VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2019 Bc. Roman Sičkaruk

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

PROGRAM PRO EXPORT VEKTOROVÝCH OBRÁZKŮ Z AUTODESK EAGLE
VECTOR IMAGE EXPORT PROGRAM FOR AUTODESK EAGLE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Roman Sičkaruk**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Pavel Hanák, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**
Ústav telekomunikací

Student: Bc. Roman Sičkaruk ***ID:*** 165518
Ročník: 2 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Program pro export vektorových obrázků z Autodesk Eagle

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a vytvořte ULP program pro export obrázků ve formátu EMF (Enhanced Windows Metafile), který bude podporovat všechny funkce systému Autodesk Eagle verze 9. Program musí exportovat aktuálně viditelné vrstvy, přičemž musí zachovat jejich orientaci, barvu i pořadí vykreslení. Při práci můžete vyjít z existujícího programu eagle2svg-1_4_1.ulp, ve kterém však nejdříve musíte odstranit chyby. Správnou funkci otestujte nejméně na 5 složitých schématech a deskách.

DOPORUČENÁ LITERATURA:

[1] Specifikace formátu Microsoft EMF, verze 13.0.

Dostupné z: https://msdn.microsoft.com/en-us/library/cc230514.aspx

[2] MATOUŠEK, D. Programování v jazyce C v příkladech. BEN, Praha, 2012. ISBN 978-80-7300-438-5

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 16.5.2019

Vedoucí práce: Ing. Pavel Hanák, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Diplomová práce se zabývá návrhem a vytvořením ULP programu pro export obrázků ve formátu EMF (Enhanced Windows Metafile). Cílem práce je umožnit podporu všech funkci systému Autodesk Eagle verze 9. Práce je významná z důvodů, že ostatní ULP programy pro export vektorových obrázků do jiných formátů obsahují chyby a nedokonalosti a jsou určené pro starší verze Autodesk Eagle. Vzhledem k nedostatečnému množství literatury o metaformátu EMF, je přínosem práce popis chování základních struktur a způsobu vytvoření metasouboru. Práce také poskytuje ukázku průchodu jednotlivých částí schémat a desek plošných spojů v jazyce ULP. Hlavní cíl práce byl splněn. Uživatel může exportovat aktuálně viditelné vrstvy, nastavit pořadí jejich vykreslení, se zachováním barev a orientace. Program byl otestovan v systémech Autodesk Eagle verze 9.1.3 a 9.3.2.

KLÍČOVÁ SLOVA

Autodesk Eagle, ULP, Windows metaformát, WMF, EMF

ABSTRACT

This master's thesis deals with design and creation of ULP program for export of vector images in EMF (Enhanced Windows Metafile) format. The main aim of this thesis is to add support of all features of Autodesk Eagle version 9. Thesis is important due to bugs and imperfections of other ULP programs for exporting of vector images to other formats also because of they aim for older versions of Autodesk Eagle. With regard to small quantity of EMF literature, thesis is beneficial as it describes behaviour of base structures and describes creation of EMF metafile. It also provides example of iteration over individual parts of schemes and printed circuit boards. Main goal of this thesis was completed. User is allowed to export currently visible layers, set order of their printing, with remain of true colors and orientation. Program was tested in Autodesk Eagle systems version 9.1.3 and 9.3.2.

KEYWORDS

Autodesk Eagle, ULP, Windows metaformat, WMF, EMF

SICHKARUK, Roman. *Program pro export obrázků z Autodesk Eagle*. Brno, Rok, 72 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Pavel Hanák, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Program pro export obrázků z Autodesk Eagle“ 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 Ing.Pavlovi Hanákovi, Ph.D. za odborné vedení, konzultace, mimořádnou trpělivost, podnětné návrhy k práci a účinnou pomoc při řešení.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Autodesk Eagle je software pro návrh desek a schémat s plošnými spoji. Jedná se o multiplatformní a uživatelský přívětivý program. Má mnoho užitečných funkcí, které jiné podobné programy postrádají. Mezi tyto funkce patří autorouting, interaktivní spojení mezi schématy a deskami, snadné vytvoření a správa knihoven, podsvícení vybraných cest a mnoho dalších. Mimořádnou výhodou Autodesk Eagle je zabudovaný uživatelský skriptovací jazyk ULP.

ULP (User Language Programming) je skriptovací jazyk, využívající syntaxi podobnou jazyku C. ULP obsahuje řídící a ovládací konstrukce, umožňující přistoupit k vnitřním strukturám schémat, desek a grafického uživatelského rozhraní. Většinu opakovaných úkonů lze tedy jednoduše systematizovat a chybějící nebo specifické požadavky od zákazníků implementovat bez nutnosti úprav zdrojových kódů. Existuje uživatelská platforma, kde jednotliví programátoři sdílí své ULP skripty a tím neustále rozšiřují vymoženosti Autodesk Eagle.Jedna důležitá vymoženost ovšem chybí - generování vektorových obrázků s podporou všech funkcí Autodesk Eagle verze 9. Právě proto se tomu věnuje tato semestrální práce. Zvoleným vektorovým formátem je metaformát EMF (Enhanced Windows Metafile) z důvodu jeho přenositelnosti a vysoké podpory v nativních aplikacích téměř všech operačních systémů.

Práce sestává z pěti kapitol. První kapitola se věnuje metaformátu WMF, jelikož je základem ostatních metaformátů Windows, včetně EMF. Ve druhé kapitole je podrobný popis a zpracování informací o metaformátu EMF. Návrh programu a vysvětlení jeho jednoduchých funkcí, spolu s odkazy na podrobný popis pro snadnou orientaci v rámci práce, je ve třetí kapitole. Zajímavým implementačním detailům a popisu programové části se věnuje čtvrtá kapitola. Poslední kapitola shrnuje dosažené výsledky a naznačuje směry budoucího vývoje.

1 WMF

V roce 1992 vydala společnost Microsoft 16 bitový formát metasouborů WMF (Windows Metafile). První podporovaný operační systém byl Windows 3.0. Pro Windows XP ovšem byla potřeba podporovat 32 bitový formát. Tak jenom o rok později vznikl formát EMF a první rozšíření EMF+ (Enhanced Metafile Format Plus Extensions). Základem je ovšem právě formát WMF a formát EMF jej pouze obaluje o další funkcionalitu, rozšíření a umožňuje použití 32 bitových příkazů rozhraní Windows.[4]

1.1 Popis

Grafický metaformát WMF slouží k uložení jak vektorové, tak i rastrové grafiky a grafických prostředků přímo v paměti nebo na disku. Rastrové obrázky mohou být uložené ve formátech:

- **DDB** (Device Dependent Bitmap)[1] – tento typ bitových map obsahuje matici s pixely a matici s mapováním pixelů na barvy výstupního zařízení.
- **DIB** (Device Independent Bitmap)[2] – obsahuje matici složenou z trojic hodnot základních barevných složek RGB pro každý pixel bitové mapy.

Hlavní výhodou Windows metasouboru je nezávislost na výstupním zařízení, jelikož jejich obsah tvoří bloky, které se mapují na posloupnost příkazů pro rozhraní GDI[3]. Právě na straně GDI totiž probíhá ošetření různých typů výstupních zařízení – monitorů, tiskáren, atd. Z hlediska uživatele tedy není práce s grafickými objekty ovlivněná výstupním zařízením, ale pouze jeho kontextem (např. nastavením hraničních oblastí, maximálních velikostí, barvy písma). Způsob mapování záznamu na příkazy je podrobně vysvětlen v sekci 1.2.

Jak je patrné z názvu, obvykle jsou metasoubory uloženy v paměti a po vykreslení uvolněny. V případě, že není dostatek paměti, nebo je nutné se souborem pracovat opakovaně, existuje možnost uložení na disk. Maximální velikost souboru je 4 GB.

I přesto, že spousta multiplatformních aplikací podporuje WMF formát, hlavním důvodem jeho existence je návaznost na vnitřní rozhraní operačních systémů Windows [2][4].

[1] https://docs.microsoft.com/en-us/windows/desktop/gdi/device-dependent-bitmaps
[2] https://docs.microsoft.com/en-us/windows/desktop/gdi/device-independent-bitmaps
[3] Graphics Device Interface

1.2 Struktura

V předchozí sekci bylo zmíněno, že metasoubor je tvořen posloupností bloků mapovaných na GDI příkazy. Úvodním blokem je hlavičkový záznam. Podle typu a struktury hlavičky, lze soubory rozdělit na [2]:

- **standard** – klasická hlavička o délce 18 B. Následuje posloupnost záznamů a ukončovací záznam. Obrázek 1.2 znázorňuje tuto strukturu.
- **placeable** – obsahuje navíc 22 B předcházejících klasické hlavičce. V těchto datech jsou ukryté informace o poloze grafických objektů v rámci zobrazovacího zařízení, mapování na logické jednotky a kontrolní součet pro ověření integrity. Pro lepší pochopení slouží obrázek 1.1.
- **clipboard** – začíná 8 B s informací o jednotkách použitých při zachycení, šířce a výšce metasouboru. Obsahuje také ukazatel na místo v paměti, kde je soubor uložen. Následují záznamy typu standard. Struktura je znázorněna na obrázku 1.3.

Obr. 1.1: Struktura souboru typu placeable.

Obr. 1.2: Struktura souboru typu standard.

Obr. 1.3: Struktura souboru typu clipboard.

Jednotlivé záznamy se skládají ze třech částí [2]:

- **Size** – celková velikost bloku včetně položek Function a Size.
- **Function** – unikátní identifikátor funkce GDI.
- **Parameters**[] – pole proměnlivé délky s parametry funkce.

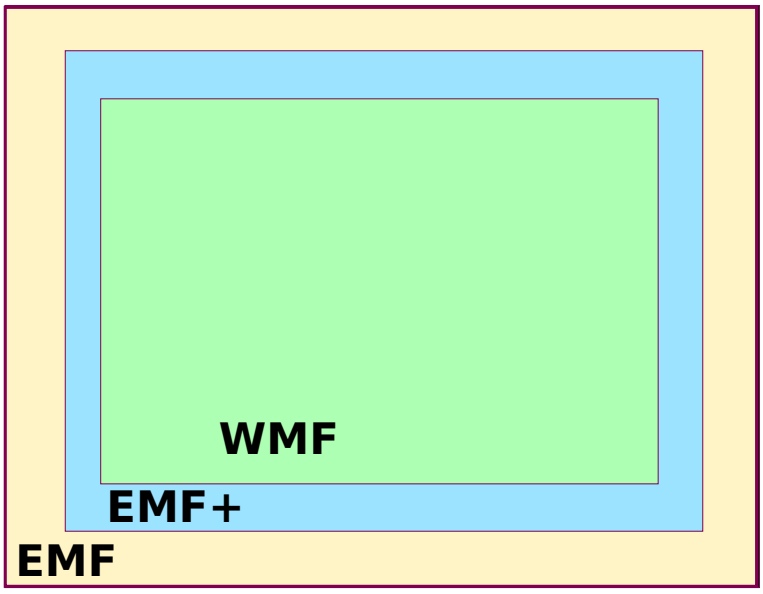
Identifikátor Function má délku 16 B. Prvních 8 B určuje počet parametrů funkce. Dalších 8 B její unikátní označení. Při mapování záznamu na funkci se dohledá GDI funkce podle identifikátoru záznamu, jednotlivé parametry se předávají v opačném

pořadí, než jsou uloženy v záznamu. Například funkce LineTo() pracuje se dvěma parametry – souřadnice na ose x (X) a souřadnice na ose y (Y). V záznamu WMF budou tedy uloženy v opačném pořadí – Y,X.

Zvláštním záznamem je ukončovací záznam. Jedná se o poslední záznam v metasouboru. Hodnota položky Function je 0x0000h. Položka Size je nastavena na hodnotu 0x0003h. Pole parametrů je prázdné. Tento záznam vždy musí být přítomen na konci metasouboru ke korektnímu ukončení čtení [2].

2 EMF

Metaformát EMF (Windows Enhanced Metafiles) je 32-bitovou verzí formátu WMF. Hlavička EMF sdružuje placeable a clipboard hlavičky. Novinkou je možnost určení své palety barev a komentáře. Podpora API Win32 také přináší nové vymoženosti a funkce. Způsob nastavení kontextu výstupního zařízení byl v mnoha ohledech přepracován. Tvůrci dokonce tvrdí, že se jedná o první formát "skutečně" nezávislý na výstupním zařízení. Při interpretaci metasouboru výsledné grafické objekty zachovávají svůj tvar, proporce a barvy. Vztah mezi metaformáty znázorňuje obrázek 2.1. V následujících podkapitolách jsou podrobně popsané klíčové záznamy, potřebné k pochopení této práce [1][4].



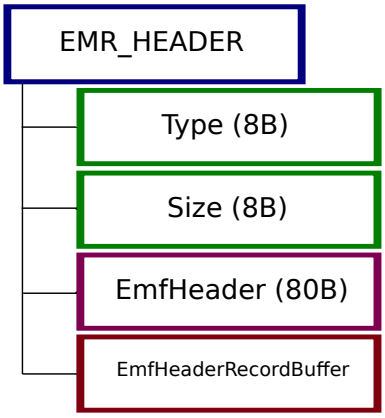
Obr. 2.1: Vztah mezi metaformáty Windows.

2.1 Hlavička

Na začátku souboru EMF je vždy umístěna jeho hlavička (EMR_HEADER). Její struktura odpovídá obecné struktuře záznamu grafického metaformátu EMF (odkaz). Ve srovnání s hlavičkou formátu WMF je mnohem složitější a rozsáhlejší. Jedná se o kontrolní bod, na kterém záleží jestli budou ostatní záznamy správně interpretovány. Právě zde jsou specifikovány nastavení vlastností zařízení při vytvoření metasouboru. Tyto informace umožňují poskytnout nezávislost na typu výstupního zařízení, jelikož se nastavení vstupu vhodně promítnou na nastavení výstupu. Prvních osm B specifikuje typ záznamu a délku bloku. Samotná délka hlavičky je

minimálně 80 B. Výsledná délka ovšem závisí na délce popisu souboru od uživatele. Jednotlivé položky záznamu jsou zobrazeny na obrázku 2.2 a poněvadž jsou doopravdy klíčové, jsou krátce vysvětlené [1]:

- **Type** – typ záznamu (0x00000001).
- **Size** – velikost hlavičky v B.
- **EmfHeader** – standardní hlavička o délce 80 B.
- **EmfHeaderRecordBuffer** – volitelná položka obsahující rozšíření hlavičky.



Obr. 2.2: Struktura záznamu EMR_HEADER.

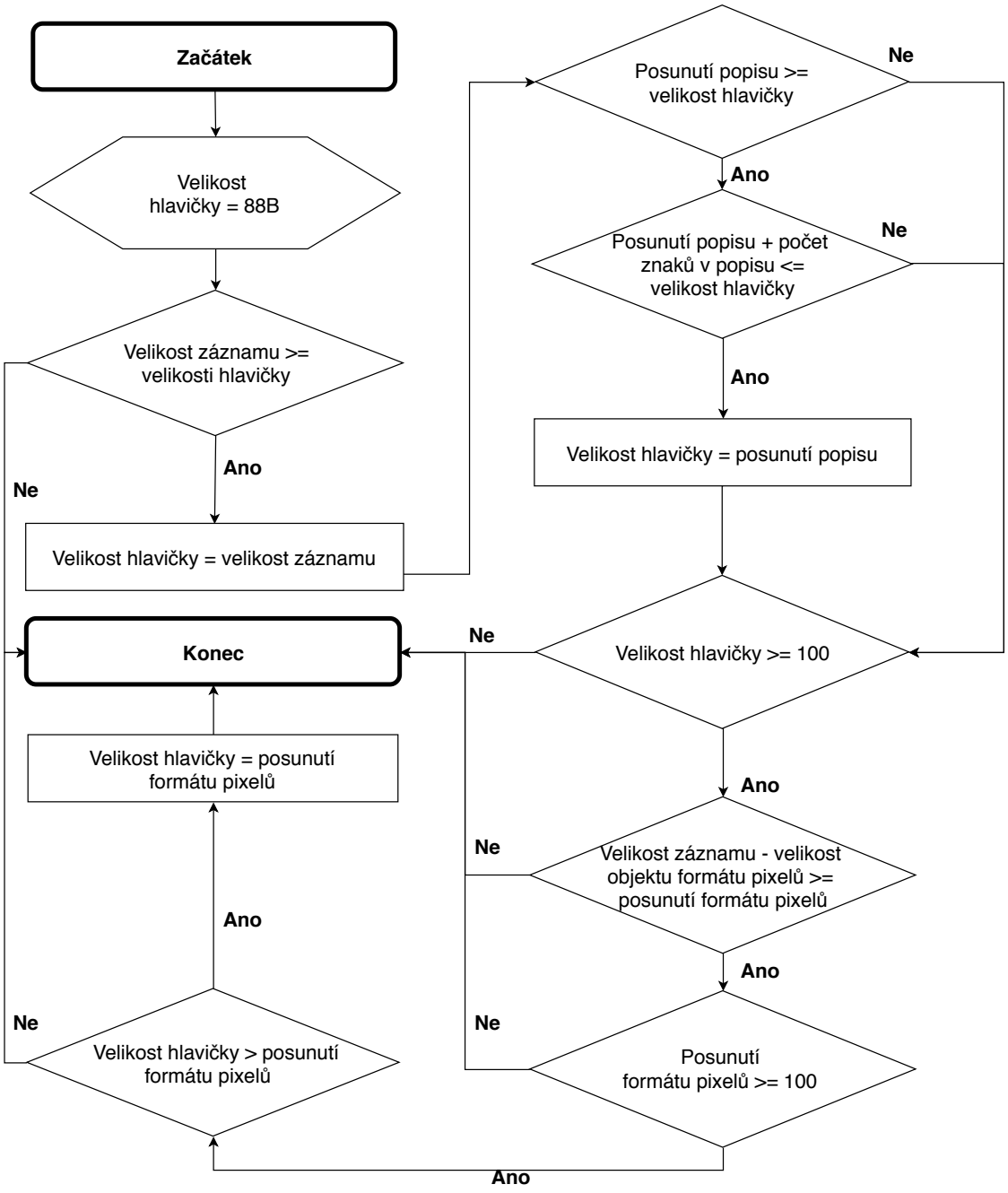
2.1.1 EmfHeaderRecordBuffer

Jak již bylo zmíněno v předchozí sekci, jedná se o pole bytů obsahující rozšíření klasické hlavičky. Samotná struktura položky se skládá ze dvou částí:

- **EmfHeaderRecordParm** – pole obsahující rozšíření.
- **AlignmentPadding** – volitelná část o délce až tří B. Tato část slouží k zarovnání, aby celková délka hlavičky byla násobkem 4 B.

Rozšíření jsou volitelná a na základě délky hlavičky GDI určí při přehrávání metasouboru typ hlavičky – **EmfMetafileHeader**, **EmfMetafileHeaderExtension1**, nebo **EmfMetafileHeader2**. Algoritmus k určení popisuje obázek 2.3.

Pokud výsledná velikost hlavičky je větší než 107 B, typ hlavičky je EmfMetafileHeaderExtension2, větší než 99 – EmfMetafileHeaderExtension1, jinak EmfMetafileHeader.



Obr. 2.3: Algoritmus k určení délky hlavičky.

2.1.2 EmfMetafileHeader

Původní hlavička EMF metasouboru dostupná s prvním vydáním obsahuje informace o struktuře metasouboru a základní nastavení kontextu výstupního zařízení. Jednotlivé položky jsou vysvětleny v tabulce 2.1[1].

