VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČASOMÍRA LETU V ULTRALEHKÉM LETADLE/VRTULNÍKU

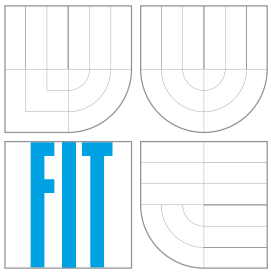
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM ŠIROKÝ

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ČASOMÍRA LETU V ULTRALEHKÉM LETADLE/VRTULNÍKU

FLIGHT TIMER IN ULTRAMICRO AIRCRAFT/HELICOPTER

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. ADAM ŠIROKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2015

Abstrakt

Tato diplomová práce se zabývá vývojem vestavěného systému pro ultralehká letadla a vrtulníky. Cílem je navrhnout systém, který by umožnoval měřit a zaznamenávat délku jednotlivých letů. Jedná se tedy o elektronický palubní deník, který by měl usnadnit a zpřehlednit administrativu spojenou s provozem sportovních letadel. Systém je postaven na mikrokontroléru z rodiny MSP430 od firmy Texas Instruments. Čtenář bude v tomto textu seznámen s použitým hardwarem, návrhem systému, jeho realizací a možnostmi využití vytvořeného systému.

Abstract

This thesis deals with the development of embedded system for ultra light airplanes and helicopters. The aim is to design a system which would allow to measure and record the length of individual flights. It is therefore an onboard electronic diary which should facilitate and streamline the administration associated with sport aircraft operation. The base of the system is created on microcontroller of the MSP430 family by Texas Instruments. A reader will be offered the used hardware, the system design, the way of function and the possibility of using of the suggested system.

Klíčová slova

Mikrokontrolér, MSP430, Texas Instruments, vestavěný systém, časomíra letu, LCD displej, Batron BTHQ128064AVD1-COG-STF-12-LED02YG, ST7565P, 6800, I^2C, Maxim DS1338Z-33+, Flytimer, Hi-Link HLK-RM04, DMA, Local Storage, palubní deník

Keywords

Microcontroller, MSP430, Texas Instruments, embedded system, flight timer, LCD display, Batron BTHQ128064AVD1-COG-STF-12-LED02YG, ST7565P, 6800, I^2C, Maxim DS1338Z-33+, Flytimer, Hi-Link HLK-RM04, DMA, Local Storage, onboard electronic diary

Citace

Adam Široký: Časomíra letu v ultralehkém
letadle/vrtulníku, diplomová práce, Brno, FIT VUT v Brně, 2015

Časomíra letu v ultralehkém letadle/vrtulníku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana profesora Pavla Zemčíka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Adam Široký
24. května 2015

Poděkování

Rád bych zde poděkoval vedoucímu mé diplomové práce za poskytnutou pomoc při tvorbě tohoto textu a cenné rady z oblasti hardwaru.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Vestavěné systémy dnes nacházejí uplatnění téměř všude kolem nás. Jinak tomu není ani u sportovních letadel, kde si tyto systému v uplynulé době vybudovaly poměrně stabilní pozici. Sportovní létání se těší celosvětové velké oblibě. Výjimkou není ani Česká republika, kde lze nalézt až překvapivě velmi početnou skupinu příznivců sportovního létání. Zároveň však platí, že se jedná o poměrně finančně i časově náročný koníček. Zvláštní kapitolou je samotný provoz letadla, což je činnost, která vyžaduje kromě značných finančních prostředků také určitou míru znalostí a zkušeností. K provozování letadla je dále nutné mít vhodné zázemí, které kromě dostatečného prostoru poskytuje také určitý standard týkající se technického vybavení. Je tedy vcelku logické, že mnoho pilotů není vlastníkem žádného letadla. Namísto toho si stroje za určitou hodinovou sazbu pronajímají.

Kromě výhod, které pramení z uvedených faktů, je také nutné zdůraznit určité nevýhody. Tou je například skutečnost, že majitel letadla nemůže přímo ovlivnit chování jednotlivých pilotů. Bez dodatečného technického vybavení dokonce nemusí mít ani přehled o tom, jak je s jeho majetkem zacházeno. Celý proces je poté založen na důvěře mezi majitelem a nájemcem. Z pohledu údržby letadel je však podstatné mít přehled například o množství skutečně nalétaných hodin, času, kdy byl motor v chodu, a úrovně namáhání draku letadla v době letu.

V praxi se velmi často setkáváme s evidencí těch nejpodstatnějších údajů pomocí různých bloků nebo tabulkových editorů. Obecně však lze říci, že proces monitorování provozních údajů sportovních letadel vykazuje velmi malou míru automatizace. Tato skutečnost je dána především tím, že v době uvedení do provozu značného množství letadel by cena zařízení, které by nabízelo automatizovaný přístup, byla příliš velká. Dnešní situace je ovšem poněkud odlišná. Cena, za kterou se dá dnes zhotovit takovéto zařízení, je podstatně nižší. Nabízí se zde tedy dostatek prostoru pro alespoň částečnou modernizaci avioniky sportovních letadel. Zavedení automatizovaného přístupu by mohlo kromě zjednodušení procesu monitorování leteckých údajů také poskytnout vhodný základ pro zařízení, které by umožňovalo zaznamenávání širšího spektra provozních údajů. Z pohledu majitele letadla by bylo například vhodné u vybraných veličin, jako je přetížení, otáčky motoru a úhel stoupání/klesání zaznamenávat maximální dosažené hodnoty během letu.

V současné době je na trhu několik zařízení, která poskytují měření některých provozních údajů. Komplexních řešení za rozumnou cenu, která by bylo možné snadno zakomponovat do současně používaných letadel, je ovšem výrazně méně. Vhodnou platformou, kterou by bylo možné použít pro vývoj takovéhoto zařízení, je například tablet. Ten má ovšem nevýhodu, která pramení z relativně malého prostoru v pilotní kabině ultralehkých letadel. Taktéž princip ovládání tabletu pomocí dotykové obrazovky může být pro řadu uživatelů

nevyhovující. Druhou cestou, kterou se lze vydat a kterou jsem zvolil já, je tvorba vestavěného systému, který lze zakomponovat do palubní desky ultralehkého letadla.

