převodníky USB ↔ I²C, USB ↔ UART a pro připojení programátoru ST-Link. Použitý USB hub má navíc rozhraní Ethernet, které je vyvedeno na přední panel hlavní karty, za účelem využití tohoto rozhraní pro testování Ethernetu na DUT.



Obr. 3.1: Blokový diagram testeru

Sběrnice I²C je vyvedena na backplane tak, aby byla přístupná na všech připojených kartách. Toto zapojení umožňuje, aby každá karta měla vlastní logiku bez

nutnosti použít MCU. Na kartách lze použít obvody, které tvoří expandéry GPIO expandéry, ADC, DAC apod. Na backplanu je pomocí třech vodičů zakódováno pořadí karty tak, aby bylo možné použít jednu kartu na různých pozicích v subracku. Použití třech adresních vodičů umožňuje použít maximálně 7 karet. Toto omezení vyplývá z množství kombinací na třech vodičích. Poslední osmá adresa je vyhrazena pro hlavní kartu, jejíž adresa je vždy 0.

Nevýhodou použití sběrnice I^2C na backplanu je problematické vytváření komunikačních rozhraní na dalších kartách. Protože rychlost sběrnice I^2C je poměrně nízká (obvykle 400 kHz), není ji možné použít pro vytváření rychlých komunikačních rozhraní. Z tohoto důvodu je dostatečné množství komunikačních sběrnic umístěno na hlavní kartě, které jsou vytvářeny pomocí USB převodníků.

Na hlavní kartě jsou dostupné celkem 3 druhy sériových komunikačních rozhraní, jejichž použití je nejrozšířenější. Jedná se o rozhraní RS232 full-duplex a dále je k dispozici UART kompatibilní s 5 V i 3,3 V logikou. Posledním rozhraním je RS485.

Na backplanu je počítáno s rozvedením napětí z řiditelného zdroje pro napájení testovaného zařízení. K tomuto účelu jsou na backplanu vyhrazeny vodiče pro výstupní napětí ze dvou napájecích zdrojů. Řiditelný zdroj napětí a proudu by měl být realizován jako karta do subracku. Ovšem v případě potřeby je možné použít i jakýkoli jiný laboratorní zdroj, který je možné ovládat z PC.

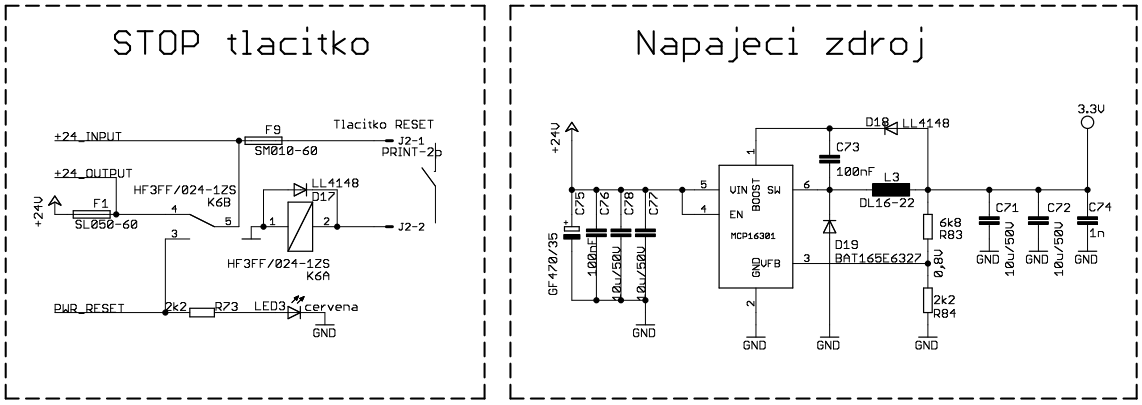
Na předním panelu se nachází 4 ovládací tlačítka. Jedno tlačítko slouží pro spuštění, případně pro zastavení testu. Další dvě tlačítka slouží pro potvrzení nebo odmítnutí aktuálního kroku. Potvrzování a odmítání se využívá, pokud je vyžadována interakce s obsluhou. Poslední tlačítko umožňuje vypnout napájení všem použitým kartám. Toto může být užitečné pokud bude třeba urychleně odpojit testované zařízení od napájení.

3.1 Napájení testeru

Napájení testeru je realizováno ze síťového adaptéru s výstupním stejnosměrným napětím 24 V. Síťový adaptér je zapojen do napájecího panelu umístěného v racku. Napájení z adaptéru je přivedeno přes vyhrazené vodiče na backplanu na hlavní kartu a odtud je znovu vyvedeno na backplane. Napětí vystupující na backplane je vedeno přes kontakty relé. Toto relé je ovládáno stop tlačítkem s aretací. Dojde-li ke stisknutí tlačítka, relé se sepne a celé napájení testeru včetně napájení hlavní karty je vypnuto. Vypnutý stav je signalizován červenou LED diodou zapojenou na NO kontakt relé.

Napájení odpínané stop tlačítkem je dále vedeno na backplane. Pomocí backplanu je napájení distribuováno na jednotlivé karty. Na jednotlivých kartách je

poté možné napájení dále upravovat podle potřeby. Na hlavní kartě se nachází měnič, který snižuje napětí z 24 V na backplanu na napětí 3,3 V, které je potřeba pro logiku hlavní karty.



Obr. 3.2: Schéma zapojení napájecích obvodů na hlavní kartě

Pro konstrukci byl vybrán obvod MCP16301 od firmy Microchip. MCP16301 je obvod určený pro snižující měniče. Pracuje se vstupním napětím v rozsahu od 4 V až do 30 V. Výstupní napětí je nastavitelné pomocí napěťového děliče v rozsahu od 2 V do 15 V. Maximální výstupní proud je 600 mA. [11]

Zapojení bylo vytvořeno na základě typického zapojení udávaného výrobcem. Hodnoty rezistorů nastavujících výstupní napětí byly vypočteny pomocí vzorce:

$$R_{83} = R_{84}\left(\frac{V_{OUT}}{V_{FB}} - 1\right) \tag{3.1}$$

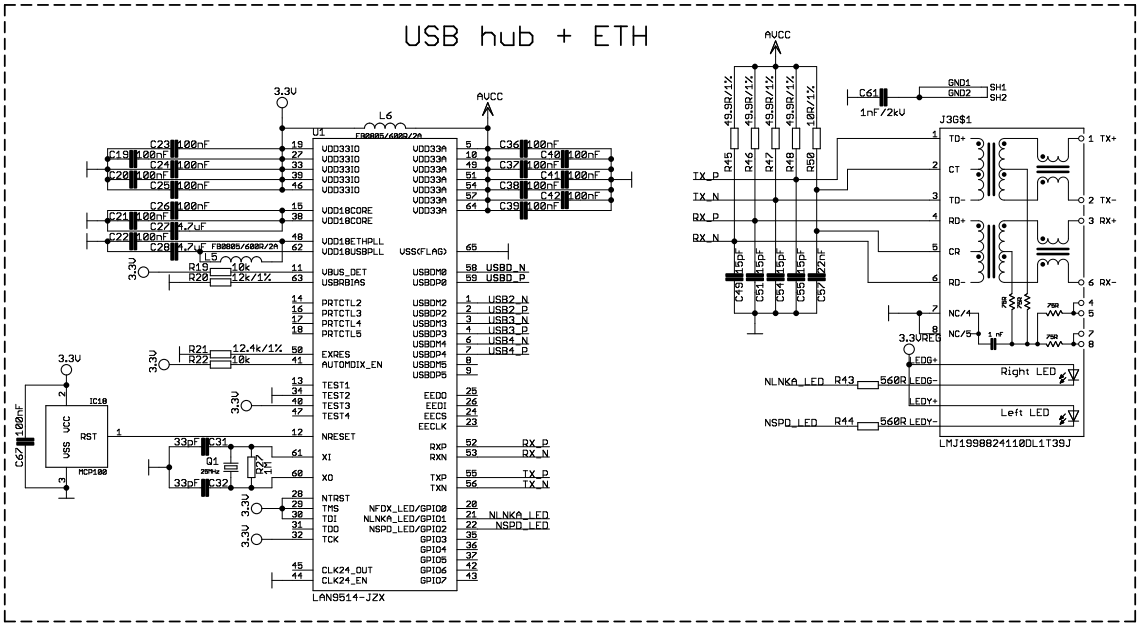
kde:

- R_{83} hodnota rezistoru v horní větvi napěťového děliče
- R_{84} hodnota rezistoru v dolní větvi napěťového děliče
- V_{OUT} výstupní napětí
- V_{FB} referenční napětí zpětné vazby 0,8 V

Výsledné hodnoty byly následně zaokrouhleny na hodnoty z řady E24. Parametry diod, tlumivky byly vybrány s ohledem na zatížení, spínací frekvenci a výstupní napětí tak jak specifikuje katalogový list. Podle parametrů byly vybrány konkrétní typy podle běžného sortimentu součástek firmy, která bude DPS osazovat.

3.2 USB hub

Na hlavní kartě je několik obvodů připojených k PC pomocí USB. Aby nebylo nutné připojovat kartu několika kabely je zde umístěn obvod realizující USB hub. Pro tento



Obr. 3.3: Schéma zapojení USB hubu

účel byl vybrán obvod LAN9514. Obvod realizuje USB hub se čtyřmi downstream porty a jedním upstream portem. Navíc obvod vytváří USB síťovou kartu s rozhraním Ethernet. [9]

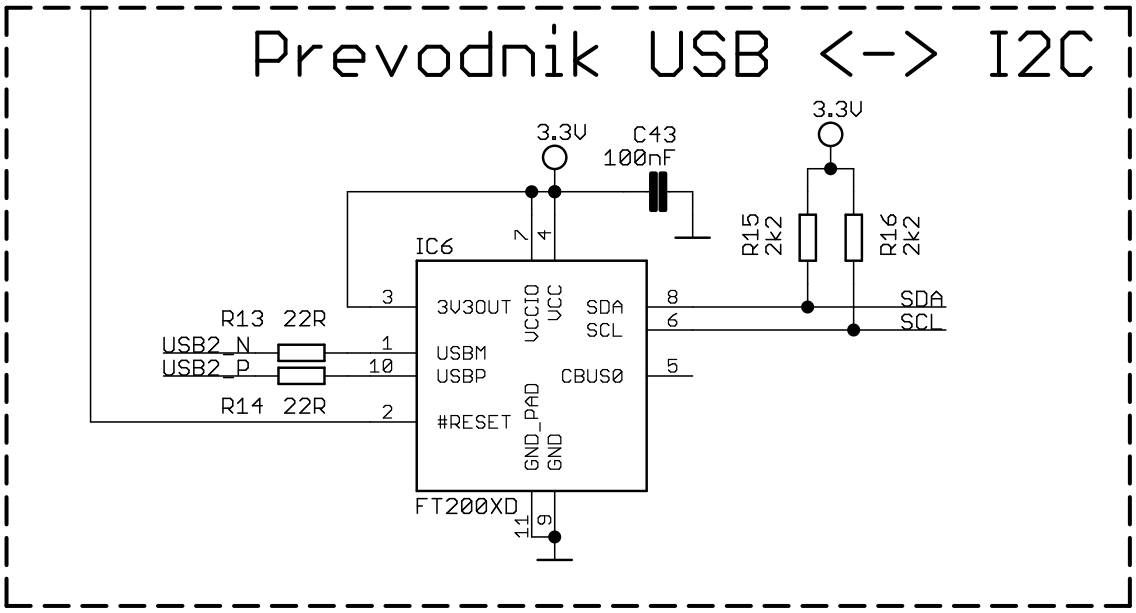
K obvodu je možné připojit EEPROM paměť, kam lze uložit MAC adresu Ethernetového rozhraní, vlastní USB deskriptory a nastavit několik parametrů (např. prohodit význam vodičů na USB lince). Paměť nebude použita, protože hub ve výchozím nastavení vyhovuje požadavkům a je tudíž zbytečné zvyšovat výrobní cenu EEPROM pamětí.

Schéma bylo vytvořeno na základě doporučení výrobce. Výjimkou je zapojení vývodů pro řízení napájení jednotlivých USB portů. Měly by zde být použity speciální obvody pro tento účel nebo obvod s diodou a vratnou pojistkou. Vzhledem k tomu, že jsou veškeré porty použity pouze na DPS a samotný tester umožňuje kompletní reset stop tlačítkem, nebyly řídicí piny použity. V případě, že by komunikace po USB nepracovala správně, může obsluha hub resetovat stiskem tlačítka totalstop.

Z hubu jsou vyvedeny 3 USB porty pro převodníky a ST-link. Ethernet je vyveden na přední panel pro účely testování konektivity. Pro Ethernet byl vybrán konektor se zabudovaným magnetickým obvodem, aby se zjednodušilo zapojení. Konektor také obsahuje dvojici LED indikujících spojení a aktivitu na lince.

Protože se jedná o citlivý obvod a je náchylný na špatnou inicializaci, byl použit resetovací obvod MCP100. Ten drží obvod hubu v resetu po dobu 300 ms od zapnutí napájení [10]. Do té doby se napájecí napětí ustálí a hub se správně inicializuje.

3.3 Převodník USB na I²C



Obr. 3.4: Schéma zapojení převodníku USB na I²C

Základním komunikačním rozhraním testeru je sběrnice I²C. Jako převodník z USB na I²C byl zvolen obvod FT200XD od firmy FTDI. Obvod komunikuje s PC přes USB pomocí protokolu D2XX. Obvod má jeden vývod sloužící pro různé účely. Vývodu lze nastavit jeho chování přes USB. Nastavení se ukládá do interní paměti a umožňuje upravit např. rychlost I²C sběrnice nebo velikost budicího proudu sběrnice. [5]

Zapojení obvodu je velice jednoduché. Jediné co obvod potřebuje je napájecí napětí v rozsahu 2,97 V až 5,5 V, přivést USB a vyvést sběrnici I²C. USB je odděleno pomocí dvojice rezistorů o hodnotě 22 Ω. Kvůli správné inicializaci je obvod resetován obvodem MCP100, který je sdílený spolu s hubem.

Sběrnice I²C má připojeny pull-up rezistory o hodnotě 2,2 kΩ. Maximální hodnota rezistorů se určuje z kapacity sběrnice, která ovšem není předem známa. Minimální hodnotu rezistorů lze určit podle vzorce: [8]

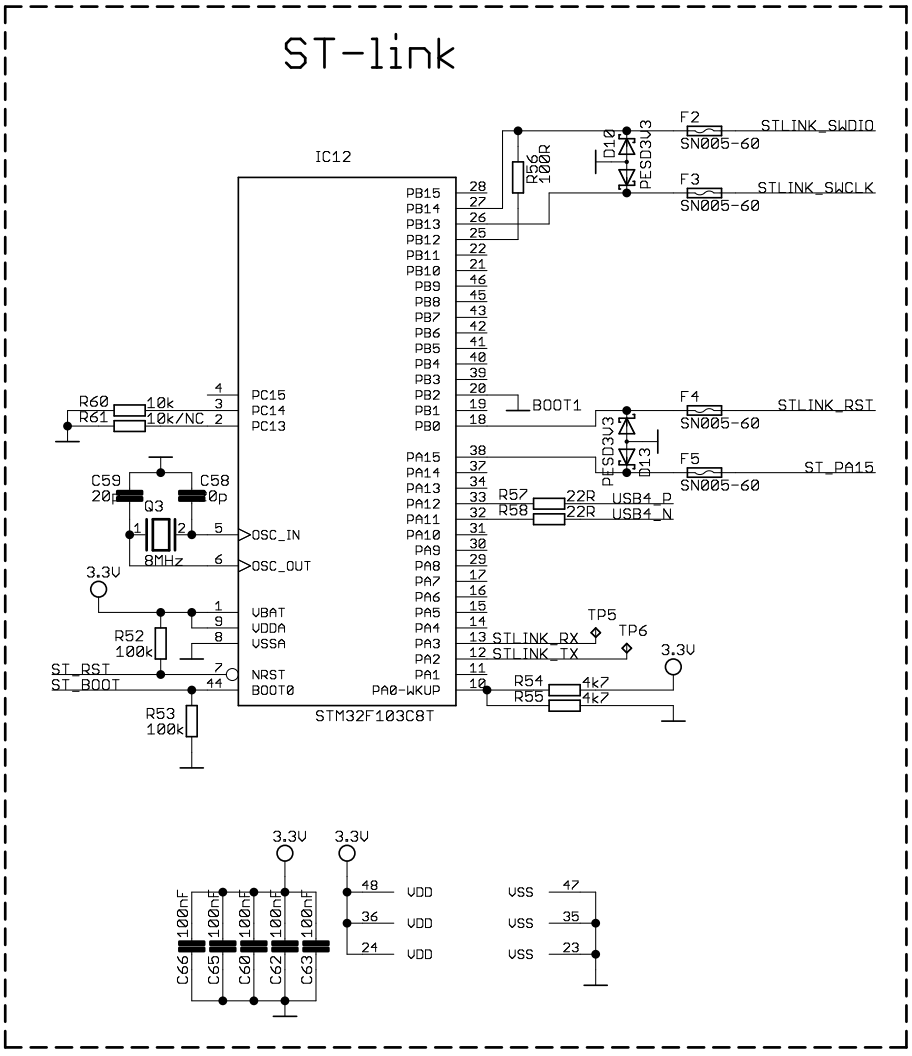
$$R_p(min) = \frac{V_{CC} - V_{OL}(max)}{I_{OL}} \tag{3.2}$$

kde:

$R_p(min)$	minimální hodnota pull-up rezistoru
V_{CC}	napájecí napětí
$V_{OL}(max)$	maximální hodnota napětí při nízké úrovni na sběrnici
I_{OL}	maximální proud pinu řídicího sběrnici

Hodnota I_{OL} je dána nastavením obvodu. Ve výchozím stavu je proud nastaven na 4 mA. Pro napájecí napětí 3,3 V je hodnota $V_{OL}(max) = 0{,}4\,\mathrm{V}$. Výsledná minimální hodnota pull-up rezistoru je 725 Ω. Hodnota 2,2 kΩ byla vybrána na základě zkušenosti tak, aby byla dodržena vypočtená minimální hodnota odporu. Hodnota rezistorů bude upravena podle výsledků testování komunikace.

3.4 Programátor ST-link



Obr. 3.5: Schéma zapojení programátoru ST-link

Součástí hlavní karty je MCU STM32F103, který slouží pro programování obvodů s rozhraním SWD. Výhodou umístění programátoru přímo na hlavní kartu je jednotné připojení DUT k testeru pomocí kabelového svazku. Není třeba další

zařízení, u kterého by se musela obsluha rozhodovat o tom, jestli ho při testování použít.

Další výhodou je odpojení programovacích vodičů pomocí relé. Díky tomu nemusí obsluha po naprogramování zařízení odpojovat programovací kabel. Po naprogramování se totiž provádí další testy, které by připojený programátor mohl ovlivňovat. Typickým případem je měření spotřeby, kde se zařízení obvykle dokáže z části napájet parazitně přes programovací rozhraní.

MCU bylo zvoleno podle MCU používaných na tzv. Discovery kitech od firmy STMicroelectronics. Kity vždy obsahují hlavní MCU a navíc druhé MCU, které slouží k jeho programování. K programovacímu MCU existuje FW, kterým jej lze updatovat. Tento FW bude nahrán do MCU na hlavní kartě. Tímto způsobem se předejde tomu, že se bude muset vytvářet vlastní implementace protokolu SWD. [15]

Aby bylo možné v případě potřeby programátor updatovat, je z MCU je vyveden pin BOOT0 a RST. Pomocí těchto pinů je možné MCU přepnout do továrního bootloaderu, který podporuje protokol DFU. Piny jsou vyvedeny na GPIO expandér.

Možnost přeprogramování MCU má ještě jednu výhodu a tou je, je možnost nahrát do MCU vlastní firmware. Díky tomu lze implementovat vlastní programovací protokoly např. ISP protokol pro programování MCU z rodiny AVR. Proto je vyvedeno více vodičů než je nezbytně nutné pro SWD rozhraní.

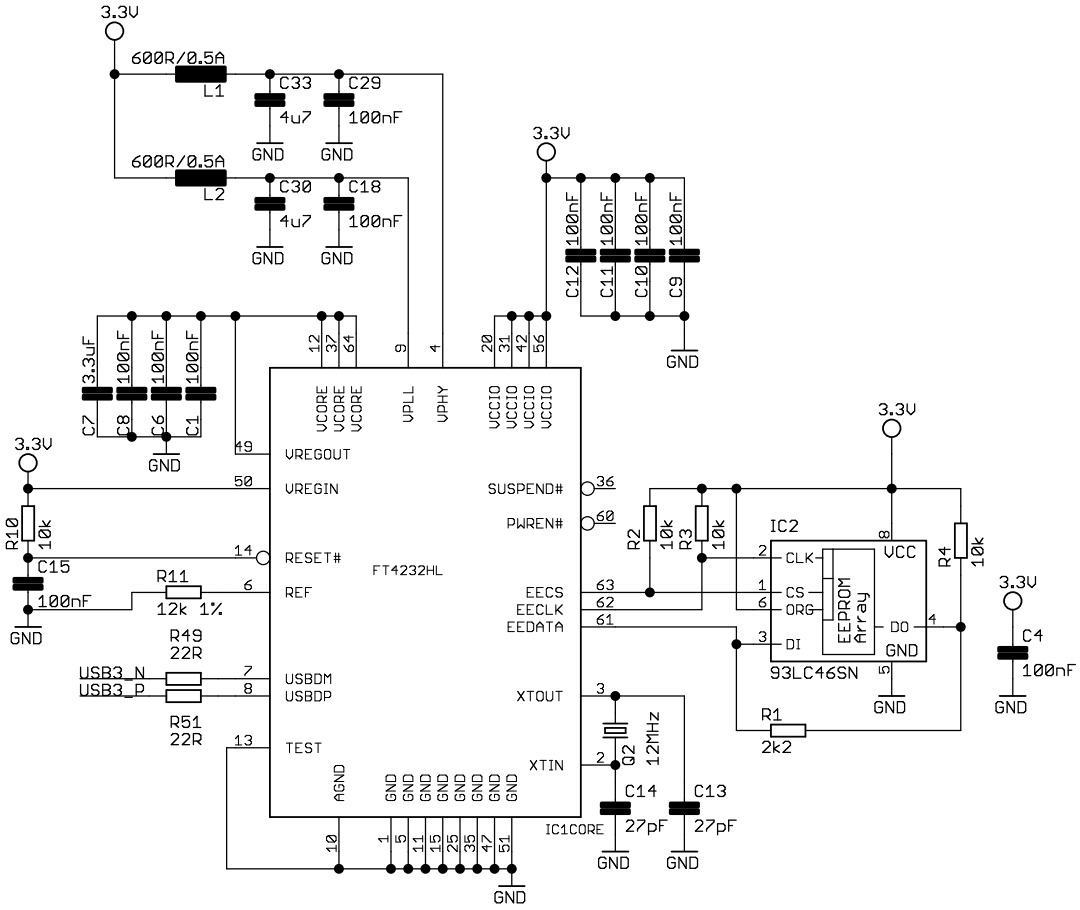
Signály, které jsou vyvedeny z DPS, jsou ošetřeny proti ESD pomocí transilu a vratné pojistky, viz obr. 3.5. Jako transil byl vybrán PESD3V3, který má pracovní napětí 3,3 V.[20] Vratná pojistka SN005-60 má maximální pracovní proud 50 mA a maximální pracovní napětí 60 V. [17]

Rozhraní SWD je vyvedeno jen na přední panel hlavní karty. SWD obvykle nepodporuje více zařízení na lince a navíc je citlivé na délku vodičů, proto nejsou signály vyvedeny na backplane.