Tab. 2.1: Význam jednotlivých položek EmfMetafileHeader [1]

Název	**Velikost [B]**	**Význam**
Bounds	16	Objekt typu RectL (2.3.2), určující obdélníkovou hraniční oblast. Prakticky se jedná o nejmenší ohraničovací rámeček okolo všech grafických objektů v souboru v logických jednotkách
Frame	16	Stejný objekt, jako Bounds, ovšem jednotkou je setina milimetru.
RecordSignature	4	Identifikátor formátu metasouboru. Pro EMF formát je identifikátor 0x464D4520
Version	4	Použitá verze EMF formátu. V rámci této práce se jedná o 0x00010000.
Bytes	4	Délka metasouboru v bytech.
Records	4	Počet záznamů v metasouboru včetně hlavičky a ukončení.
Handles	2	Počet grafických objektů, které jsou použité při rekonstrukci souboru.
Reserved	2	Rezervováno do budoucna. Hodnota musí být nastavená na 0x0000.
nDescription	4	Počet znaků v řetězci popisu.
offDescription	4	Počet B od začátku souboru do umístění řetězce s popisem souboru.
nPalEntries	4	Počet záznamů v bloku s uživatelsky definovanou paletou barev (2.3.2).
Device	8	Objekt typu SizeL (2.3.2), určující rozměry vstupního zařízení v pixelech.
Millimeters	8	Objekt typu SizeL (2.3.2), určující rozměry vstupního zařízení v milimetrech.

2.1.3 EmfMetafileHeaderExtension1

První vydané rozšíření ke standardní hlavičce přináší podporu příkazů OpenGL a popis formátu pixelů. Obsahuje [1]:

- **cbPixelFormat** – 4 B. Velikost PixelFormatDescriptor objektu. Pokud je velikost 0, PixelFormatDescriptor nebyl použit.

- **offPixelFormat** – 4 B. Posunutí od začátku souboru k začátku objektu PixelFormatDescriptor.
- **bOpenGL** – přepínač o délce 4 B. V případě, že je nastaven na 0x00000001, signalizuje o přítomnosti příkazů OpenGL v metasouboru. Pokud je hodnota 0x00000000, rozhraní OpenGL nebylo použito.

2.1.4 EmfMetafileHeaderExtension2

Druhé vydané rozšíření ke standardní hlavičce umožňuje nastavit velikost zařízení v mikrometrech. Specifikuje horizontální a vertikální rozměry zařízení [1]:

- **MicrometersX (4 B)** – horizontální velikost výstupního zařízení.
- **MicrometersY (4 B)** – vertikální velikost výstupního zařízení.

2.2 Ukončení

Záznam EMR_EOF ukončuje metasoubor. Jedná se poslední přečtený záznam. Veškeré bloky po ukončení budou ignorovány. EMR_EOF se skládá z [1]:

- **Type (4 B)** – 0x0000000E.
- **Size (4 B)** – délka bloku v bytech. 16 B v případě, že nebyla specifikována uživatelská paleta, jinak je délka proměnlivá.
- **nPalEntries (4 B)** – počet barev v uživatelské paletě.
- **offPalEntries (4 B)** – posunutí od začátku záznamu do začátku specifikace palety. Pokud je použita systémová paleta, je tato hodnota 0x00000010.
- **PaletteBuffer (proměnlivá délka)** – pole objektů typu LogPaletteEntry 2.3.2.
- **SizeLast (4 B)** – poslední platná položka v metasouboru. Hodnota musí být stejná, jako hodnota položky Size.

2.3 Použité výčty a objekty

Tato podkapitola popisuje použité objekty a výčty, jelikož jsou podstatné pro pochopení jednotlivých položek záznamu.

2.3.1 Výčty

Výčty v metaformátu EMF slouží k uchování předem definovaných hodnot pro položky některých záznamů.

BackgroundMode

Hodnoty výčtu BackgroundMode určují chování pozadí při renderování textu, šrafování a kreslení nespojitých čár. Tabulka 2.2 znázorňuje možné hodnoty a jejich význam [1].

Tab. 2.2: Význam jednotlivých položek výčtu BackgroundMode

Název	**Hodnota**	**Význam**
TRANSPARENT	0x0001	Pozadí zůstává neměnné.
OPAQUE	0x0002	Pozadí je vyplněno aktuální barvou předtím, než se renderují texty a grafické objekty s použitím šrafovaných štětců nebo pera.

MapMode

Nastavená hodnota z výčtu MapMode definuje způsob mapování logických jednotek zařízení na fyzické jednotky. Nutným předpokladem je stejná pozice počátku. Tabulka 2.3 uvádí významy nastavení [1].

Tab. 2.3: Význam jednotlivých položek výčtu MapMode

Název	**Hodnota**	**Význam**
MM_TEXT	0x01	1:1 mapování. Počátek je nejlevější horní bod.
MM_LOMETRIC	0x02	1:0.1mm mapování. Počátek je nejlevější dolní bod.
MM_HIMETRIC	0x03	1:0.01mm mapování. Počátek je nejlevější dolní bod.
MM_LOENGLISH	0x04	1:0.01palec mapování. Počátek je nejlevější dolní bod.
MM_HIENGLISH	0x05	1:0.001palec mapování. Počátek je nejlevější dolní bod.
MM_TWIPS	0x06	1:twip mapování. Twip je 1/1440 palce. Počátek je nejlevější dolní bod.
MM_ISOTROPIC	0x07	1:n mapování. Faktor n a počáteční bod je nutné specifikovat.
MM_ANISOTROPIC	0x08	1:n mapování. Faktor n a počáteční bod je nutné specifikovat pro každou osu zvlášť.

BinaryRasterOperation

Tento výčet slouží k nastavení binární operace, která se provede při překrytí stejných pixelů různými objekty. Následující tabulka vysvětluje povolené operace [1]:

Tab. 2.4: Význam jednotlivých položek výčtu BinaryRasterOperation

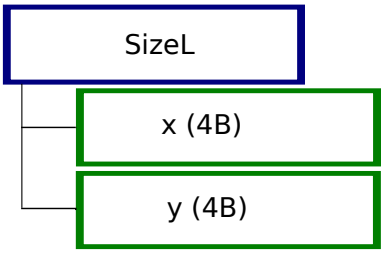
Název	**Hodnota**	**Operace**
R2_BLACK	0x0001	Nastavení na 0.
R2_NOTMERGEPEN	0x0002	Negace původního pixelu a binární OR s novým pixelem.
R2_MASKNOTPEN	0x0003	Původní pixel a binární AND s negací nového pixelu.
R2_NOTCOPYPEN	0x0004	Negace nového pixelu.
R2_MASKPENNOT	0x0005	Negace původního pixelu a binární AND s novým pixelem.
R2_NOT	0x0006	Negace původního pixelu.
R2_XORPEN	0x0007	Binární XOR hodnot.
R2_NOTMASKPEN	0x0008	Binární AND negace původního pixelu a nového pixelu.
R2_MASKPEN	0x0009	Binární AND hodnot.
R2_NOTXORPEN	0x000A	Binární XOR negace původního pixelu a nového pixelu.
R2_NOTXORPEN	0x000B	Původní pixel.
R2_MERGENOTPEN	0x000C	Negace nového pixelu a binární OR s původním pixelem.
R2_COPYPEN	0x000D	Nový pixel.
R2_MERGEPENNOT	0x000E	Původní pixel a binární OR s negací nového pixelu.
R2_MERGEPEN	0x000F	Nový pixel a binární OR s původním pixelem.
R2_WHITE	0x0010	Nastavení na 1.

2.3.2 Objekty

Metaformát EMF nabízí určitou úroveň abstrakce za použití objektů. Objekty mohou být položkami záznamu a většinou jsou zděděny z metaformátu WMF.

SizeL

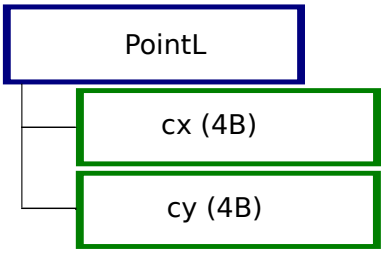
V objektu SizeL jsou uloženy šířka a výška [1].



Obr. 2.4: Struktura objektu SizeL. První položka x má délku 4 B a uchovává šířku. Druhá položka – výšku.

PointL

Tento objekt představuje bod zobrazovacího zařízení [1].



Obr. 2.5: Struktura objektu PointL. Souřadnice osy x je uložena v položce cx a souřadnice osy y v cy.

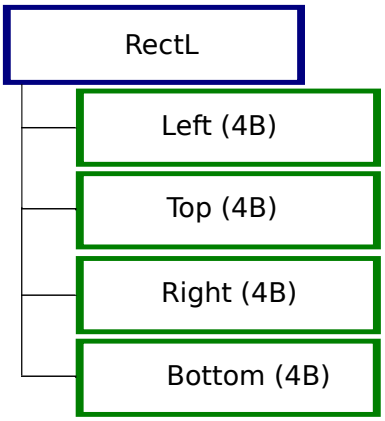
LogPaletteEntry

Znázorňuje barvu. Skládá se ze 4 položek o délce 4 B [1]:

- **Reserved** – připraveno pro budoucí využití.
- **Blue** – intenzita modré barevné složky.
- **Green** – intenzita zelené barevné složky.
- **Red** – intenzita červené barevné složky.

RectL

Blok RectL je abstrakcí obdélníku. Používá se například ke specifikaci ohraničovacího obdélníku [1].



Obr. 2.6: Položky bloku RectL. Položky Left a Top tvoří levý horní roh. Položky Right a Bottom tvoří pravý spodní roh.

2.4 Použité záznamy

Vzhledem k tomu, že práce je založena na zpracování metaformátu, je ke správnému pochopení důležité vysvětlit použité záznamy a důvody jejich použití.

2.4.1 Stavové záznamy

Stavové záznamy mění výstupní kontext zařízení. Pomocí nich lze nastavit způsob vykreslení jednotlivých pixelů přes sebe, barvu a zarovnání textu, velikost a způsob přenesení grafických objektů z nahrávacího zařízení na výstupní. Umožňují také uložit celý kontext v určitém okamžiku a následně se k němu vrátit.

EMR_SETMAPMODE

Tento záznam určuje způsob mapování jednotek rozměrů stránky na jednotky výstupního zařízení. Kromě typu (0x00000011) a velikosti (0x00000010) obsahuje položku MapMode, jejíž hodnota je vybrána z výčtového typu MapMode (2.3.1)[1].

EMR_SETVIEWPORTEXTEX a EMR_SETWINDOWEXTEX

Jedná se o záznamy o velikosti 24 B, obsahující položku typu SizeL (2.3.2) pro nastavení poměru zvětšení pro horizontální a vertikální osy. EMR_SETVIEWPORTEXTEX nastavuje zvětšení ve fyzických jednotkách a záznam EMR_SETWINDOWEXTEX v logických[1].

EMR_SETVIEWPORTORGEX a EMR_SETWINDOWORGEX

V případě použití mapování jednotek MM_ISOTROPIC, nebo MM_ANISOTROPIC je nutné specifikovat počáteční bod. Ke specifikaci ve fyzických jednotkách slouží záznam EMR_SETVIEWPORTORGEX. EMR_SETWINDOWORGEX specifikuje počátek v logických jednotkách. Oba tyto záznamy, kromě typu a velikosti bloku, obsahují počáteční bod typu PointL (2.3.2)[1].

EMR_SETROP2

Záznam EMR_SETROP2 nastavuje způsob určení výsledné barvy pixelů při překreslení již vyplněné oblasti. Jeho třetí položka ROP2Mode o délce 4 B musí být nastavená na jednu z hodnot výčtu BinaryRasterOperation (2.3.1), který obsahuje všechny kombinace logických funkcí a operátoru negace nad původním pixelem a barvou použitého pera [1].

EMR_SETBKMODE

Tento záznam určuje, jak se zachová pozadí při kreslení. Kromě typu a velikosti obsahuje hodnotu z výčtu 2.3.1.

EMR_SETTEXTALIGN

Před zápisem textu je nutné nastavit zarovnání. K tomu slouží položka TextAlignmentMode v záznamu EMR_SETTEXTALIGN. TextAlignmentMode je maska příznaku TextAlignmentMode Flags[1].

EMR_SETTEXTCOLOR

Kromě písma a zarovnání, se před zápisem textových dat nastavuje jeho barva. Nastavení probíhá pomocí záznamu EMR_SETTEXTCOLOR. Jeho položky, kromě typu a velikosti, tvoří objekt ColorRef, který má totožnou strukturu s objektem LogPaletteEntry (2.3.2).

EMR_SAVEDC a EMR_RESTOREDC

K uložení kontextu zařízení na zásobník stavu slouží záznam EMR_SAVEDC. Blok EMR_RESTOREDC obsahuje jeden parametr – záporný index stavu na zásobníku (-1 je naposledy uložený stav) a provádí načtení a obnovení kontextu zařízení [1].

[1] https://msdn.microsoft.com/en-us/library/cc669453.aspx

2.4.2 Grafické záznamy

Grafické záznamy se mapují na funkce GDI k samotnému vykreslení grafických objektů. K vykreslení ohraničení se používá aktuálně nastavené pero. Výplň objektů udává aktuální štětec.

EMR_POLYGON

Záznam typu EMR_POLYGON definuje polygonální objekt. Polygon je tvořen minimálně dvěma body spojenými přímou čarou. K uzavření polygonu dochází automaticky, propojením prvního a posledního bodu. K vykreslení ohraničení se použije aktuálně nastavené pero a k vyplnění aktuálně nastavený štětec. Struktura záznamu se skládá z [1]:

- **Type (4 B)** – 0x00000003.
- **Size (4 B)** – velikost bloku. Minimální velikost je 44 B.
- **Bounds (16 B)** – ohraničovací obdélník typu RectL (2.3.2) v logických jednotkách.
- **Count (4 B)** – počet bodů.
- **aPoints (Count * 8 B)** – pole bodů typu PointL (2.3.2), zadaných opět v logických jednotkách.

EMR_POLYPOLYGON

Blok EMR_POLYPOLYGON sdružuje několik polygonů v rámci jednoho objektu. Záznam se skládá z [1]:

- **Type (4 B)** – 0x00000008.
- **Size (4 B)** – velikost záznamu.
- **Bounds (16 B)** – ohraničovací obdélník typu RectL (2.3.2) v logických jednotkách.
- **NumberOfPolygons (4 B)** – počet jednotlivých polygonů.
- **Count (4 B)** – celkový počet bodů ve všech dílčích polygonech.
- **PolygonPointCount (4 B * NumberOfPolygons)** – pole počtů bodů pro každý polygon.
- **aPoints (Count * 8 B)** – pole bodů typu PointL (2.3.2). Každý bod je specifikován v logických jednotkách. Počáteční a koncové body dílčích polygonů musí být totožné.

EMR_POLYLINE

Záznam EMR_POLYLINE vytváří posloupnost čar propojením jednotlivých bodů. Položky záznamu jsou totožné s předchozím záznamem EMR_POLYGON. Jedinou odlišností je typ, který má hodnotu 0x00000004 a to, že se nepoužívá vyplnění štětcem a nedochází k automatickému uzavření [1].

EMR_RECTANGLE

Blok EMR_RECTANGLE slouží k vykreslení obdélníka. V případě, že nastavené pero je typu PS_NULL, rozměry obdélníka jsou zkráceny o 1 pixel. Je nutné podotknout, že vykreslení nastavuje aktuální polohu na zobrazovacím zařízení. Položky záznamu jsou následující [1]:

- **Type (4 B)** – 0x0000002B.
- **Size (4 B)** – velikost bloku (24 B).
- **Box (16 B)** – obdélník k vykreslení typu RectL (2.3.2) v logických jednotkách.

EMR_ROUNDRECT

Jedná se o obdélník se zaoblenými okraji. První tři položky jsou shodné se záznamem EMR_RECTANGLE. Poslední položkou je objekt typu SizeL (2.3.2) s rozměry elipsy, která slouží k zaoblení rohů.

EMR_ELLIPSE

EMR_ELLIPSE je, jak je patrné z názvu, abstrakcí elipsy. Je tvořen třemi položkami [1]:

- **Type (4 B)** – 0x0000002A.
- **Size (4 B)** – velikost bloku (24 B).
- **Box (16 B)** – ohraničovací obdélník (RectL 2.3.2), ve kterém je vepsaná elipsa.

EMR_ARC

K abstrakci oblouku metaformát EMF obsahuje záznam EMR_ARC. Ten se skládá z [1]:

- **Type (4 B)** – 0x0000002D.
- **Size (4 B)** – velikost bloku (40 B).
- **Box (16 B)** – ohraničovací obdélník (RectL 2.3.2).

- **Start (4 B)** – počáteční bod typu PointL (2.3.2) v logických jednotkách.
- **End (4 B)** – koncový bod typu PointL (2.3.2) v logických jednotkách.

EMR_EXTTEXTOUTW

Při vykreslení textu se používá aktuálně nastavené písmo, zarovnání, barva a samotný text v záznamu EMR_EXTTEXTOUTW. Jeho strukturu tvoří [1]:

- **Type (4 B)** – 0x00000054.
- **Size (4 B)** – velikost bloku.
- **Bounds (16 B)** – ohraničení (RectL 2.3.2).
- **iGraphicsMode (4 B)** – mod k určení velikosti. V rámci práce nebyl použit.
- **exScale (4 B)** – zvětšení šířky (nepoužito).
- **eyScale (4 B)** – zvětšení výšky (nepoužito).
- **wEmrText**[2] – objekt obsahující samotný textový řetězec.

2.4.3 Záznamy pro vytvoření objektů

Záznamy v této sekci vytváří nástroje k renderování jiných objektů. Tabulka objektů metasouboru obsahuje indexy jednotlivých nástrojů, což zaručuje možnost opětovného použití.

EMR_CREATEPEN

Záznam EMR_CREATE vytváří nové pero. Kromě typu a velikosti může uživatel nastavit tyto položky [1]:

- **ihPen (4 B)** – unikátní index do tabulky objektů metasouboru.
- **PenStyle (4 B)** – jeden z předem definovaných stylů PenStyle[3].
- **Width (8 B)** – objekt PointL (2.3.2), ve kterém x-ová souřadnice určuje šířku pera.
- **ColorRef (4 B)** – barva (2.3.2).

EMR_CREATEBRUSHINDIRECT

S pomocí záznamu EMR_CREATEBRUSHINDIRECT lze vytvořit nový štětec. Parametry bloku jsou následující [1]:

- **ihPen (4 B)** – unikátní index do tabulky objektů metasouboru.

[2]https://msdn.microsoft.com/en-us/library/cc230576.aspx
[3]https://msdn.microsoft.com/en-us/library/cc231188.aspx

- **BrushStyle (4 B)** – jeden z předem definovaných stylů. V rámci této práce se použily štětce s nastavením stylu BS_NULL (průhledný štetec) a BS_SOLID (pravidelná výplň). Styl BS_HATCH nebyl použit kvůli absenci nutnosti práce se šrafováním.
- **Color (4 B)** – barva (2.3.2).
- **BrushHatch (4 B)** – šrafování.

EMR_EXTCREATEFONTINDIRECTW

K vytvoření písma se používá záznam EMR_EXTCREATEFONTINDIRECTW. Po nastavení indexu ihPen následuje nastavení objektu LogFontExDv[4]. Tento objekt obsahuje název písma, nastavení váhy a výšky, rozestupů mezi znaky a mnoho dalších nastavení [1].

2.4.4 Záznamy pro manipulaci s objekty

Manipulační záznamy pracují s objekty, jejichž vytváření bylo popsáno v předchozí části této kapitoly. Cílem je umožnit nastavení těchto objektů do kontextu výstupního zařízení [1].

EMR_SELECTOBJECT

EMR_SELECTOBJECT nastavuje aktuální objekt určitého typu v kontextu. Kromě typu a velikosti, obsahuje položku ihObject, která je indexem do interní tabulky objektů kontextu. Například před vykreslením grafických objektů se musí pomocí tohoto bloku nastavit pero a štětec [1].