Tuto práci lze označit za volné pokračování mé bakalářské práce na Fakultě informačních technologií VUT v Brně, v níž jsem se věnoval možnostem využití mikropočítače v ultralehkém letadle. Výsledkem této práce bylo jednoduché zařízení, které je schopné zaznamenávat trajektorii letu. Díky této práci se mně povedlo navázat užší spolupráci s leteckým klubem v Kotvrdovicích. Tato spolupráce mně přinesla nabídku v podobě možnosti realizace již dlouho připravovaného zařízení s názvem Flytimer. Zařízení by mělo sloužit jako jakýsi elektronický palubní deník letadla.

Cílem práce je realizovat a přizpůsobit návrh zařízení Flytimer dnešním požadavkům. Ty se poněkud změnily od doby, kdy bylo zařízení poprvé navrhnuto. Kromě samotného zařízení Flytimer je cílem navrhnout a vytvořit vhodné uživatelské rozhraní, které by vyhovovalo potřebám a technickým možnostem místního aeroklubu.

Atraktivitu zadání vidím v tom, že mně umožňuje vytvořit zařízení s reálným komerčním potenciálem v oblasti, ve které se nepohybuji poprvé, a která je terčem mého zájmu. Nedílnou součástí řešení tohoto projektu je aktivní komunikace s piloty, leteckými instruktory a odborníky v oblasti vývoje vestavěných systémů. Všechny tyto skutečnosti, včetně značné atraktivity cílového prostředí, mě vedly k výběru tohoto zadání.

Kromě úvodu, závěru a příloh se tato práce skládá ze tří kapitol. První kapitola je věnována popisu vybraných hardwarových obvodů a technologií včetně možností jejich využití. Veškeré tyto obvody a související technologie se přímo vztahují k zařízení Flytimer a jejich správné pochopení bylo z pohledu realizace klíčové. Závěr této kapitoly patří popisu několika existujích zařízení, které mají podobný nebo stejný charakter. Náplní druhé kapitoly je stručný popis existujícího návrhu zařízení Flytimer, stanovení požadavků na zařízení a návrh způsobu realizace těchto požadavků. V poslední části se věnuji popisu vlastní implementace palubního přístroje a aplikace, která slouží pro správu dat a samotného přístroje. Kromě samotného popisu způsobu realizace je zde uvedena i využitelnost navrženého systému.

Kapitola 2

Dostupná řešení a použité technologie

Tato kapitola shrnuje veškerá podstatná fakta, která bylo nutné zohlednit při návrhu diplomové práce. Samotný rozsah tohoto dokumentu však nedovoluje zmínit veškerá zařízení a všechny dostupné technologie včetně možností jejich využití.

Samotná kapitola je rozdělena na osm částí, kdy první je věnována mikrokontrolérům z rodiny MSP430. Následující dvě části se zabívají problematikou LCD[1] displejů a obvodům reálného času. Kromě popisu a příkladů vybraných obvodů se zde dozvíme konkrétní způsoby připojení těchto periférií k mikrokontrolérům. Následující kapitola ve stručnosti představuje velmi populární framework Qt. Čtvrtá část je zaměřena na protokoly transportní vrstvy síťového modelu TCP/IP, včetně krátkého popisu možnosti realizace síťové aplikace v Qt přes konkrétní protokol transportní vrsty. Obsahem další části je bezdrátová komunikace. Tato podkapitola přináší stručný přehled vybraných technologií a detailnější pohled na možnost převodu sériové komunikace na komunikaci bezdrátovou. Šestá část pro změnu náleží relačním databázím a možnosti využití databáze SQlite v aplikacích, které využívají frameworku Qt. Konečně poslední část představuje vybraná zařízení na trhu, která mají podobný nebo stejný charakter jako popisované zařízení.

2.1 Mikrokontroléry MSP430

MSP430 je označení velmi početné rodiny šestnáctibitových mikrokontrolérů od firmy Texas Instruments, která čítá více než 400 zařízení. Tyto mikrokontroléry byly navrženy s velkým důrazem na nízkou cenu a co možná nejnižší příkon. Tento fakt ovšem neznamená, že by výrobce neinvestoval do vývoje nových technologií. Naopak jako první výrobce představil řadu mikrokontrolérů, které využívají technologie FRAM[2]. Jedná se o nový typ paměti, která nahrazuje doposud využívanou paměť Flash. Tato nová technologie se pyšní tím, že kombinuje výhody pamětí Flash a SRAM[3]. Ve výsledku to znamená, že zákazník dostává nevolatilní paměť, která je mnohem rychlejší, má nižší spotřebu a výrazně delší životnost než paměť typu Flash [35, 28].

Z těchto důvodu se výrobce pyšní tím, že se jedná o mikrokontroléry, které nabízí co možná nejnižší spotřebu, přijatelnou cenu a vyváženou kombinaci integrovaných periférií,