3.5 Převodník USB na UART

Většina zařízení, která se při výrobě testují, využívá pro komunikaci některý druh sériového rozhraní. Pokud zařízení sériové rozhraní nepoužívá pro výslednou funkcionalitu, je k dispozici alespoň pro servisní účely a ladění při vývoji. Z tohoto důvodu bylo nutné zajistit možnost komunikace mezi DUT a testerem pomocí sériové komunikace.

Pro vytvoření dostatečného množství sériových komunikačních linek byl využit integrovaný obvod FT4232HL. Obvod vytváří na USB sběrnici 4 VCP kanály, které poté převádí na UART nebo na univerzální rozhraní MPSSE. [6]



Obr. 3.6: Schéma zapojení převodníku USB na UART

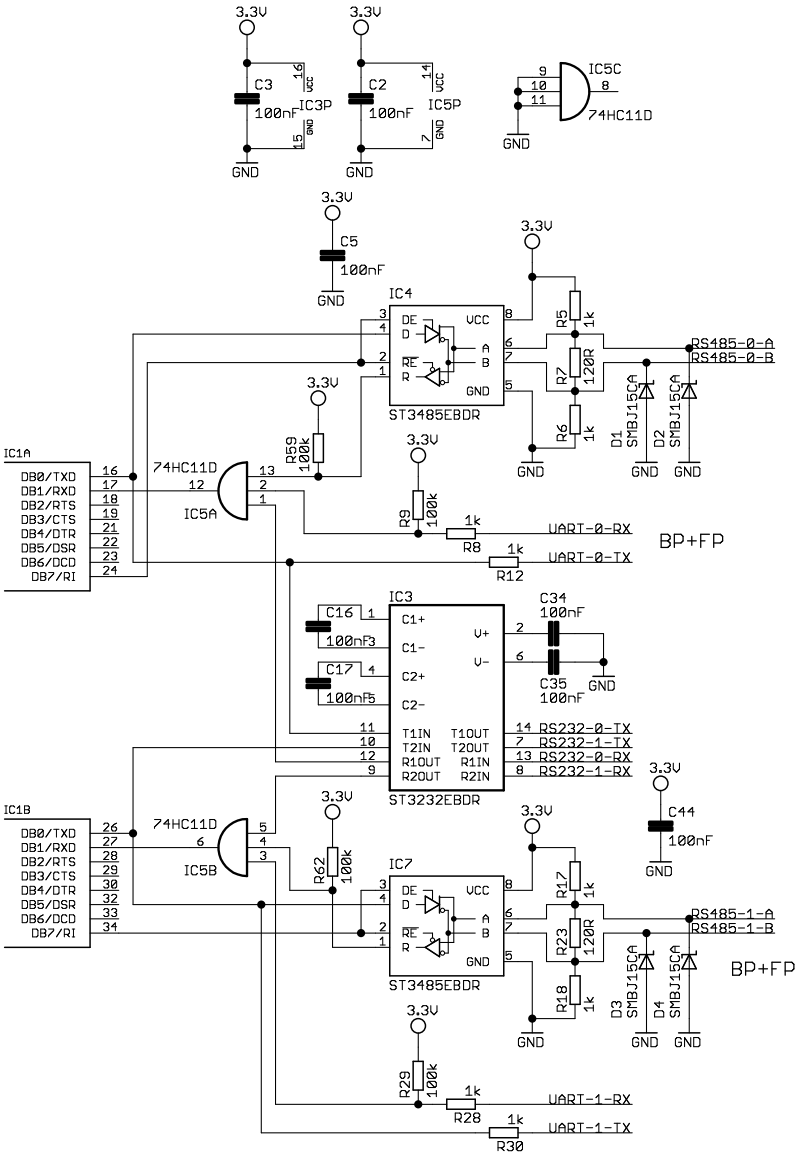
Pro účely testeru byl obvod zapojen pro vytvoření rozhraní UART. Obvod vyžaduje precizně provedené napájení zejména pro interní obvody PLL. Proto jsou navrženy LC filtry pro odrušení. Dále je k obvodu připojen 12 MHz krystal a paměť EEPROM.

Paměť slouží pro ukládání nastavení obvodu. Obvod umožňuje činnost i bez paměti. Protože je však UART dále převáděn na sběrnici RS485, je nutné nastavit obvod FT4232 tak, aby na jednom z pinů generoval signál přepínající vysílání a příjem na RS485. Toto nastavení je možné uložit pouze do externí EEPROM.

Jako EEPROM byl vybrán obvod 93LC46SN podle doporučení výrobce čipu FT4232. EEPROM má kapacitu 1 KiB. Do paměti je možné uložit vlastní USB deskriptory, sériové číslo atd.

V praxi se kromě rozhraní UART častěji používají rozhraní RS485 a RS232. Obě rozhraní používají jiné napěťové úrovně než UART. Z toho důvodu bylo třeba doplnit stávající převodník ještě obvody, zajišťujícími převod UART na jednotlivé rozhraní.

Pro převod na rozhraní RS485 byl použit obvod ST3485EBDR. Obvod pracuje s napájecím napětím v rozmezí od 3 V do 3,6 V. Rychlost přenosu na RS485 může s použitím daného obvodu dosáhnout až 12 Mbps. [19]



Obr. 3.7: Schéma zapojení převodníku UART na RS485 a RS232

Obvod má dva signály pro povolení vysílače a přijímače. Protože signály mají opačnou logiku, jsou spojeny do jednoho signálu pro řízení směru a tento je připojen na převodník FT4232. Sběrnice RS485 je zakončena rezistorem o hodnotě 120 Ω, jehož hodnota je dána specifikací RS485. [21] K interním ochranám proti ESD, které jsou zabudované v obvodu ST3485ECDR, byly přidány ještě dva transily pro zvýšení odolnosti.

Pro převod UART na RS232 byl použit obvod ST3232EBDR, který zajišťuje

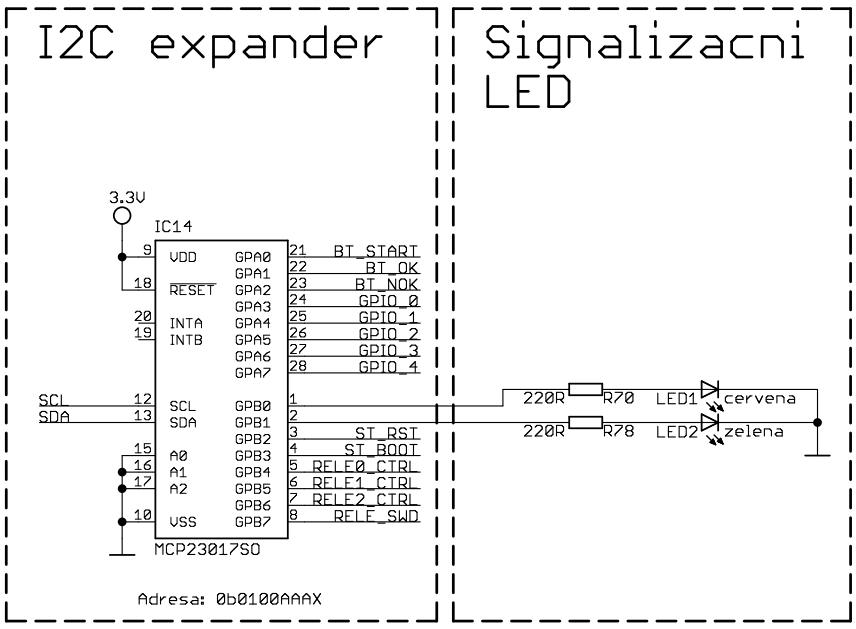
převod úrovní na úrovně dané specifikací RS232. Použitý obvod je dvojitý a proto je sdílen mezi dvěma kanály, viz obrázek 3.7. [18]

Vysílací část každého obvodu je zapojena přímo na vstupy převodníků, případně vyvedena na konektor. Přijímací část je sloučena pomocí třívstupového hradla AND. Signály na UART jsou v klidovém stavu v log. 1. Pokud začnou po některém ze vstupů přicházet data (log. 0), dojde k přepnutí výstupu AND hradla do log. 1. Hradlo AND tedy slučuje všechny přijímací signály do jednoho, který je veden do převodníku FT4232.

Všechny rozhraní jsou zapojeny neustále a je možné je kdykoli využít bez přepínání. Z toho plyne omezení, že nepoužitá rozhraní na každém kanálu musí být v neaktivním stavu. Tento stav je vynucen přímo zapojením. Je tedy dostačující nepoužité vstupy nechat nezapojené.

Na obrázku 3.7 je schéma pouze jedné dvojice kanálů. Druhá dvojice má topologii zapojení stejnou. Všechny signály všech komunikačních rozhraní jsou vyvedeny na konektory. Protože mají konektory omezenou velikost jsou dva kanály přivedeny na backplane i na konektor na předním panelu. Druhé dva kanály jsou rozděleny a jsou zapojeny každý na jeden z konektorů.

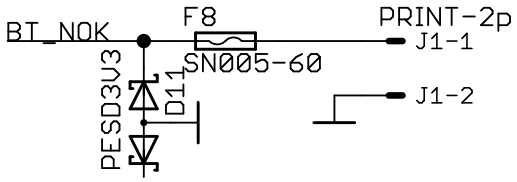
3.6 Vstupy a výstupy



Obr. 3.8: Schéma zapojení I^2C expandéru a signalizačních LED

Tester pro svou činnost potřebuje vstupy a výstupy. K tomuto účelu je na hlavní

kartě použit I^{2}C expandér MCP23017. Obvod vytváří dva osmibitové obousměrné porty. Každý pin obvodu lze nezávisle nastavovat jako vstup nebo výstup. Na každém portu lze také využít přerušení, které aktivuje výstupy INTA a INTB. Adresu lze nastavit pomocí třech pinů A0-A2. Jak bylo popsáno výše, na hlavní kartě je adresa napevno nastavena na hodnotu 0.

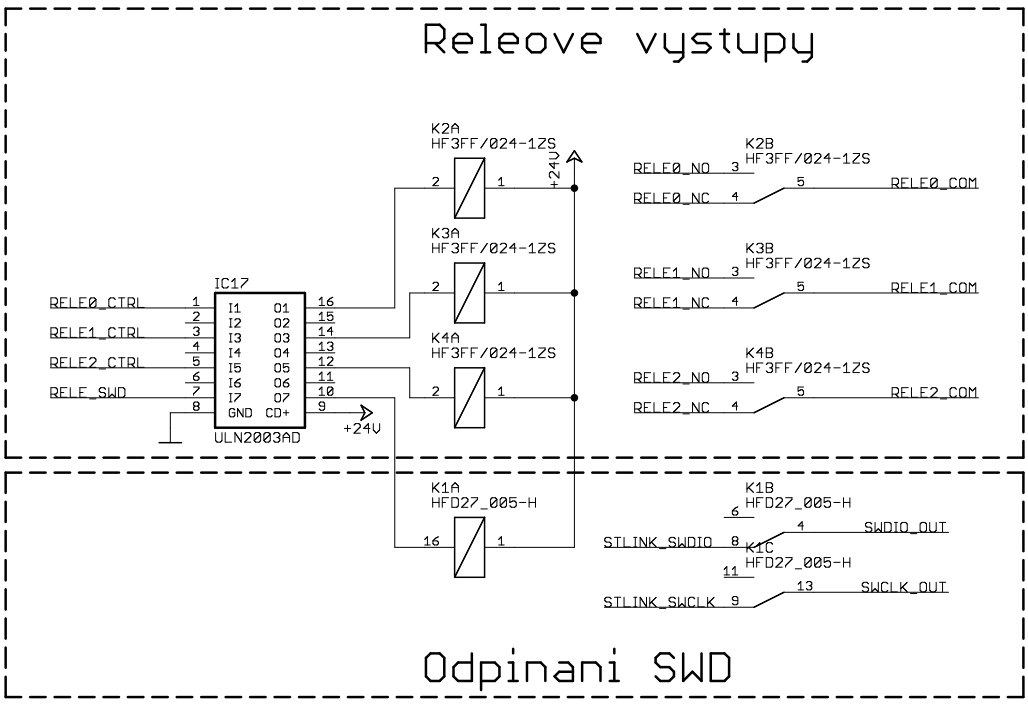


Obr. 3.9: Schéma zapojení ovládacích tlačítek

Signalizační LED jsou připojeny přímo na vývody expandéru, který je schopen na každým pinu dodávat proud až 20 mA. [12] Hodnoty předřadných rezistorů byly vypočteny na základě znalosti napětí na diodě v rozsvíceném stavu a proudu, který by jí měl protékat. Výpočet byl proveden podle ohmova zákona:

$$R = \frac{U_Z - U_D}{I} = \frac{3.3 - 2.1}{0.005} = 240\,\Omega \tag{3.3}$$

Nejbližší běžně vyráběná hodnota odporu je 220 Ω.



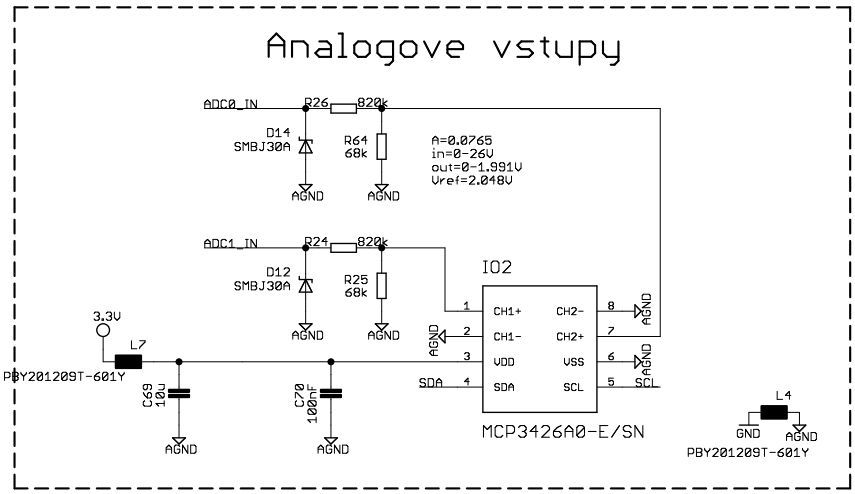
Obr. 3.10: Schéma zapojení relé

K expandéru jsou připojena 3 tlačítka. Schéma zapojení tlačítek je vidět na obrázku 3.9. Vstupy pro připojení tlačítek jsou chráněny proti ESD vratnou pojistkou a transilem. Pro správnou funkčnost tlačítek musí mít expandér pro příslušné vývody povoleny pull-up rezistory.

Expandér ovládá trojici relé, jejichž kontakty jsou vyvedeny jak na backplane, tak na konektor na předním panelu. Čtvrté relé slouží pro odpínání programovacího rozhraní SWD. Schéma je na obrázku č. 3.10.

Relé jsou spínána pomocí obvodu ULN2003, který obsahuje tranzistory v darlingtonově zapojení. Tímto obvodem byl nahrazen klasický obvod s tranzistorem, rezistorem a diodou. Důvodem je urychlení výroby a její celkové zlevnění. Jediný obvod zde dokáže nahradit celkem 12 součástek, které by se musely osazovat samostatně.

3.7 Analogové vstupy



Obr. 3.11: Schéma zapojení analogových vstupů

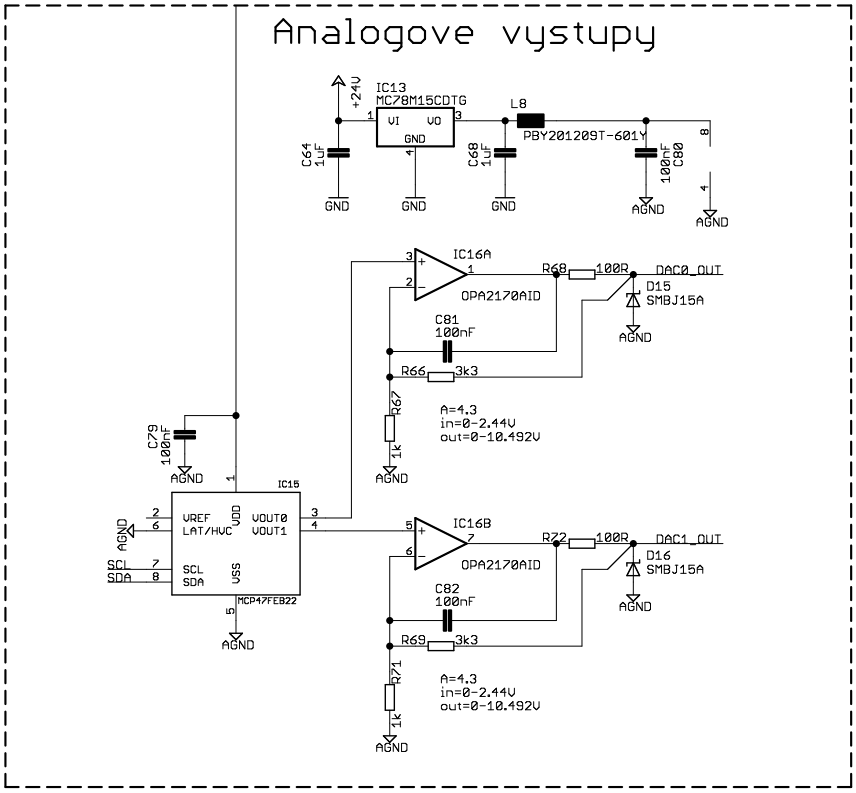
Pro možnost měření analogových hodnot hlavní karta disponuje analogově digitálním převodníkem. Obvod MCP3426A0 je 18 bitový, dvoukanálový analogově digitální převodník. Převodník převádí metodou sigma-delta s maximální vzorkovací frekvencí 240 Hz. Převodník má programovatelné zesílení ve čtyřech úrovních. Vybraný obvod má adresu z výroby nastavenou na 0. [13]

Zapojení je vytvořeno tak, aby bylo možné měřit napětí o maximální velikosti 26 V. Přičemž vstup převodníku bude vybuzen napětím 1,991 V. Referenční napětí je 2,048 V. Proti ESD jsou vstupy převodníku chráněny transily SMBJ30A, které mají maximální pracovní napětí 30 V.

Napájení převodníku je odděleno od napájecí větve pro digitální obvody LC článkem. Analogové vstupy jsou vyvedeny na konektor na předním panelu i na backplane.

3.8 Analogové výstupy

Aby mohl tester generovat analogové budicí signály, je hlavní karta vybavena digitálně analogovým převodníkem. Obvod MCP47FEB22 je 12 bitový, dvoukanálový DAC převodník. Obvod má integrovanou referenci 1,22 V, možnost měnit zesílení ve dvou krocích a možnost synchronizace obou výstupů. [14]



Obr. 3.12: Schéma zapojení analogových výstupů

Zapojení je vytvořeno tak, aby bylo možné generovat napětí v rozsahu 0 – 10 V. Proto je za převodník zařazen operační zesilovač v neinvertujícím zapojení. Maximální výstupní napětí převodníku může být 2,44 V při zesílení 2 a interní referenci 1,22 V. Zesílení OZ je nastaveno tak, aby při plném vybuzení převodníku bylo výstupní napětí přibližně 10 V. Z hodnot vstupního a výstupního napětí je možné určit požadované zesílení OZ:

$$A = \frac{U_{OUT}}{U_{IN}} = \frac{10}{2.44} = 4{,}098\,\mathrm{V} \tag{3.4}$$

Známe-li požadované zesílení, je možné navrhnout hodnoty rezistorů. Hodnoty musí splňovat:

$$A = 1 + \frac{R_{66}}{R_{67}} \tag{3.5}$$

Ze vzorce zjistíme, že kombinací rezistorů $R_{66} = 3{,}3\,\mathrm{k\Omega}$ a $R_{67} = 1\,\mathrm{k\Omega}$ se zesílení dostatečně přiblíží požadované hodnotě. Výsledné zesílení s použitím navržených hodnot je 4.3. Maximální dosažitelné napětí při použití interní reference je 10,492 V.

Aby nedocházelo k rozkmitání zesilovače, je do zpětné vazby přidán kondenzátor. Kondenzátor zpomaluje rychlost přeběhu výstupního napětí, a tím zajišťuje stabilitu obvodu. Na výstupu zesilovače je omezen proud pomocí sériového rezistoru.

Zesilovač pro svou činnost potřebuje napětí vyšší než je maximální výstupní napětí. Bylo nutné vytvořit nový napájecí zdroj s výstupním napětím alespoň 12 V. Toho bylo dosaženo pomocí obvodu MC78M15. Obvod je lineární stabilizátor s výstupním napětím 15 V. Stabilizátor je napájen z napájecího napětí 24 V. Na výstupu stabilizátoru je zařazen LC filtr tak, aby se snížilo rušení vnikající do analogové části obvodu.

Pro zesilovač byl vybrán obvod operačního zesilovače OPA2170. Vybraný OZ pracuje při nesymetrickém napájení v rozsahu 2,7 do 36 V. Obvod má rail-to-rail vstup i výstup, což je důležité pro napájení z jediného zdroje. Vstupní napěťová nesymetrie je maximálně 1,8 mV. Obvod má v jednom pouzdře dva OZ. [16]

3.9 Zapojení konektorů

Na hlavní desce jsou celkem 3 konektory. První konektor typu DIN 41612 je určen pro propojení s backplanem. Jedná se o dvouřadý konektor s 64 piny. Existují i verze třířadé s 96 piny, díky kterým by bylo možné vyvést více signálů. Hlavní výhodou menšího konektoru je existence protikusů, které lze nakrimpovat na plochý IDC kabel. Toho je využito při výrobě backplanu. Backplane se vyrábí v plně propojené verzi jako DPS s konektory, jeho cena je však vysoká. Proto bude backplane vytvořen krimpováním konektorů na plochý kabel. Díky tomu bude také možné bezproblémově vytvořit požadované adresování karet.