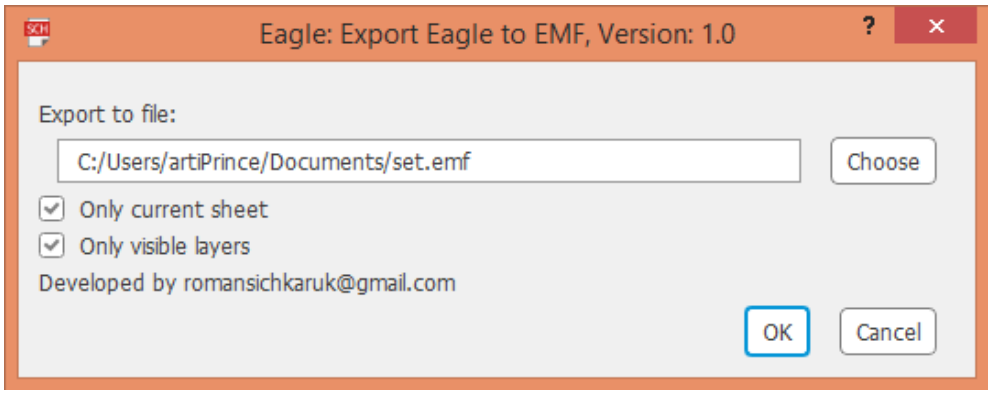
EMR_DELETEOBJECT

K odstranění objektů se používá záznam EMR_DELETEOBJECT. K určení objektů se opět použije položka ihObject, která obsahuje index do tabulky objektů. Po odstranění objektu, který je nastaven, jako aktuální, je vždy nutno nastavit výchozí objekt [1].

[4]https://msdn.microsoft.com/en-us/library/cc230577.aspx

3 Návrh implementace

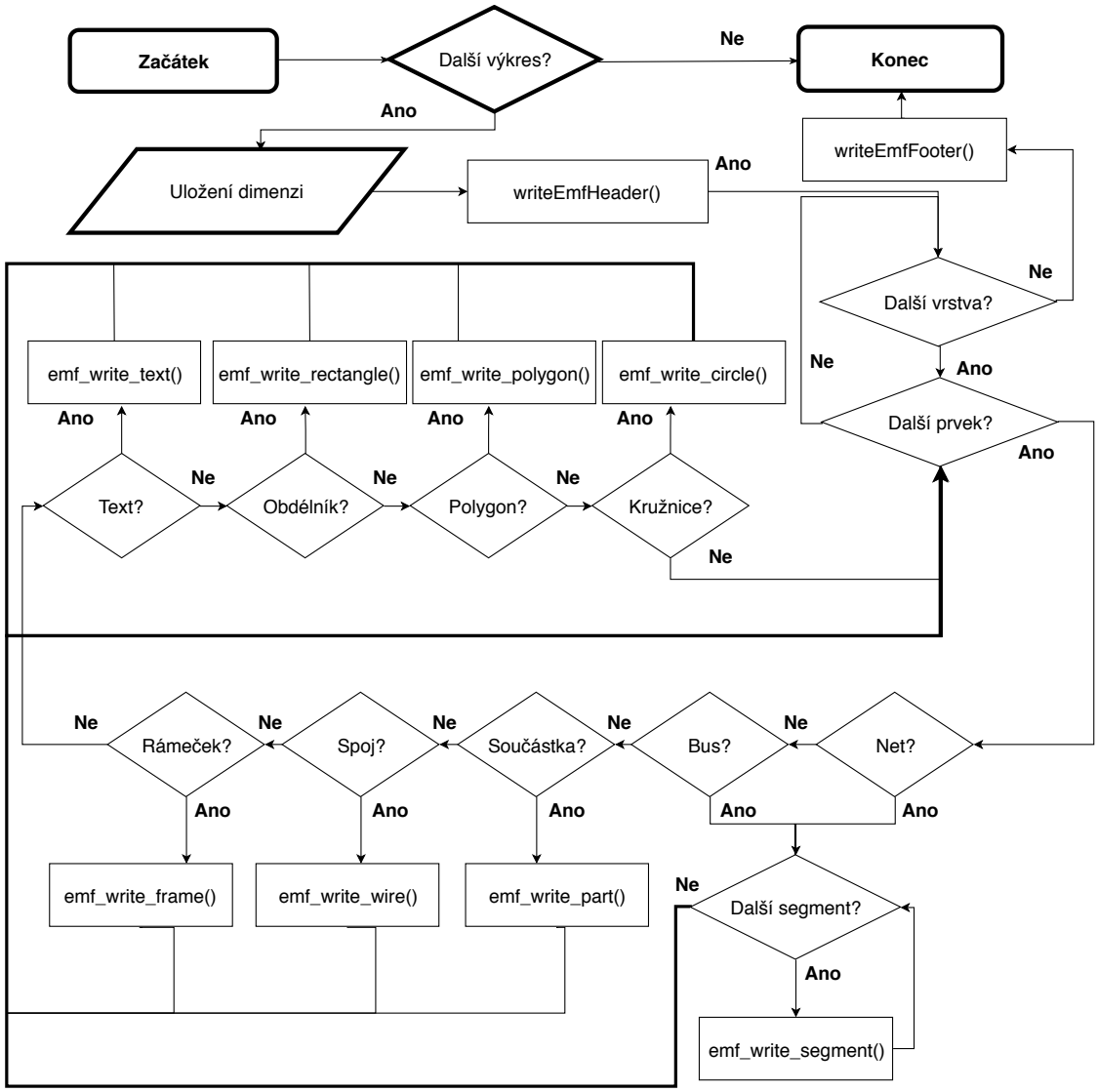
Cílem diplomové práce je implementovat program, umožňující exportovat obrázky ve formátu EMF (Enhanced Windows Metafile) s podporou všech funkcí programu Autodesk Eagle verze 9. V rámci práce aplikace musí podporovat desky a schémata. Návrh aplikace vychází z programu pro export obrázků ve formátu SVG[1], doporučeného vedoucím práce v zadání. Po spuštění skriptu se objeví uživatelský dialog jako na obrázku 3.1.



Obr. 3.1: Uživatelský dialog programu.

Nejdříve je uživatel vyzván k určení cílového souboru. Tlačítko Choose (česky vybrat) otevře průzkumníka souborů pro jednodušší nalezení správné cesty. V případě, že uživatel vyžaduje export schématu, zobrazí se přepínač Only current sheet (česky pouze aktuální výkres). Přepínač umožní učinit rozhodnutí pro vytvoření EMF metasouboru pouze pro aktuální výkres nebo všechny výkresy v sadě. Dalším přepínačem je Only visible layers (česky pouze viditelné vrstvy), který, jak napovídá název, zaručuje průchod všemi vrstvami nebo pouze viditelnými. Samotná logika programu je poněkud odlišná pro export ze schémat a desek. Vývojový diagram 3.2 znázorňuje chování v případě schémat. Diagram pro vykreslení desek vypadá velice obdobně. Místo součástek jsou navíc zpracovány elementy, obrysy desky (angl. dimensions), díry a signály.

[1]Scalable Vector Graphics - format škálovatelné vektorové grafiky



Obr. 3.2: Vývojový diagram programu pro export schémat.

Z diagramu je patrné, že v rámci implementace byly použity dvě notace pojmenování funkcí. První z nich je hadí notace (angl. snake case[1]), která označuje funkce s vnitřní logikou a voláním dalších funkcí s použitou velbloudí notací (angl. camel case[2]) pro přímý zápis EMF objektů do souboru. Jednotlivé funkce jsou vysvětleny v tabulkách 3.1 a 3.2. Společnou částí logiky všech funkcí je ověření, zda objekt má být vykreslen na aktuální vrstvě.

[1] https://en.wikipedia.org/wiki/Snake_case

[2] https://techterms.com/definition/camelcase

Tab. 3.1: Význam jednotlivých funkcí vývojového diagramu 3.2

Funkce	**Význam**
`writeEmfHeader`	Funkce pro zápis hlavičky souboru. Podrobněji vysvětleno v podkapitole 4.2.
`writeEmfFooter`	Funkce pro zápis ukončení souboru.
`emf_write_text`	Zapisuje text do souboru. Podrobněji vysvětleno v podkapitole 4.3.4.
`emf_write_rectangle`	Obstarává zápis obdélníku do souboru. Podrobněji vysvětleno v podkapitole 4.3.4.
`emf_write_circle`	Zajišťuje zápis kruhu do souboru. V případě potřeby nastavuje barvu štětce na průhlednou pro vykreslení kružnice.
`emf_write_polygon`	Uloží do souboru polygonální objekt. Podrobný popis lze nalézt v sekci 4.3.2
`emf_write_frame`	Zpracuje rameček, který je tvořen množinou textů a spojů.
`emf_write_symbol`	Tělo součástky (symbol) může obsahovat texty, kruhy, připojovací body (piny), polygony, spoje, rámečky a obdélníky. Rámečky jsou zpracované funkcí `emf_write_frame()`, která přistupuje k jednotlivým spojům a textům rámečků a vykresluje je pomocí funkcí popsaných dříve. Přípojné plochy (piny) jsou zpracované pomocí funkce `emf_write_pin()`, která přistupuje ke stejným objektům jako funkce `emf_write_frame()` a navíc ke kruhům typu `UL_CIRCLE`.
`emf_write_part`	Obsahuje základní logiku pro součástky složené z textů a symbolů.
`emf_write_segment`	Pro zápis segmentů funkce `emf_write_segment()` provádí průchod a zpracování jednotlivých spojů, textů a kruhových uzlů (angl. junctions). Uzel se vykreslí funkcí `emfWriteDot()`, vytvářející kruh o nulové šířce pera a o průměru daného atributem uzlu.

Tab. 3.2: Význam funkcí volaných pouze pro desky plošných spojů

Funkce	**Význam**
`emf_write_hole`	Funkce pro zápis děr. Nejprvé se aktuální vrstva nastaví na zápornou hodnotu pro výběr barvy pozadí (podrobněji 4.1.3). Díra je vykreslená funkcí pro kruh s nastavením nulové šířky pera a průměru získaného z atributu díry.
`emf_write_element`	Určeno k vykreslení elementů tvořených z mnoha dalších tvarů. Elementy obsahují vlastní posunutí a rotaci. Ovšem ty jsou již aplikované na jednotlivé dílčí objekty a tak iteraci těmito objekty v ULP získáme vždy jejich absolutní transformace. Logika funkce je vysvětlena v sekci 4.3.6.
`emf_write_dimension`	Zapisuje informace o obrysech desky (angl. dimensions). Ty jsou většinou tvořeny texty a spoji, jelikož jejich hlavním účelem je například zobrazení vzdálenosti mezi objekty, nebo ohraničení desek.
`emf_write_signal`	Funkce obsahuje logiku zpracování signálů. Popis lze nalézt v sekci 4.3.5.

4 Implementace

Implementace vychází ze dvou uživatelských programů - `sch2svg-1_4_1.ulp`[6] a `sch2wmf.ulp`[7]. První problém, který bylo nutno vyřešit při implementaci, je absence ucelené informace o samotné struktuře metasouboru. V dokumentaci jsou popsány struktury jednotlivých záznamů, ale chybí sekvence záznamu pro vygenerování doopravdy multiplatformního řešení. Pro vyřešení tohoto úkolu byla použita metoda zpětné analýzy. Byly dekodovány výstupy z různých aplikací ve formátu EMF. Jako první referenční program byl zvolen OpenOffice Draw[1] kvůli jeho dostupnosti. První nevýhodou ovšem bylo, že čáry s tloušťkou více než 1 pixel, jsou vykresleny, jako vyplněné polygony. Následně se ukázalo, že ve většině případů ostatní aplikace vnímaly výstup jako poškozený. Jako další vhodný vzor se jevila aplikace Inkscape[2]. Problém ovšem nastal při pokusu o generování textů. Dokonce samotná aplikace má potíže s nastavením polohy a rozestupu mezi znaky při čtení jí vygenerovaného metasouboru. Objevily se také chyby v určení ohraničovacího rámečku. Právě z těchto důvodů byl, z doporučení vedoucího práce, za referenci zvolen Adobe Illustrator[3].

Vedoucím práce byly poskytnuty referenční metasoubory, pomocí nichž bylo docíleno řešení. Způsob implementace diplomové práce je díky tomu vysvětlen v této podkapitole. Důraz je kladen na objasnění základní programové logiky a vstupních, výstupních funkcí.

4.1 Pomocné funkce

Tato sekce popisuje funkce, které slouží k pomocným úkonům. Mezi tyto úkony patří transformace bodů, nastavení barev pro štětce a pera v rámci různých vrstev a objektů. Také jsou zdůrazněny funkce pro zápis dat do metasouboru EMF.

4.1.1 Funkce pro zápis

Vzhledem k povaze struktury záznamů EMF, je nutné zapisovat data o délce 1, 2 a 4 B do souborů a také do paměti. Zápis do paměti probíhá z důvodu určení velikosti bloku před samotným zápisem do souboru. Níže jsou uvedeny ukázky zápisových funkcí pro uložení hodnot o délce 2 a 4 B:

[1] https://www.openoffice.cz/draw
[2] https://inkscape.org/cs/
[3] https://www.adobe.com/cz/products/illustrator.html

```
...
    void PrintDWordToFileContent(int dw) {
      fileContent[fileContentSize++]=dw&0xFF;
      fileContent[fileContentSize++]=(dw>>8)&0xFF;
      fileContent[fileContentSize++]=(dw>>16)&0xFF;
      fileContent[fileContentSize++]=((dw>>16)>>8)&0xFF;
    }
...
```

Výpis 4.1: Funkce pro zápis 4 B do řetězce fileContent. Položka fileContentSize slouží jako index v řetězci a je inkrementována kvůli nutnosti zpětné identifikace velikosti bloku.

```
...
    void PrintWord(int w) {
        printf("%c%c",w&0xFF,(w>>8)&0xFF);
    }
...
```

Výpis 4.2: Funkce pro zápis 2 B dat do výstupního souboru.

4.1.2 Transformace a převody jednotek

Co se týče transformací, musí se aplikovat zrcadlení podle horizontální osy a také posunutí do počátku. Vzhledem k možnosti schémat používat i záporný souřadnicový prostor, bylo nutné aplikovat následující transformaci na všechny body:

```
...
    // Funkce pro transformaci x-ové složky
    int pointTransformX(int x){
      return x+transformX;
    }

    // Funkce pro transformaci y-ové složky
    int pointTransformY(int y){
      return dimensionY - (y+transformY);
    }
...
```

Výpis 4.3: Funkce pro transformaci horizontálních a vertikálních složek bodu.

Kromě posunutí bodů je zapotřebí jej orotovat kolem určitého bodu. Například při orotování obdélníku je nutno orotovat všechny body kolem středu obdélníku a uložit orotované body do EMF metasouboru. Metaformát EMF totiž neuvádí možnost specifikace rotace u použitých záznamů v této diplomové práci. Rotaci zprostředkovávají funkce `rotatePointX()` a `rotatePointY()`. V ukázce kódu 4.4 je definice funkce pro zisk horizontální složky bodu při rotaci. Vertikální složku lze získat prohozením funkcí *sin* a *cos* a přičtením složky `cy` místo `cx`.

```
...
   int rotatePointX(int x, int y, int cx, int cy,
                    real angle)
   {
     return ((x-cx)*cos(angle*PI/180) -
             (y-cy)*sin(angle*PI/180))+cx;
   }
...
```

Výpis 4.4: Funkce pro rotaci bodu a zisk horizontální složky.

Jak již bylo zmíněno, logické a fyzické jednotky jsou v poměru 1:10000. Poměr je uložen v globální proměnné `TOLOGIC` a lze ji navýšit v případě potřeby. V ukázce 4.5 je uvedena funkce `u2logic()`.

```
...
   int u2logic(int x){
     int logicUnits=u2mm(x)*TOLOGIC;
     return logicUnits;
   }
...
```

Výpis 4.5: Funkce pro mapování fyzických jednotek na logické jednotky.

Funkce `u2logic()` nejprve převede interní jednotky Autodesk Eagle na milimetry a následně je vynásobí hodnotou `TOLOGIC`. Tím je zaručena přesnost na desetiny mikrometru.

4.1.3 Nastavení barev

Barva v Autodesk Eagle vždy přísluší jednotlivé vrstvě. Objekt (např. spoj, obdélník) obsahuje číslo vrstvy. S pomocí čísla vrstvy, funkce `setColor()` aplikuje průchod všemi vrstvami a v případě, že je vrstva definována v rámci desky, či schématu, získá

její barvu. Průhlednost je v tomto případě ignorována, jelikož ji metaformát EMF nepodporuje. Na ukázce kódu 4.6 je úsek pro zisk odpovídající barvy u desek plošných spojů. V případě, že barva nebyla nalezena, použije se bílá barva pro schéma a černá pro desky. Tyto barvy jsou výchozí pro pozadí.

```
...
   board(B){
      B.layers(Lay){
         if(layer == Lay.number){
            int lcolor=palette(Lay.color)&0x00ffffff;
            printColor(lcolor);
            return;
         }
      }
   }
   printColorToFile(0,0,0);
...
```

Výpis 4.6: Kód pro zisk barvy pro desku plošných spojů.

Způsob uložení barev v strukturách ULP se liší od uložení v metaformátu EMF. Druhý nejvyšší oktet v 32 bitovém integeru popisujícím ULP barvu, obsahuje hodnotu modré složky barvy RGBA barevného modelu[4]. V metaformátu EMF je modrá složka poslední slabikou záznamu `EMR_COLOR`. Mapování barev je k nahlédnutí v ukázce 4.7.

```
...
   void printColor(int color){
      int red = color >> 16;
      int green = (color >> 8) & 0xff;
      int blue = color & 0xff;
      printColorToFile(red,green,blue);
   }
...
```

Výpis 4.7: Funkce pro mapování barev Autodesk Eagle na barvy EMF.

Výslednou barvu zapíše funkce `printColorToFile()`. Její definice je na ukázce zdrojového kódu 4.8.

[4]R - červená složka, G - zelená složka, B - modrá složka, A - průhlednost

```
...
   void printColorToFile(int r, int g, int b){
      PrintDWordToFileContent((r) + (g << 8) +
                              (b  << 16));
   }
...
```

Výpis 4.8: Zápis barvy do metasouboru EMF.

4.2 Implementace hlavičky a nastavení kontextu

Nastavení kontextu přehrávání metasouboru EMF se skládá z několika kroků. Prvním je uložení rozměrů schématu nebo desky do globálních proměnných `dimensionX)` a `dimensionY)`. Kromě rozměrů se uloží také posunutí od počátku souřadnicového systému. Toto posunutí je použito k transformaci jednotlivých klíčových bodů při zpracování ULP struktur.

Dalším krokem je volání funkce `emfWriteHeader()`. Tato funkce je společná pro schémata a desky. Zápis hlavičky se skládá ze dvou částí. Nejprve je do souboru uložen typ záznamu (0x00000001). Následně se funkce přepne na zpracování obsahu schématu a pokračuje na začátku volání funkce ukončení - `emfWriteFooter()`. Nyní se uloží velikost souboru, počet záznamů a počet ukazatelů na grafické objekty v paměti. Grafickými objekty jsou štětce, pera, písma, atd.

V rámci hlavičky se do paměti zapíše ohraničení zařízení v logických jednotkách a milimetrech. Také se uloží rozměry v pixelech, milimetrech a mikrometrech. Následuje nastavení mapovacího módu na hodnotu `MM_ANISOTROPIC`. Toto nastavení umožňuje zvýšit přesnost vykreslení, avšak vyžaduje nastavení koeficientů zvětšení pro jednotlivé osy ve fyzických a logických jednotkách. Způsob nastavení bude vysvětlen níže. Hlavička pokračuje určením stylu práce s pozadím, při renderování textů a šrafování, či kreslení přerušovaných čar - pozadí zůstává nevyplněným. Také se nastaví způsob řešení překrytí pixelů. V rámci diplomové práce je tato hodnota nastavena na přepsání původního pixelu. V tomto okamžiku se musí uložit kontext výstupního zařízení, aby se mohl před ukončením řádně obnovit. Poslední částí zpracování hlavičky je vytvoření štětce s barvou pozadí a vyplnění stránky tímto štětcem pomocí vyplněného polygonu.

Jak již bylo zmíněno, kvůli zvýšení přesnosti je nutné nastavit koeficienty zvětšení pro fyzické a logické jednotky. Hodnoty těchto zvětšení byly zjištěny metodou zpětné analýzy, při procházení EMF metasouborů generovaných programem libreOffice.

```
...
   PrintDWordToFileContent(0x0000000B);
   PrintDWordToFileContent(0x00000010); //velikost bloku
   PrintDWordToFileContent(100); // osa X
   PrintDWordToFileContent(100); // osa Y
   recordsCount++;
...
```

Výpis 4.9: Ukázka uložení struktury typu EMR_SETVIEWPORTEXTEX pro nastavení zvětšení v jednotkách zařízení.