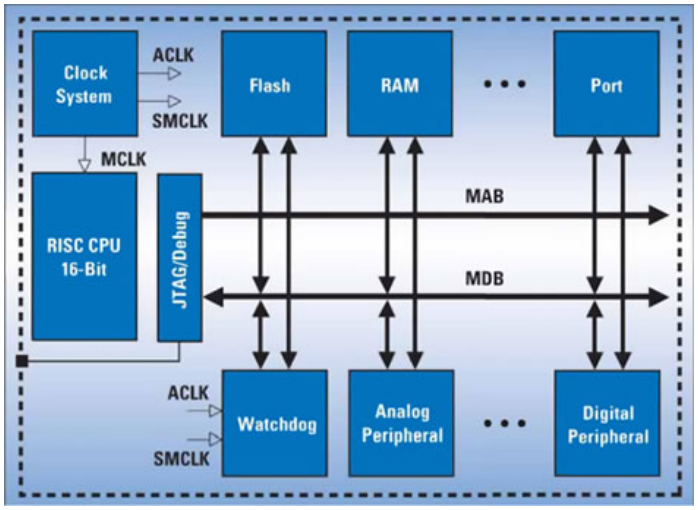
[1] LCD (Liquid Crystal Display)
[2] FRAM (Ferroelectric Random Access Memory)
[3] SRAM (Static Random Access Memory)

jakými jsou například časovače, A/D převodníky a moduly pro komunikaci s periferiemi. A právě těmto modulům je mimo jiné věnována podstatná část této kapitoly. Dále se zde dozvíme něco o samotné architektuře mikrokontrolérů z rodiny MSP430 [28].

2.1.1 Architektura

Rodina mikrokontrolérů MSP430 je rozdělena do několika řad, které sdružují mikrokontroléry s podobnými vlastnostmi. Příkladem této vlastnosti je typ hlavní paměti.

Společným rysem celé rodiny je architektura mikrokontroléru. Konkrétně se jedná o von Neumannovu architekturu, která má společnou paměť pro program i data. Základní blokové schéma mikrokontrolérů z této rodiny je na obrázku 2.1. Z něho je však patrné, že mikrokontroléry této rodiny obsahují dva paměťové elementy. Konkrétně bloky Flash a RAM[4]. Paměť typu Flash slouží k uložení kódu a dat. V závislosti na konkrétní řadě mikrokontrolérů může být tato paměť i typu FRAM nebo ROM[5]. Paměť RAM slouží primárně k ukládání proměnných [28].



Obrázek 2.1: Blokové schéma mikrokontroléru MSP430 [6]

Skutečnost, že data i kód jsou fyzicky ve stejné paměti zabraňuje tomu, abychom mohli zároveň přistupovat k datům v paměti a načítat další instrukce kódu. Za určitých okolností však může být žádoucí, aby mikrokontrolér mohl vykonávat určitou činnost v době, kdy probíhá například ukládání dat do paměti Flash. Právě z tohoto důvodu je možné využít paměťi RAM, která primárně slouží pro uložení proměnných, i pro uložení části kódu. Ten pak může být vykonáván i v době, kdy řadič hlavní paměti provádí paměťovou operaci [14].

Kromě již zmíněných paměťových bloků a procesoru (CPU[6]) může být mikrokontrolér dále osazen různými moduly jako jsou například moduly pro: řízení hodinových signálů, komunikaci s analogovými a digitálními periferiemi, programování mikrokontroléru, časovače apod. Jednotlivé části mikrokontroléru jsou propojeny pomocí datové sběrnice MDB[7] a sdílené adresové sběrnice MAB[8] [2].

[4]RAM (Random Access Memory)
[5]ROM (Read Only Memory)
[6]CPU (Central Processing Unit)
[7]MDB (Memory Data Bus)
[8]MDB (Memory Adsress Bus)

CPU

Výrobce dále udává, že mikrokontroléry MSP430 jsou postaveny na šestnáctibitovém RISC[9] procesoru. Na toto téma lze však nalézt celou řadu diskuzí v odborné literatuře, které poukazují na to, že né všechny principy architektury RISC jsou úplně dodrženy. Typickým rysem procesorů s architekturou RISC je to, že přístup do paměti je realizován výhradně pomocí instrukcí LOAD a STORE. Pro názornost si uvedeme příklad, jehož cílem je vymazat dva bity registru TACTL (mapován do hlavní paměti) [4].

V procesoru RISC, kde je tento princip přesně dodržován, by byla tato operace realizována pomocí sekvence instrukcí, které jsou na obrázku 2.2.

```
load.w  #TACTL,R4          ; load address of TACTL   [2w, 2c]
load.w  @R4,R5             ; load value of TACTL     [1w, 2c]
load.w  #MC0|MC1,R6        ; load immediate operand  [2w, 2c]
bic.w   R6,R5              ; perform operation       [1w, 1c]
store.w R5,@R4             ; store result for TACTL  [1w, 2c]
```

Obrázek 2.2: Příklad využití instrukční sady RISC procesorů [4]

První instrukce načte adresu registru TACTL do registru R4. Následně je vyčtena hodnota z hlavní paměti z adresy uložené v registru R4 do registru R5. Následuje načtení operandů a modifikace načteného slova pomocí dvou instrukcí. Poslední instrukce uloží výsledek z registru R5 zpět do paměti na adresu, která je uložena v registru R4. Doba vykonání této posloupnosti instrukcí je devět hodinových cyklů [4].

S přihlédnutím k faktu, že operace tohoto typu jsou u mikrokontrolérů velmi časté, se vývojáři mikrokontrolérů MSP430 rozhodli nedodržet tento charakteristický rys architektury RISC. Namísto toho umožnili přímo adresovat místo v paměti. Díky tomu je možné provést stejnou operaci pomocí jedné instrukce, která je znázorněna na obrázku 2.3. Doba vykonání operace díky tomu klesne na pět hodinových cyklů [4, 2].