Rozmístění signálů na konektoru a tedy i na backplanu je v tabulce 3.1. Signály USB + a USB - budou vedeny jen na první konektor, který je vyhrazen pro hlavní kartu. Pomocí signálů A0 až A3 budou adresovány jednotlivé pozice. Ostatní signály jsou rozvedeny přes celý backplane.

Druhý konektor je umístěn na předním panelu. Jedná se o dvouřadý konektor IDC se 40 piny. Na konektor jsou vyvedeny komunikační linky, ST-link GPIO a jeden z výstupů regulovatelného zdroje. Výstup regulovatelného zdroje je zde umístěn

	A	C
1	USB +	USB -
2	GND	GND
3	24V IN	24V IN
4	24V OUT	24V OUT
5	GND	GND
6	PWR RESET	ADDR 0
7	ADDR 1	ADDR 2
8	I2C SDA	I2C SCL
9	REG PWR 0	REG PWR 0
10	REG PWR 1	REG PWR 1
11	GND	GND
12	P0 RS485 A	P0 RS485 B
13	P0 RS232 RX	P0 RS232 TX
14	P0 UART RX	P0 UART TX
15	P1 RS485 A	P1 RS485 B
16	P1 RS232 RX	P1 RS232 TX

	A	C
17	P1 UART RX	P1 UART TX
18	P2 RS485 A	P2 RS485 B
19	P2 RS232 RX	P2 RS232 TX
20	P2 UART RX	P2 UART TX
21	GND	GND
22	ADC 1	ADC 0
23	DAC 0	DAC 1
24	GND	GND
25	GPIO 0	GPIO 1
26	GPIO 2	GPIO 3
27	GPIO 4	GND
28	RELE 0 COM	RELE 0 NC
29	RELE 0 NO	RELE 1 COM
30	RELE 1 NC	RELE 1 NO
31	RELE 2 COM	RELE 2 NC
32	RELE 2 NO	GND

Tab. 3.1: Rozvržení signálů na backplanu

proto, aby bylo možné využít jen jediný konektor pro připojení DUT. Pro jednoduchá zařízení nemusí být potřeba nic víc než je na jednom z konektorů. Regulovatelný zdroj je potřeba pro každé testování a je tedy vhodné aby byl přístupný na všech konektorech. Rozvržení signálů na konektoru je popsáno v tabulce 3.2.

Poslední konektor je typu Canon 25. Konektor je umístěn na předním panelu. Jsou na něm vyvedeny signály, které se nevešly na konektor IDC 40. Nutnost použití dvojice konektorů na předním panelu je dána omezenou velikostí karty. Jednotný konektor běžného typu by se na přední panel nevešel. Bylo by tedy nutné použít netypický konektor nebo více běžných konektorů. Vzhledem k ceně atypických konektorů byla zvolena druhá možnost. Aby nemohlo dojít k vzájemné záměně byly použity konektory dvou typů.

Konektory jsou umístěny nad sebou. Aby bylo toto uspořádání možné realizovat, je konektor Canon připojen plochým kabelem na DPS. Na DPS se nachází IDC konektor do nějž se Canon připojí. Díky tomu se uvolnil prostor na panelu a zároveň jsou vyvedeny všechny potřebné signály.

40	P0 RS485 A	P0 RS485 B	**39**
38	P0 RS232 RX	P0 RS232 TX	**37**
36	P0 UART RX	P0 UART TX	**35**
34	P1 RS485 A	P1 RS485 B	**33**
32	P1 RS232 RX	P1 RS232 TX	**31**
30	P1 UART RX	P1 UART TX	**29**
28	P3 RS485 A	P3 RS485 B	**27**
26	P3 RS232 RX	P3 RS232 TX	**25**
24	P3 UART RX	P3 UART TX	**23**
22	REG PWR 0	REG PWR 0	**21**
20	GND	GND	**19**
18	ADC 0	ADC 1	**17**
16	DAC 0	DAC 1	**15**
14	GND	GND	**13**
12	GPIO 0	GPIO 1	**11**
10	GPIO 2	GPIO 3	**9**
8	GPIO 4	GND	**7**
6	STLink SWDIO	STLink SWCLK	**5**
4	STLink reset	STLink PA15	**3**
2	GND	GND	**1**

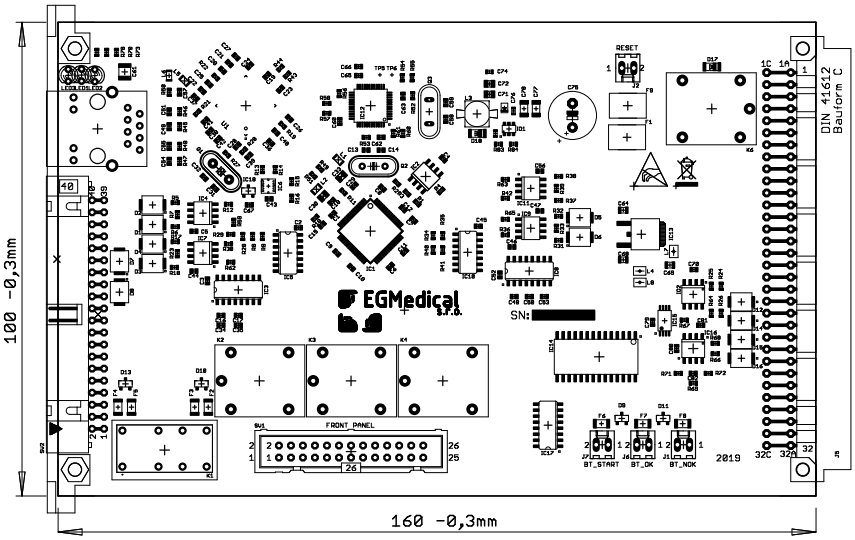
Tab. 3.2: Rozvržení signálů na IDC 40 konektoru na předním panelu

1	24V OUT	24V OUT	**14**
2	GND	GND	**15**
3	REG PWR 0	REG PWR 0	**16**
4	GND	GND	**17**
5	REG PWR 1	REG PWR 1	**18**
6	GND	GND	**19**
7	RELE 0 COM	RELE 0 NC	**20**
8	RELE 0 NO	RELE 1 COM	**21**
9	RELE 1 NC	RELE 1 NO	**22**
10	RELE 2 COM	RELE 2 NC	**23**
11	RELE 2 NO	GND	**24**
12	GND	GND	**25**
13	GND	GND	**26**

Tab. 3.3: Rozvržení signálů na konektoru Canon 25 na předním panelu

3.10 Deska plošných spojů

Deska plošných spojů byla navržena v návrhovém programu Eagle. Deska je oboustranná s rozměry $100 \times 160\,\text{mm}$. Formát desky je také označován jako eurokarta. Velikost desky je daná specifikací systému rack.



Obr. 3.13: Rozmístění součástek na DPS

Při návrhu DPS bylo třeba dbát na signálovou integritu USB a Ethernetu. Všechny tyto signály jsou na DPS vedeny jako diferenciální páry. Jejich délka je poté upravena pomocí meandrů tak, aby vodiče byly stejně dlouhé.

Při návrhu bylo třeba zajistit oddělení digitální a analogové části. To je provedeno pomocí tlumivek a rozlévané mědi pro jednotlivé části separátně. Tyto části jsou spojeny jen v jednom bodě tak aby případným protékajícím proudem nebylo ovlivněno měření.

Rozložení součástek je ukázáno na obrázku č. 3.13.

3.11 Tvorba aplikačně specifických karet

Jelikož se jedná o rozšiřitelné zařízení je možné vytvářet karty podle potřeb testovaných zařízení. Aby karty s testerem správně fungovaly, musí být navrhovány s ohledem na tuto skutečnost.

Nejdůležitější je, aby specifické karty měly kompatibilní rozložení konektoru na backplanu. Tyto karty nemají na backplanu žádný prostor pro vlastní signály, protože by snadno vznikla kolize mezi různými kartami. Musí být ovšem zajištěno spojení napájecích vývodů a I^2C sběrnice. Specifické karty mohou signály, které jsou na

backlanu vyvedeny, využívat a dodávat jim tak novou funkcionalitu. Je však nutné zajistit, aby se karty při vkládání do testeru navzájem neovlivňovaly. Mohlo by k tomu dojít v případě že alespoň dvě karty budou používat stejné signály. Naopak v některých případech může být takové sdílení signálů vyžadováno.

Adresování na backplanu je zajištěno přivedením nulového napětí na piny konektoru, kde má být logická 0. Pokud má být na adresním pinu logická 1 není vývod konektoru zapojen. Proto musí každá karta mít adresovací signály taženy k napájecímu napětí 3,3 V pull-up rezistorem o vhodné hodnotě (např. 10 kΩ). Pokud I^2C zařízení na kartě nepoužívá všechny adresní vodiče, měly by být primárně použity vodiče s nižším číselným označením.

Protože hlavní karta má tlačítko totalstop, které při stisku odpojí vodiče 24V_OUT od napájení, je vhodné, aby se karta napájela právě z těchto signálů. Výjimkou může být karta u které je vyžadováno jiné chování než odpojení napájení. V takovém případě může být napájena z vodičů 24V_IN. Pro reakci na totalstop je poté nutné použít vodič PWR_RESET, který bude v případě stisku tlačítka připojen na napájecí napětí 24 V.

Dodržením výše zmíněných doporučení lze vytvářet karty pro testování velkého množství různých zařízení jedním testerem. Protože karta má standardizovaný rozměr je možné koupit prototypovací desku určenou do subracku. Cena prototypovací desky je násobně nižší než výroba DPS s vlastním vzorem. Díky tomu lze ušetřit při vytváření jednodušších karet.

4 Softwarové vybavení testeru

Pro úspěšný návrh realizace SW je nutné si nejprve ujasnit co musí SW dělat a jakým způsobem se budou definovat testy.

Způsoby definování testů jsou dva. Test může být přímo součástí kódu testeru nebo může být oddělen. Z předchozích zkušeností s testery se zabudovaným postupem testování vyplynulo, že bude použito odděleného způsobu definování testu.

Výhodou tohoto způsobu je snadné vytváření nových testů, kde stačí jen vytvořit nový předpis, který tester provede. Další výhodou je výborná udržovatelnost, zvláště pokud je test definován pomocí speciálně navrženého zápisu, který je k tomuto účelu uzpůsoben. Nevýhodou je vyšší pracnost implementace takového způsobu definování testů.

Z výše uvedených požadavků bylo navrženo, že testovací předpisy se budou definovat pomocí značkovacího jazyka XML. XML je jazyk určený zejména pro ukládání/sdílení dat ve strukturované podobě. Protože XML nedefinuje konkrétní formát dokumentu, je s jeho pomocí možné vytvářet vlastní specializované značkovací jazyky.

Požadavky na SW testeru jsou:

- komunikace s HW,
- načtení a vykonání testovacího předpisu,
- informovat obsluhu o průběhu testování,
- umožnit obsluze jednoduchým způsobem tester ovládat,
- ukládat reporty z testování.

Pro tvorbu SW testeru byl vybrán programovací jazyk Python. Jedná se o vysokoúrovňový skriptovací jazyk. Hlavním důvodem proč byl zvolen jazyk Python je právě to, že se jedná o skriptovací jazyk a ne o jazyk kompilovaný. Díky tomu je možné v rámci běžícího programu spouštět kód, který je definován mimo něj a případně byl načten až po jeho spuštění. V kompilovaných jazycích je to možné také, ale za cenu mnohem většího úsilí.

Možnost spouštět externí kód bude použita v testovacích předpisech, kam bude možné zapisovat kód. Tímto způsobem se zajistí dostatečná flexibilita testeru.

4.1 Testovací předpis

Pro návrh testovacího předpisu je nejprve třeba si ujasnit jak každý test probíhá. Postup testování každého zařízení lze rozdělit na jednotlivé kroky. Každý krok testuje jednu funkcionalitu, např.: reakci na vstup, měření spotřeby, schopnost komunikace po určitém rozhraní apod. Během jednotlivých kroků se provádějí akce potřebné

pro správné otestování dané funkcionality, např.: nastavení napětí na vstupu zařízení, propojení signálů zařízení apod. Jedná se tedy o jistý druh měření odezvy zařízení na vstupní signály. Přičemž se kontroluje zda se zařízení chová tak jak je očekávané. Následující krok se začne vykonávat až po splnění podmínek kroku právě probíhajícího, tj. když se zařízení chová tak je očekávané.

Jelikož tester není schopen zajistit některé činnosti sám, typicky připojení / odpojení zařízení nebo stisky příslušných tlačítek, musí být testování přítomna obsluha. Aby obsluha mohla vykonat požadovanou akci, musí o tom být informována. Z tohoto důvodu musí být součástí jednotlivých kroků pokyny pro obsluhu. Pokyny musí být v přehledném formátu s možností zvýraznění kriticky důležitých částí.

Testovací předpis je XML dokument s kořenovým elementem *<test>*. V atributech kořenového elementu je definováno jméno testovaného výrobku, datum poslední změny testovacího předpisu, jméno autora a verze předpisu. Kořenový element může obsahovat jeden element *<user-commands>* a několik elementů *<step>*.

Element *<user-commands>* slouží pro načtení uživatelských příkazů. Element může obsahovat pouze elementy *<module>* s atributem `name`, který odkazuje na jméno modulu v Pythonu který bude načten jako nový příkaz. Podrobnosti o vytváření uživatelských příkazů viz 4.3.8.

Element *<step>* slouží pro popis průběhu jednoho kroku. V atributech jsou obsaženy obecné informace o daném kroku a jeho nastavení:

Atribut	**Význam**	**Hodnoty**	**Výchozí**
skippable	Povoluje možnost pokračovat v testu i v případě selhání daného kroku.	true, false	false
disabled	Zakázání vykonání kroku.	true, false	false
title	Název kroku.	řetězec	prázdné
report-message	Text zobrazovaný v reportu.	řetězec	prázdné
timeout	Omezení doby vyhodnocování kroku. Po uplynutí dané doby krok selže.	čas	žádný timeout
duration	Doba po kterou musí být podmínka splněna.	čas	žádný
timing	Základní časování kroku. Příkazy kroku jsou volány s touto periodou.	čas	100 ms

Tab. 4.1: Význam atributů elementu *<step>*

Element *<step>* obsahuje tři typy vnořených elementů: instrukce pro obsluhu, bloky a příkazy.

Instrukce pro obsluhu se zapisují tagem *<instruction>*. Instrukce jsou ve formátu Markdown. Instrukce by měly obsahovat informace o tom co se po obsluze žádá, co nesmí zapomenout udělat, co musí zkontrolovat, potvrdit apod.

```
<?xml version='1.0' encoding='utf-8'?>
<test product="název výrobku" created="datum vytvoření"
    author="jméno a příjmení autora" version="verze">
    <user-commands>
        <module name="jméno modulu s uživatelsky
            definovanými příkazy"/>
    </user-commands>
    <step skippable="lze překakovat?"
        disabled="vykonat tento krok?" title="název kroku"
        report-message="text do reportu"
        timeout="maximální doba trvání"
        timing="perioda vykonávání">
        <instruction>
            instrukce pro obsluhu ve formátu HTML
            včetně interního stylopisu
        </instruction>
        <at-begin>
            příkazy k vykonání na začátku kroku
        </at-begin>
        <at-end>
            příkazy k vykonání na začátku kroku
        </at-end>
        <condition delay="zpoždění vyhodnocování"
            period="perioda vyhodnocování" duration="
            minimální doba trvání pravdivosti podmínky">
            podmínky ukončení kroku
        </condition>
    </step>
</test>
```

Výpis 4.1: Struktura testovacího předpisu

Markdown byl zvolen s ohledem na to, že testovací předpis je ve formátu XML. Není tedy dobré, aby uvnitř byl formát, který je na XML založen. Mohlo by docházet k různým duplicitám. HTML formát, který byl také uvažován, nebyl z tohoto důvodu použit, i když by z praktického hlediska byl výhodnější. Další důvod proč nepoužít HTML formát je to, že jeho specifikace není natolik striktní ohledně chyb. Například v XML je za vážnou chybu, kvůli které bude parser přerušen, považováno neuvedení lomítka v nepárovém tagu. Naproti tomu to HTML dovoluje. Kvůli těmto rozdílům by musel být HTML obsah escapován nebo obalen do sekce `CDATA`. Toto je velice nepraktické z hlediska vytváření testovacích předpisů.

Bloky mají význam jen ve spojení s příkazy, které jsou do bloků vnořeny. Bloky

určují, jak se bude s danými příkazy zacházet. Bloky jsou celkem 4. Blok at-begin se zapisuje jako *<at-begin>* a obsahuje příkazy, které se provedou jako první po spuštění daného kroku. Obdobně příkazy v bloku at-end (*<at-end>*) se provedou jako poslední po ukončení kroku ještě před tím, než se spustí nový krok nebo uloží report. Oba příkazy se provedou pouze jednou.

Příkazy v bloku periodic (*<periodic>*) se provádějí periodicky. Bloků může být v jednom kroku víc s různě nastaveným časováním. Blok condition se zapisuje jako tag *<condition>* a slouží pro zápis podmínky ukončení testu. Příkaz umožňuje zadat minimální dobu po kterou musí být podmínka platná aby se krok ukončil. Minimální doba pravdivosti podmínky se zapisuje jako atribut `duration` s hodnotou času.

Časování se jednotlivým blokům condition a periodic nastavuje pomocí dvojice atributů `period` a `delay`. První atribut specifikuje periodu vykonávání příkazů v bloku. Druhý atribut (`delay`) určuje zpoždění vykonání příkazů od začátku vykonávání kroku. Je-li tedy nastavena perioda na 1 s a zpoždění na 5 s, začnou se příkazy vykonávat až po 5 s od přechodu na aktuální krok. Příkazy se budou provádět jedenkrát za sekundu. Oba atributy nejsou povinné.

Veškeré časy v testovacím předpisu se zapisují ve formátu celého čísla, které je následováno příponou. Povolené přípony jsou h, min, s, ms, us, ns. Validním vstupem času tedy je `1s`, `1000ms` a nevalidním např. `0.5s`, `0,5s`, `1 s`.

Poslední typ elementů vnořených do elementu *<step>* jsou příkazy. Jak už název napovídá, příkazy vykonávají určitou akci. Příkazy jsou dvojího druhu. Podmínkové, které jsou určeny pro vložení do podmínkového bloku a akční, které něco vykonávají. Podmínkové mohou vykonávat nějakou akci, ale vždy vracejí výsledek, který může být podmínkovým blokem vyhodnocen. Akční příkazy jen vykonávají jim zadanou činnost.

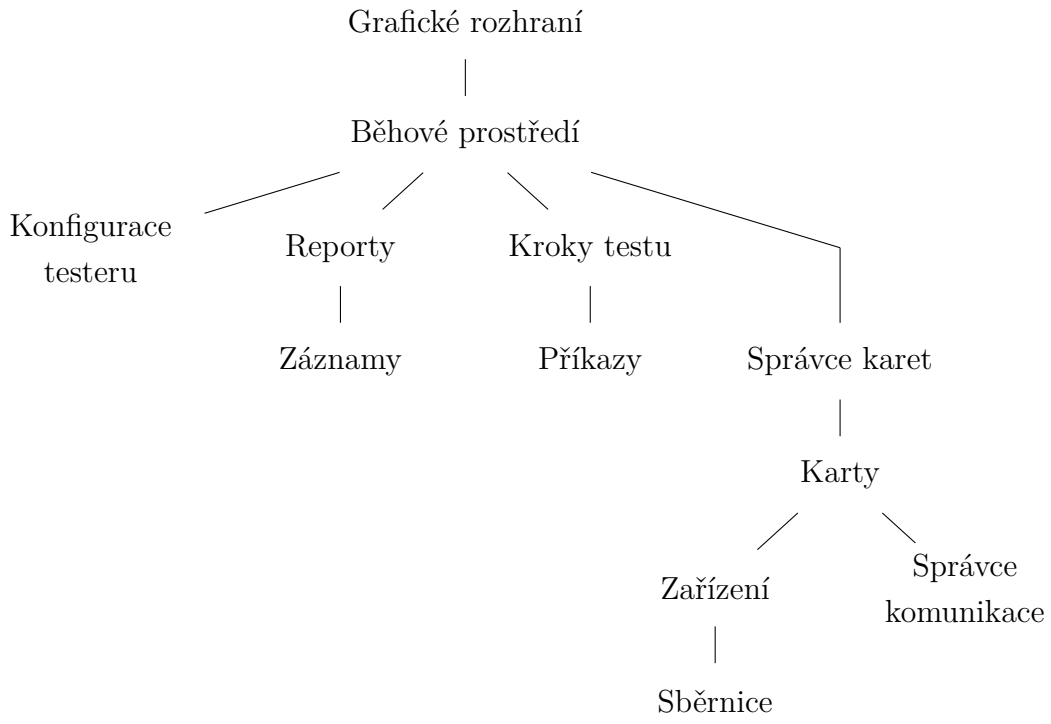
4.2 Struktura softwaru

Software testeru je rozdělen do několika logických celků (modulů) podle jejich funkce. Na obrázku 4.1 je zobrazena orientační struktura SW.

Srdcem testeru je běhové prostředí, které se stará o načítání testovacího předpisu, běh testu a zprostředkovává rozhraní pro jeho ovládání. Nad běhovým prostředím je grafické rozhraní, které zajišťuje lidsky přívětivý způsob ovládání běhového prostředí.

Běhové prostředí je závislé na nižších vrstvách SW. Jedná se o správce karet, který detekuje přítomnost karet na sběrnici a podle konfiguračních souborů poskládá karty z jednotlivých vstupně výstupních zařízení a komunikačních zařízení.

Každé zařízení musí mít svůj driver, který umožňuje ovládání daného zařízení. Drivery mají jednotné rozhraní pro komunikaci mezi s nadřazenými vrstvami SW.



Obr. 4.1: Struktura softwaru

Díky tomu je možné drivery dynamicky načítat z konfiguračního souboru. Další důležitou vlastností je možnost vytvářet další drivery podle potřeb jednotlivých karet bez zásahu do nadřazených vrstev SW.

Jednotlivá zařízení komunikují po sběrnici I^2C. Protože ovládání rozhraní I^2C může být na různých strojích různé. Je toto odděleno pomocí SW vrstvy. Tuto vrstvu je možné snadno upravit bez potřeby modifikovat cokoli dalšího v SW testeru. Tímto způsobem je možné snadno zajistit, aby PC komunikovalo s testerem přes jakýkoli I^2C převodník.

Jelikož je test v testovacím předpisu rozdělen na kroky, které dále obsahují jednotlivé příkazy, software obsahuje podobné dělení. Modul kroků testeru zajišťuje vykonávání příkazů včetně časování a případného vyhodnocení.