Jak je patrné z ukázky kódu 4.10, zvětšení pro logické jednotky je vždy násobkem hodnoty 25.4, která odpovídá hodnotě 1000 mil v milimetrech. V nápovědě není ani zmínka o této hodnotě. V rámci práce bylo vyzkoušeno několik variant mapování logických a fyzických jednotek (například 1:1, 1:10000, atd), ovšem pouze tato implementace se jeví, jako funkční.

```
...
   PrintDWordToFileContent(0x00000009);
   PrintDWordToFileContent(0x00000010); //velikost bloku
   PrintDWordToFileContent(25.4 * TOLOGIC); // osa X
   PrintDWordToFileContent(25.4 * TOLOGIC); // osa Y
   recordsCount++;
...
```

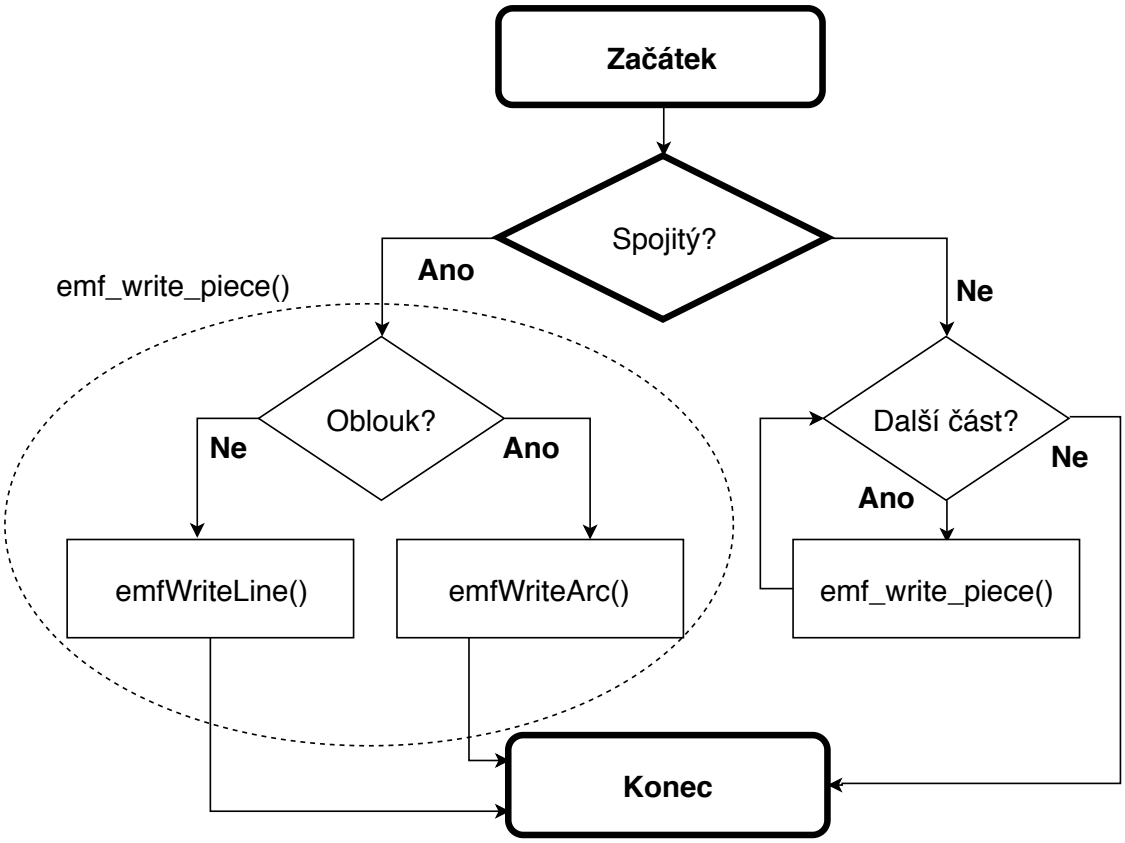
Výpis 4.10: Ukázka uložení struktury typu EMR_SETWINDOWEXTEX pro nastavení zvětšení v logických jednotkách.

4.3 Zpracování jednotlivých částí desek a schémat

Tato sekce popisuje logiku a implementaci jednotlivých funkcí pro zpracování struktur ULP Eagle. Kromě jednoduchého průchodu struktur, pojednává o řešení problémů jejich mapování na záznamy metasobouru EMF.

4.3.1 Vykreslení spojů

Plošný spoj je v ULP zastoupen strukturou `UL_Wire`, uchovávající informaci o počátečním a koncovém bodě, šířce spojů, vrstvě a stylu spojů. V případě, že spoj

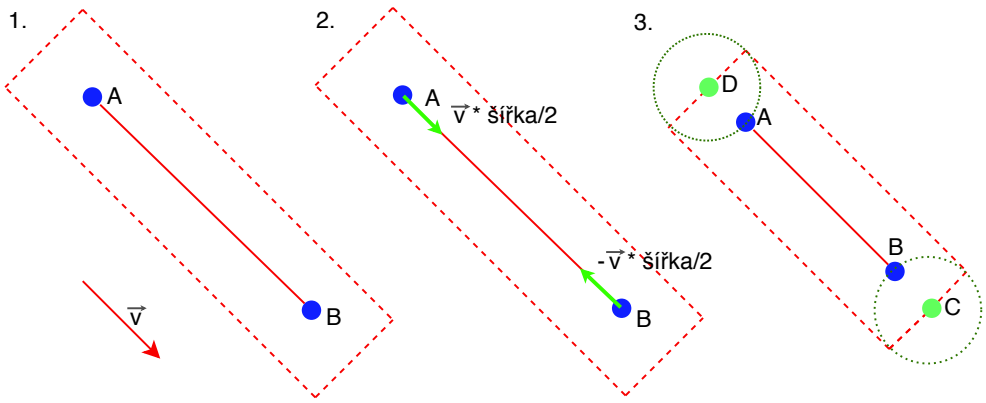


Obr. 4.1: Vývojový diagram programu pro export spojů.

není spojitý, lze k jednotlivým částem přistoupit pomocí iterátoru `pieces()`. Logiku zpracování spojů lze lépe znázornit pomocí vývojového diagramu 4.1.

Nejprve se určí, zda se spoj skládá z několika částí. V případě, že ano, je pro každou část volána funkce `emf_write_piece()`. Spoj, nebo jeho část, je určena obloukem nebo úsečkou. Funkce `emfWriteArc()` zapisuje oblouk typu `EMR_ARC` do metasouboru. Oblouk je v tomto případě definován ohraničovacím obdélníkem, počátečním a koncovým bodem. Pokud je spoj tvořen úsečkou, vykreslí jej funkce `emfWriteLine()`. Metaformát EMF nemá objekt pro úsečku se zaoblenými konci. Právě proto je nutné vykreslit kratší úsečku a přidat dva kruhy s průměrem o šířce spoje. Postup vytvoření zaobleného spoje je na obrázku 4.2.

Ke zkrácení úsečky je nejprve vypočten její jednotkový vektor $\vec{v}$. Následně se násobek poloviny šířky spoje přičte k počátečnímu bodu A a odečte od koncového bodu B. Algoritmus končí vykreslením dvou kruhů se středy v původních řídících bodech, nově označených C a D.



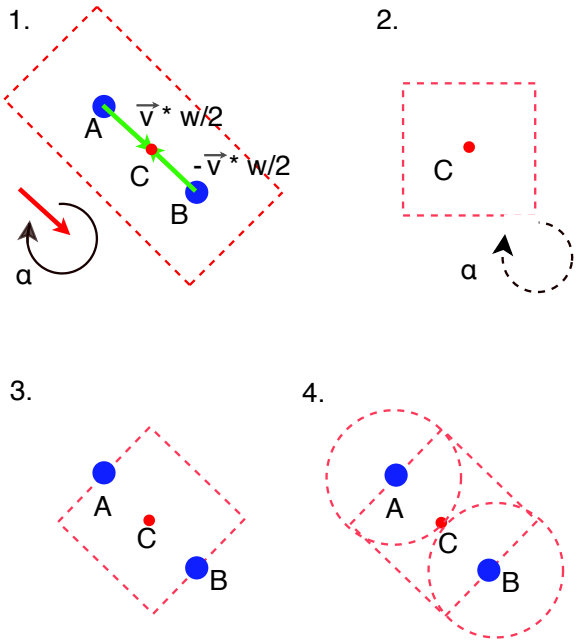
Obr. 4.2: Postup vytvoření zaoblené úsečky. V prvním kroce se určí jednotkový vektor $\vec{v}$. V druhém kroce se odečtením násobku vektoru $\vec{v}$ naleznou nové koncové body. Následuje vykreslení dvou kruhů se středy v původních koncových bodech.

Za předpokladu, že máme šířku spoje w a původní koncové body A, B, můžeme určit tři typické situace při zkrácení úsečky:

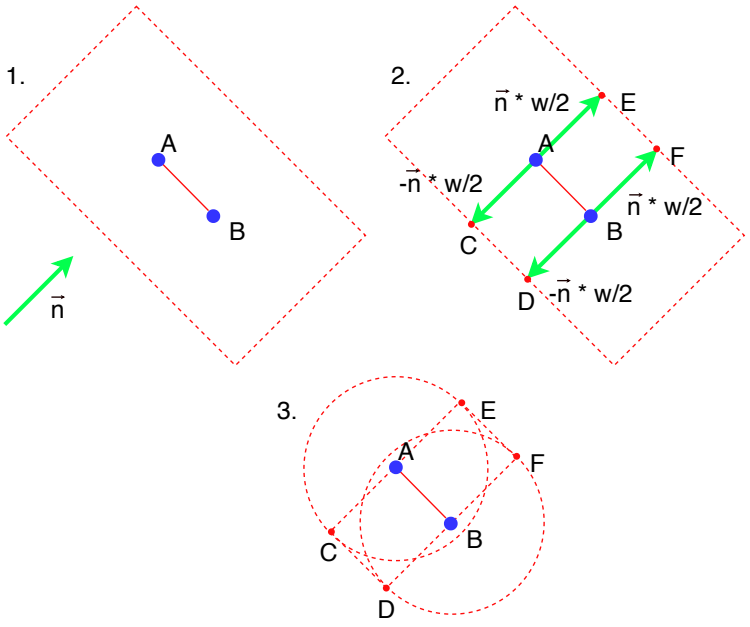
- $|AB| > w$ – situace přesně vystižená obrázkem 4.2. Délka spoje je větší jak jeho šířka.

- $|AB| == w$ – případ, kdy po zkrácení úsečky zůstanou dva totožné body. EMF metaformát nepodporuje vykreslení čáry s nulovou délkou. Algoritmus tedy pokračuje výpočtem úhlu úsečky. Ve vzniklém bodě se vykreslí obdélník pomocí funkce `emfWriteRect()`. Šířka pera se nastaví na nulu a jednotlivé body obdélníku se vypočítají posunutím bodu o polovinu původní šířky spoje ve čtyřech základních směrech horizontální a vertikální osy. Jako parametr funkce je předán úhel úsečky a obdélník je orotován o tento úhel kolem jeho středu. Nakonec se v původních řídících bodech úsečky vykreslí dva kruhy.

- $|AB| < w$ – délka úsečky je menší, než šířka spoje. V tomto případě po zkrácení úsečky vzniknou body, které leží ve vzdáleností kratší, než polovina šířky spoje od opačného koncového bodu. Výsledkem by tedy byla úsečka, která by po vykreslení s původní šířkou přesahovala původní koncové body. Právě z těchto důvodů je nutné určit normálový vektor $\vec{n}$. Pomocí přičtení a odečtení násobku poloviny šířky spoje a vektoru $\vec{n}$, se získají body obdélníku. V posledním kroce se v původních koncových bodech spoje vykreslí dva kruhy. Tento algoritmus je znázorněn na obrázku 4.4.



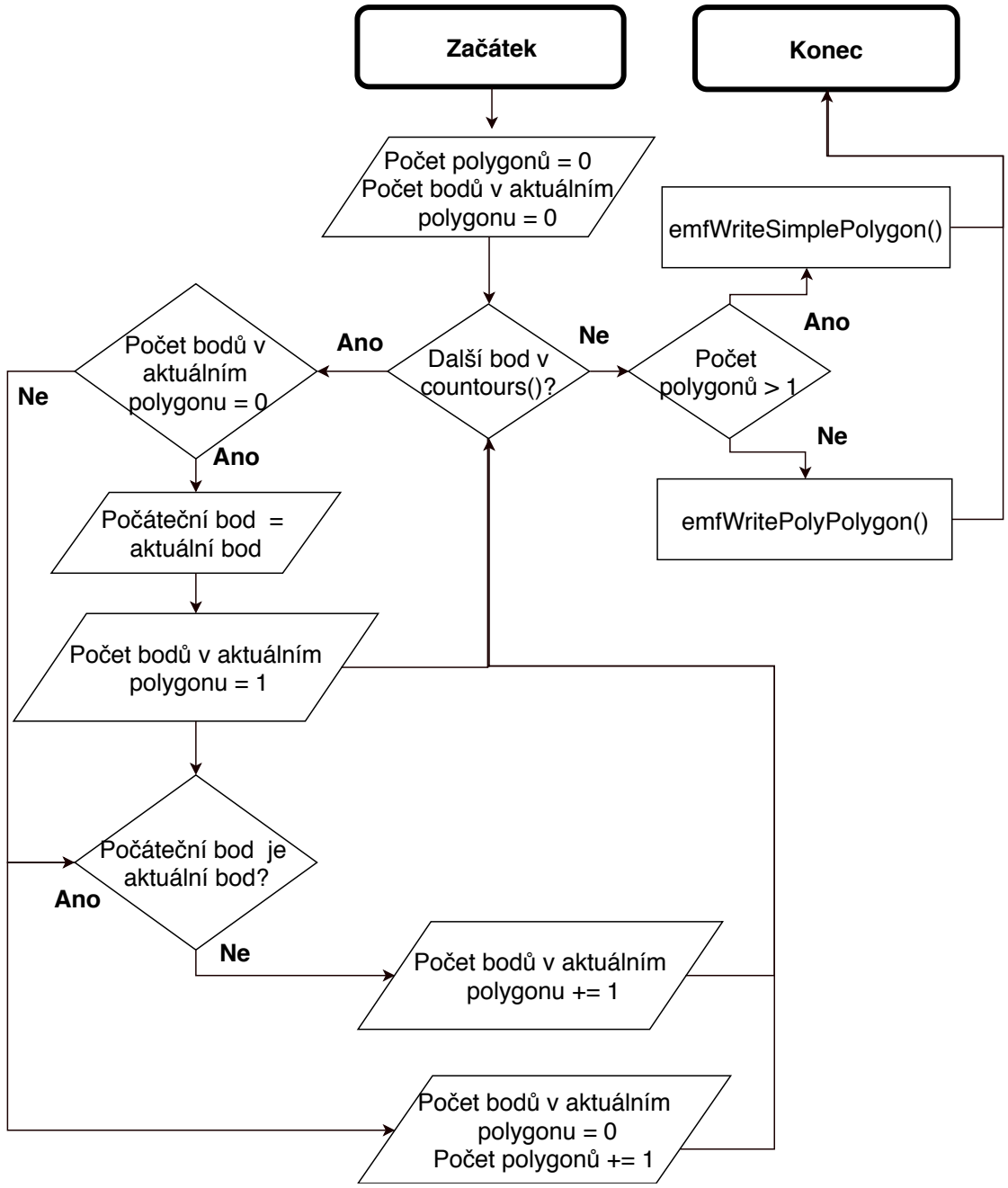
Obr. 4.3: Postup vytvoření zaoblené úsečky v případě, že délka úsečky je rovna jeho šířce.



Obr. 4.4: Postup vytvoření zaoblené úsečky v případě, že délka úsečky je kratší než šířka spoje.

4.3.2 Vykreslení polygonů

Logiku zpracování polygonu funkcí `emf_write_polygon()` lze vysvětlit pomocí vývojového diagramu .



Obr. 4.5: Vývojový diagram funkce emf_write_polygon().

V případě zpracování polygonů je možné přistoupit ke třem iteratorům struktury `UL_POLYGON` - `contours()`, `fillings()` a `wires()`. Iterator vždy vrací spoje `UL_WIRE`. V případě schématu, iterator `contours()` vrací stejné úsečky jako `wires()`,

a `fillings()` nevrací nic. V případě desek plošných spojů je situace jiná. Iterator `wires()` vrací všechny úsečky. U iterátoru `fillings()` je uložena informace o vyplnění polygonu horizontálními úsečkami. V rámci práce je důležitý průchod pomocí `contours()`. Obrysové linie, vrácené tímto iterátorem, vytvářejí polygony s vlastním průnikem, vnořené polygony a polygony s překryvy. Každý polygon začíná a končí stejným bodem. EMF metaformát abstrahuje polygon pomocí záznamu `EMR_POLYGON`. Tento záznam podporuje vlastní průnik, ale nepodporuje vnořené polygony. Právě proto existuje další záznam `EMR_POLYPOLYGON`, který sdružuje několik polygonů.

Funkce `emf_write_polygon()` začíná uložením prvního bodu získaného atributem `countours()`. Postupně iteruje body, než narazí na shodný s uloženým bodem. V případě, že existují další body, je polygon tvořen několika dílčími polygony, což znamená, že je nutné zavolat funkci `emfWritePolyPolygon()`. Tato funkce uloží jednotlivý počet bodů v každém polygonu a při zápisu do struktury `EMR_POLYPOLYGON` je uloží všechny. Narozdíl od funkce `emfWriteSimplePolygon()`, kde se nesmí ukládat poslední bod, je nutné uložit i koncové body jednotlivých polygonů.

4.3.3 Vykreslení obdélníků

O vykreslení obdélníků se stará funkce `emfWriteRect()`. Tato funkce zapisuje záznam typu `EMR_RECTANGLE`. Při implementaci bylo zjištěno, že záznam podporuje pouze obdélníky orotované o ortogonální úhel. Funkce `emfWriteRect()` tedy začíná zjištěním, jestli úhel je celočíselným násobkem 90°. Pokud ano, pokračuje funkce zápisem záznamu `EMR_RECTANGLE`. V opačném případě je obdélník vykreslen jako polygon voláním funkce `emfWritePolygonRect()`. Podobně se chová i funkce `emfWriteRoundedRect`, která vykresluje obdélník se zaoblenými okraji. Důležité je správně vypočítat elipsu, která tvoří tyto okraje. V ULP je uloženo procento zaoblení. Hlavní a vedlejší osy elipsy tvoří úsečky o délce rozměrů obdélníku vynásobených procentem zaoblení.

4.3.4 Vykreslení textů

V rámci konzultací s vedoucím práce bylo zjištěno, že vykreslení textu může být problém. Příčinou je absence dostatečných informací o ohraničení textů, písmu a způsobu zalomení řádků. Bylo tedy rozhodnuto jednotlivé texty zpracovávat ve vektorovém stylu. Každý objekt `UL_TEXT` obsahuje iterator `wires()`, skrze který lze přistupovat k jednotlivým částem textů (úsečkám, obloukům). Pomocí funkce `emf_write_wire()` se tyto části vykreslí do souborů EMF.