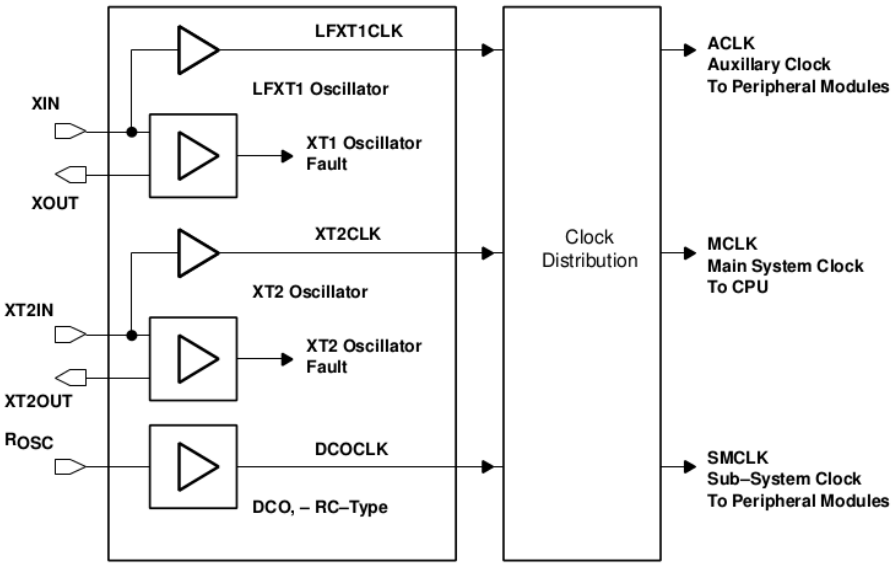
```
bic.w   #MC0|MC1,&TACTL ; stop timer [3 words, 5 cycles]
```

Obrázek 2.3: Příklad využití instrukční sady procesoru mikrokontroléru MSP430 [4]

2.1.2 Modul BCM[10]

Jedním z typů hodinových modulů, se kterým se lze u mikrokontrolérů z rodiny MSP430 setkat, je modul BCM. Jeho úlohou je generovat a distribuovat hodinové signály do zbytku mikrokontrolérů. Konkrétně je tento typ modulu použit i u mikrokontrolérů z řadu MSP430x1xx. Blokové schéma tohoto modulu je na obrázku 2.4.

[9] RISC (Reduced Instruction Set Computing)
[10] BCM (Basic Clock Module)



Obrázek 2.4: Blokové schéma modulu BCM [19]

Hodinové signály

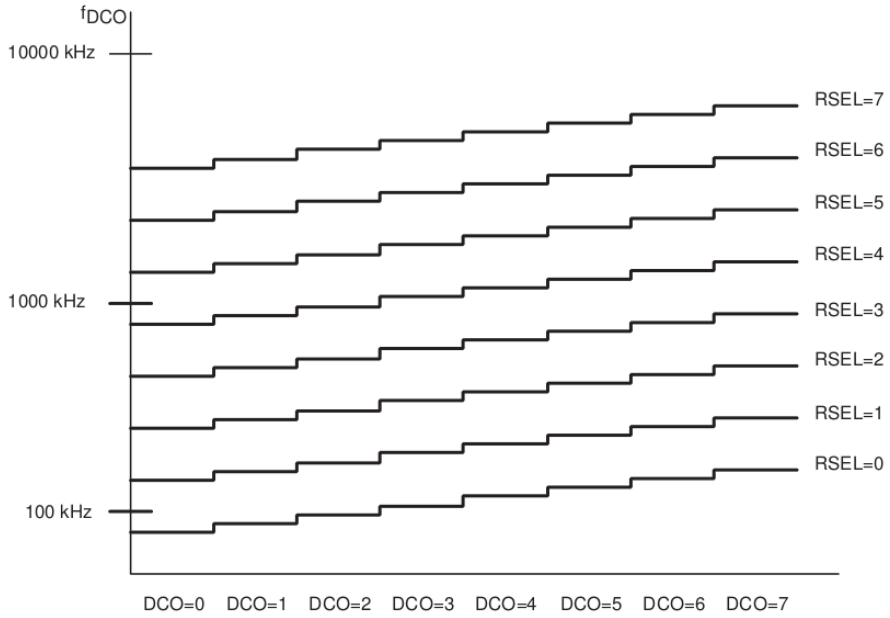
Výstupem modulu jsou hodinové signály MCLK[11], SMCLK[12] a ACLK[13]. Hodinový signál MCLK je téměř výhradně určen pro CPU. Signály ALCK a SMCLK jsou naopak určeny pro jednotlivé moduly mikrokontroléru. Obvykle platí, že frekvence signálů MCLK a SMCLK je v řádu jednotek MHz, zatímco frekvence signálů ACLK bývá v desítkách až stovkách kHz. Tato hodnota je dána hodnotami taktovacích frekvencí většiny modulů, kterými mikrokontroléry z rodiny MSP430 mohou disponovat. Výjimku tvoří třeba modul ADC12, který jako jeden z mála může být taktován hodinovým signálem s frekvencí v řádu jednotek MHz [4].

Oscilátory

Zdrojem jednotlivých hodinových signálů jsou oscilátory. Jejich počet a typ vždy závisí na konkrétním mikrokontroléru. Celkem se můžeme setkat až se čtyřmi oscilátory, přičemž mikrokontroléry řady MSP430x1xx využívající modulu BCM mívají dva až tři oscilátory. Konkrétně se jedná o oscilátory:

DCO[14]: RC oscilátor, který se nachází u všech mikrokontrolérů z rodiny MSP430 bez ohledu na konkrétní typ hodinového modulu. Jeho největší předností je velice krátká doba ustálení frekvence a cena v porovnání s krystalovými oscilátory. Například u řady MSP430x1xx dojde k ustálení frekvence do jedné mikro sekundy. Výstupní frekvence oscilátoru f_{DCO} je nastavována čistě softwarově pomocí bitů DCOx a registru DCOCTL a bitů RSELx z registru BCSCTL2. Závislot hodnoty těchnto bitů na výstupní frekvenci oscilátoru můžeme vidět na obrázku 2.5. Nevýhodou tohoto oscilátoru je jeho menší přesnost v porovnání s krystalovými oscilátory. Ta je přímo úměrná napájecímu napětí, zvolené výstupní frekvenci a také teplotě okolí.