Posledním modulem jsou reporty. Reporty se skládají z jednotlivých záznamů, které obsahují informace o průběhu testu. Informace vkládá běhové prostředí na základě uživatelských akcí a reakcí testovaného zařízení. Modul také umožňuje odesílat reporty na servery.

4.3 Implementace SW

Software je navržen jako vrstevnatá struktura. Vyšší vrstvy jsou závislé na vrstvách nižší nebo stejné úrovně. Z důvodu návaznosti je implementace jednotlivých částí popisována od nejnižší vrstvy směrem k vrstvám vyšším.

4.3.1 Obsluha sběrnice

Pro obsluhu sběrnice je vytvořena třída `backplane_bus`, která zajišťuje obsluhu komunikace po sběrnici I²C. K tomu se využívá balíček `smbus2`, který kromě SMBus podporuje také I²C. Sběrnice SMBus je odvozena od sběrnice I²C a proto je možné balíček používat.

Při vytváření instance třídy, je třeba specifikovat identifikaci (ID) sběrnice v systému. Sběrnice I²C se v linuxu obvykle objevuje jako zařízení */dev/i2c-<id>*. Toto ID si knihovna najde mezi dostupnými zařízeními a otevře si jej.

Třída `backplane_bus` definuje metody pro uzavření sběrnice, detekování zařízení na sběrnici, zápis a čtení jednoho nebo více bytů.

4.3.2 Drivery vstupně výstupních zařízení

Základem pro tvorbu driverů je třída `io_driver_template`, která definuje rozhraní pro drivery. Každý driver musí být odvozen od této třídy, aby bylo jasně specifikované rozhraní.

Třída `io_driver_template` implementuje možnost pojmenovávat vstupy a výstupy alternativními jmény, která lze přiřadit v konfiguraci karty. Alternativní jména tzv. aliasy se ukládají jako slovník spojující nový název (klíč) a původní název (hodnota). O přiřazování aliasů ke jménům pinů se stará metoda `set_alias`.

Drivery vykonávají tři základní činnosti. Hlavním úkolem je nastavovat a číst stav vstupu / výstupu (V/V). Dále umožňují konfigurovat V/V a přiřazovat jim alternativní jména tzv. aliasy.

Nastavování a čtení V/V je zajištěno metodami `set_value`, `get_value`, `get_value_direct`. Metody přebírají jako parametry název pinu a v případě metody pro nastavení i novou hodnotu. Jako název pinu může být použit alias i jeho původní název. Metody jsou specifické pro každé zařízení a musí být implementovány v odvozené třídě.

Nastavování a čtení je koncipováno tak, že ve slovníku původních názvů pinů (`pin_names`) je obsažena informace o tom o který pin se jedná, tak aby ostatní funkce poté věděly co a kde nastavovat a číst. Například pro driver obvodu MCP23017 je v tomto slovníku uložen port a číslo pinu. Tuto informaci si mohou ostatní metody dohledat. Čtení nebo zápis pro MCP23017 probíhá tak, že se podle zadaného jména

nalezne port a číslo pinu na jejichž základě se vygeneruje název registru, kterého se akce týká. Na základě jména registru poté metody pro čtení a zápis na sběrnici naleznou jeho adresu ve slovníku adres registrů (`register_address`) a provedou danou akci.

Konfigurace pinu se provádí s podporou slovníku validních konfiguračních voleb (`config_params`). Slovník obsahuje pro každou volbu informace o tom co a v jakém registruje třeba změnit. Metoda `configure` přebírá název pinu a konfigurační položky. Pro každou konfigurační položku zavolá metodu `get_pin_settings`, která vrátí název registru a novou hodnotu. Metoda je specifikována v odvozené třídě. Její důležitost spočívá v možnosti zajistit specifické nastavení registrů podle potřeb daného obvodu a dané volby. Např. obvod MCP23017 lze konfigurovat jen jednobitovými položkami zatímco obvod MCP3426 už má konfiguraci dvoubitovou.

Některé obvody také mohou vyžadovat kalibraci. Toto je možné implementovat pomocí slovníku `calibration_settings`, který může obsahovat informace o zesílení a offsetu pro každý konkrétní pin. Tento slovník je nastavován na základě konfiguračního souboru pro karty.

4.3.3 Komunikace

Komunikace je založena na třídě `com_manager`, která definuje základní rozhraní společná pro všechny typy komunikace. Oproti rozhraní driverů je rozhraní komunikace definováno minimalisticky. Je to kvůli velké rozdílnosti v přístupu k různých typům komunikačních rozhraní.

Každé komunikační rozhraní musí umět definovat alternativní názvy pro každý port. Portem je zde myšlen jeden koncový bod používaný pro komunikaci. Například pro sériovou komunikaci to bude fyzický port a pro síťovou komunikaci to bude socket. Dále musí každé komunikační rozhraní implementovat metody `transmit`, `get_received`, `start_receive`, `stop_receive` a `configure`.

Nastavení portů slouží metoda `configure`, která přebírá jako parametry jméno portu a jeho nastavení. Pro odeslání dat je definována metoda `transmit`, která přebírá jako parametry jméno portu, kudy mají být data odeslána a samotná data, která se mají odeslat. Pro příjem dat je to složitější, protože některá komunikační rozhraní vyžadují aktivní vyčítání dat. Proto je specifikováno rozhraní tak, že se nejdříve zapne příjem metodou `start_receive` a poté je možné přijatá data vyčítat pomocí metody `get_received`. Příjem je možné zastavit pomocí metody `stop_receive`.

V modulu `serial_com` je implementována obsluha sériových rozhraní. Základem je třída `serial_manager`, která spravuje jednotlivé sériové porty (instance třídy `serial_port`. Základní třída pouze zajišťuje přidávání nových portů a přípravu konfigurace před samotnou aplikací na konkrétní port.

Třída implementující obsluhu jednotlivých portů implementuje přímé vykonání jednotlivých příkazů. Pro vysílání je zajištěno fyzické odeslání dat. Metoda `start_receive` spouští vlákno, které se pokouší vyčítat data a ukládá je do bufferu. Buffer přijatých dat je možné vyčíst pomocí metody `get_received`. Po vyčtení jsou data z bufferu vymazána. K vymazání bufferu dojde také pokud v bufferu bude více jak 10 kB dat.

4.3.4 Správa karet

Základem správy karet je třída `card_manager` jejímž úkolem je sestavit tester z jednotlivých karet. Správce také zajišťuje prohledávání karet a volání příslušných metod driverů podle toho, na které kartě se vstup nebo výstup nachází. Správce má implementovánu metodu `update`, jejímž voláním se z jednotlivých zařízení na všech kartách vyčtou aktuální hodnoty vstupů a výstupů.

Při inicializaci třídy `card_manager` se nejprve vytvoří instance třídy `backplane_bus` a následně se detekují všechna zařízení, která jsou připojena na sběrnici. Protože na každé kartě musí být přítomna EEPROM s adresou která má nejvyšší 4 bity 1010, je možné detekovat všechny karty. Prvních 16 B v paměti EEPROM obsahuje název karty. Tento název je vyčten a na jeho základě se sestavuje seznam dostupných karet. Pro každou kartu ze seznamu je vytvořena instance třídy `card`. Při inicializaci instance jsou předávány tři parametry. První je název konfiguračního souboru pro danou kartu, druhý obsahuje referenci na objekt obsluhy sběrnice I^2C a poslední určuje pozici karty v racku.

Třída `card_manager` má implementovány metody pro nastavení hodnoty na výstup, získání hodnoty ze vstupu / výstupu a pro jejich konfiguraci. Metody prohledávají zařízení na všech dostupných kartách a podle zadaného názvu pinu volají příslušnou metodu odpovídajícího driveru.

Třída `card` se při inicializaci poskládá z jednotlivých zařízení a komunikačních rozhraní. Ke každé kartě existuje Konfigurační soubor, který popisuje z čeho se karta skládá, včetně nastavení jednotlivých komponent. Skladba karty probíhá na základě konfiguračního souboru viz výpis č. 4.2.

Konfigurační soubor karty je XML dokument, jehož kořenovým elementem je *<card>*. Kořenový element zapouzdřuje elementy *<device>* a *<communication>*. Elementy *<device>* obsahují informace o zařízení, které se na kartě nachází. Atribut `name` udává název zařízení a také jméno driveru, který umožní komunikaci se zařízením.

Atribut `address` specifikuje masku adresy zařízení. Maska je udána v binární podobě a na pozici kde může být adresa změněna pomocí adresování na backplanu

je zapsán znak otazník. Takto specifikovaná maska adresy slouží pro identifikaci zařízení na sběrnici nezávisle na její poloze.

Každé zařízení může obsahovat definici pinů s jejich výchozí konfigurací, kalibrací a alternativním názvem (aliasem). Definice pinu se provádí pomocí elementu *<pin>*. Atribut `name` specifikuje výchozí název pinu podle driveru, atribut `alias` určuje nový alternativní název pinu a `conf` udává výchozí konfiguraci pinu. Atribut `value` obsahuje výchozí hodnotu pinu. Tato hodnota se nastavuje na výstupy v okamžiku, kdy neprobíhá testování.

Obsahem elementu *<pin>* může být *<calibration>* s položkami *<gain>* a *<offset>*. Jak už název napovídá jedná se o kalibrační strukturu, podle níž může driver přepočítat hodnoty na reálné.

```
<card>
    <device name="MCP47FEB22" address="11000??">
        <pin name="DAC0" alias="DA_0"
            conf="GAIN2 REFBGP"
            value="0">
            <calibration>
                <offset>0</offset>
                <gain>0.0765</gain>
            </calibration>
        </pin>
    </device>
    <communication>
        <serial>
            <port name="PORTA" baudrate="9600"
                config="8N1" xonxoff="false"
                rtscts="false" dsrdtr="false"
                device="TesterMainCardConverter"
                interface="1.0"/>
            <port name="PORTB" path="/dev/ttyUSB0" />
        </serial>
    </communication>
</card>
```

Výpis 4.2: Struktura konfigurace karty

Další komponentou ze které může být karta sestavena je komunikace. Komunikace je zabalena do elementu *<communication>* jehož obsahem je specifikace typu komunikace. Typ komunikace určuje který komunikační driver, který bude nahrán. V závislosti na typu komunikace lze specifikovat jednotlivé porty, ze kterých se komunikační rozhraní skládá. Např. pro sériovou komunikaci lze specifikovat libovolné množství sériový linek, buď zadáním cesty k zařízení v Linuxu, nebo pomocí specifikace jména zařízení a identifikace interfacu na USB. Součástí specifikace každého portu může být také výchozí konfigurace (přenosová rychlost, apod.).

4.3.5 Konfigurace testeru

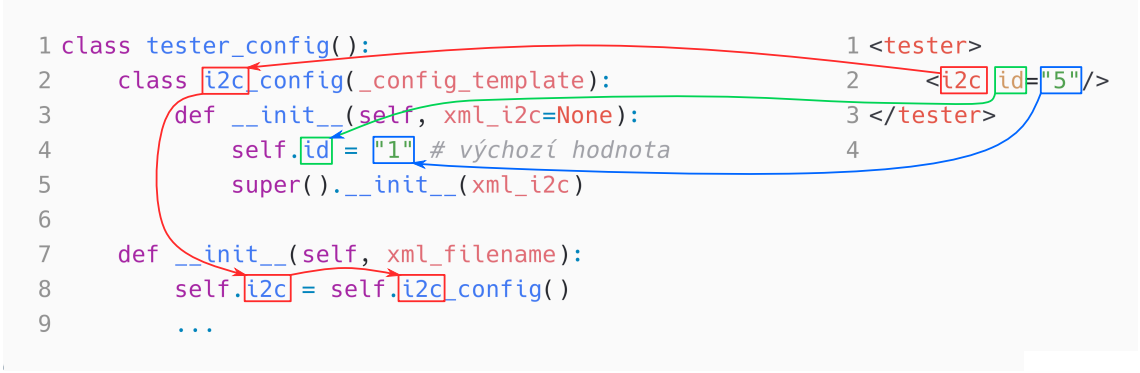
Konfigurace testeru se provádí pomocí XML souboru, který je při startu běhového prostředí načten. Konfigurace obsahuje informace, které jsou specifické pro konkrétní tester, který je umístěn v konkrétní firmě. Nyní konfigurace obsahuje identifikaci I^2C sběrnice v rámci PC, adresář, kam se ukládají reporty, formát ve kterém se budou ukládat a nastavení serverů včetně jejich metody přístupu. Parsování konfigurace bylo implementováno s ohledem na rozšiřitelnost.

```
<tester>
    <i2c id="5"/>
    <log directory="reports" output_format="PLAIN_TEXT"/>
    <servers>
        <server address="http://192.168.0.2/is/qual/add"
            access_type="EGMIS"/>
    </servers>
</tester>
```

Výpis 4.3: Konfigurace testeru

Parsování konfigurace provádí třída `tester_config` ve stejnojmenném modulu. Pro vytvoření struktury konfigurace stačí vytvořit instanci třídy `tester_config`. Při inicializaci instance je jako jediný parametr přebírán název konfiguračního souboru.

Každá položka, která se může v konfiguraci nacházet je určena jedním třídním atributem a podtřídou. Název atributu musí být stejný jako je název tagu, který se bude do tohoto atributu parsovat. Podtřída, jejíž název je stejný jako název tagu doplněný o „_config“ slouží jako kontejner pro data a zároveň jako parsovací třída.



Obr. 4.2: Propojení scriptu v Pythonu s konfigurací v XML

Parsovací třída je odvozena od třídy `_config_template` a musí obsahovat třídní atributy jejichž název je shodný s atributy daného tagu v XML souboru. Pokud odvozená třída při inicializaci volá inicializační funkci třídy rodičovské, budou automaticky všechny atributy XML načteny do atributů třídy. Není-li parsovací třídě

předán žádný argument vznikne instance s výchozí hodnotou. Tato vlastnost je důležitá pro případ, kdy není některý z tagů v XML přítomen. Pokud není některý atribut přítomen je mu v datové struktuře ponechána výchozí hodnota.

Pokud je třeba vytvořit pole několika prvků vnořených do nějakého elementu, musí se danému atributu v třídě `tester_config` zadat typ list (např. prázdný seznam). Označí-li se takto některý atribut, budou do něj automaticky přidávány všechny instance, které se vytvoří na základě tagů vnořených.

Na obrázku 4.2 je vidět propojení XML a parsovacího skriptu v Pythonu. Červená šipka určuje, kde všude musí být jméno tagu, aby došlo k propojení. Podobně zelená šipka označuje, kde musí být název atributu. Modrá šipka značí kam se po vyparsování uloží hodnota z konfigurace.

4.3.6 Reporty

Správu výsledků testů včetně jejich ukládání zajišťuje modul reportů. Základ modulu tvoří třída `report`, která záznamy sdružuje a zajišťuje jejich přidávání a ukládání.

Při vytváření instance třídy `report` je nutné zadat jméno operátora, odpovědného za test, sériové číslo testovaného výrobku a cestu k testovacímu předpisu. Z testovacího předpisu se načítá jméno autora testu, datum jeho vytvoření a aktuální verzi.

Informace o testovacím předpisu jsou ukládány zejména pro případ, kdy se testovací předpis během života výrobku mění. Pokud by došlo k reklamaci otestovaného výrobku, je možné dohledat jakým testovacím předpisem byl otestován a jestli byla reklamovaná funkcionalita otestována či nikoli.

Při vytváření instance je taktéž možné zadat výstupní formát reportu, cestu ke složce s reporty a callback informující o novém záznamu. Výstupní formát záznamu lze volit mezi prostým textem (`PLAIN_TEXT`) a formátem XML. Volba je platná pouze pro ukládání do souboru a lze ji změnit pomocí metody `set_out_format`.

Cesta ke složce s reporty je ve výchozím stavu nastavena na složku `log` v adresáři, kde je umístěn testovací předpis. Pokud složka v daném umístění neexistuje je automaticky vytvořena. Callback informující o novém záznamu je určen zejména pro účely GUI. Callback je volán v případě, že je do reportu přidán nový záznam. Jeho parametrem je řetězec s obsahem nového záznamu.

Pro účel rychlého přístupu k výsledkům jednotlivých kroků v testu, obsahuje třída `report` atribut `steps`. Atribut obsahuje seznam všech kroků s jejich názvy a výsledky. Atribut je instancí třídy `OrderedDict`, která se chová jako seřazený slovník. Klíčem je titulek kroku a hodnotou je jeho výsledek. Výsledky mohou být:

- krok neproveden (None),
- krok selhal (False),

- krok prošel (True).

Po vytvoření instance jsou načteny všechny kroky, které se budou vykonávat a je jim přiřazena hodnota None. Při přidání nového záznamu s výsledkem kroku je tento výsledek automaticky příslušnému klíči přiřazen.

Jednotlivé záznamy do reportu se vkládají pomocí metody `log`, která přebírá jeden povinný parametr a tři nepovinné. Povinný parametr obsahuje text záznamu v reportu. První nepovinný parametr určuje jestli krok, kterému záznam přísluší prošel nebo ne. Pokud není zadán jedná se pouze o informativní záznam. Druhý nepovinný parametr obsahuje text, který by měl detailněji popisovat okolnosti záznamu. Typicky se jedná o informace o tom proč byl test přerušen (timeout, přerušeno uživatelem, ...) nebo informace o tom, za jakých podmínek krok prošel (hodnoty analogových veličin, ...). Poslední parametr obsahuje titulek kroku, ke kterému se záznam váže. Titulek je svázán s atributem `steps`, kde slouží jako klíč pro přístup k výsledkům.

Metoda `log` vytváří instanci třídy `report_item` s parametry které byly metodě předány. Metoda ukládá vytvořenou instanci do seznamu `logs`. Seznam obsahuje všechny záznamy v chronologickém pořadí a umožňuje vytvoření kompletního reportu.

Pro uložení reportu na disk je určena metoda `save_to_file`. Metoda přebírá jako volitelný parametr jméno souboru do kterého má být report uložen. Pokud není jméno specifikováno je generováno automaticky ve formátu:
`<sériové číslo>_YYYYMMDDhhmmss.log`.

```
1 Testing protocol
2 ==================================================
3 Product: tester
4 S/N: SX00001
5 Operator: Radim Vítek
6 Test file name: /home/test/tests/selftest.xml
7 Test file version: 0.1
8 Date and time: 01.05.2019 17:34:20
9 ==================================================
```

Výpis 4.4: Hlavička reportu ve formátu prostého textu

```
1 <product>tester</product>
2 <serial_number>SX00001</serial_number>
3 <operator>Radim Vítek</operator>
4 <test_file_name>/home/test/tests/selftest.xml</test_file_name>
5 <test_file_version>0.1</test_file_version>
6 <date>01.05.2019 17:34:20</date>
```

Výpis 4.5: Hlavička reportu ve formátu XML

O samotné ukládání do souboru se starají metody `write_plain` a `write_xml`. Kromě uložení záznamů do požadovaném souboru metody ještě vytváří hlavičku. Hlavička obsahuje obecné informace o testu viz výše. Hlavičky v obou formátech jsou ve výpisech 4.4 a 4.5

Ukládání reportů na disk není jednou možností jak report zaznamenat permanentně. Je také možné odesílat jej na server. K tomu slouží metoda `send_to_servers`. Metoda na základě typu přístupu k serveru vybere ze slovníku funkci která zajistí odeslání reportu na server. Slovník spojuje typ přístupu k serveru s odesílací funkcí. Argumentem odesílací funkce je konfigurace daného serveru. Z konfigurace se načte adresa a případné další parametry a odešle se report na server.

Je implementován jeden typ systému se kterým tester umí komunikovat. Server přijímá data ve formátu JSON, do nějž se report převede a odešle se pomocí HTTP POST na server. Jelikož se jedná o jednoduchý způsob komunikace není třeba aby měla konfigurace jiné položky než adresu a způsob komunikace serverem.

4.3.7 Záznamy v reportu

Jednotlivé záznamy se skládají z vlastního textu, informace o selhání nebo neselhání testu, detailů a titulky. Při vytvoření nového záznamu se také poznamenává čas. Záznamy mají implementované rozhraní pro převod záznamu do určitého formátu. Základním formátem je prostý text který je implementován jako metoda `__str__`, která umožňuje převod objektu na řetězec. Dalším formátem je XML, který je generován jako objekt `Element` modulu `etree`.

```
1 [11:54:05] Kontrola LED diod (Stopped) ... <Failed>
2 [11:54:25] Kontrola LED diod ... <Passed>
3 [11:54:45] Test zahájen
4 [11:54:55] Nevyžádaná akce obsluhy (stisk tlačítka ANO)
```

Výpis 4.6: Formát prostého textu pro záznamy v reportu

Výsledný formát záznamu v prostém textu lze vidět na výpisu 4.6. Na prvním řádku je vidět kompletní záznam, kde je v hranatých závorkách čas vytvoření záznamu, následovaný vlastním textem. V kulatých závorkách jsou obsaženy detaily a na konci záznamu je v ostrých závorkách informace o výsledku.

```
1 <step time="11:54:05" details="Stopped" result="Failed">
2   Kontrola LED diod
3 </step>
4 <step time="11:54:25" result="Failed">
5   Kontrola LED diod
6 </step>
```

```
7  <step time="11:54:45">
8    Test zahájen
9  </step>
10 <step time="11:54:55" details="stisk tlačítka ANO">
11   Nevyžádaná akce obsluhy
12 </step>
```

Výpis 4.7: Formát XML pro záznamy v reportu

Druhý řádek je zkrácený o detaily. Poslední dva řádky jsou informativní záznamy bez výsledku. Ty slouží pro zaznamenání událostí, které se přímo netýkají výsledků kroků. I tyto záznamy mohou obsahovat detaily, které se zobrazují v kulatých závorkách.

Formát záznamu v XML je vždy zaobalen do tagu *<step>*. Text uvnitř je samotný text záznamu. Detaily připojené k záznamu jsou zapsány do atributu `details` tagu *<step>*. Čas záznamu je v atributu `time`. Výsledek kroku je zaznamenán pomocí atributu `result`. Atributy `details` a `result` nejsou povinné a mohou být vynechány.

Ve výpisu 4.7 jsou zaznamenány stejné záznamy jako ve výpisu 4.6 ovšem ve formátu XML namísto ve formátu prostého textu.

4.3.8 Příkazy

Všechny příkazy, které tester podporuje, jsou po skupinách implementovány jako třídy, které jsou odvozeny od třídy `test_cmd_template`. V okamžiku, kdy některá třída dědí z třídy `test_cmd_template` jsou automaticky příkazy, které implementuje, zařazeny mezi známé příkazy a mohou být použity v testovacím předpisu. Jedinou podmínkou je, že musí být soubor, kde je třída definována, importován. Mechanismus funguje na základě možnosti zjistit seznam tříd, které dědí z třídy jiné, bez nutnosti znát jejich název, nebo příslušnost k modulu. Díky tomuto mechanismu je možné snadno přidávat vlastní příkazy pouhým importováním daného modulu do programu testeru.