4.3.5 Zpracování signálů

Signál je v Autodesk Eagle tvořen množinou spojů (vykreslených pomocí funkce `emf_write_wire()`), polygonů (zpracovaných `emf_write_polygon()` a prokovů. Funkce `emf_write_via()` provádí vyhodnocení prokovů a volá funkce pro vykreslení jednotlivých částí. Logiku je, vzhledem k jednoznačnosti kódu, možné vysvětlit pomoci ukázky kódu 4.11.

```
...
void emf_write_via(UL_VIA V)
{
  if (V.flags&VIA_FLAG_STOP)
  {
    if(current_layer == LAYER_TSTOP){
        emf_write_restring(V.x, V.y,
                           V.diameter[LAYER_TSTOP],
                           V.shape[LAYER_TSTOP],
                           0, 0.0, LAYER_TSTOP);
    }
    else if(current_layer == LAYER_BSTOP){
        emf_write_restring(V.x, V.y,
                           V.diameter[LAYER_BSTOP],
                           V.shape[LAYER_BSTOP],
                           0, 0.0, LAYER_BSTOP);
    }
  }
  if (current_layer == LAYER_VIAS)
  {
    emf_write_restring(V.x, V.y, V.diameter[V.end],
      V.shape[V.end], 0, 0.0, current_layer);

    emfWriteDot(V.x, V.y, V.drill, -1);
  }
}
...
```

Výpis 4.11: Kód funkce emf_write_via().

Nejprve se otestuje příznak `VIA_FLAG_STOP`. Pokud je příznak nastavený a aktuální vrstva je `LAYER_TSTOP` nebo `LAYER_BSTOP`, je nutné vykreslit prokov na těchto

vrstvách. Odpovídající tvar a rozměr lze získat z polí `diameter` a `shape`. Indexem je číslo vrstvy a příslušné tvary a rozměry se generují na základě množiny návrhářských pravidel pro celou desku plošných spojů. Samotná via je propojením některých vrstev, avšak vykresluje se pouze na vrstvě `LAYER_VIAS` s tvarem odpovídajícím poslední propojené vrstvě. Toto zjištění bylo získáno pomocí předchozích zkušeností vedoucího práce a nahlédnutím do jiných ULP skriptů. Kromě vyplnění tvaru, je pomocí funkce `emfWriteDot()` (s nastavením neexistující vrstvy pro tisk barvy pozadí) vykreslena i díra (angl. drill).

V ukázce kódu jednotlivé tvary vykresluje funkce `emf_write_restring()`. Její parametry jsou tvar, rozměr (průměr) a středový bod. Exituje několik možných tvarů:

- `PAD_SHAPE_SQUARE` – obdélník. Pomocí středového bodu a poloměru jsou vypočteny jednotlivé řídící body. Obdélník se vykreslí funkcí `emfWriteRect()`.
- `PAD_SHAPE_OCTAGON` – osmiúhelník. Funkce `emfWritePolygonOctagon` zapíše do metasouboru osmiúhelník, jako vyplněný polygon.
- `PAD_SHAPE_LONG` – zakulacený obdélník. Výsledný tvar je totožný se situací při vykreslení spoje na obrázku . Pro vykreslení je nutno nejdříve aplikovat rotaci na středové body kruhů. Následně se jednotlivé části vykreslí příslušnými funkcemi.
- další – mezi další možné tvary patří `PAD_SHAPE_OFFSET`, `PAD_SHAPE_ANNULUS`, `PAD_SHAPE_THERMAL` a `PAD_SHAPE_ROUND`. Tyto tvary jsou vykresleny jako kruhové uzly pomocí funkce `emfWriteDot()` a aktuální vrstvy.

4.3.6 Zpracování elementů

Funkce `emf_write_element()` vyhodnocuje elementy tvořené texty a pouzdry typu `UL_PACKAGE`. Pouzdro se skládá z polygonů, spojů, kruhů, textů, děr, pájecích plošek a SMD prvků. Pájecí plošky jsou vykresleny funkcí `emf_write_restring()` na vrstvě `LAYER_PADS`. Prvky SMD jsou vykresleny na vrstvách `LAYER_TOP`, `LAYER_BOTTOM` a na vrstvách `LAYER_TSTOP`, `LAYER_BSTOP` pokud je nastaven příznak `SMD_FLAG_STOP`, na vrstvách `LAYER_TCREAM`, `LAYER_BCREAM` pokud je nastaven příznak `SMD_FLAG_CREAM`. K vykreslení je použita funkce `emfWriteRoundedRect()` (4.3.3).

5 Dosažené výsledky

Cíle práce byly splněny. Program je funkční a produkuje vektorové obrázky odpovídající schématům a deskám plošných spojů. V přílohách B až F je porovnání schémat, přičemž první obrázek je vždy vygenerován přímo systémem Autodesk Eagle a druhý je odpovídající obrázek EMF vytvořený programem v rámci práce. V přílohách G až K je porovnání desek plošných spojů. V příloze G je ukázka výstupu referenčního skriptu sch2svg-1_4_1.ulp. Z ukázky je patrné, že byly vylepšeny nastavení šířky pera, pořadí vykreslení jednotlivých vrstev, vykreslení pájecích plošek a prokovů. Jednotlivé spoje jsou zaoblené podle vzorů zpracování v Autodesk Eagle. Pájecí plošky a prokovy navíc obsahují díry s odpovídajícím průměrem. Polygonální objekty nejsou tvořené horizontálními páskami, ale vyplněnými polygony. Toto porovnání bylo součástí zadání.

Mezi další vylepšení patří možnost nastavení pořadí vykreslení vrstev přímo v hlavičce kódu a určení potřebné přesnosti vykreslení (výchozí přesnost je na desetiny mikrometru). Optimalizované bylo i nastavení barev a šířky per a štětců. Tato optimalizace způsobuje v některých případech zmenšení velikosti souboru na jednu třetinu. Rychlost vytvoření metasouboru nepřekročila, ani u větších desek plošných spojů, jednotky vteřin. Zdrojový soubor programu obsahuje včetně komentářů 2410 řádků a je členěn do 75 funkcí. Funkčnost programu byla otestována v systémech Autodesk Eagle verze 9.1.3 a 9.3.2.

Program ovšem není dokonalý. Mezi možné další směry vývoje patří zakulacení konců oblouků a vyřešení problémů s absencí průhlednosti v metaformátů EMF.

Závěr

Cílem diplomové práce bylo vytvořit program pro export obrázků do vektorového formátu EMF ze schémat a desek plošných spojů systému Autodesk Eagle s podporou všech funkcí verze 9. Program byl úspěšně implementován a otestován na pěti složitějších schématech a deskách plošných spojů, které lze najít v přílohách. Samotný vývoj proběhl na operačním systému Windows 8.1, za použití volně dostupného editoru zdrojového kódu Notepad++ s nastavením syntaxe jazyka C. Kromě samotné implementace je práce přínosná z hlediska podrobného rozboru použitých struktur a samotných metaformátů Windows.

První kapitola této diplomové práce se věnuje popisu metaformátu WMF, který je základem pro další metaformáty Windows. K pochopení práce je vhodné se seznámit s tímto formátem, jelikož tvoří základ a mnoho výčtových typů a objektů bylo použito v implementaci. Zároveň kapitola pojednává o celkové myšlence a původu tohoto metaformátu, čímž naznačuje jeho výhody v rámci přenositelnosti a jednoduchosti vykreslení dat na různých zařízeních.

Pokračováním popisu metaformátu Windows se zabývá druhá kapitola. Uvádí metaformát EMF a popisuje jeho části, které jsou relevantní pro diplomovou práci.

Ve třetí kapitole je čtenář seznámen se zjednodušenou logikou programu, jeho návrhem a základními funkcemi pro jednotlivé vymoženosti Autodesk Eagle 9. Jednoduché funkce jsou přímo vysvětleny. U složitějších funkcí je přiložen odkaz na podrobný popis kvůli snadné orientaci v rámci práce.

Implementačním detailům a získaným zkušenostem se věnuje čtvrtá kapitola. Tato část práce pojednává o problémech v částech nepodporovaných metaformátem EMF, způsobech řešení a možnostech náhrady nedostačující funkcionality. Nechybí ani ukázky kódů, vývojové diagramy a vizuální interpretace algoritmů.

Poslední kapitolou je shrnutí dosažených výsledků – plného splnění zadání z hlediska podpory výstupu pouze viditelných vrstev, správného pořadí vykreslení a zachování barev, rozměrů a orientace. Kapitola také naznačuje možná vylepšení a směry dalšího vývoje.

Literatura

[1] Microsoft: *[MS-EMF]: Enhanced Metafile Format* [online]. Dokumentace formátu. 9. 12. 2018, [cit. 14. 12. 2018]. Dostupné z URL: <`https://msdn.microsoft.com/en-us/library/cc230514.aspx`>.

[2] Microsoft: *[MS-WMF]: Windows Metafile Format* [online]. Dokumentace formátu. 9. 8. 2018, [cit. 14. 12. 2018]. Dostupné z URL: <`https://msdn.microsoft.com/en-us/library/cc250370.aspx`>.

[3] HEROUT, P. *Učebnice jazyka C:*. 1. díl. České Budějovice: Kopp, 2008. ISBN 978-807-2323-517.

[4] Microsoft: *About Metafiles* [online]. 31. 5. 2018, [cit. 14. 12. 2018]. Dostupné z URL: <`https://docs.microsoft.com/cs-cz/windows/desktop/gdi/about-metafiles`>.

[5] Autodesk: *EAGLE EASILY APPLICABLE GRAPHICAL LAYOUT EDITOR* [online]. User Language, version 5.4. 2017, [cit. 14. 12. 2018]. Dostupné z URL: <`http://dl36mmdz94630.cloudfront.net/uploads/eagle_resources/files/000/002/364/original/ulp822_en.pdf?1500495692`>.

[6] xtitoris: *eagle2svg-1.4.1.ulp* [online]. User Language Program, Version 1.4. 2012, [cit. 8. 5. 2019]. Dostupné z URL: <`http://eagle.autodesk.com/eagle/download/1864`>.

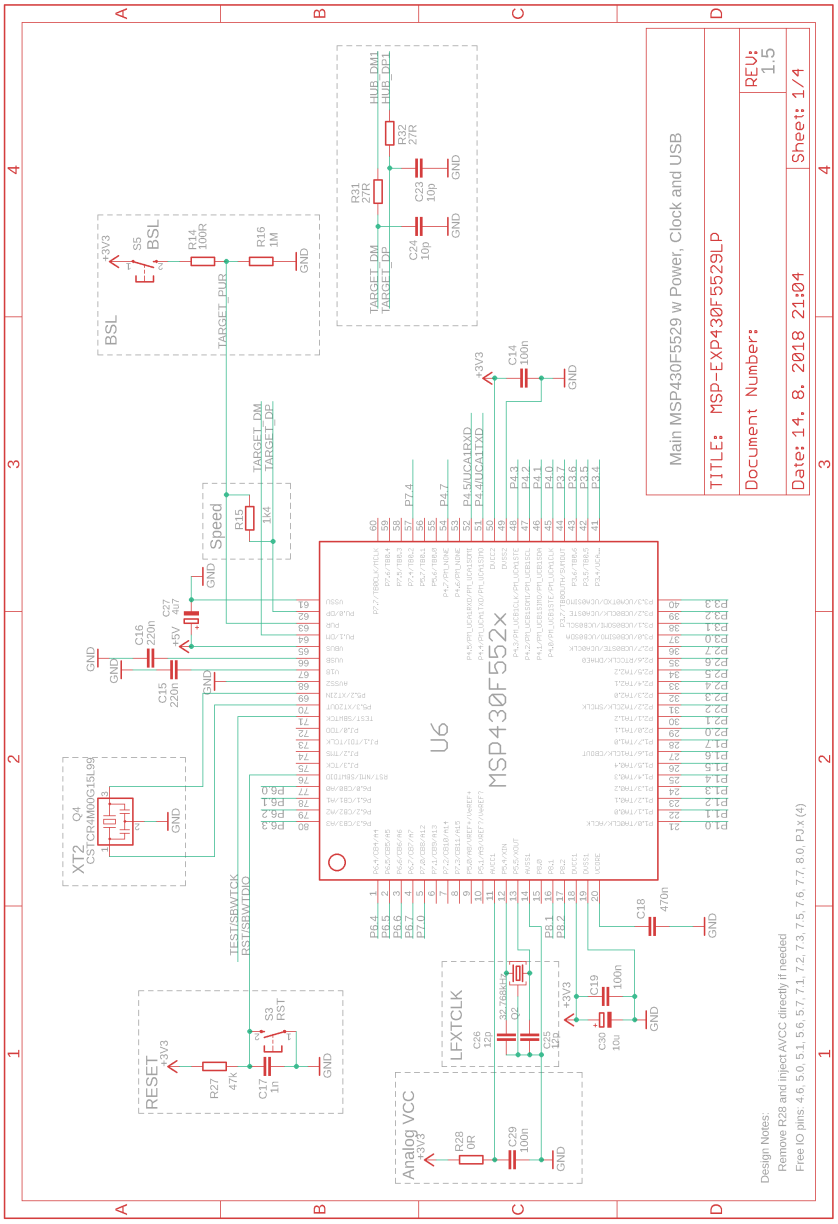
[7] Henrik Haftmann: *sch2wmf.zip* [online]. User Language Program. 2008, [cit. 8. 5. 2019]. Dostupné z URL: <`http://eagle.autodesk.com/eagle/download/1979`>.

Seznam příloh

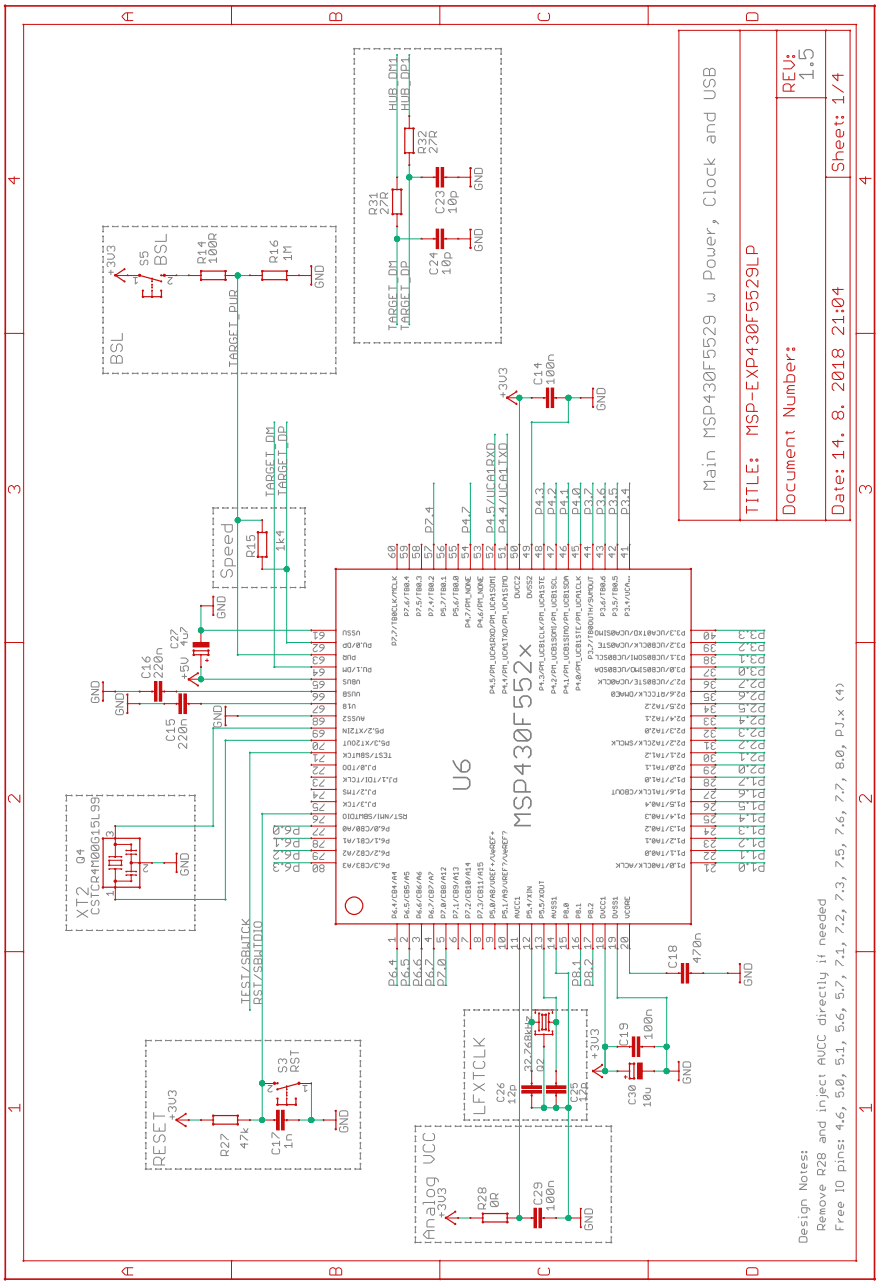
A Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `tex.zip` ..... archiv se zdrojovými textovými soubory
  - `práce.pdf` ..... tato diplomová práce
  - `eagle2emf.ulp` ..... zdrojový kód programu vytvořeného v rámci práce
  - `emf.zip` ..... archiv s EMF obrázky z příloh

B Porovnání schéma MSP-EXP430F5529LP

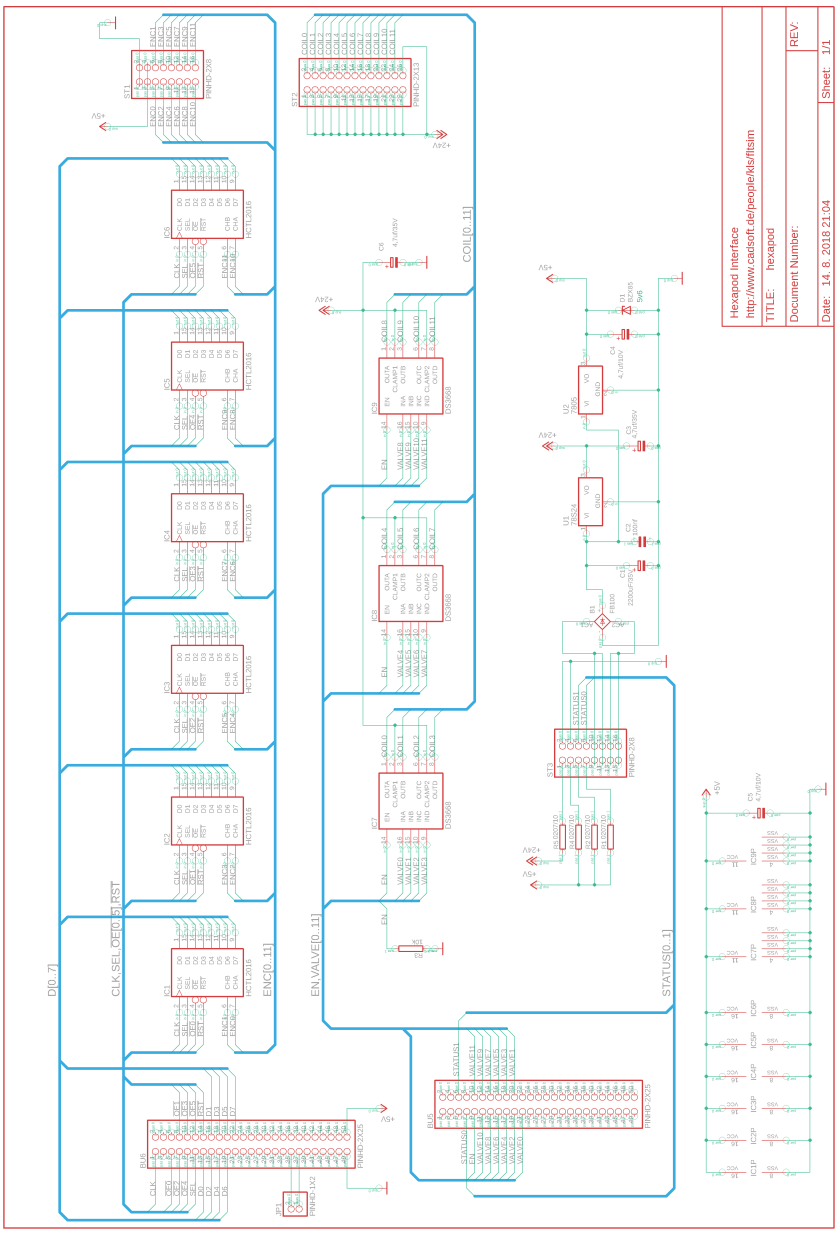


Obr. B.1: Vzorový PDF soubor č.1.