[11] MCLK (Main System Clock)
[12] SMCLK (Sub-system Clock)
[13] ACLK (Auxillary Clock)
[14] DCO (Digitally Controlled Oscillator)



Obrázek 2.5: Závislot hodnot bitů DCOx a RSELx a frekvence f_{DCO} [20]

LFXT1[15]**:** Krystalový oscilátor, který vyžaduje připojení externího krystalu na piny XIN a XOUT. Oscilátor pracuje vždy v jednom ze dvou režimů. Prvním je nízko frekvenční režim, kdy je nutné k oscilátoru připojit krystal o frekvenci 32 kHz. Žádné další komponenty není nutné na vstup oscilátoru připojovat. V případě vysokofrekvenčního režimu (450 kHz až 8 MHz) je nutné připojit kromě příslušného krystalu i patřičně velké kondenzátory. Ty se připojují mezi krystal a oba vstupní piny oscilátoru (XIN a XOUT), přičemž jejich velikosti je nutné dohledat v dokumentaci. Doba ustálení frekvence oscilátoru může dosahovat až stovek milisekund. Důležitým faktem je to, že mikrokontrolér v tuto dobu nemůže vykonávat žádnou užitečnou činnost. Tuto skutečnost je nutné zohlednit v případě, kdy je využito některého z úsporných režimů, který pozastavuje činnost tohoto oscilátoru.

XT2[16]**:** Opět krystalový oscilátor, který je dostupný jen v některých zařízeních. Zařízení, která nedisponují tímto typem oscilátoru, suplují jeho výstup výstupem oscilátoru LFXT1. Princip činnosti a využití tohoto oscilátoru je shodný s oscilátorem LFXT1 ve vysokofrekvenčním režimu [4, 19].

Jednotlivé oscilátory mohou být pomocí několika bitů, které ovládají příslušné multiplexory v obvodu, mapovány na jednotlivé hodinové signály. Kromě samotného mapování je zde také několik děliček hodinového signálu, přes které prochází jednotlivé hodinové signály uvnitř modulu. Díky tomu je například možné použít jeden oscilátor jako zdroj více hodinových signálů, kdy každý hodinový signál může mít odlišnou frekvenci. K nastavení těchto a mnoha dalších funkcí modulu slouží registry BCSCTL1, BCSCTL2 a IE1 [19].

[15] LFXT1 (Low- or High-frequency Crystal Oscillator)
[16] XT2 (High-frequency Crystal Oscillator)

2.1.3 Modul DMA[17]

Modul DMA slouží k přenosu dat mezi dvěma adresami bez účasti CPU. Příkladem užití je uložení výstupu analogově digitálního převodníku do paměti RAM. Výhodou využití tohoto modulu je zvýšení propustnosti jednotlivých periférií (zkrácení doby přenosu vstupních a výstupních dat) a teoretické snížení spotřeby elektrické energie. K reálnému snížení spotřeby může dojít pouze za předpokladu, že se mikrokontrolér nachází v jednom z úsporných režimů, který by opustil pouze z důvodu provedení paměťové operace. Namísto toho lze tuto činnost vykonat pomocí řadiče DMA, přičemž zdroj hodinového signálu tohoto modulu musí být po celou dobu ponechán aktivní.

V závislosti na konkrétním zařízení se lze setkat s různými verzemi modulu DMA, které se od sebe liší například v množství kanálů. Princip činnosti je ovšem shodný. Tento text je věnován verzi, se kterou se lze setkat například u zařízení MSP430F1611 [20].

Architektura

Modul DMA je tvořen řadičem, který umožňuje využití až tří zcela nezávislých kanálů. Každý kanál může pracovat v jednom ze čtyř adresovacích a v jednom ze šesti přenosových režimů. Priorita jednotlivých kanálů je konfigurovatelná a jejich činnost je povolována pomocí řídícího bitu DMAEN.

K nastavení modulu slouží kontrolní registry DMACTL0 a DMACTL1. Dále je modul vybaven čtyřmi registry pro každý kanál, které slouží k nastavení zdrojové adresy (DMAxSA), cílové adresy (DMAxDA), velikosti bloku (DMAxSZ) a režimu činnosti (DMAxCTL) [20].

Režimy adresování

Pomocí každého kanálu je možné provádět přenos *bajtu na bajt*, *slova na slovo*, *bajtu na slovo* nebo *slova na bajt*. V případě přenosu *slova na bajt* je přenesen pouze spodní bajt slova. Pro přenos *bajtu na slovo* platí, že zdrojový bajt je umístěn do spodní poloviny slova a horní polovina slova je automaticky nulována.

K určení velikosti zdroje a cíle slouží bity DMASRCBYTE a DMADSTBYTE kontrolního registru DMAxCTL. Každý přenos probíhá v rámci jednoho z dostupných adresovacích režimů:

- pevná adresa na pevnou adresu
- pevná adresa na blok adres
- blok adres na pevnou adresu
- blok adres na blok adres

Ten určuje rozsah zdrojových a cílových adres a velikost bloku (množství dat). Velikost bloku je určena hodnotou registru DMAxSZ. Počáteční zdrojová a cílová adresa je určena již zmíněnými registry DMAxSA a DMAxDA. U obou adres je možné nastavit automatickou inkrementaci/dekrementaci pomocí bitů DMASRCINCRx (zdrojová adresa) a DMADSTINCRx (cílová adresa) z řídícího registru kanálu [20].