Import uživatelských příkazů lze provést v testovacím předpisu. K importu slouží párový tag *<user-commands>*, umístěný v kořenovém elementu. Všechny moduly specifikované v tomto elementu jsou při načítání testu importovány a tím zařazeny mezi známé příkazy. Seznam známých příkazů se vytváří postupným dotazováním na jednotlivé třídy, které implementují nějaké příkazy (dědí od třídy `test_cmd_template`). Dotaz probíhá pomocí metody `get_supported_commands`. Metodu musí být možné zavolat bez nutnosti vytváření instance dané třídy. Proto je dekorovaná jako `@classmethod`.

Metoda `get_supported_commands` musí vracet slovník, jehož klíči jsou řetězce s názvy příkazů a hodnotou reference na funkci pro vykonání příkazu. Pokud jednotlivé příkazy vyžadují nějaké dodatečné informace (např. počet parametrů), je možné aby hodnota byla typu tuple. Tuple v první položce obsahuje funkci pro vykonání a za ní může následovat libovolný počet parametrů libovolného typu. Názvy příkazů specifikovaných ve slovníku se shodují s tagy příkazů, které lze použít v testovacím předpisu.

Výchozí třída `test_cmd_template` implementuje základní strukturu a metody, které jsou sdílené pro všechny příkazy. Pro inicializační funkci je zde definováno základní chování a argumenty, které musí akceptovat všechny příkazy. Pro vytvoření instance je nutné zadat referenci na správce karet, aby příkazy měly přístup ke kartám a jejich prostřednictvím ke vstupům a výstupům. Dále se novému příkazu předává XML element, na jehož základě je vytvořen. Nepovinné parametry se týkají časování příkazů a jsou převzaty z bloků, kterým příkazy náleží.

Všechny parametry jsou uloženy jako atributy instance. Dále se z XML elementu vyparsují a do atributů uloží některé často používané informace. Atribut `type` ukládá jméno příkazu, `params` obsahuje slovník s parametry příkazu (XML atributy). Atribut `next_run` slouží pro časování. Je v něm uložen čas, kdy má dojít k dalšímu vykonání příkazu. Atribut `cond_state` je příznak výsledku příkazu. Příznak je kompletní výsledek příkazu a je nastavován s ohledem na dobu duration, která specifikuje minimální dobu platnosti než příkaz skutečně projde. Příznak může nastavovat příznak na hodnoty:

- neznámý výsledek nebo příkaz není podmínkový (None),
- příkaz selhal (False),
- příkaz neselhal (True).

Při vytváření instance je na základě známého typu příkazu a slovníku podporovaných příkazů zjištěna funkce, kterou je třeba volat pro vykonání příkazu. Tato funkce je uložena do atributu `run_function`. Tento atribut (funkce) je volán při každém vykonávání metodou `run`. Metoda `run` se stará o správné časování daného příkazu. Pokud ještě nenastal čas příkaz vykonat, příkaz se nevykoná. Pokud ovšem čas na vykonání už nastal, funkce se postará o jeho vykonání a poznačí, kdy má být příkaz vykonán znovu.

Výchozí třída také implementuje metodu pro resetování časování, která je volána v okamžiku spuštění kroku. Tím je zajištěno správné časování relativně k začátku kroku.

Jako příklad tvorby uživatelských příkazů byla implementována dvojice příkazů, umožňující ovládání externího napájecího zdroje. Jedná se o zdroj TP30005, který podporuje vzdálené ovládání. Zdroj lze připojit k PC pomocí sběrnice USB a komunikovat s ním pomocí protokolu SCPI přes virtuální sériový port. První implemen-

tovaný příkaz `psu-extern-init` slouží pro navázání spojení se zdrojem. Příkazu je v testovacím předpisu zadávána identifikace sériového portu, po kterém bude se zdrojem komunikovat a jeho ID pro identifikaci v rámci testu. Druhý příkaz `psu-extern` slouží pro nastavení výstupu zdroje. Příkaz má čtyři atributy. Prvním je ID jako identifikace konkrétního zdroje, druhý je povolení výstupu a posledními atributy jsou požadované napětí a proud. Aby bylo možné tyto příkazy používat, je nutné je naimportovat v testovacím předpisu.

4.3.9 Kroky testu

Modul `test_steps.py` sdružuje příkazy do bloků a zajišťuje jejich volání. Modul také zajistí správné parsování jednotlivých příkazů.

Základem je třída `test_step`, která při vytváření instance získává referenci na správce karet a element z modulu `etree`, který obsahuje konfiguraci jednoho kroku v testovacím předpisu. Třída obsahuje a uchovává obecné informace o kroku, které jsou specifikované v atributech kroku a také ukládá pokyny pro obsluhu v HTML formátu. Dále skladuje příkazy v blocích podle toho, jak jsou popsány v testovacím předpisu.

Pokyny pro obsluhu jsou v testovacím předpisu uloženy ve formátu Markdown, který je nutné převést na HTML. HTML je formát, který lze snadno v každém GUI frameworku zobrazit v prvku, který implementuje webový prohlížeč. Pro převod je použita knihovna `markdown`. Při parsování je převedené HTML obaleno běžnou strukturou HTML dokumentu a je přilinkován stylopis. Stylopis byl vygenerován pomocí Pythonové utility Pygments.

Během vytváření instance se nejprve vytvoří slovník všech dostupných příkazů. Klíči slovníku jsou názvy příkazů a jeho hodnotami jsou odkazy na třídy, které daný příkaz implementují. Slovník je vytvářen jen jednou při vytvoření první instance `test_step`.

Bloky jsou načítány na základě slovníku podporovaných bloků, který má jako klíče názvy bloků a jako hodnoty tuple. V tuple je na první položce odkaz na seznam, který skladuje daný typ bloků. Druhá položka je metoda, která daný blok umí parsovat. Celé načtení kroku spočívá v projití všech elementů v XML a zavolání příslušné parsovací funkce. Parsovací funkce pro bloky prochází všechny elementy uvnitř bloku, vytváří instance příslušných příkazů a ukládá je do příslušného seznamu.

Protože příkazy mohou mít v rámci různých bloků různé časování, je během parsování bloků volána metoda `load_block_timing`. Metoda z XML načte hodnoty jednotlivých časování, převede je na číselnou reprezentaci a vrátí slovník s názvem parametru (klíč) a jeho hodnotou (hodnota). Takto získaný slovník se poté rozbalí do parametrů při vytváření instance jednotlivých příkazů. Časování je zajištěno na

úrovni příkazů a ne na úrovni bloků i přes to, že se časování specifikuje blokům a ne příkazům.

Třída `test_step` hlídá, jestli právě probíhající krok nepřekročil čas, který mu byl vyhrazen. Metoda `timouted` vrací booleovskou hodnotu na základě času začátku kroku a aktuálního času. Poslední metodou, která tvoří rozhraní pro kroky je `reset`, která poznačí nový čas začátku vykonávání kroku a zajistí, aby všechny příkazy provedly reset také.

Třída `test_step` neimplementuje nic co se týká běhu samotného testu. To je úkolem běhového prostředí, pro které je tato třída jen datovým kontejnerem s rozšířenou funkcionalitou.

4.3.10 Běhové prostředí

Běhové prostředí je srdcem celého testeru. Jeho úkolem je zajistit načtení testu, jeho vykonání a následné vyhodnocení. Zaobaluje tedy všechny nižší vrstvy SW a zajišťuje jejich vzájemné propojení. Vše je implementováno ve třídě `test_runtime`. Běhové prostředí obsahuje vlákno, které zajistí vykonávání vlastního testu jako nekonečnou smyčku. Definuje také API pro ovládání průběhu testu a samotného běhu vlákna.

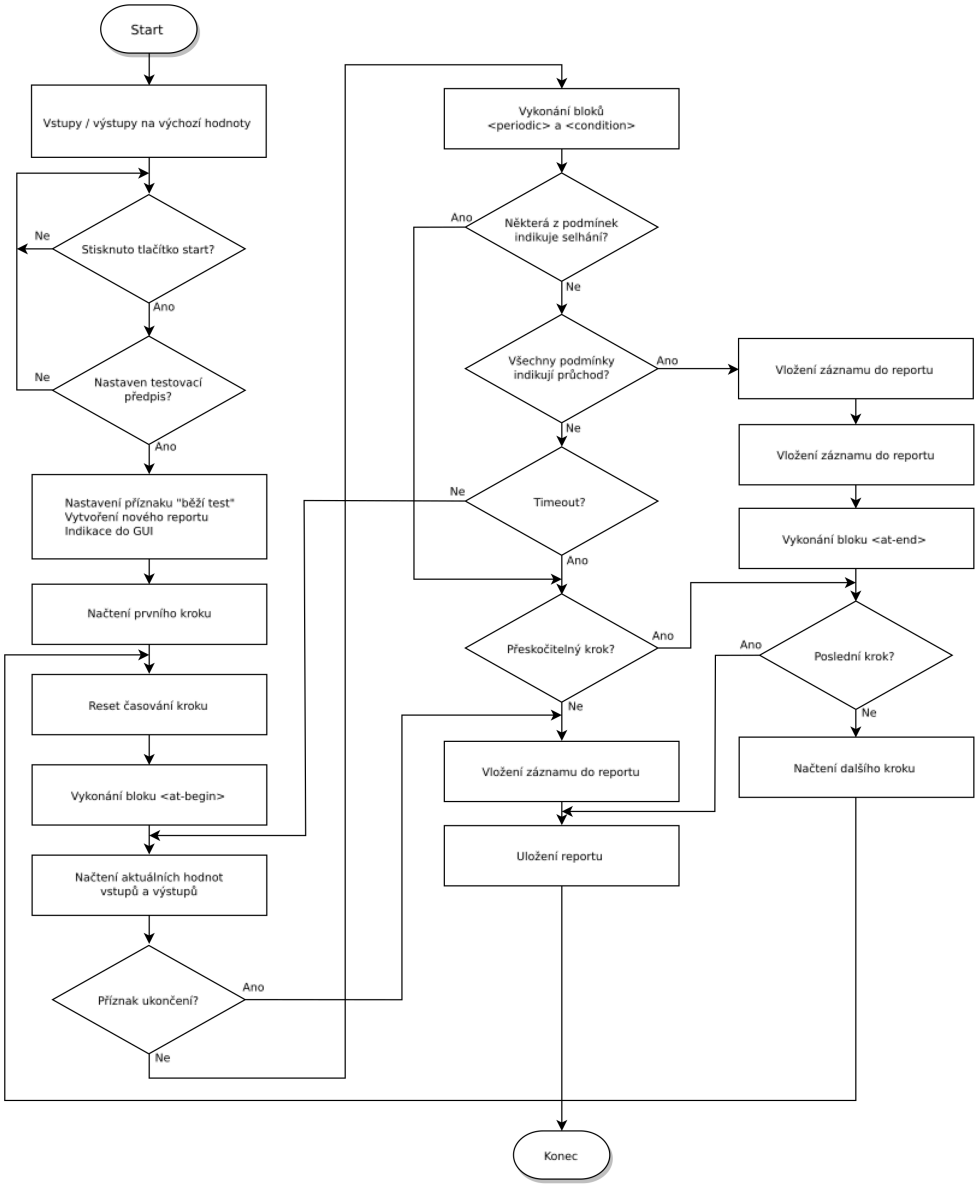
Při vytváření instance běhového prostředí je předávána inicializační metodě cesta k testovacímu předpisuje je také možné předávat referenci na objekt, který implementuje metody (callbacky) zajišťující komunikaci s nadřazenou částí programu (nejčastěji GUI). Callbacky se volají jen pokud je objekt nastaven. Je tedy možné, aby vše běželo bez další nadřazené vrstvy.

Inicializační metoda vytváří strukturu dat potřebnou pro správný běh testu. Jako první se načte konfigurace testeru do atributu `tester_config`. Konfigurace je uložena jako instance třídy `tester_config`, která zajistí načtení a vyparsování konfigurace, ze které je vytvořena struktura používaná v běhovém prostředí. Podrobněji je konfigurace testeru popsána v oddíle 4.3.5.

Dále je na základě konfigurace testeru, která obsahuje mimo jiné také identifikaci sběrnice I^2C v systému, vytvořena instance správce karet. Následně se připraví prázdné atributy pro skladování jména operátora a sériového čísla produktu. Tyto atributy jsou nastavovány z vnějšku pomocí property. Tento přístup zajistí, že se nezmění sériové číslo za běhu testu. Pokud by se o to někdo pokusil, běhové prostředí by vyvolalo výjimku. Atributy také obsahují název produktu, který slouží zejména pro GUI, kde je zobrazen. Aby GUI poznalo, že se název změnil, je při každé změně názvu vyvolán callback do GUI.

Tester obsahuje tři tlačítka pro ovládání běhu programu. Konkrétně se jedná o tlačítko pro spuštění nebo zastavení testu a dvojici tlačítek pro potvrzení nebo

odmítnutí dotazu v testovacím předpisu. Aby tato tlačítka mohla být i v GUI je v běhovém prostředí implementován mechanismus, který to umožní. GUI při stiscích kláves volá metody `user_action_ok`, `user_action_nok` `test_set_event`. Metody vyvolávají události pro hlavní vlákno. Vlákno tyto události kontroluje a na jejich základě provede příslušnou akci.



Obr. 4.3: Vývojový diagram testovacího vlákna běhového prostředí

Testovací předpis je načítán pomocí metody `load_test`. Metoda jako jediný pa-

rametr přebírá název testovacího předpisu. Součástí názvu testovacího předpisu smí být i relativní nebo absolutní cesta. Metoda rozparsuje daný XML soubor a postupně z něj načte jeho jednotlivé části. Jako první se načítají případné uživatelské příkazy. Načtení musí být provedeno jako první, aby byly nové příkazy dostupné při vytváření jednotlivých kroků.

Načtení uživatelsky definovaných příkazů se provádí pomocí metody `import_user_commands`, která přebírá kořenový element testovacího předpisu. Metoda postupně projde všechny moduly a pokusí se je naimportovat pomocí knihovny importlib. Naimportováním se příkazy zařadí mezi známé a mohou být použity v testovacím předpisu.

Další činností, která se provádí při načítání testovacího předpisu je načtení názvu produktu. Název je načten metodou `load_test_information`, která do property `product` uloží nový název. Property `product` při každém zápisu vyvolá callback `set_productname` pro GUI.

Na konec se při načítání testovacího předpisu načtou jednotlivé kroky. Každý krok je instancí třídy `test_step`, uložený jako jedna položka v seznamu `steps`. Při vytváření instance je inicializační metodě předána reference na správce karet a konkrétní element step v testovacím předpisu.

Je-li kompletně načten testovací předpis, je možné spustit test. Samotné testovací vlákno může běžet i bez testovacího předpisu, protože čeká, dokud nedostane příkaz ke spuštění testu a dokud není testovací předpis nastaven. Vlákno začne zpracovávat načtený předpis až po tomto příkazu. Vlákno je spouštěno pomocí metody `start_testing` a zastavit ho je možné pomocí `stop_testing`. Metoda pro zastavení zajistí korektní zastavení testovacího vlákna. Pošle vláknu událost `_terminate_thread_event`, které uloží report, nastaví výstupy na výchozí hodnoty a přeruší vykonávání nekonečné smyčky, čímž se vlákno ukončí.

Testovací vlákno se skládá z několika do sebe vnořených smyček. První smyčka je nekonečná zajišťuje možnost test opakovat pouhým vyvoláním příslušné události. Do této „supersmyčky“ je vnořena smyčka, která zajistí, že se budou postupně vykonávat jednotlivé kroky testu v pořadí v jakém jsou v testovacím předpisu uvedeny. Poslední smyčka zajišťuje správné časování kroků.

Po spuštění vlákna jsou výstupy nastaveny do výchozího stavu a začne se čekat na stisk tlačítka (HW i SW v GUI), které test spustí. Je-li tlačítko stisknuto, spustí se testování a je nastaven příznak běžícího testu. Ihned poté se vytvoří nový report, který přebírá jméno operátora, sériové číslo, apod. Pokud je nastaven objekt s callbacky, provede se callback, který indikuje začátek testování, následně callback, který vytvoří seznam kroků a smažou se staré záznamy v GUI.

Po této inicializaci testu se spustí vykonávání jednotlivých kroků. Každý dostupný krok je načten a jako první je resetováno jeho časování. Následně se vykonají

příkazy v bloku *<at-begin>*. V časovací smyčce se provádí načtení stavů všech vstupů/výstupů ze všech dostupných karet. Dále se kontroluje, jestli nebyl test přerušen stiskem tlačítka, které je k tomu určeno. Pokud nedošlo k přerušení začnou se vykonávat periodické příkazy a vyhodnocovat podmínky. Pokud některá podmínka selže, automaticky selže i celý test. Naopak pokud jsou všechny podmínky splněny, je aktuální krok označen jako úspěšný. V obou případech výsledku je přidána položka do reportu s detaily, ve kterých je specifikován stav podmínkových příkazů v okamžiku selhání nebo průchodu kroku.

V okamžiku, kdy je krok označen za dokončený, tzn. úspěšný nebo neúspěšný, jsou provedeny příkazy v bloku *<at-end>*. Pokud byl krok úspěšný, nebo mu byla specifikována vlastnost „skippable“, dojde k přechodu na další krok v pořadí. Další krok se začíná vykonávat od resetu časování. Pokud ovšem krok selhal a vlastnost „skippable“ nemá, dojde k ukončení celého testu.

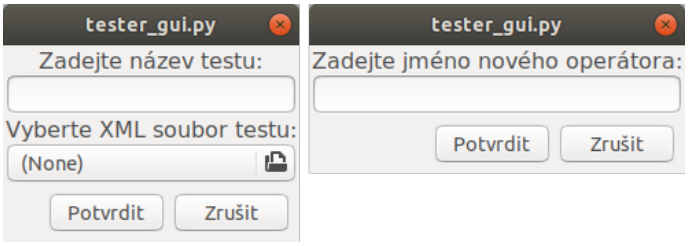
Při ukončení testu jsou nejdříve nastaveny výstupy do výchozího stavu. Následuje ukládání reportu na disk, příp. jeho odesílání na servery. Následně je zrušen příznak běžícího testu a vlákno bude čekat na další spuštění. Do GUI je indikováno ukončení testu pomocí callbacku `test_indication`.

4.3.11 Grafické rozhraní

Úkolem grafického rozhraní je zajistit možnost pohodlně ovládat tester. Grafické prostředí je vytvořeno v Pythonu s pomocí knihovny GTK+. Rozmístění komponent bylo navrženo v editoru Glade.

Obr. 4.4: Grafické rozhraní testeru

Grafické prostředí je rozděleno na dva panely. V horní části levého panelu jsou umístěna políčka pro výběr testu a operátora. Obě políčka jsou rozbalovací seznamy s položkami definovanými v CSV souborech. Pro přidávání položek do seznamu slouží tlačítka „Add“ vpravo od rozbalovacích seznamů. Tato tlačítka vyvolají dialogové okno, kam se zadává nová položka. V případě nového operátora, je v okně vstupní pole pro zadání jména. Pro zadávání nového testu se vyvolá dialog, kde je možné vybrat cestu k testovacímu předpisu a zadat jméno testu v přívětivějším formátu než je název souboru. Dialogová okna viz obrázek 4.6. Použití rozbalovacích seznamů zajišťuje rychlé a pohodlné zadávání jména, čímž jsou jména v reportu vždy uvedena ve stejném formátu. Díky tomu je lze snadno strojově zpracovat.



Obr. 4.5: Dialogová okno pro zadávání nového testu a operátora

Pod rozbalovacími seznamy pro zadávání testu a operátora je umístěno vstupní pole pro zadávání sériového čísla. Vpravo od něj se nachází tlačítko „+1“. Toto tlačítko zajistí inkrementování sériového čísla. Je-li ve vstupním poli zadáno sériové číslo, lze tímto tlačítkem rychle zvýšit jeho hodnotu. Formát sériového čísla podporuje i nečíselné znaky. Tlačítko pro inkrementaci funguje jen v případě, že jsou na konci řetězce sériového čísla umístěny číslice. V tomto případě je inkrementována pouze poslední sekvence číslic. Je-li na konci sériového čísla nečíselný znak, je při stisku inkrementačního tlačítka vyvolán chybový dialog s vysvětlením. Běží-li test, je část pro zadávání údajů neaktivní.

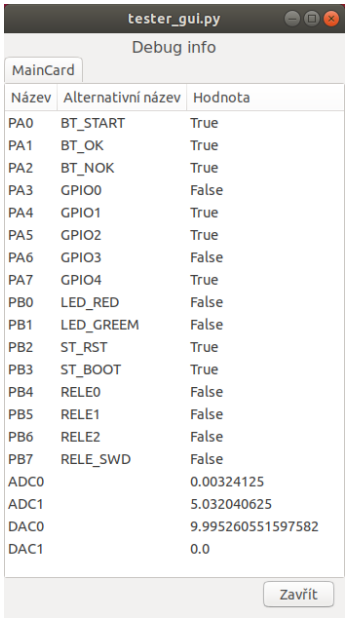
Uprostřed levého panelu jsou tlačítka pro spuštění, přerušení testu a pro zjišťování odpovědi obsluhy. Chování tlačítek je shodné s chováním HW tlačítek na předním panelu testeru. Tlačítko „Spustit“ / „Přerušit“ umožňuje spustit test a pokud běží tak jej přerušit. Při přerušení testu je test automaticky označen stejně, jako kdyby selhal. Tlačítko „ANO“ značí kladnou odpověď na dotaz testeru. Pokud testovací předpis nepředpokládá uživatelskou reakci, tlačítko nic neprovede. Tlačítko „NE“ značí zápornou odpověď. Na rozdíl od tlačítka „ANO“ je možné tímto tlačítkem daný krok kdykoli přerušit. Pokud je krok označen jako přeskočitelný, bude test po selhání daného kroku pokračovat krokem následujícím.

Pod tlačítky je umístěn seznam jednotlivých kroků. U každého kroku je ikona, která označuje jestli krok selhal (červený křížek) nebo uspěl (zelené zatržítko). U aktuálně probíhajícího kroku je zobrazena zelená šipka. Pod seznamem je umístěn vý-

sledek testu. Pokud test selže, je zde o tom obsluha, červeným políčkem a textem „Test neuspěl“, informována. Pokud test uspěje, je políčko zelené, s textem „Test proběhl úspěšně“. Probíhající test je označen světlým políčkem s textem „Test probíhá ...“.