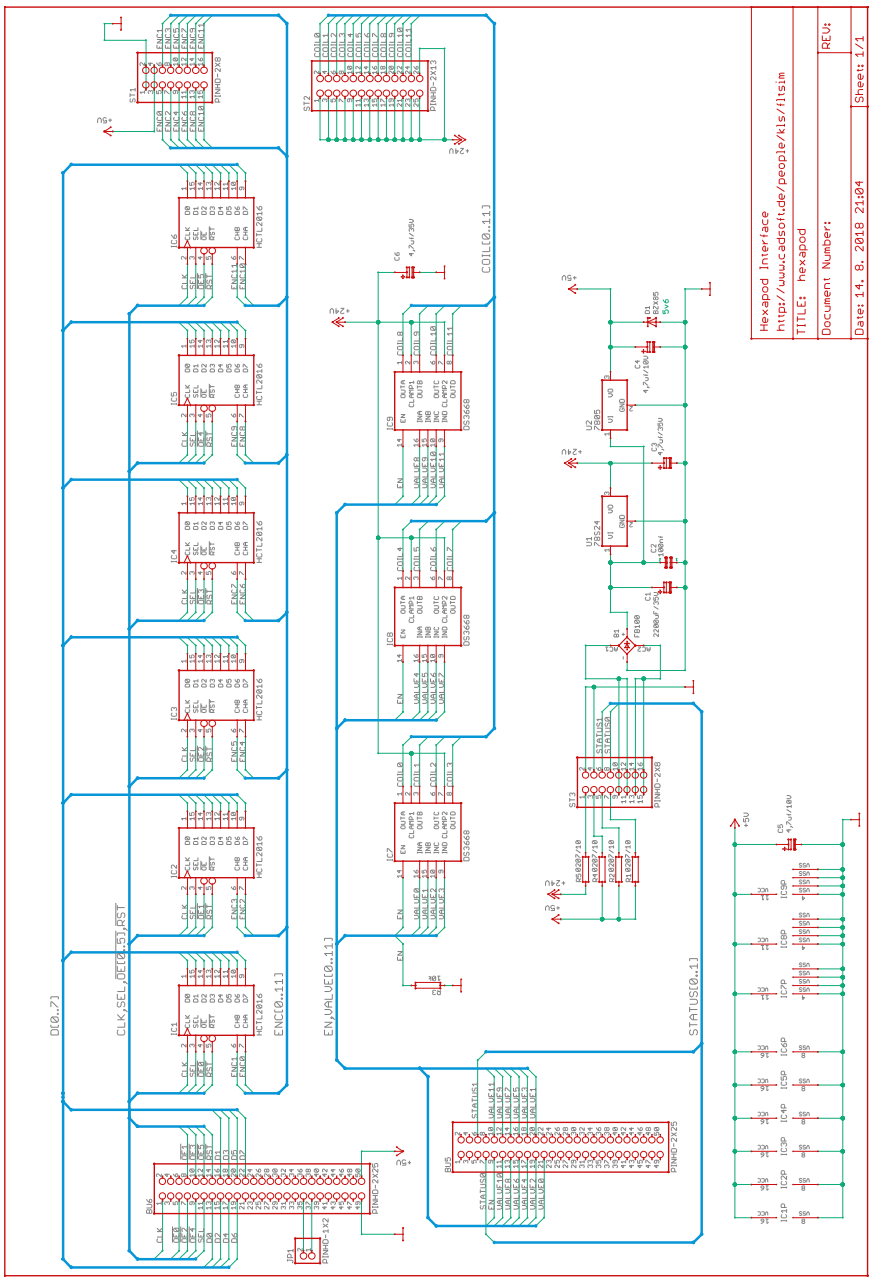


Obr. B.2: Vygenerovaný EMF soubor č.1.

C Porovnání schéma Hexapod

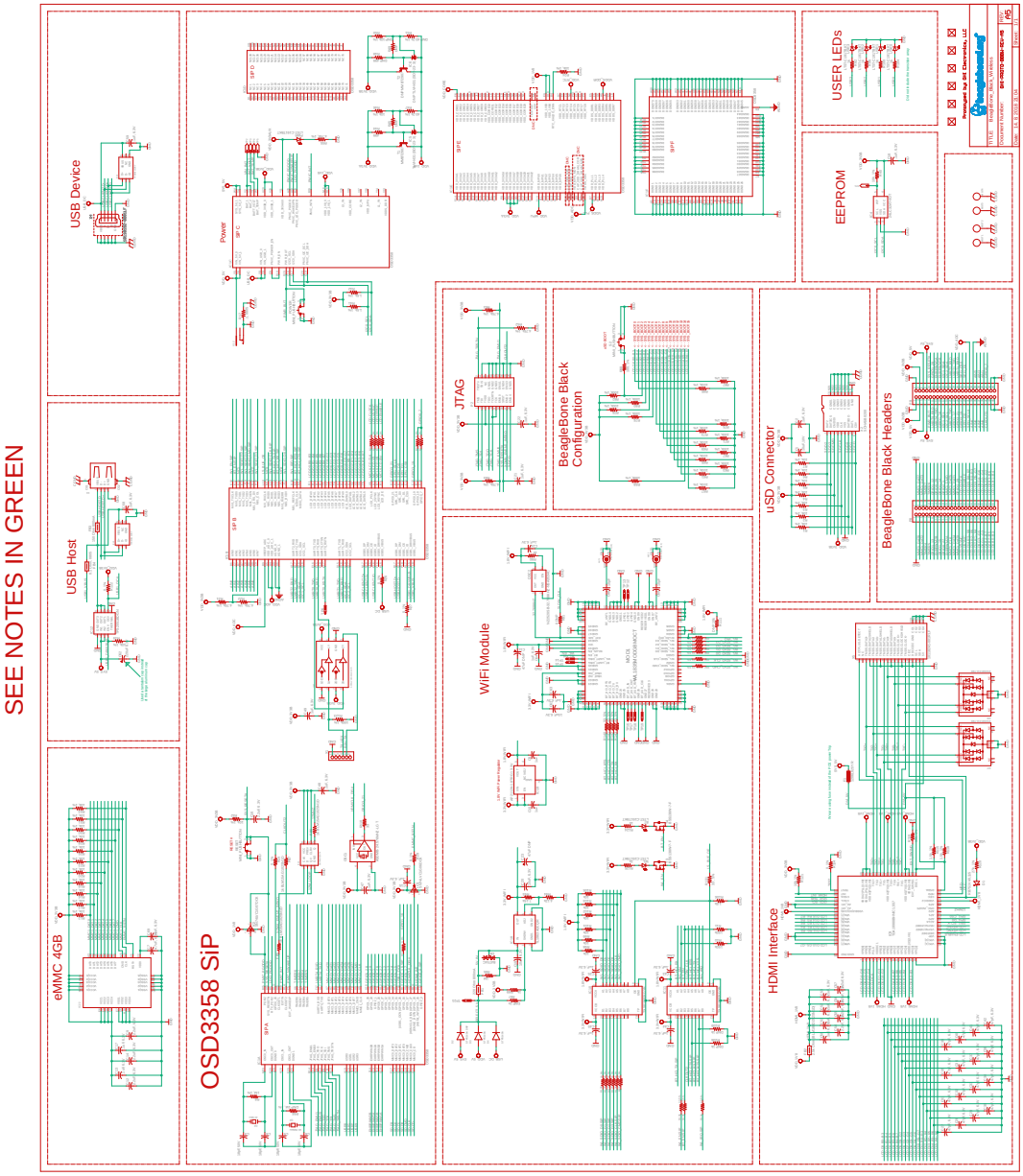


Obr. C.1: Vzorový PDF soubor č.2.

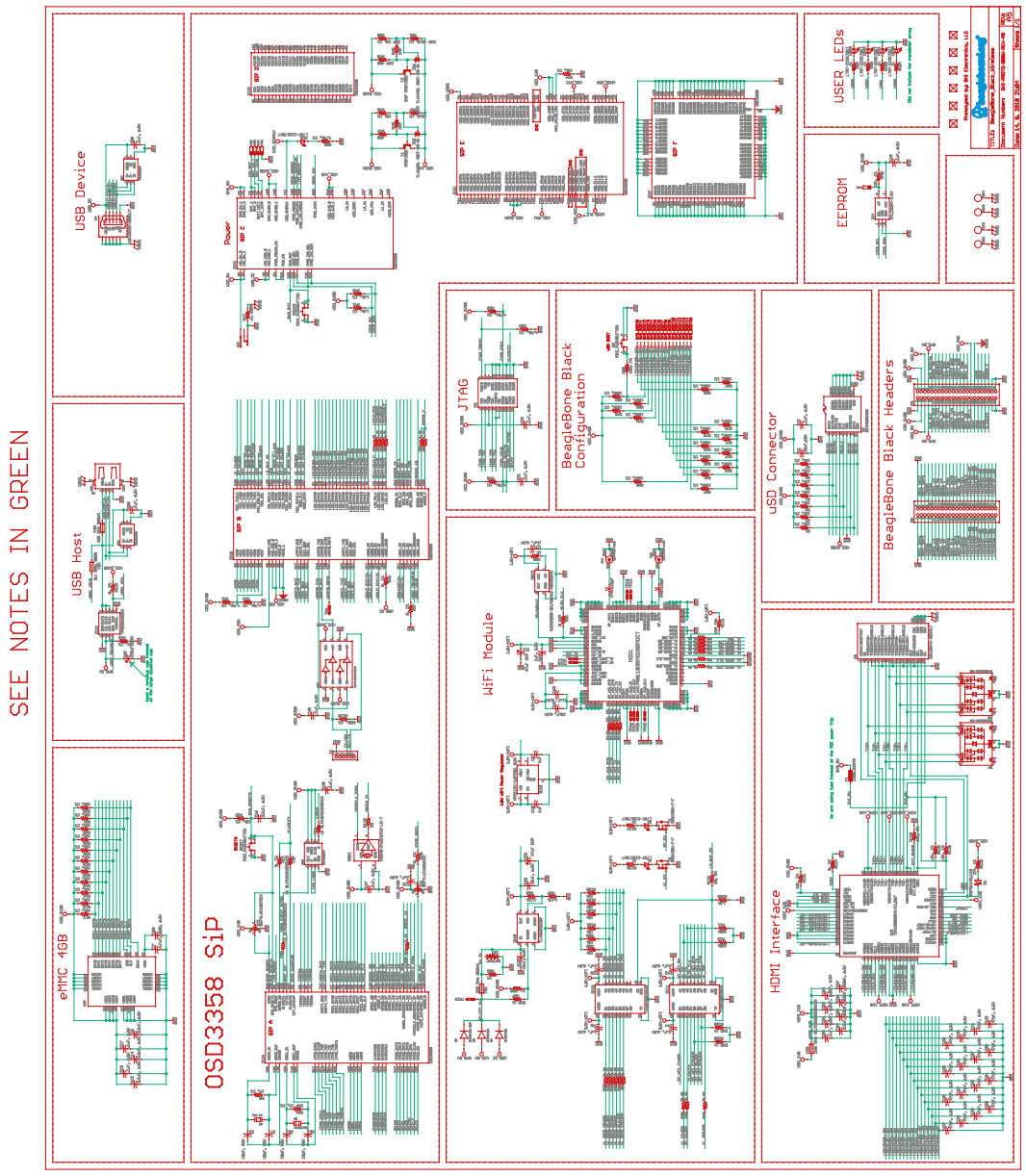


Obr. C.2: Vygenerovaný EMF soubor č.2.

D Porovnání schéma BeagleBone_Black

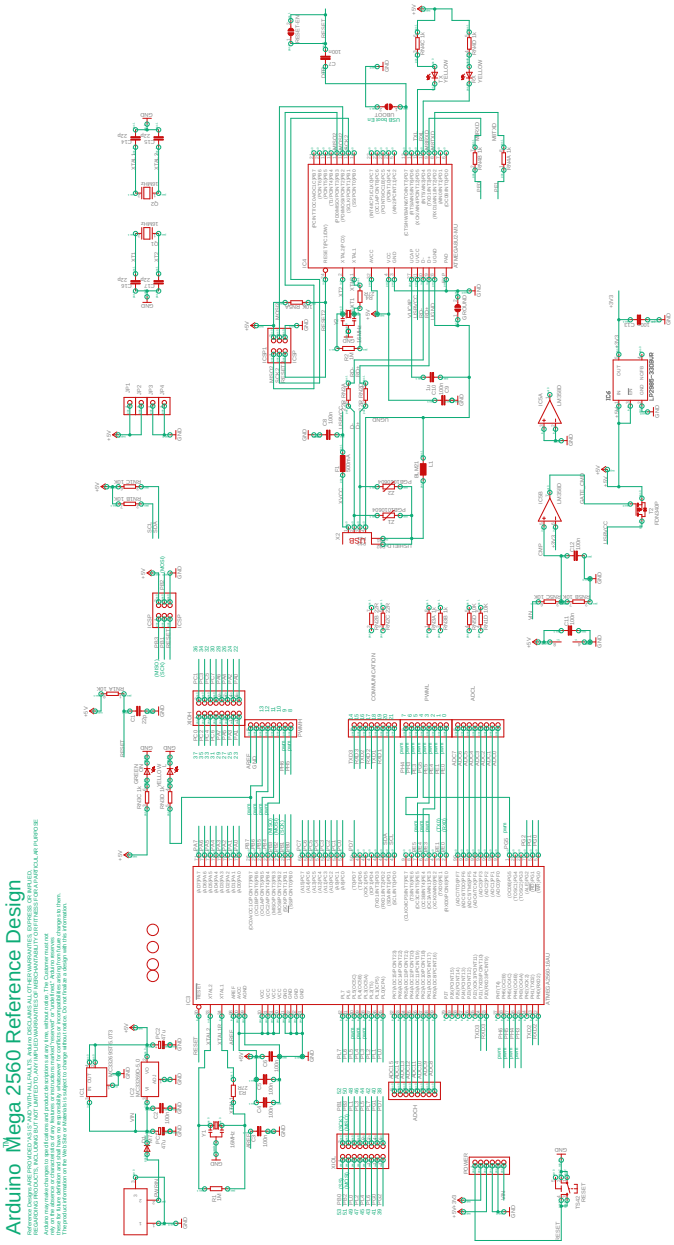


Obr. D.1: Vzorový PDF soubor č.3.

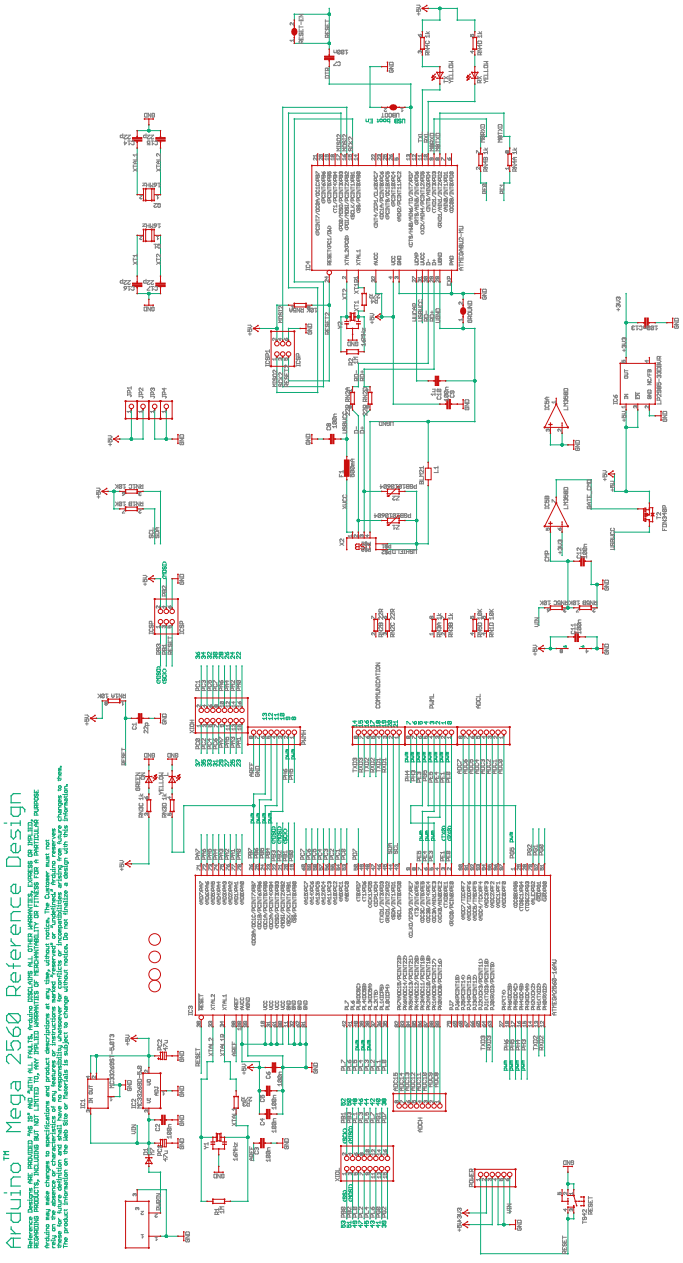


Obr. D.2: Vygenerovaný EMF soubor č.3.

E Porovnání schéma Arduino_MEGA2560

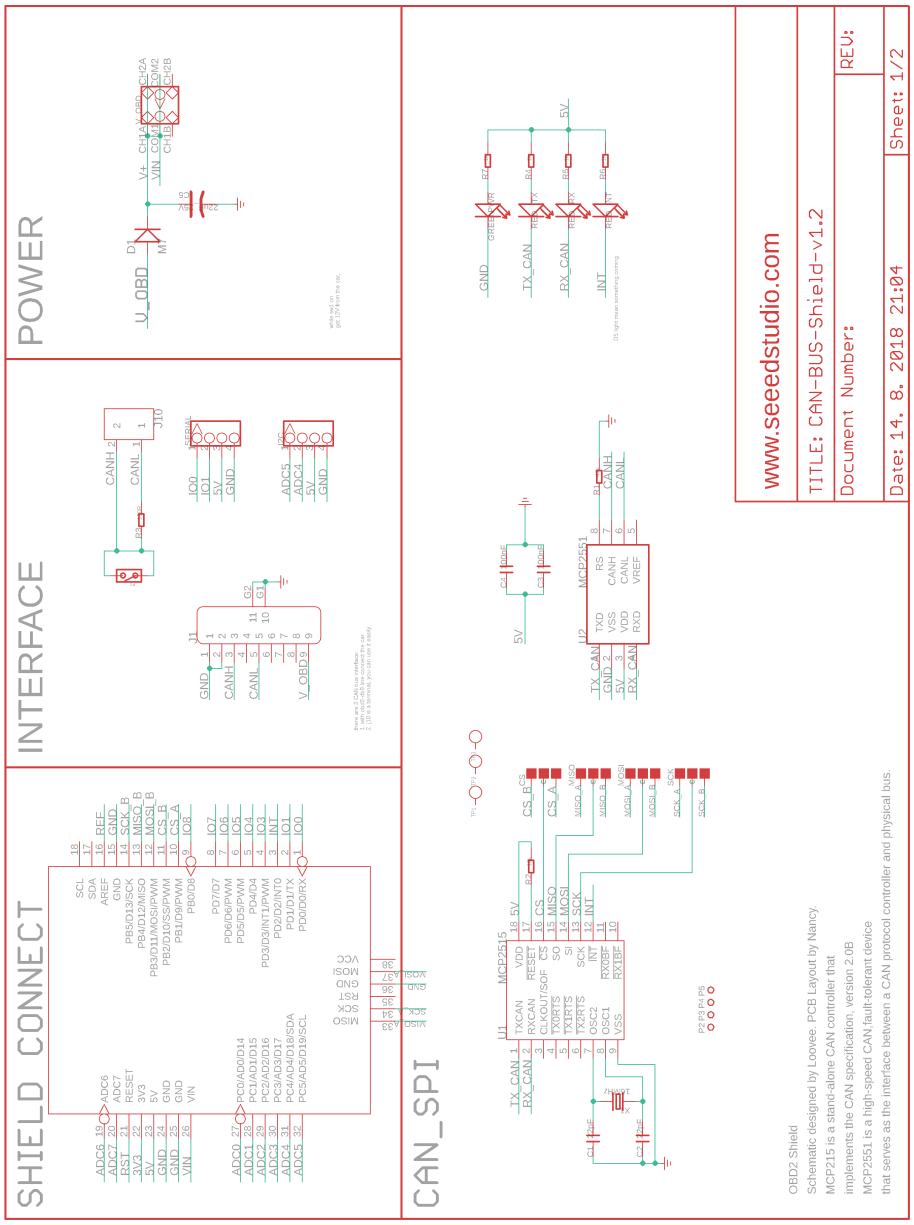


Obr. E.1: Vzorový PDF soubor č.4.