[17] DMA (Direct Memory Access)

Režimy přenosu

Každý kanál může pracovat nezávisle na adresovacím režimu v jednom ze šesti níže popsaných režimů přenosu, který se nastavuje bity DMADTx řídícího registru kanálu. Na začátku každého přenosu jsou hodnoty registrů DMAxSA a DMAxDA uloženy do zvláštních regitrů, odkud jsou na konci přenosu obnoveny. Dostupné režimy přenosu jsou:

jeden přenos : Přenos každého bajtu/slova je zahajován zvlášť. Hodnota registru DMAxSZ udává počet bajtů/slov, které se mají přenést. S každým přeneseným bajtem/slovem je obsah registru DMAxSZ dekrementován. Při nulové hodnotě registru nebude zahájen žádný přenos. Po dokončení přenosu (hodnota registru DMAxSZ byla dekrementována na nulu) dojde k vymazání bitu DMAEN a nastavení příslušného příznaku DMAIFG. Přenos každého bajtu/slova trvá dva takty hodinového signálu MCLK.

jeden opakovaný přenos : Stejně jako předchozí případ s výjimkou toho, že bit DMAEN zůstává nastaven i po dokončení přenosu.

blokový přenos : Přenos celého bloku o velikosti dané registrem DMAxSZ stačí zahájit pouze jednou. Obsah registru DMAxSZ je opět po každém přeneseném bajtu/slově dekrementován, přičemž jeho aktuální hodnota udává počet zbývajících bajtů/slov v bloku. Po dokončení přenosu celého bloku dojde k vymazání bitu DMAEN a nastavení příslušného bitu DMAIFG. Při nulové hodnotě registru DMAxSZ a zahájení přenosu nebude žádný přenos realizován. Přenos jednoho bajtu/slova zabere dva takty hodinového signálu MCLK. Přenos celého bloku o velikosti DMAxSZ tedy zabere 2×DMAXSZ taktů signálu MCLK. Po celou dobu přenosu je činnost CPU pozastavena.

opakovaný blokový přenos : Opět stejné jako předchozí případ s výjimkou ponechání hodnoty bitu DMAEN i po skončení přenosu.

dávkový blokový přenos : Jedná se o modifikaci *blokového přenosu*. Rozdíl mezi variantami spočívá ve způsobu odesílání dat. Tento režim pracuje tak, že odešle první čtyři bajty/slova bloku (tzv. dávku) a poté povolí činnost procesoru na dva takty hodinového signálu MCLK. Po vypršení této doby je odeslána další dávka. Tento cyklus trvá do doby, než dojde k odeslání celého bloku. Po dokončení odesílání bloku je opět povolena činnost CPU, vymazán bit DMAEN a nastaven příslušný příznak DMAIFG.

opakovaný dávkový blokový přenos : Modifikace předchozího režimu, kdy bit DMAEN zůstává nastaven i po dokončení přenosu bloku. K zahájení dalšího dávkového blokového přenosu není nutný žádný externí podnět. Přenos začne neprodleně po dokončení předchozího přenosu. Ukončení opakování je možné pouze manuálním vymazáním bitu DMAEN nebo pomocí nemaskovatelného přerušení (NMI) [20].

K získání celkového počty taktů hodinového signálu MCLK na realizaci datového přenosu je nutné k výše uvedeným časům ještě přičíst jeden až dva takty pro synchronizaci před zahájením přenosu a jeden extra takt pro ukončení přenosu [20].

Zahájení přenosu

Každý datový přenos je zahájen v reakci na nějaký podnět/zdroj. Ten je nastavován pro každý kanál modulu zvlášť pomocí bitů DMAxTSELx registru DMACTL0. Tento registr je

možný modifikovat pouze pokud není nastaven příslušný bit DMAEN. V opačném případě je chování modulu nepředvídatelné. Příkladem možného zdroje, který lze namapovat na jeden z kanálů, je příznak přerušení časovače, analogově digitálního převodníku nebo příznak přerušení jednoho ze zbývajících kanálů modulu. Kromě samotného zdroje je možné také určit pomocí bitu DMALEVEL registru DMAxCTL, zda má být kanál citlivý na nástupnou hranu nebo úroveň zdrojového signálu [20].

Priorita kanálů

Každý kanál má svou prioritu, která slouží pro určení pořadí vykonávání jednotlivých datových přenosů při současném splnění dvou a více podmínek zahájení přenosu. Zároveň však platí, že přenos jednoho kanálu nemůže být přerušen ani výskytem podnětu pro zahájení datového přenosu kanálu s vyšší prioritou.

Ve výchozím nastavení jsou priority jednotlivých kanálů definovány tak, že kanál nula má nejvyšší a kanál tři naopak nejnižší prioritu. Alternativně lze nastavit určování priorit pomocí bitu ROUNDROBIN z registru DMACTL1 tak, že kanál, jehož datový přenos byl právě ukončen, dostává nejnižší prioritu [20].

Přerušení

Přenos, který je realizován pomocí tohoto modulu, není možné ukončit pomocí systémového přerušení. Veškeré žádosti o přerušení, které vznikly v době přenosu, budou obslouženy na základě priorit až po dokončení přenosu.

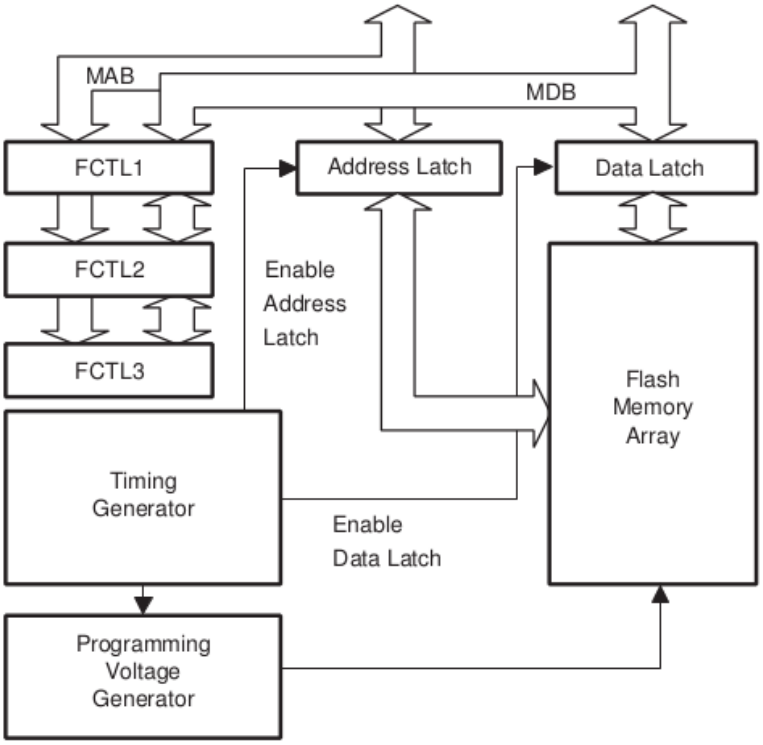
V souvislosti s režimy přenosu již byly zmíněny příznaky DMAIFG, kdy každý kanál vlastní právě jeden tento příznak. Ten je uložen v registru DMAxCTL spolu s bitem DMAIE, který slouží pro povolení přerušení. V případě nastavení obou těchto bitů a současném nastavení bitu GIE[18] dojde k vyvolání přerušení, přičemž přerušovací vektor je pro všechny kanály společný. Je tedy nutné v rámci obslužné rutiny přerušení určit zdroj přerušení. Bity DMAIFG je nutné vymazat ručně v rámci obslužné rutiny [20].