V pravé části GUI se nachází pokyny pro obsluhu. Pod tímto panelem se vypisují jednotlivé záznamy reportu. Pokyny pro obsluhu zůstávají zobrazeny i po ukončení testu. Takto je možné zobrazit pokyny, které se netýkají testu a je třeba je provést po otestování. Typicky se jedná o zobrazení pokynů pro balení. Záznamy z reportů ve spodní části panelu jsou určeny spíše pro opravy. Obsluha zde může vyčíst některé hodnoty, které jí pomohou při opravě závady.

Pro snazší tvorbu testovacích předpisů a případné řešení komplikovanějších závad, je možné otevřít dialogové okno s ladicím výpisem, viz obrázek 4.6. V ladicím výpisu jsou zobrazeny jednotlivé vstupy a výstupy všech dostupných karet. Tuto nabídku je možné zobrazit pomocí tlačítka „Debug info“, které je umístěno v pravém horním rohu okna. Výpis je aktualizován pouze za běhu testu přibližně každou sekundu.



Obr. 4.6: Ladicí dialogové okno

GUI je implementováno jako třída `app`. Při inicializaci je načteno rozvržení okna, vytvořeného pomocí editoru Glade, a soubor se styly, které modifikují standardní vzhled. Poté musí být zavolána metoda `run`, která zajistí běh celé aplikace. Metoda zobrazí hlavní okno a vytvoří instanci běhového prostředí s referencí na sebe sama (instance třídy `app`). Reference zajistí propojení callbacků běhového prostředí a GUI.

Callbacků je celkem 8. Jejich deklarace je možné vidět ve výpisu 4.8. Callback

`set_productname` vyvolává běhové prostředí při každé změně názvu výrobku. Dochází k tomu při načítání testovacího předpisu. Callback `settings_refresh` informuje GUI o tom, že se test zastavil nebo rozběhl. Je to využito pro deaktivaci části GUI, která je určena pro zadávání údajů. Callback `get_operator_name` je volán z běhového prostředí v okamžiku, kdy je spuštěn test. Jeho návratovou hodnotou je jméno operátora. Běhové prostředí se tímto způsobem dozví o tom, kdo právě testuje. Pokud není jméno vybráno, je vyvolána výjimka. Stejným způsobem funguje callback `get_serial_number`, který vrací sériové číslo.

Pro určení jestli byl test spuštěn nebo ukončen, slouží callback `test_indication`. Argument „finished“ určuje zda šlo o ukončení nebo ne. Pro předávání seznamu jednotlivých kroků je určen callback `create_tasklist`, který je volán z běhového prostředí a jako parametr předává seznam kroků i s jejich výsledky. Jak už název napovídá, `clear_logs` zajistí smazání všech záznamů v GUI. Toto je provedeno vždy při spuštění nového testu. Při každém novém kroku je volán callback `display_instructions`, který předává do GUI instrukce pro obsluhu ve formátu HTML.

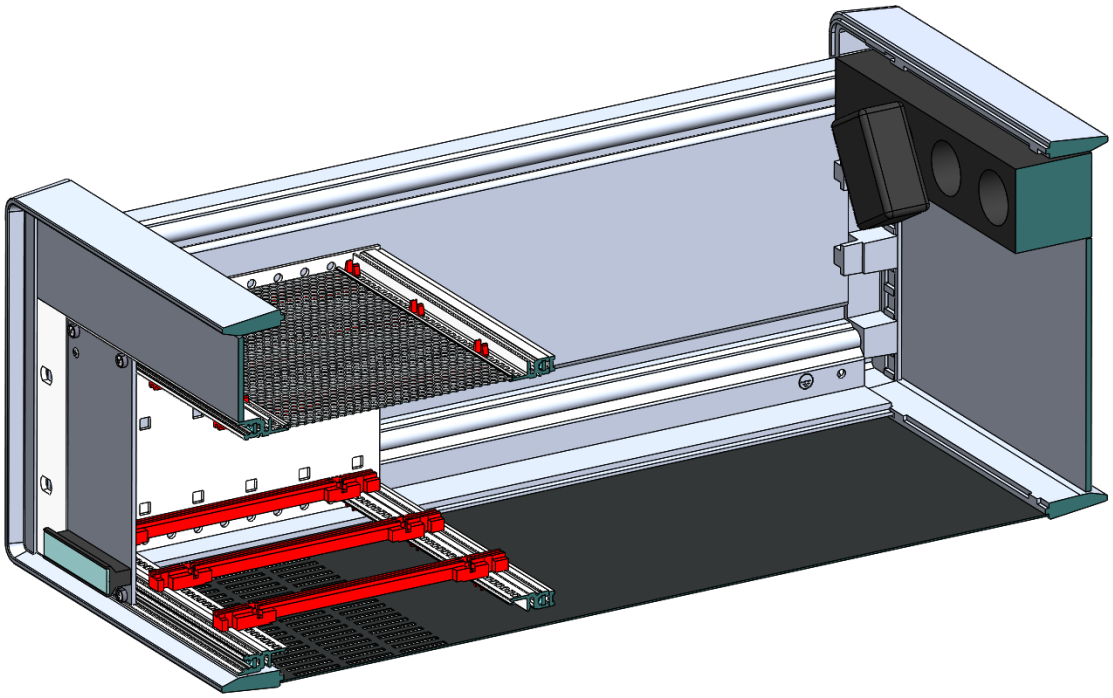
```
1 set_productname(produkt)
2 settings_refresh()
3 get_operator_name()
4 get_serial_number()
5 test_indication(report_steps, finished=False)
6 create_tasklist(report_steps)
7 clear_logs()
8 display_instructions(step_instruction)
```

Výpis 4.8: Callbacky propojující GUI a běhové prostředí

5 Mechanická konstrukce

Tester je z mechanického hlediska sestaven z racku, subracku, jednotlivých karet a backplanu. V racku je umístěn subrack ,do kterého jsou zasunuty jednotlivé karty s roztečí 12 HP (HP je jednotka udávající šířku karty v rackovém systému). Aby byly karty mezi sebou propojeny, je na zadní straně subracku umístěn backplane. Backplane kromě propojení také zajišťuje adresování karet na sběrnici a je využit pro připojení napájení. Celá sestava, skládající se ze subracku, z karet a backplanu, je zasazena v rackové krabici.

Pro připojení napájení je do racku zabudován zásuvkový panel. V panelu jsou umístěny adaptéry, které zajišťují napájení celého testeru. Pro vyvázání kabeláže je na subracku umístěna mřížka. Konstrukce testeru je patrná z obrázku 5.1. Pro přehlednost není na obrázku zakreslen backplane.



Obr. 5.1: Konstrukce testeru

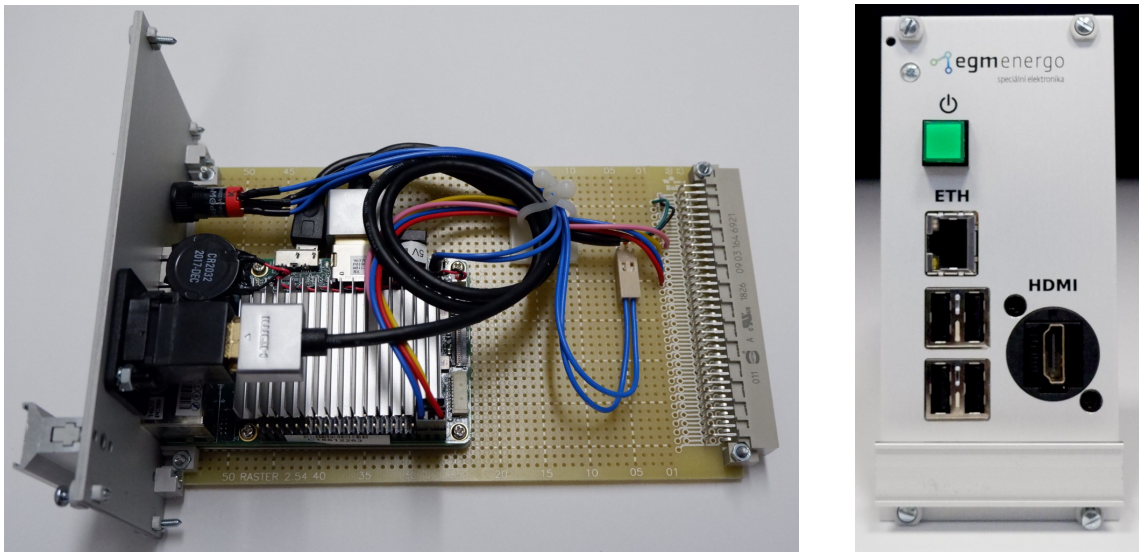
5.1 PC karta

Z důvodu kompaktnosti celého zařízení bylo rozhodnuto, že řídicí PC bude zabudováno do testeru. Aby byl tester snadno rozebiratelný a konstrukčně jednoduchý, bylo PC umístěno jako karta do subracku. Pro tento účel byl vybrán jednodeskový

počítač UP-CHT01-A12-0432 od firmy AAEON. Jedná se o jednodeskový počítač s procesorem Intel Atom x5-Z8350, 4 GB RAM a 32 GB FLASH.

Tento typ PC byl vybrán z důvodu rozměrů, které umožňují montáž na kartu. Další důvod proč byl zvolen tento typ PC je, že se jedná o platformu x86. Díky tomu bude veškerý SW testeru kompatibilní se stolními PC. Také je možné použít běžně dostupný systém. Není třeba kompilovat jej kompilovat, jak je zvykem pro většinu embedded linuxových počítačů.

Aby nebylo nutné kreslit a vyrábět DPS, byla pro montáž PC použita prototypovací deska určená do subracku (eurokarta). PC je na desku přišroubováno a je z něj vyvedeno napájení, USB a I^2C na konektor, který vede na backplane. V předním panelu jsou vyfrézované otvory pro čelní panel PC, na kterém jsou vyvedeny USB konektory a jeden ethernetový konektor. Dále je na přední panel vyveden HDMI výstup z PC, který je určen pro připojení monitoru. Na předním panelu je také tlačítko s podsvětlením, určené pro spouštění PC a pro indikaci jeho běhu. Podsvětlení pro tlačítko je řízeno napájením USB, které je vypnuto, když PC neběží.

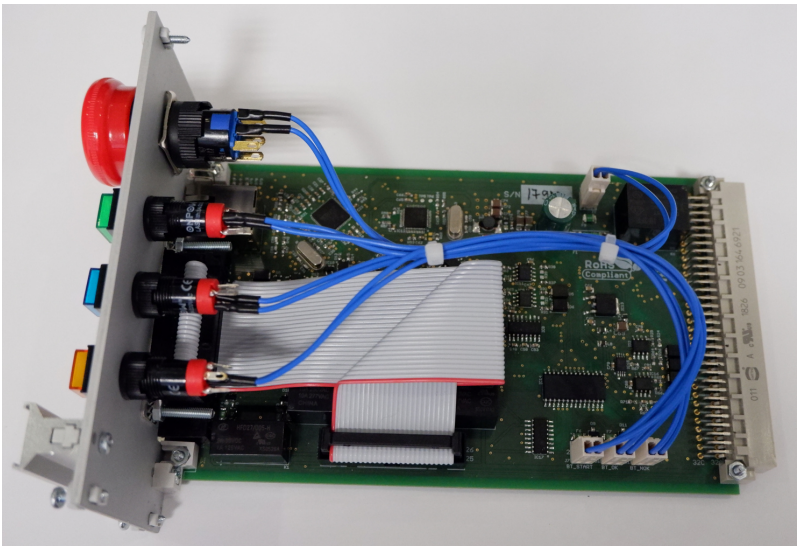


(a) Zapojení karty (b) Přední panel

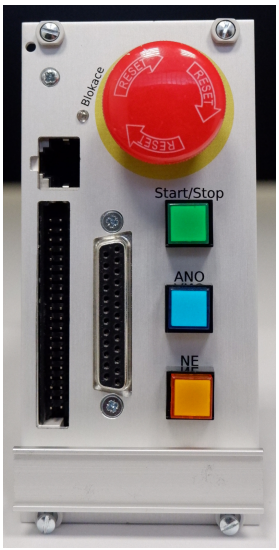
Obr. 5.2: PC karta

5.2 Hlavní karta

Hlavní karta slouží jako expandér vstupů výstupů, převodník komunikačních rozhraní a jako základní ovládací panel. Karta musí být v testeru osazena vždy a vždy na druhé pozici. Je to z toho důvodu, že je sem přivedena sběrnice USB, kterou karta využívá.



(a) Zapojení karty



(b) Přední panel

Obr. 5.3: Hlavní karta

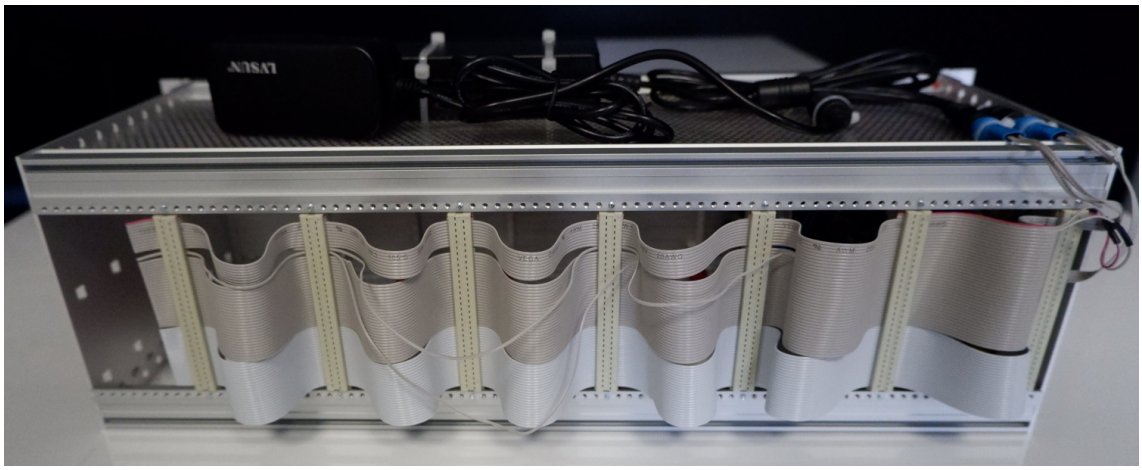
Přední panel karty má vyfrézované otvory pro konektory a pro ovládací tlačítka. Je zde také vyvedena LED dioda, která indikuje, že jsou karty blokovány tlačítkem totalstop. LED slouží pro urychlení identifikace problémů s testerem.

5.3 Backplane

Backplane testeru je sestaven z konektorů nakrimpovaných na plochém kabelu. Tento způsob vytváření backplanu byl zvolen kvůli potřebě zakódovat adresy pro jednotlivé pozice a zajistit propojení jen určitých signálů. Toto by nesplňovaly backplany, které lze běžně koupit, protože bývají zpravidla plně propojené. Druhou možností, bylo vytvořit vlastní backplane na DPS. Ovšem cena desky by kvůli svým rozměrům byla neakceptovatelná.

Zvolený způsob tvorby backplanu má oproti verzím s DPS jednu velkou výhodu. Je možné snadno změnit rozteč mezi jednotlivými kartami. Naopak jako nevýhodu, je možné označit nutnost použití konektoru s 64 piny. Konektor s 96 piny určený pro krimpování na plochý kabel existuje ale není běžně dostupný. Použití většího konektoru by umožnilo zajistit lepší rozmístění signálů na backplanu.

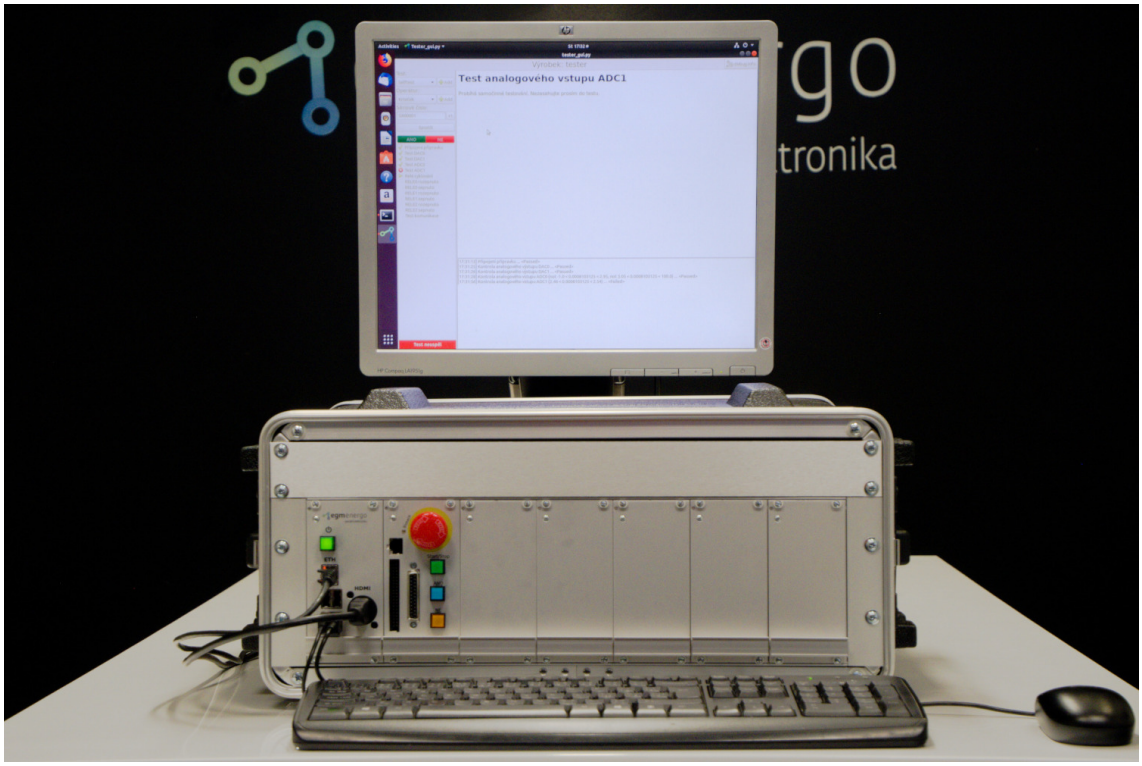
Výsledek tvorby backplanu ukazuje obrázek 5.4. Jeho zapojení je příloze.



Obr. 5.4: Zkompletovaný backplane připevněný na subracku

5.4 Výsledná konstrukce

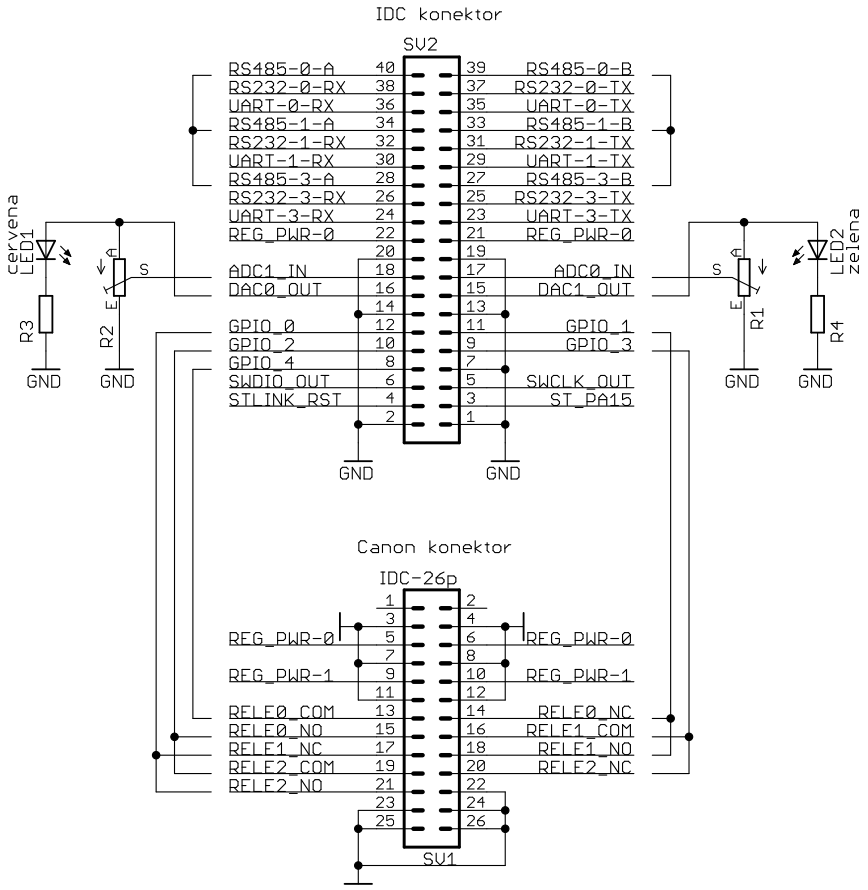
Zkompletovaný a běžící tester zachycuje obrázek 5.5. Subrack s kartami je zasazen do rackového kufru s madly a víky. Na zadní straně je umístěn vypínač a přívodní šňůra.



Obr. 5.5: Zkompletovaný tester

6 Ověření funkčnosti testeru

Funkčnost testeru byla ověřena na jednoduchém přípravku, který demonstruje možnosti testeru. Přípravek je konstruován jako prosté propojení vstupů s výstupy. Propojení je provedeno tak, aby bylo možné zjistit jestli tester funguje podle očekávání. Zapojení přípravku je na obrázku 6.1.



Obr. 6.1: Zapojení testovacího přípravku

Ve vzorovém testovacím předpisu je popsáno testování všech propojení. Předpis je koncipován spíše jako ukázka možností testeru a použití jednotlivých příkazů. Celý test se skládá ze 13 kroků.