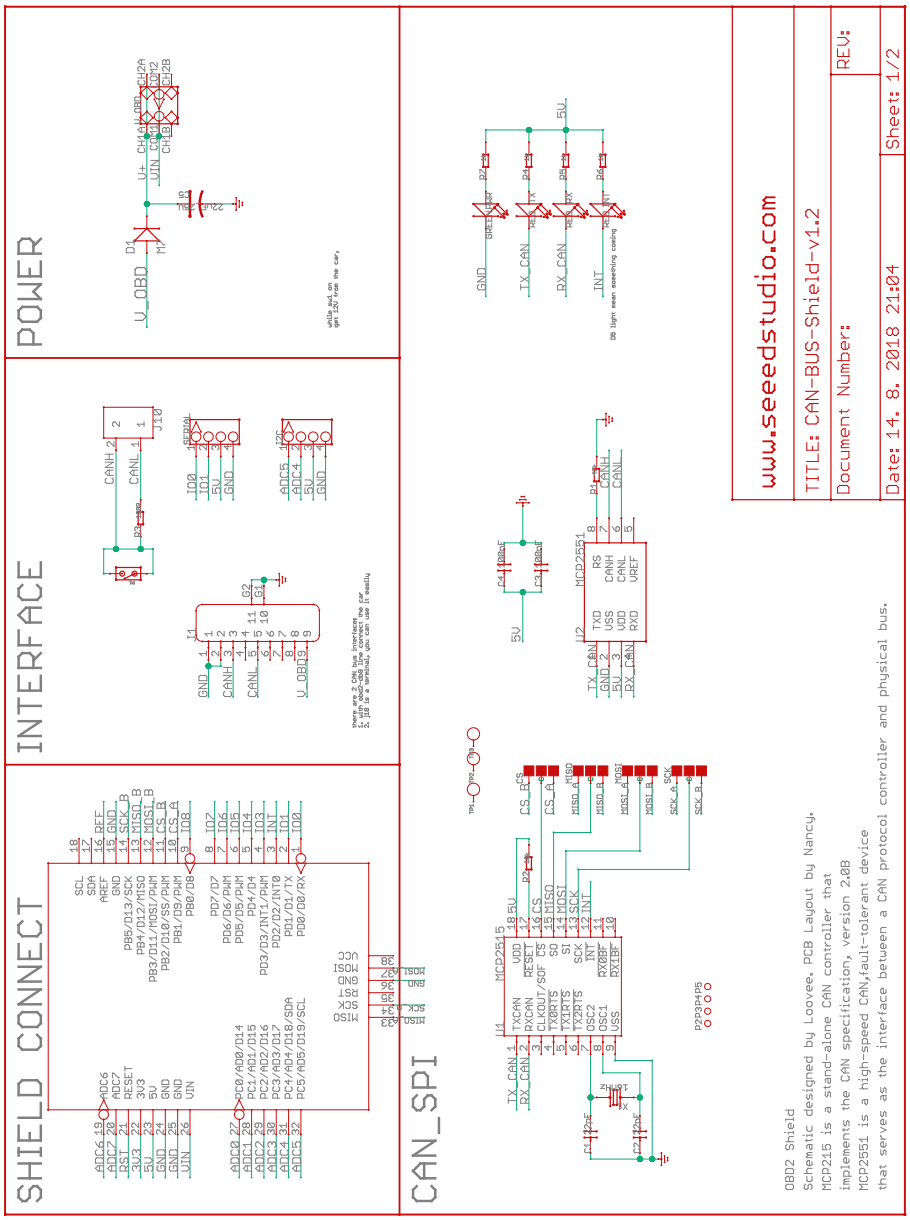


Obr. E.2: Vygenerovaný EMF soubor č.4.

F Porovnání schéma CAN-BUS-Shield-v1.2

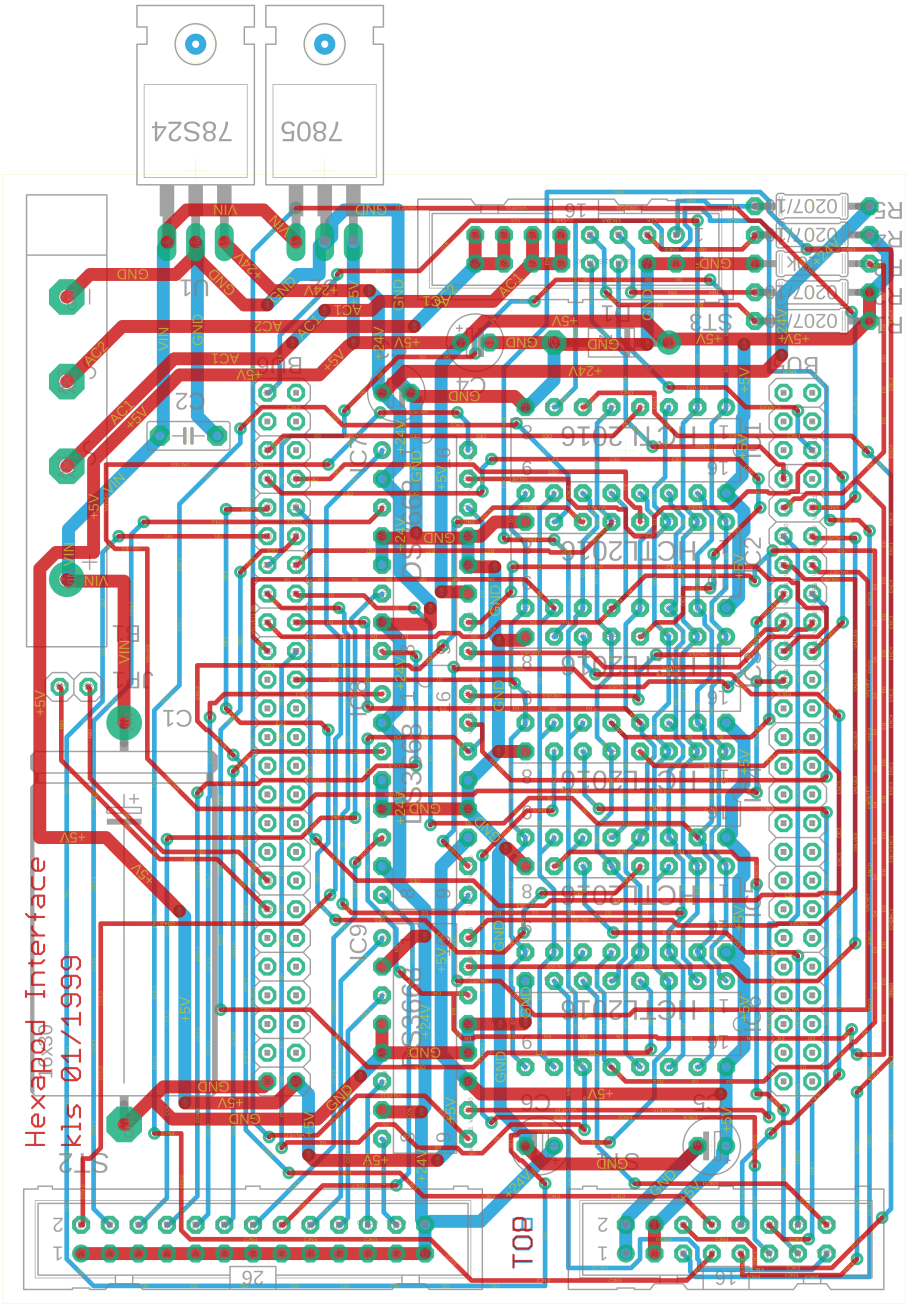


Obr. F.1: Vzorový PNG soubor č.5.

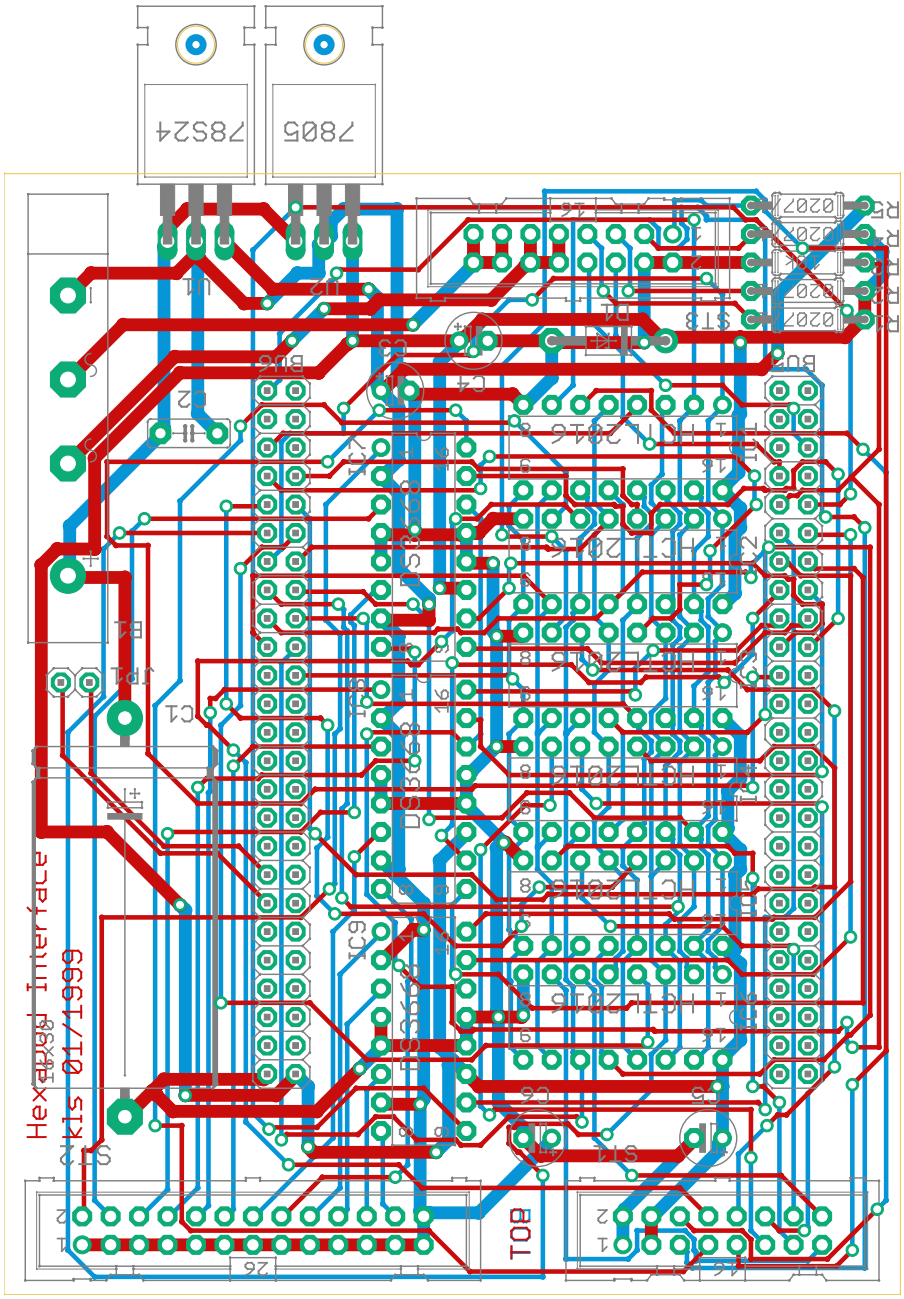


Obr. F.2: Vygenerovaný EMF soubor č.5.

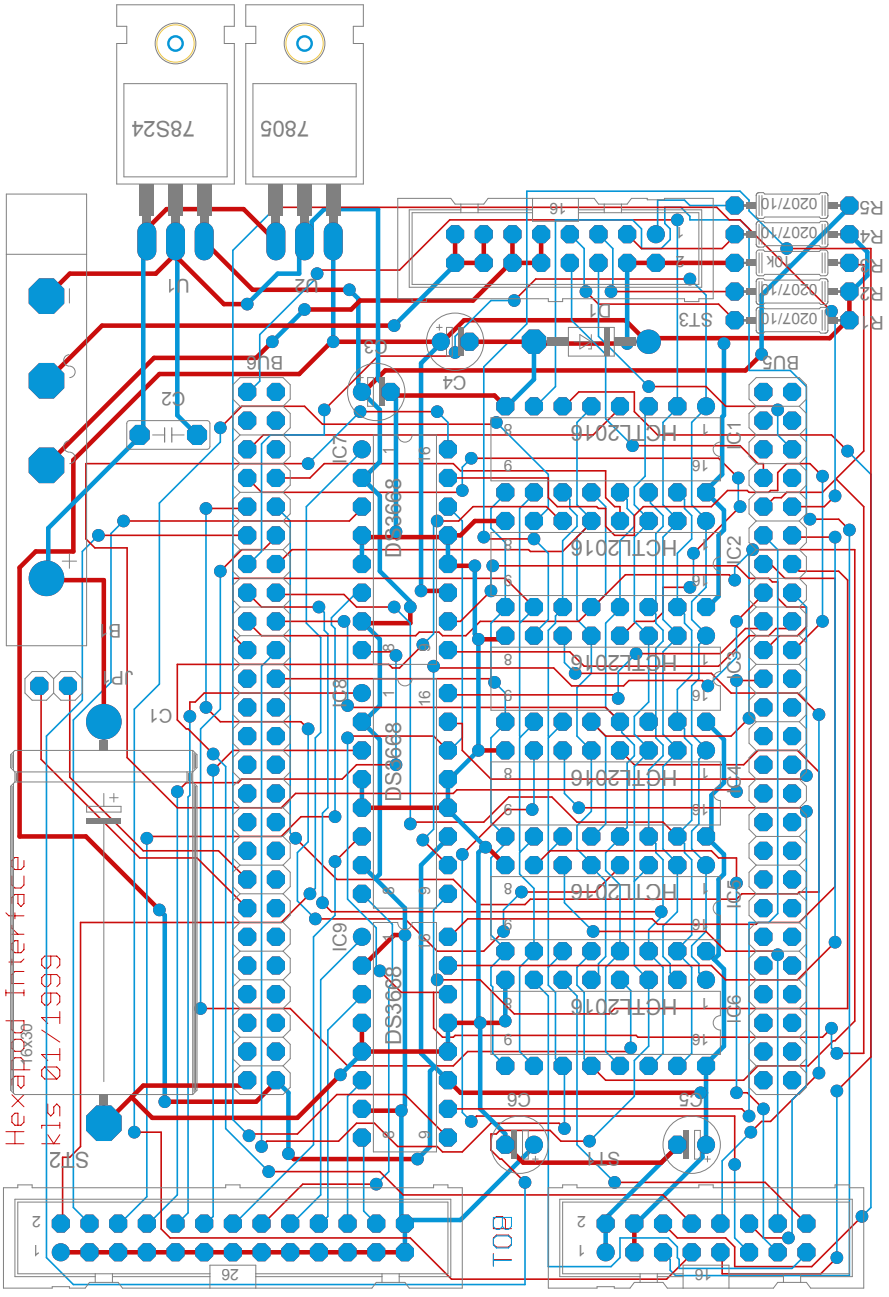
G Porovnání desky plošných spojů Hexapod



Obr. G.1: Vzorový PNG soubor č.6.

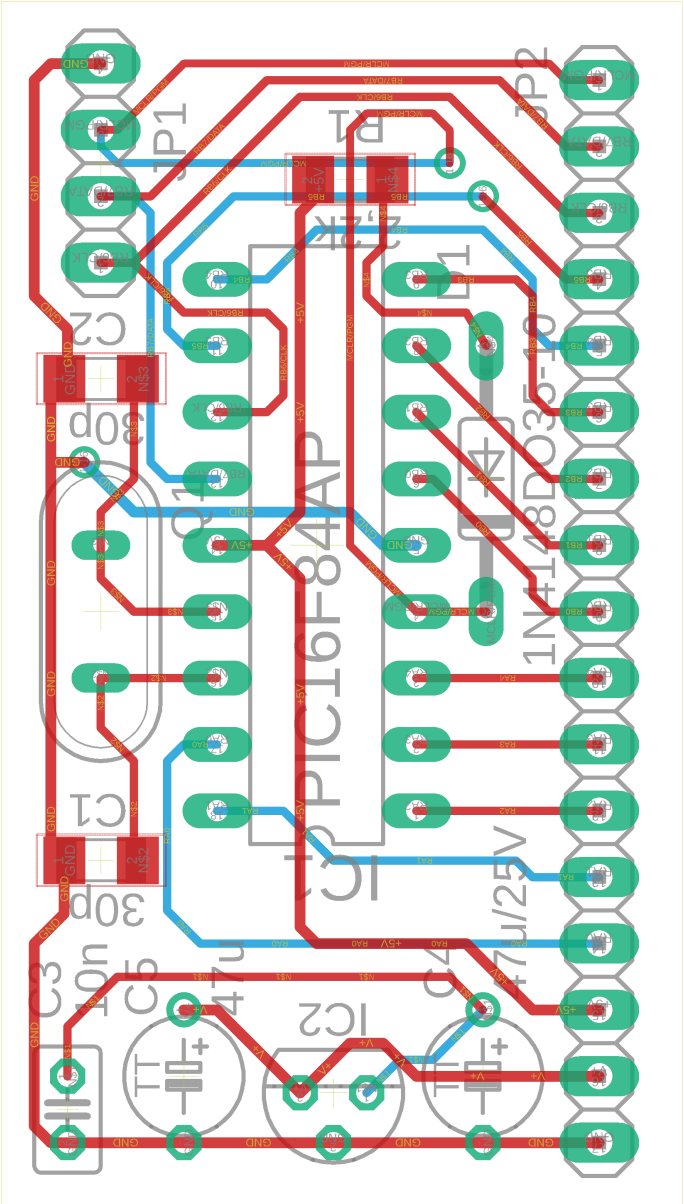


Obr. G.2: Vygenerovaný EMF soubor č.6.

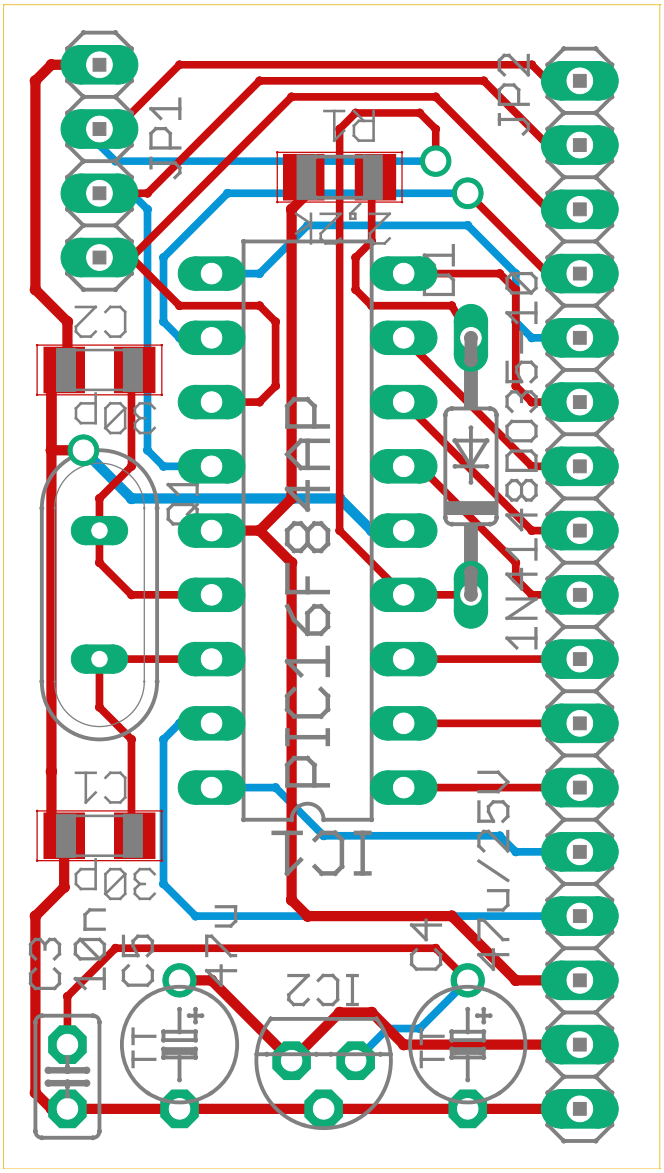


Obr. G.3: Vygenerovaný SVG soubor č.6.