2.1.4 Vnitřní paměť Flash

V úvodu této části již bylo řečeno, že konkrétní typ hlavní paměti vždy záleží na konkrétní řadě mikrokontroléru. Tento text se věnuje paměťovému modulu s pamětí typu Flash, který lze nalézt například u mikrokontroléru z rodiny MSP430x1xx. Obecně lze říci, že architektura modulu je pro všechny mikrokontroléry s pamětí typu Flash velmi podobná a liší se například množstvím registrů. Princip činnosti je ovšem totožný. Výhodou, která pramení z architektury mikrokontrolérů, je možnost přeprogramovat mikrokontrolér za běhu [20].

Modul je tvořen nevolatilní pamětí typu Flash, která v tomto konkrétním případě umožňuje adresování jednotlivých bitů, bajtů nebo slov. Dále obsahuje integrovaný řadič, který realizuje jednotlivé paměťové operace. Samotný řadič dále obsahuje tři šestnáctibitové registry FCTLx, generátor hodinového signálu (Timing Generator) a generátor napětí (Programming Voltage Generator). Registry FCTLx jsou z bezpečnostních důvodů chráněny heslem. Každý přístup do nich tedy musí začít zadáním hesla FWKEY. Neoprávněný přístup k těmto registrům vyvolá softwarový reset zařízení a nastavení příznaku KEYV z registru FCTL3. Blokové schéma modulu mikrokontrolérů řady MSP430x1xx je na obrázku 2.6.

[18] GIE (General Interrupt Enable)



Obrázek 2.6: Blokové schéma paměťového modulu mikrokontrolérů řady MSP430x1xx [20]

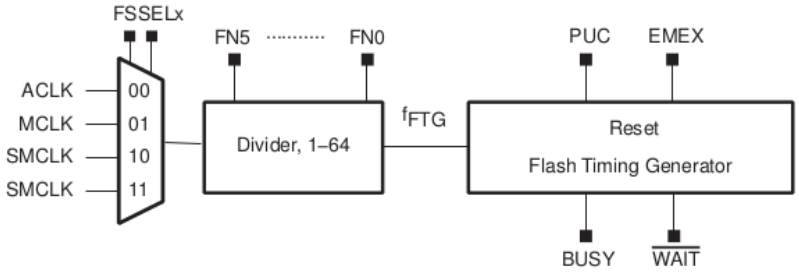
Rozdělení paměti

Samotná paměť je rozdělena na hlavní a informační část. Tyto části jsou dále členěny na segmenty, jejichž velikost je v rámci jedné části uniformní. U informační paměti je velikost segmentu 128 bajtů a u hlavní paměti je velikost segmentu 512 bajtů. Segment je zároveň nejmenší možný kus paměti, který lze u MSP430 vymazat. Zápis je však možný po bitech, bajtech nebo slovech. Každý segment je dále rozdělen na bloky, jejichž velikost je 64 bajtů [20].

Generátor hodinového signálu

Operace zápisu a mazání paměti, které budou představeny níže, vyžadují zdroj hodinového signálu o frekvenci 257 kHz až 476 kHz. Přesná hodnota vždy závisí na konkrétním mikrokontroléru. Řadič paměti je proto vybaven modulem, který je svou strukturou velmi podobný části hodinového modulu BCM. Blokové schéma generátoru je na obrázku 2.7.

Z obrázku je patrné, že generátor neobsahuje žádný oscilátor. Namísto toho obsahuje obvody pro mapování a úpravu frekvence externího hodinového signálu. Konkrétně se jedná o multiplexor (bity FSSELx registru FCTL2) a děličku hodinového signálu (bity FNx téhož registru). Díky tomu lze jako zdroj hodinového signálu vybrat jeden ze tří hodinových signálu mikrokontroléru, jehož frekvenci je možné upravit až uvnitř paměťového modulu. Při nedodržení požadované frekvence není zaručena správnost provedení a spolehlivost paměťových operací [20].



Obrázek 2.7: Blokové schéma generátoru hodinového signálu paměťového modulu [20]

Princip činnosti

Standardně se paměť chová jako paměť typu ROM. Generátory hodinového signálu a napětí jsou v tuto dobu neaktivní, což přispívá k nižší spotřebě elektrické energie mikrokontroléru. Naopak v případě mazání paměti nebo zápisu do paměti řadič aktivuje oba generátory, kdy je nutné počítat s určitým zpožděním, než dojde k ustálení výstupů obou generátorů. Do této doby nelze provádět žádné paměťové operace.

Každá paměťová buňka nabývá logické hodnoty *nula* nebo *jedna*. Logickou hodnotu *nula* lze nastavit pomocí operace zápisu. Naopak nastavení logické hodnoty *jedna* je možné pouze pomocí operace mazání. Ta může být prováděna pouze po segmentech.