První krok vybízí uživatele k připojení testovaného přípravku k testeru. Krok vyžaduje jen potvrzení od uživatele, že dané pokyny provedl. Uživatelská akce je zapsána do podmínek jako tag *<user-action/>*

```
1 <step title="Připojení přípravku"
2     report-message="Připojení přípravku">
3     <instruction>
4 # Připojení přípravku
```

```
Přípravek připojte k~testeru.

**Potvrďte stiskem tlačítka ANO.**
    </instruction>
    <condition>
        <user-action/>
    </condition>
</step>
```

Výpis 6.1: Krok výzvy obsluhy k připojení zařízení

Po potvrzení je testováno jestli svítí LED dioda. Dioda je připojena na analogový výstup a je znovu vyžadováno potvrzení od operátora. Při začátku kroku je nastaven analogový výstup na napětí 10 V. Po skončení kroku je výstup nastaven zpět na napětí 0 V.

```
<step disabled="false" title="Test DAC0"
    report-message="Kontrola analogového výstupu DAC0">
    <instruction>
# Test analogového výstupu DAC0

**Svítí na přípravku červená LED?**
    </instruction>
    <at-begin>
        <pin id="DAC0" value="10"/>
    </at-begin>
    <at-end>
        <pin id="DAC0" value="0"/>
    </at-end>
    <condition>
        <user-action/>
    </condition>
</step>
```

Výpis 6.2: Krok testování DAC s podporou obsluhy

Následuje kontrola druhé LED a poté je testován analogový vstup. Pro tento krok je analogový výstup nastaven na hodnotu 5 V. Napětí je pomocí trimru sníženo na polovinu a toto je měřeno pomocí analogového vstupu. Pro vyhodnocení jestli je hodnota v dovolených mezích je použita podmínka *<in-range>*. Podmínka vyžaduje tři hodnoty. První je dolní limit hodnoty, následuje kontrolovaná hodnota a poslední je horní limit hodnoty. Pokud bude naměřená hodnota v rozsahu specifikované v podmínce dojde k přechodu na další krok. Aby bylo zajištěno ustálení výstupní hodnoty na DAC, je vyhodnocení podmínky pozdrženo pomocí atributu „delay“ o 2 s. Pro případ, že by hodnota kolísala, je atributem „duration“ vynuceno minimální trvání splněné podmínky na 1 s.

```
<step disabled="false" title="Test ADC0"
    report-message="Kontrola analogového vstupu ADC0">
    <instruction>
# Test analogového vstupu ADC0

Probíhá samočinné testování. Nezasahujte prosím do testu.
    </instruction>
    <at-begin>
        <pin id="DAC1" value="6"/>
    </at-begin>
    <at-end>
        <pin id="DAC1" value="0"/>
    </at-end>
    <condition delay="2s" duration="1s">
        <out-range>
            <value value="-1"/>
            <pin id="ADC0"/>
            <value value="2.95"/>
        </out-range>
        <out-range>
            <value value="3.05"/>
            <pin id="ADC0"/>
            <value value="100"/>
        </out-range>
    </condition>
</step>
```

Výpis 6.3: Krok samočinného testování ADC

Po kontrolách analogových vstupů následuje kontrola spínání relé. Tento krok ukazuje plné možnosti časování periodicky vykonávaných bloků. Při kroku je relé spínáno s periodou 2 s. Krok je opět ukončen uživatelskou akcí. V tomto kroku je také ukázáno ovládání digitálních výstupů pomocí příkazů *<pin>*.

Pro střídavé spínání relé je třeba použít dva bloky periodic. První blok bude mít nastavenu jen požadovanou periodu a bude relé např. spínat. Druhý blok periodic bude mít nastavenu periodu stejnou a navíc bude mít nastaveno zpoždění na hodnotu např. na polovinu periody (střída 50 %). V tomto bloku bude relé odpínáno. Při běhu testu se poté bude vykonávání bloků periodic střídat se zvolenou periodou a střídou.

Protože má obsluha za úkol zjistit, jestli k přepínání relé došlo a není možné, aby to vyhodnotila dříve než po uběhnutí alespoň jedné periody cyklu, je vyhodnocení podmínky pozdrženo o 2 s.

```
<step disabled="false" title="Relé cyklování"
    report-message="Cyklování relé">
    <instruction>
# Test relé
```

```
Relé cyklují s~periodou 2s.

**Je slyšet přepínání relé?**
    </instruction>
    <at-begin>
        <pin id="RELE0" value="False"/>
    </at-begin>

    <at-end>
        <pin id="RELE0" value="False"/>
    </at-end>

    <periodic period="2s">
        <pin id="RELE0" value="True"/>
    </periodic>

    <periodic delay="1000ms" period="2s">
        <pin id="RELE0" value="False"/>
    </periodic>

    <condition delay="2s">
        <user-action/>
    </condition>
</step>
```

Výpis 6.4: Krok cyklování s výstupy

Dále následuje několik kroků, které testují vstupně výstupní piny. Pro tento účel je nutné vždy nakonfigurovat každý pin podle potřeby. V následujícím výpisu z testovacího předpisu je ukázáno, jakým způsobem jsou piny konfigurovány.

Při začátku kroku je pin GPIO4 nakonfigurován jako vstup bez pull-up rezistoru, piny GPIO1 a GPIO2 jsou nakonfigurovány jako výstupy. Podmínkou ukončení kroku je, že pin GPIO4 musí být v logické 0 po dobu minimálně 1 s. Vyhodnocování podmínky je spuštěno 1 s po začátku kroku.

```
<step disabled="false" title="RELE0 rozepnuto"
    report-message="Kontrola funkce GPIO4">
    <instruction>
# Kontrola rozepnutého RELE0 a~GPIO4 jako vstupu

Probíhá samočinné testování. **Nezasahujte prosím do testu.**
    </instruction>
    <at-begin>
        <pin-conf id="GPIO4" config="IN NOPULL"/>
        <pin-conf id="GPIO1" config="OUT"/>
        <pin-conf id="GPIO2" config="OUT"/>
        <pin id="GPIO1" value="False"/>
        <pin id="GPIO2" value="True"/>
```

```
14      </at-begin>
15      <condition delay="1s" duration="1s">
16          <equal>
17              <value value="false"/>
18              <pin id="GPIO4"/>
19          </equal>
20      </condition>
21  </step>
```

Výpis 6.5: Krok testování vstupně výstupních vývodů

Poslední krok, který je v rámci testu vykonán je kontrola komunikační sběrnice RS485. Všechny porty dostupné na předním panelu jsou vzájemně spojeny a kontroluje se, jestli jsou data odeslaná z jednoho portu přijata všemi zbývajícími porty. Výpis č. 6.6 ukazuje zkrácený testovací předpis pro tento krok. Z předpisu jsou pro přehlednost vynechány duplicity týkající se kontroly dvou portů a instrukce pro obsluhu. Popisuje tedy příjem jen z jednoho portu.

Na začátku kroku je nejdříve spuštěn příjem z portu B. Spuštění příjmu je provedeno voláním funkce `start_receive`. Pro volání funkce byl použit kód v Pythonu zabudovaný přímo do testovacího předpisu. Od okamžiku zavolání této funkce, vybraný port port přijímá všechna data, která jsou na něj poslána. Aby nějaká data došla, je třeba je také odeslat. K tomu slouží funkce `transmit`.

Pro vytvoření podmínky je možné také použít zabudovaný kód. Výsledek každého podmínkového příkazu je zaznamenán v atributu `cond_state`. Je-li nastaven na hodnotu `True`, je podmínka splněna. Pokud je nastaven na hodnotu `None` tak podmínka není splněna, ale neznačí selhání testu. Naproti tomu hodnota `False` způsobí okamžité ukončení kroku a jeho selhání. Tento atribut je nastavován podle dat, která jsou přijata. Pro zaznamenání podrobnějších informací do reportu je možné nastavit atribut `details`. Obsah atributu bude doplněn k záznamu z tohoto kroku.

```
1   <step title="Test komunikace"
2       report-message="Test komunikace">
3       <at-begin>
4           <run-embed>
5   self.card_man.cards["MainCard"].com["serial"].start_receive("
        PORTB")
6           </run-embed>
7       </at-begin>
8       <at-end>
9           <run-embed>
10  self.card_man.cards["MainCard"].com["serial"].stop_receive("
        PORTB")
11          </run-embed>
12      </at-end>
13      <periodic period="1s">
```

```
14          <run-embed>
15 self.card_man.cards["MainCard"].com["serial"].transmit("PORTA
       ", b'ahoj ja jsem PORTA')
16 print("odeslano")
17          </run-embed>
18      </periodic>
19
20      <condition delay="5s">
21          <run-embed>
22 prijato = bytes(self.card_man.cards["MainCard"].com["serial
       "].get_received("PORTB"))
23 print("prijato B ", prijato)
24 if b'ahoj ja jsem PORTA' == prijato:
25   self.cond_state = True
26   self.details = "PORTB - ok"
27 elif b'ja nejsem PORTA' == prijato:
28   self.cond_state = False
29 else:
30   self.cond_state = None
31          </run-embed>
32      </condition>
33 </step>
```

Výpis 6.6: Krok testování komunikace

7 Zhodnocení a další možný rozvoj práce

V průběhu práce zařízení mírně změnilo svou plánovanou konstrukci. Původním plánem bylo, že PC bude umístěno jako zásuvka do racku o výšce 1 U. Se sběrnicí I^2C na backplanu mělo být propojeno přes převodník z USB na I^2C. Tento mezičlánek byl vynechán. Namísto toho bylo použito PC, jehož rozměry dovolují montáž na kartu do subracku. Díky tomu se zjednodušila konstrukce, kdy nebylo třeba žádné signály propojovat v rámci několika komponent (subrack s kartami a PC). K propojení dojde jen v rámci subracku, které je zajištěno backplanem.

Použité PC má sběrnici I^2C vyvedenu přímo na konektor, který je určený pro připojení periferií. Tím se zjednodušila také komunikace přes I^2C převodník, která by při jeho použití vyžadovala implementaci D2XX driverů. Později se také ukázalo, že vybraný obvod, tvořící I^2C převodník nepodporuje režim master, ale pouze slave. Nebylo by jej možné pro spojení PC a testeru použít.

Výsledná konstrukce testeru je dostatečně robustní a přitom snadno rozebiratelná. Především možnost vyndat subrack jako monoblok obsahující jak karty tak i PC je velice užitečná. Zejména pro případy, kdy je nutné mít přístup k backplanu za běhu testu.

Přípravek pro ověření funkčnosti testeru se při testování osvědčil, protože odhalil chybu v osazení hlavní karty. Na hlavní kartě byly nedopatřením osazeny budiče sběrnice RS485, která nepracují s napájením 3,3 V. Po výměně budičů za správný typ byly další testy již úspěšné.

Do budoucna je možné tester rozšiřovat o další karty. Pravděpodobně první doplňující kartou která vznikne, bude karta s napájecími zdroji a měřiči spotřeby. Absence karty je velmi citelná a bez ní je třeba k testeru vždy přidat externí zdroj.

8 Závěr

Úkolem této práce bylo popsat metody testování používaných při výrobě elektroniky. Dále měl být vytvořen návrh testeru včetně schémat a desky plošných spojů. Součástí práce bylo také vytvoření softwarového vybavení testeru a ověření činnosti zařízení v praxi.

V první části práce jsou popsány významné metody testování DPS a celých zařízení při výrobě. Jsou popsány jejich výhody i nevýhody.

Ve druhé části jsou popsány požadavky, které byly na tester kladeny a které musely být zohledněny při návrhu. Kapitola zmiňuje, jak požadavky na hardware tak i požadavky na SW a jeho použití a konfiguraci.

Ve třetí části je podrobně popsán návrh HW pro testovací platformu. Platforma je navržena jako systém v rackové skříni, který je dále rozšiřitelný pomocí zásuvných karet. Návrh se zabývá zejména hlavní kartou, která zajišťuje základní funkcionalitu testeru. V kapitole je podrobně popsána koncepce testeru. Dále je také popsán výběr komponent a podrobný návrh jednotlivých obvodů na hlavní kartě.

Další část popisuje softwarové vybavení. Rozebírá požadavky na testovací předpisy, které popisují celý test na jehož základě tester kontroluje výrobky. Je zde popis vrstevnaté struktury softwaru a jejích vzájemných vazeb, následovaný podrobným popisem implementace softwaru a jeho ovládáním.

Pátá kapitola popisuje jakým způsobem je tester mechanicky konstruován. Zabývá se jednotlivými částmi konstrukce, jejich výrobou a montáží.

Šestá kapitola ukazuje jakým způsobem se tester konfiguruje. Využívá k tomu testovací předpis, použitý pro demonstraci funkčnosti. Postupně rozebírá stěžejní části testovacího předpisu, vysvětluje jakým způsobem je daný krok popsán. Také se zmiňuje o důvodech použití různých možností testovacího předpisu.

Poslední kapitola rozebírá návrh z hlediska změn a kompromisů, které bylo nutné v průběhu práce udělat. Také popisuje jakým směrem by bylo možné tester dále rozvíjet.

V rámci práce bylo navrženo a zkonstruováno zařízení určené pro funkcionální testování elektronických výrobků. Průběh testu je popsán pomocí XML dokumentu, který tvoří komplexní testovací předpis, jak pro obsluhu testeru, tak pro samotný tester. Aby bylo možné tester pohodlně ovládat vzniklo grafické rozhraní, které umožňuje operátorovi zadávat informace, které tester pro svou činnost potřebuje.

Činnost zařízení byla ověřena pomocí testovacího přípravku. Při testech byl tester schopen odhalit chyby a vytvářel o tom reporty podle požadavků. Zařízení splnilo veškerá očekávání.

Literatura

[1] AMTECH. Automatická optická inspekce plně ve 3D. *DPS Elektronika od A do Z*. Liberec: CADware, 2013, **2013**(2). ISSN 1805-5044.

[2] KLAUZ, Milan. Elektrický test osazených desek metodou Flying Probe. *DPS Elektronika od A do Z*. Liberec: CADware, 2016, **2016**(3), 32-33. ISSN 1805-5044.

[3] MICRO SYSTEMS S.R.L. FLYING PROBE MACHINE. *MICRO-SYSTEMS* [online]. Modena, 2016 [cit. 2018-12-28]. Dostupné z: `http://www.micro-systems.it/en/flying-probe-test-system/`

[4] Flying Scorpion FLS980Dxi — Flying Probe Tester. *Acculogic* [online]. Ontario: Acculogic [cit. 2018-10-20]. Dostupné z: `https://acculogic.com/products/flying-probe-testers/flying-scorpion-fls980/`

[5] FUTURE TECHNOLOGY DEVICES INTERNATIONAL. *FT200XD USB I2C SLAVE IC* [online katalogový list]. In: . 2017 [cit. 2018-12-27]. Dostupné z: `https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT200XD.pdf`

[6] FUTURE TECHNOLOGY DEVICES INTERNATIONAL. *FT4232H QUAD HIGH SPEED USB TO MULTIPURPOSE UART/MPSSE IC* [online katalogový list]. In: . 2018 [cit. 2018-12-27]. Dostupné z: `https://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT4232H.pdf`

[7] Inspection and test techniques. In: *MicroElectronics Cloud Alliance – MECA* [online]. [cit. 2018-12-28]. Dostupné z: http://www.ett.bme.hu/meca/Courses/AIT/8_3.html

[8] TEXAS INSTRUMENTS INCORPORATED. *I2C Bus Pullup Resistor Calculation* [online aplikační poznámka]. In: . 2015 [cit. 2018-12-27]. Dostupné z: `http://www.ti.com/lit/an/slva689/slva689.pdf`

[9] MICROCHIP TECHNOLOGY INC. *LAN9514/LAN9514I* [online katalogový list]. In: . 2016 [cit. 2018-12-26]. Dostupné z: `http://ww1.microchip.com/downloads/en/devicedoc/00002306a.pdf`

[10] MICROCHIP TECHNOLOGY INC. *MCP100/101* [online katalogový list]. In: . 2001 [cit. 2018-12-26]. Dostupné z: `http://ww1.microchip.com/downloads/en/DeviceDoc/11187f.pdf`

[11] MICROCHIP TECHNOLOGY INC. *MCP16301/H* [online katalogový list]. In: . 2015 [cit. 2018-12-26]. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/20005004d.pdf

[12] MICROCHIP TECHNOLOGY INC. *MCP23017/MCP23S17* [online katalogový list]. In: . 2016 [cit. 2018-12-27]. Dostupné z: http://ww1.microchip.com/downloads/en/devicedoc/20001952c.pdf

[13] MICROCHIP TECHNOLOGY INC. *MCP3426/7/8* [online katalogový list]. In: . 2009 [cit. 2018-12-27]. Dostupné z: https://ww1.microchip.com/downloads/en/DeviceDoc/22226a.pdf

[14] MICROCHIP TECHNOLOGY INC. *MCP47FEBXX* [online katalogový list]. In: . 2015 [cit. 2018-12-28]. Dostupné z: http://ww1.microchip.com/downloads/en/DeviceDoc/20005375A.pdf

[15] STMICROELECTRONICS. NUCLEO-F411RE. *STMicroelectronics* [online]. 2018 [cit. 2018-12-27]. Dostupné z: https://www.st.com/en/evaluation-tools/nucleo-f411re.html

[16] TEXAS INSTRUMENTS INCORPORATED. *OPAx170 36-V, Single-Supply, SOT553, Low-Power Operational Amplifiers Value Line Series* [online katalogový list]. In: . 2018 [cit. 2018-12-28]. Dostupné z: http://www.ti.com/lit/ds/symlink/opa4170.pdf

[17] NXP SEMICONDUCTORS. *PESDxS2UT series* [online katalogový list]. In: . 2014 [cit. 2018-12-27]. Dostupné z: https://www.tme.eu/cz/Document/efa69d9c2b7084c0ec85cad1371d68e3/PESD3V3S2UT.pdf

[18] STMICROELECTRONICS. *ST3232EB, ST3232EC* [online katalogový list]. In: . 2018 [cit. 2018-12-27]. Dostupné z: https://www.st.com/resource/en/datasheet/st3232eb.pdf

[19] STMICROELECTRONICS. *ST3485EB, ST3485EC, ST3485EI, ST3485EIY* [online katalogový list]. In: . 2017 [cit. 2018-12-27]. Dostupné z: https://www.st.com/resource/en/datasheet/st3485eb.pdf

[20] ECE. *SURFACE MOUNT PTC: SN (1206) MODEL* [online katalogový list]. In: . [cit. 2018-12-27]. Dostupné z: https://www.tme.eu/cz/Document/0302dcd94dd6f5e9a9ded1eefb7cf5df/20151120103220.pdf

[21] KUGELSTADT, Thomas. *The RS-485 Design Guide* [online aplikační poznámka]. In: . 2016 [cit. 2018-12-27]. Dostupné z: http://www.ti.com/lit/an/slla272c/slla272c.pdf

Seznam symbolů, veličin a zkratek

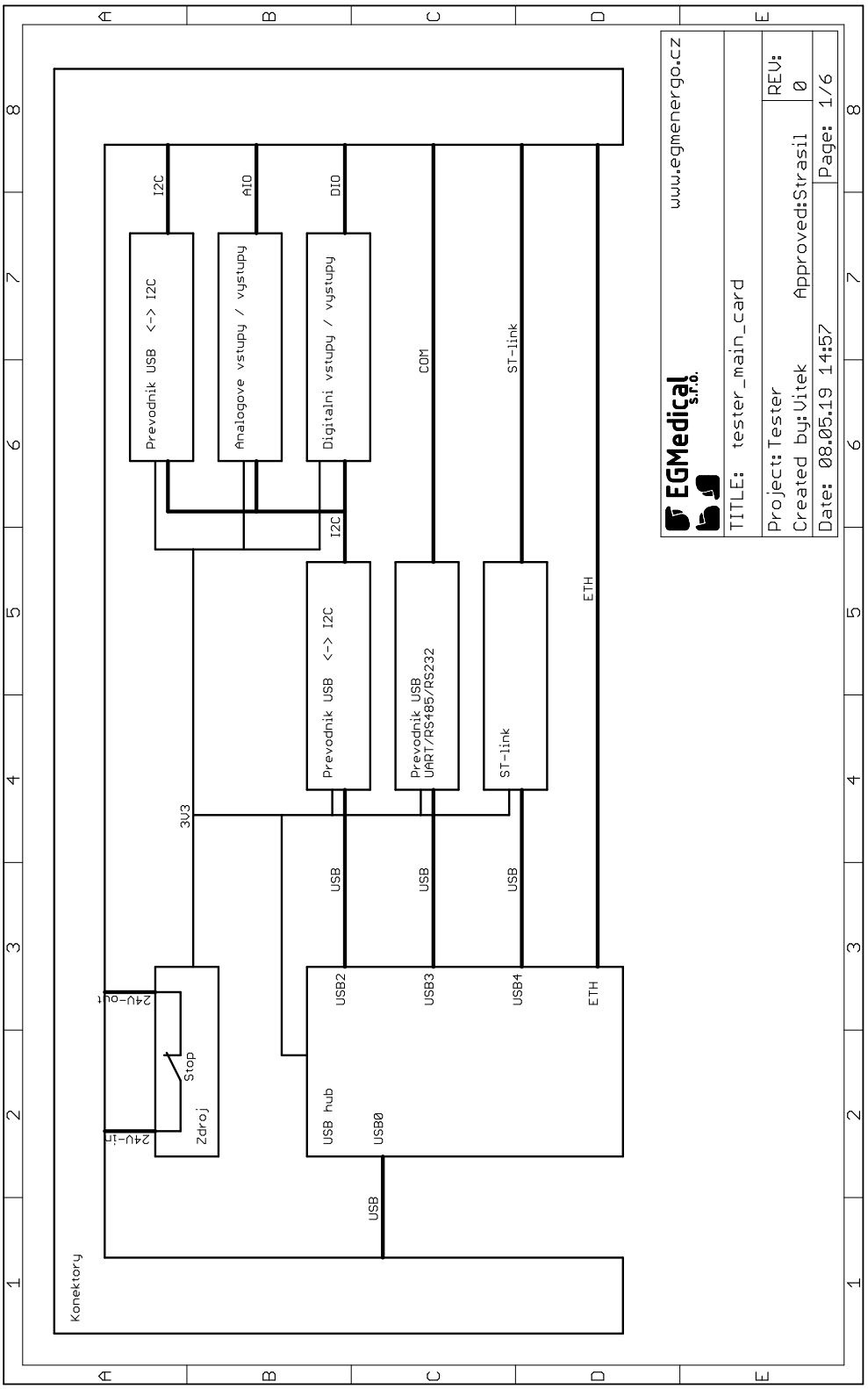
AOI	Automatická optická inspekce
DUT	Testované zařízení – device under test
DPS	Deska plošných spojů
CNC	Počítačem řízené stroje
RF	Rádiový kmitočet – radio frequency
JTAG	Joint Test Action Group – standard pro programování a ladění elektronických komponent
FW	Programové vybavení mikrokontroléru – firmware
MCU	Mikrokontrolér
PC	Počítač
ADC	Analogově digitální převodník
DAC	Digitálně analogový převodník
ESD	Elektrostatický výboj
PLL	Fázový závěs
GPIO	Vstupy a výstupy pro obecné použití
VCP	Virtuální sériový port