H Porovnání desky plošných spojů demo2

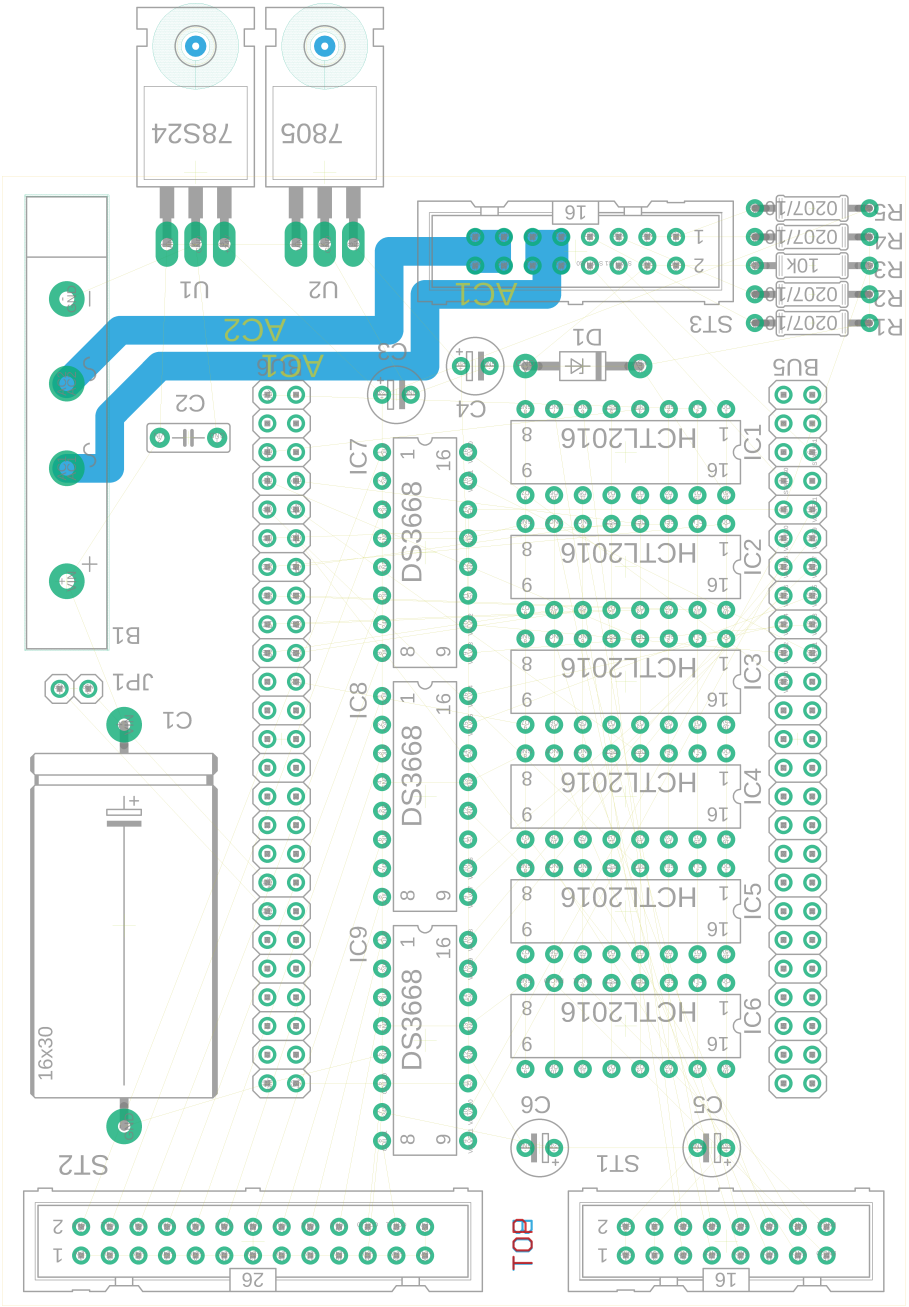


Obr. H.1: Vzorový PNG soubor č.7.

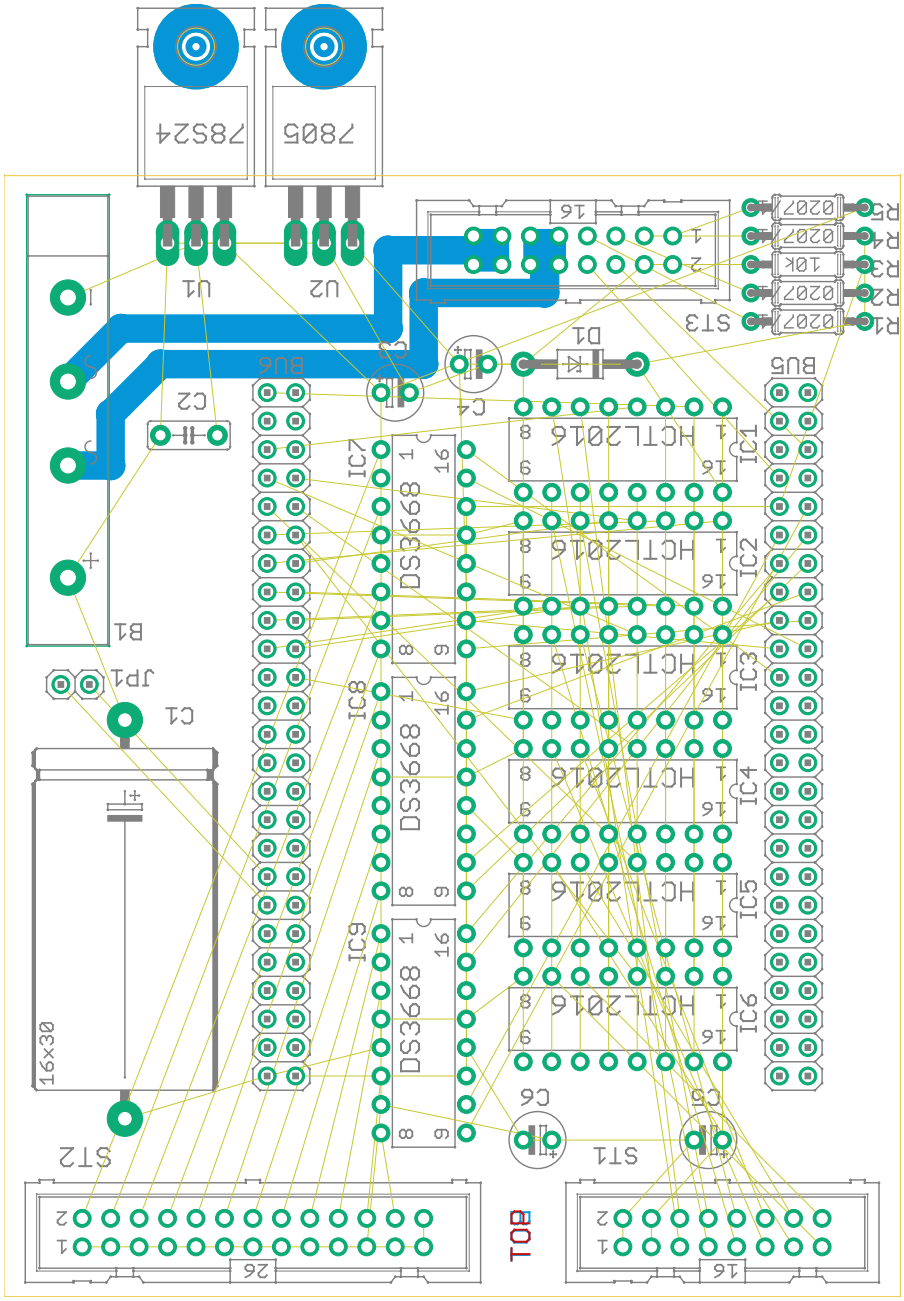


Obr. H.2: Vygenerovaný EMF soubor č.7.

I Porovnání desky plošných spojů hexapodu

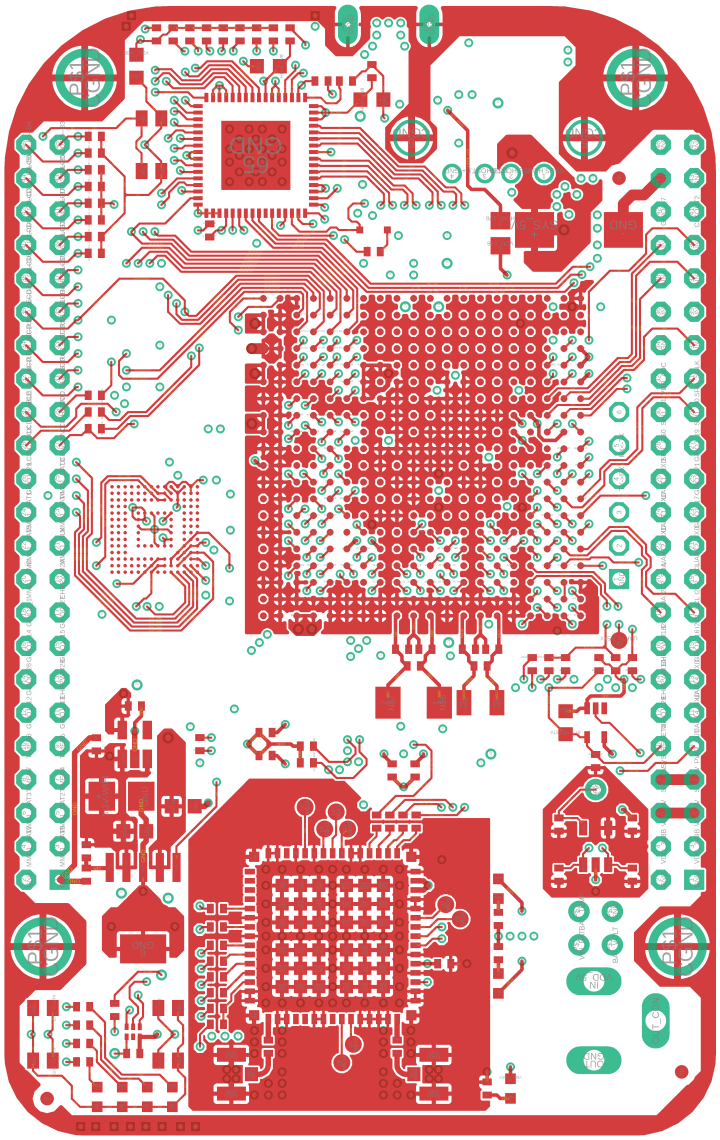


Obr. I.1: Vzorový PNG soubor č.8.

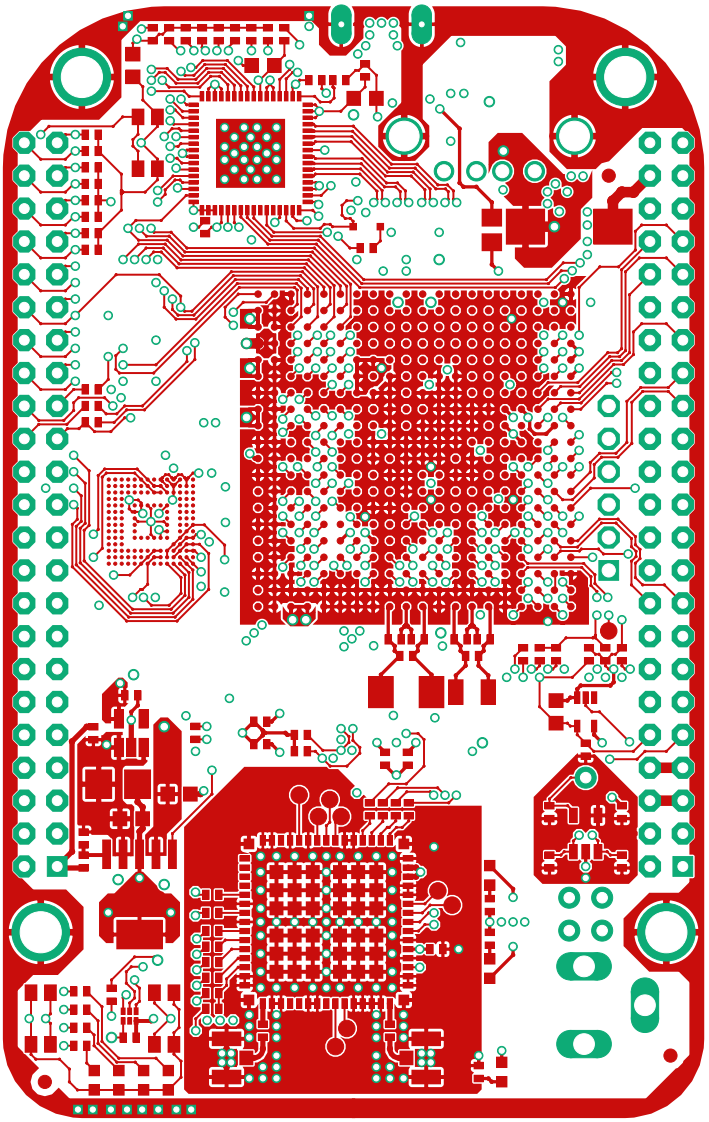


Obr. I.2: Vygenerovaný EMF soubor č.8.

J Porovnání desky plošných spojů BeagleBone Black Wireless, vrstvy TOP, PADS a VIAS

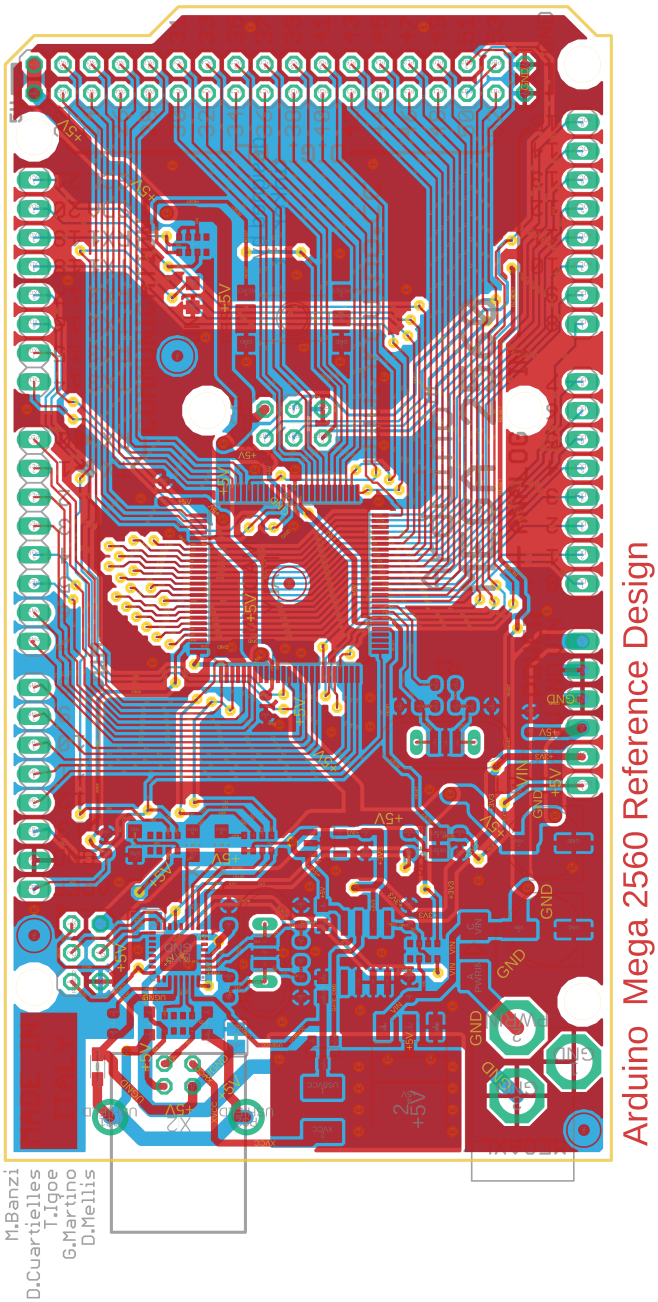


Obr. J.1: Vzorový PNG soubor č.9.

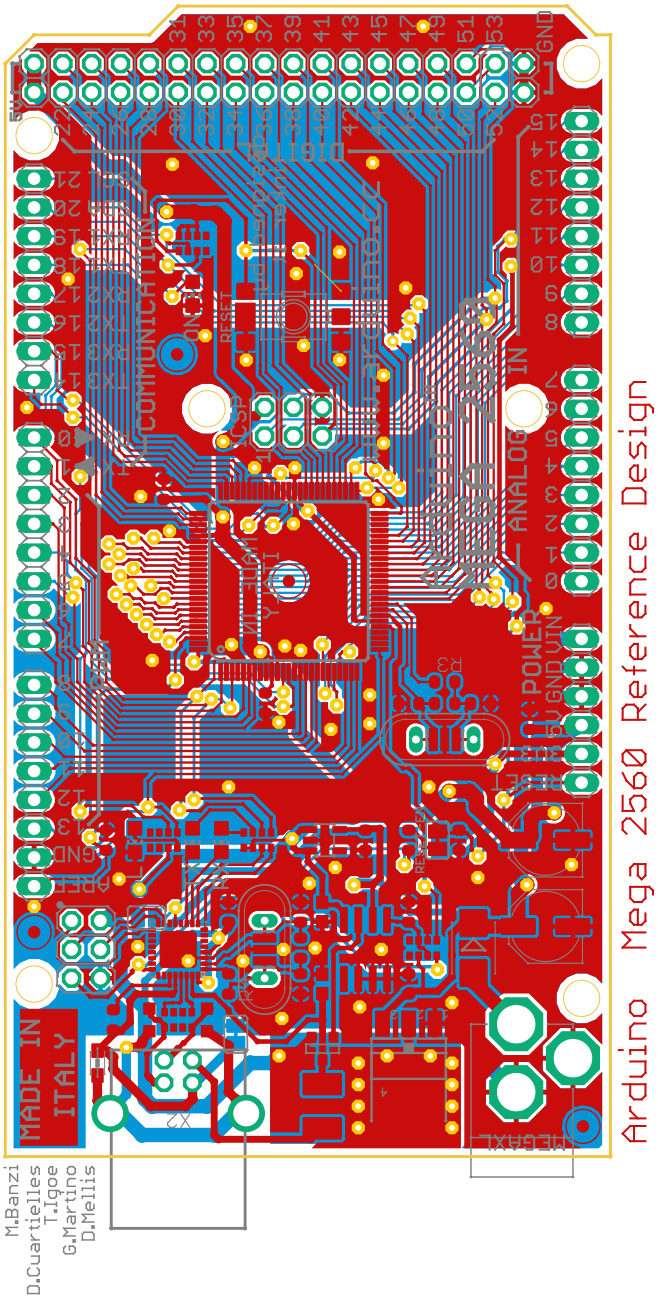


Obr. J.2: Vygenerovaný EMF soubor č.9.

K Porovnání desky plošných spojů Arduino MEGA2560 ref



Obr. K.1: Vzorový PNG soubor č.10.



Obr. K.2: Vygenerovaný EMF soubor č.10.