Seznam příloh

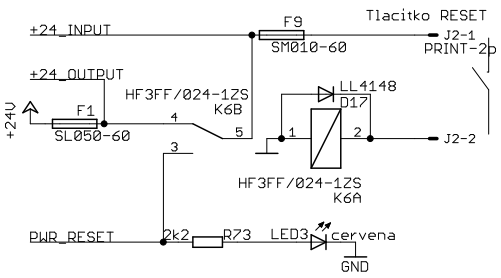
A Obsah elektronických příloh

- /
  - diplomova_prace.pdf ..... text práce v elektronické podobě
  - HW/ ..... dokumentace HW
    - hlavni_karta/ ..... dokumentace hlavní karty
      - lbr/ ..... knihovny programu Eagle
      - vyrobni_podklady/ ..... výrobní podklady hlavní karty
    - celni_panely/ ..... výrobní podklady pro čelní panely
    - backplane_PC_karta.pdf ..... zapojení backplanu a PC karty
  - SW/ ..... softwarové vybavení testeru
    - card_config/ ..... konfigurační soubory karet
    - icons/ ..... ikony pro GUI
    - operators/ ..... seznam operátorů
    - tests/ ..... testovací předpisy
  - fotografie/ ..... fotografická dokumentace

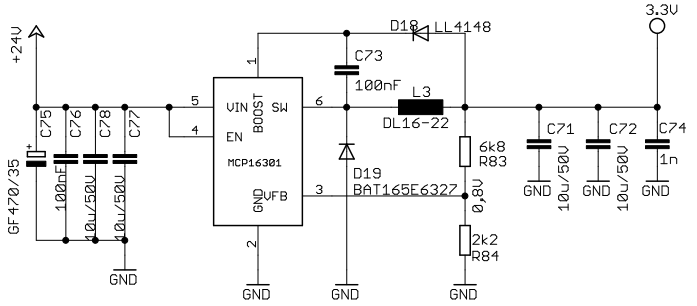
B Schéma hlavní karty



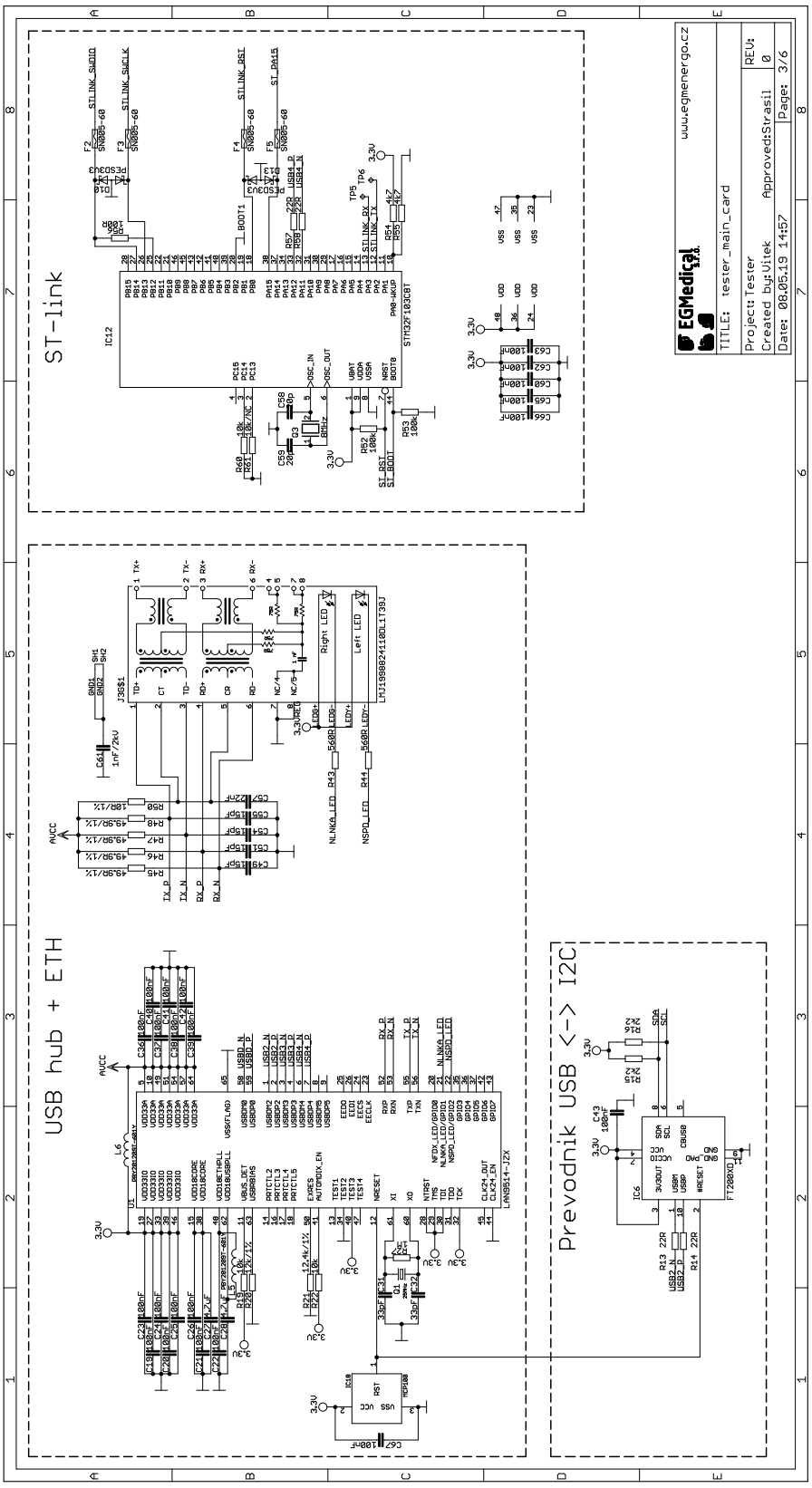
STOP tlacitko

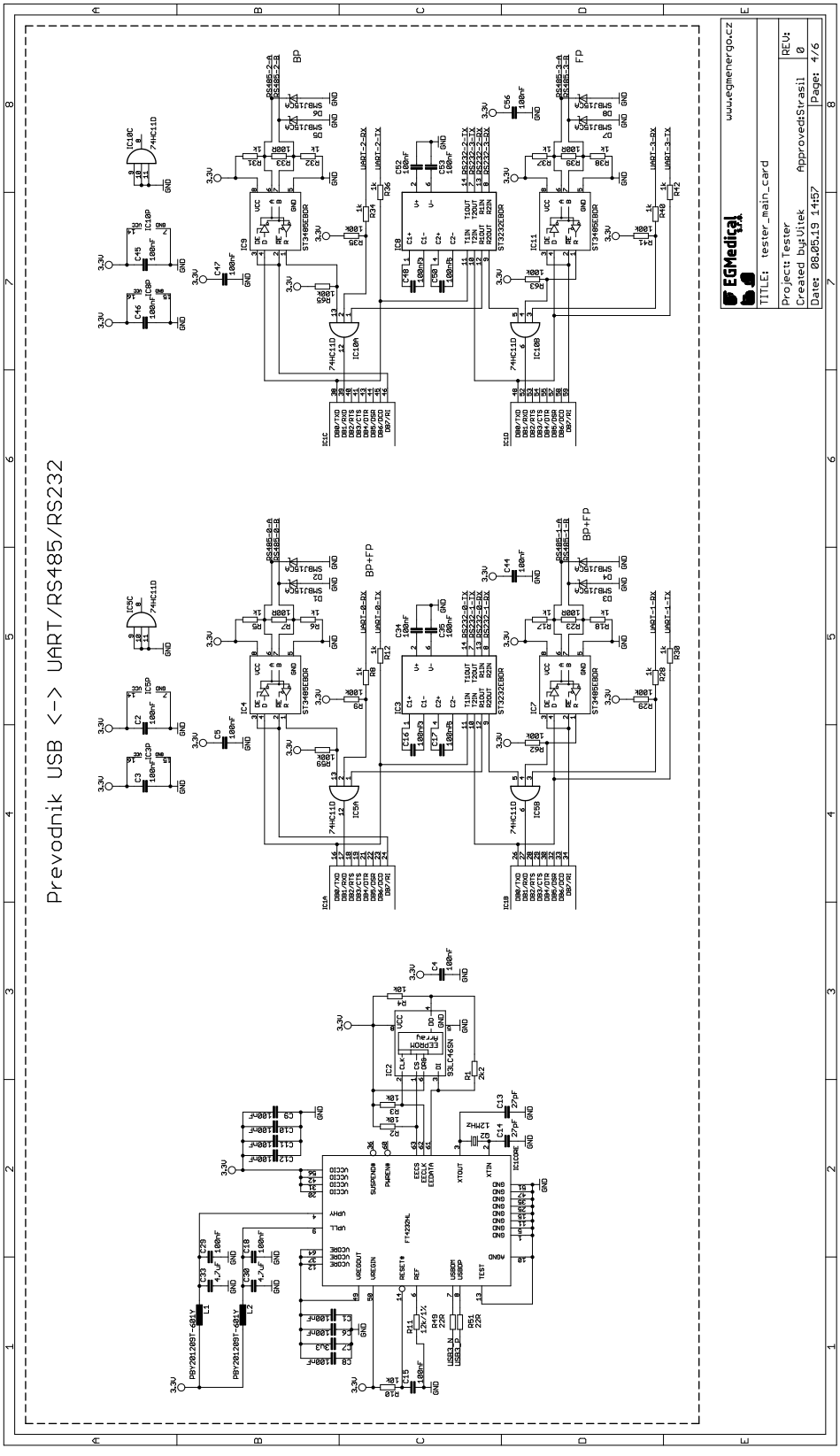


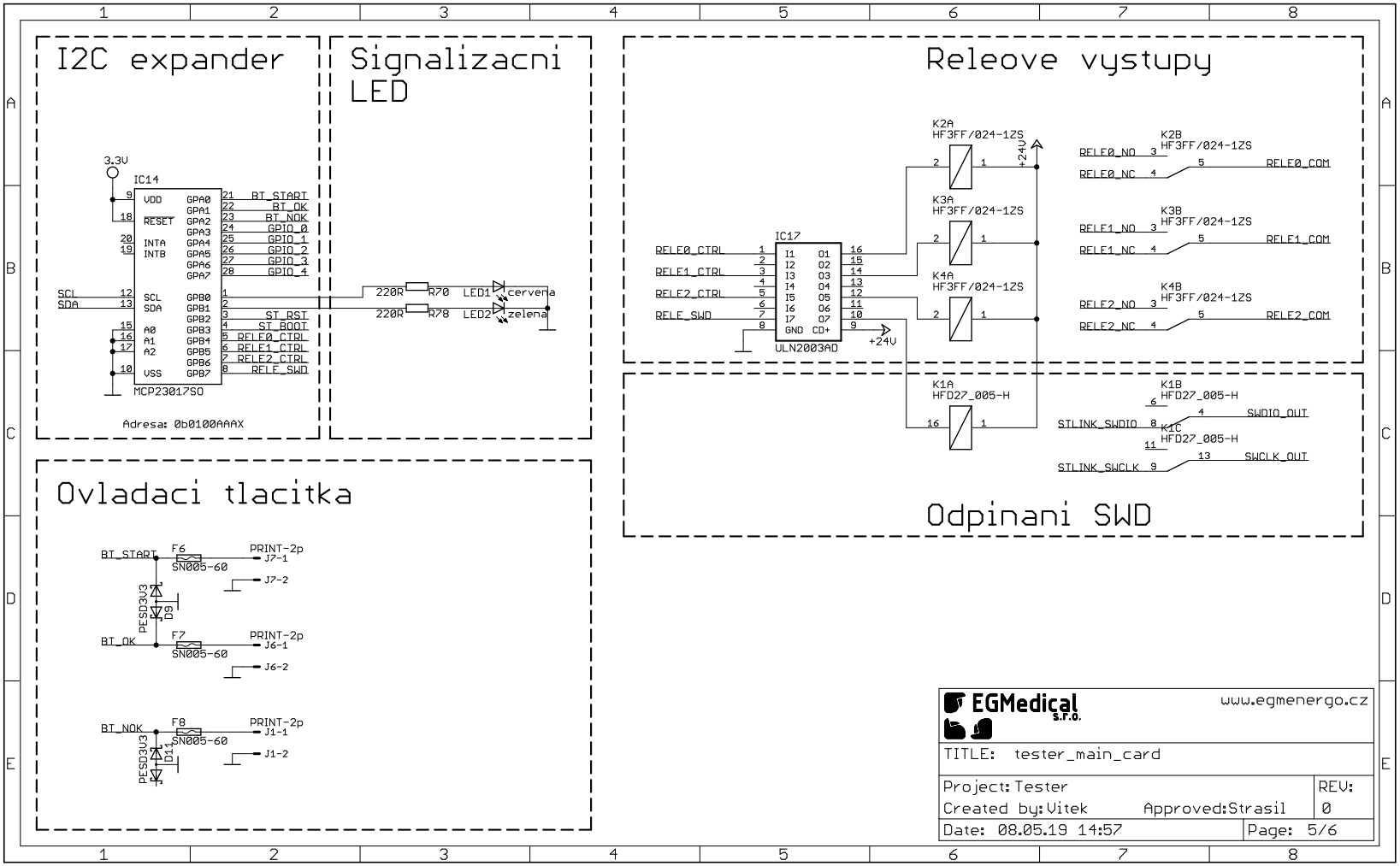
Napajeci zdroj

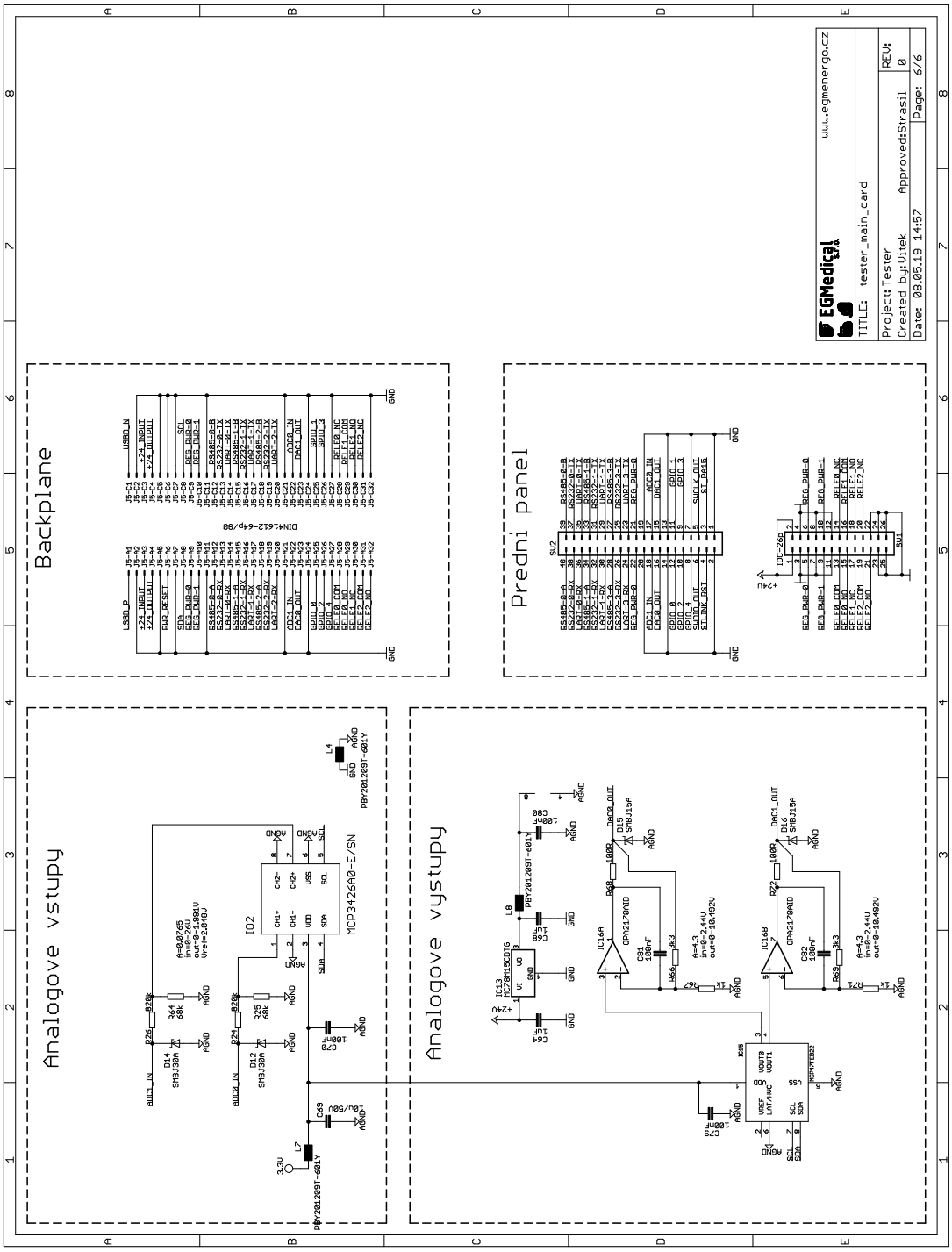


EGMedical S.R.O.		www.egmenergo.cz
TITLE: tester_main_card		
Project: Tester		REV:
Created by: Vitek	Approved: Strasil	0
Date: 08.05.19 14:57		Page: 2/6

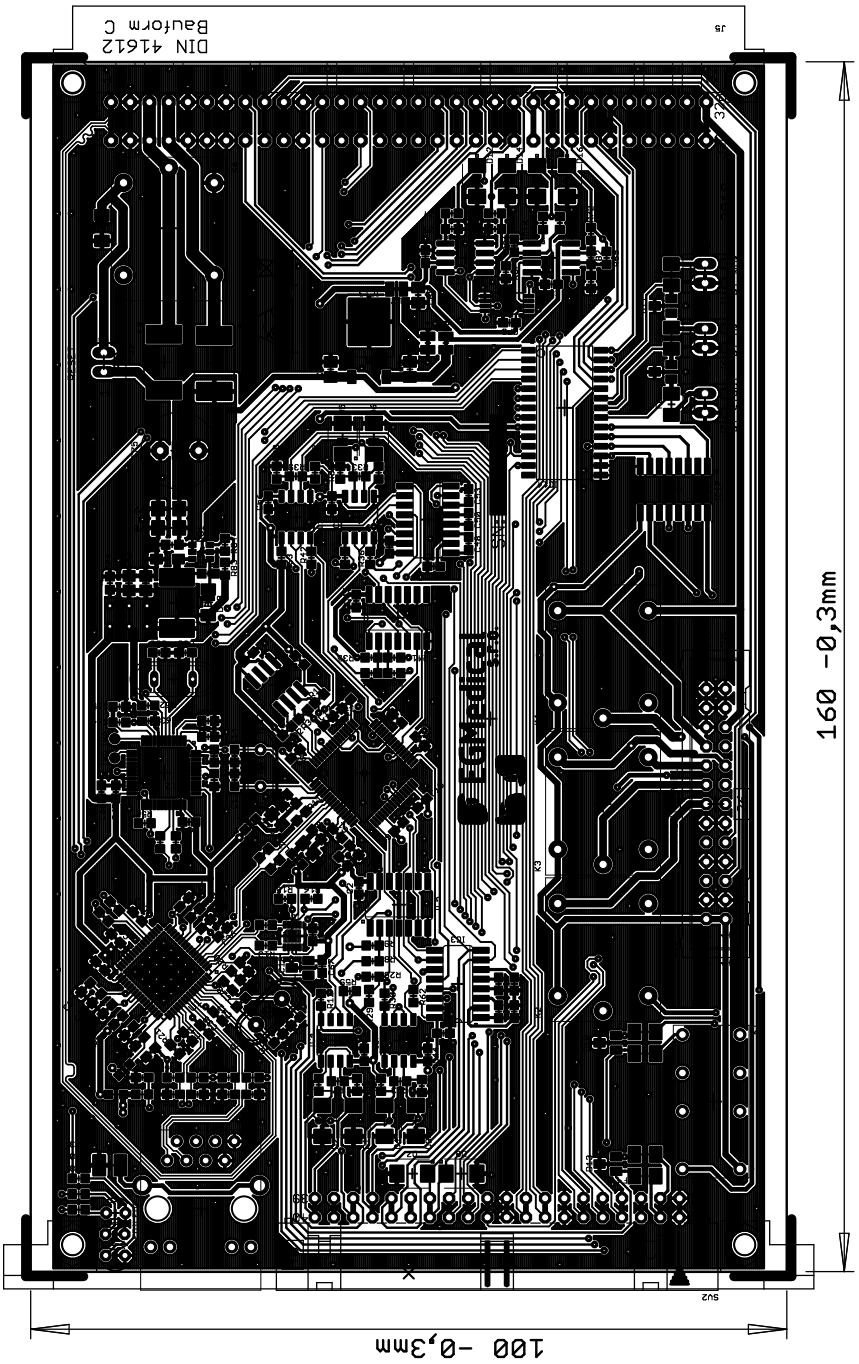

ST-link
USB hub + ETH
Prevodnik USB <-> I2C
www.egmenergo.cz
EGMedical s.r.o.
TITLE: tester_main_card
Project: Tester
Created by: Vitek
Approved: Strasil
Date: 08.05.19 14:57
REV: 0
Page: 3/6


Prevodnik USB <-> UART/RS485/RS232
BP
FP
BP+FP
www.egmenergo.cz
EGMedical
TITLE: tester_main_card
Project: Tester
Created by: Ultek
Approved: Strasil
Date: 08.05.19 14:57
REV: 0
Page: 4/6


I2C expander
Signalizacni LED
Releove vystupy
Ovladaci tlacitka
Odpinani SWD
3.3V
IC14
MCP23017SO
Adresa: 0b0100AAAX
SCL
SDA
220R
R70
LED1
cervena
R78
LED2
zelena
IC17
ULN2003AD
+24V
K2A HF3FF/024-1ZS
K3A HF3FF/024-1ZS
K4A HF3FF/024-1ZS
K2B HF3FF/024-1ZS
K3B HF3FF/024-1ZS
K4B HF3FF/024-1ZS
K1A HFD27_005-H
K1B HFD27_005-H
K1C HFD27_005-H
RELE0_CTRL
RELE1_CTRL
RELE2_CTRL
RELE_SWD
RELE0_NO
RELE0_NC
RELE0_COM
RELE1_NO
RELE1_NC
RELE1_COM
RELE2_NO
RELE2_NC
RELE2_COM
STLINK_SWDIO
STLINK_SWCLK
SWDIO_OUT
SWCLK_OUT
BT_START
BT_OK
BT_NOK
F6
F7
F8
SN005-60
PRINT-2p
PESD3V3
D9
D11
EGMedical s.r.o.
www.egmenergo.cz
TITLE: tester_main_card
Project: Tester
REV: 0
Created by: Vitek
Approved: Strasil
Date: 08.05.19 14:57
Page: 5/6


Backplane
Predni panel
Analogove vstupy
Analogove vystupy
EGMedical
www.egmenergo.cz
TITLE: tester_main_card
Project: Tester
Created by: Vitek
Approved: Strasil
REV: 0
Date: 08.05.19 14:57
Page: 6/6

C Horní vrstva DPS



D Dolní vrstva DPS