Z pohledu realizace těchto operací je zcela zásadní, zda je kód vykonáván z paměti Flash, nebo paměti RAM (více v části 2.1.1). V případě vykonávání kódu z paměti Flash je veškeré časování v roli řadiče, který pozastaví další vykonávání kódu po dobu, kdy je řadič zaneprázdněn (bit BUSY registru FCTL3). Druhá varianta, kdy je kód vykonáván z paměti RAM, je poněkud složitější. Veškerou zodpovědnost za právě probíhající a nastávající paměťové operace má na starosti programátor. Ten musí ručně testovat bit BUSY, kdy je nutné počkat se zahájením paměťové operace do doby, než je jeho úroveň na hodnotě logická nula. Tento krok je nutné zopakovat i před ukončením paměťové operace. Vynechání těchto kroků vede k nastavení příznaku ACCVIFG z registru FCTL3, přičemž výsledek paměťové operace by byl nepředvídatelný [20].

Čtení z paměti

Operace čtení z paměti je velmi rychlá, protože není nutné aktivovat generátory napětí a hodinového signálu. Samotná operace čtení je prováděna pouhým vyčtením hodnoty z adresy v paměti [20].

Mazání paměti

Již bylo zmíněno, že změna hodnoty paměťové buňky z logické hodnoty *nula* na logickou hodnotu *jedna* je uskutečnitelná pouze pomocí vymazání příslušného segmentu paměti. Celkem máme k dispozici tři režimy, které se liší množstvím vymazaných segmentů. K nastavení režimu slouží bity ERAS a MERAS registru FCTL1, jehož obsah může být modifikován pouze, pokud neprobíhá žádná paměťová operace (bit BUSY registru FCTL3). V závislosti na kombinaci těchto bitů je možné vymazat:

- jeden segment
- všechny segmenty hlavní paměti

- všechny segmenty hlavní a informační paměti

Samotná operace mazání paměti je realizována pomocí tzv. fiktivního zápisu (dummy write), který uvede v činnost generátory napětí a hodinového signálu. Po dobu vykonávání operace je nastaven bit BUSY, který je po dokončení automaticky vymazán [20].

Zápis do paměti

Tato operace se používá pro změnu hodnoty paměťové buňky z logické hodnoty *jedna* na logickou hodnotu *nula*. Stejně jako v případě mazání máme na výběr více režimů zápisu. Konkrétně lze pomocí bitů BLKWRT a WRT registru FCTL1 vybrat zápis jednoho bajtu/slova, nebo celého bloku dat (64 bajtů).

Při každé operaci zápisu je nutné aktivovat generátor napětí a hodinového signálu. Po dokončení zápisu jsou oba generátory automaticky deaktivovány. Při blokovém zápisu jsou jednotlivé generátory aktivovány a deaktivovány pouze jednou. Díky tomu může být blokový zápis až dvakrát rychlejší v porovnání se zápisem bloku po jednotlivých bajtech. Blokový režim je možné použít pouze, pokud je kód vykonáván z paměti RAM. Zápis jednoho bajtu trvá v obou režimech 32 hodinových cyklů. Bit BUSY je po dobu vykonávání operace opět nastaven na logickou hodnotu jedna [20].

Přerušení

V popisu modulu zazněly v různém kontextu příznaky KEYV a ACCIVG. Jedná se o jediné dva příznaky přerušení, kterými modul disponuje. První příznak je spojen s neoprávněným přístupem do registrů řadiče. Příznak ACCIVG naopak značí chybu paměťové operace.

Po nastavení příznaku KEYV okamžitě dochází k softwarovému resetu zařízení. Nastavení příznaku ACCIVG vyvolá žádost o nemaskovatelné přerušení (NMI[19]). To znamená, že přerušení bude obslouženo bez ohledu na hodnotu globální masky přerušení GIE [20].

2.1.5 Modul USART[20]

Jedná se o modul, který umožňuje realizovat komunikaci skrze rozhraní UART[21], SPI[22] a v některých případech dokonce i I^2C. U novějších řad mikrokontrolérů z rodiny MSP430 je tento modul již nahrazen modulem USCI[23], nicméně u dříve vydaných a dodnes běžně dostupných zařízení (např. MSP430x1xx) se lze s tímto modulem stále setkat. V závislosti na konkrétním typu může být mikrokontrolér vybaven jedním nebo dvěma moduly USART. Ty se poté označují jako USART0 a USART1 [20].

Konfigurace

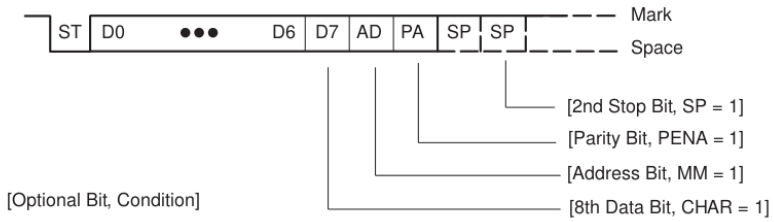
Ke konfiguraci modulu slouží několik řídících registů, jejichž obsah je resetován buď při softwarovém resetu zařízení (PUC[24]), nebo při nastavení bitu SWRST z řídícího registru UxCTL. Obecně se doporučuje provádět konfiguraci modulu jako sekvenci, která se skládá z: nastavení bitu SWRST, modifikace bitů kontrolních registrů, povolení činnosti modulu, vymazání bitu SWRST a povolení přerušení modulu [20].

[19] NMI (Non-Maskable Interrupt)
[20] USART (Universal Synchronous/Asynchronous Receiver/Transmitter)
[21] UART (Universal Asynchronous Receiver/Transmitter)
[22] SPI (Serial Peripheral Interface)
[23] USCI (Universal Serial Communication Interface)
[24] PUC (Power-up Clear)

USART v režimu UART

V tomto režimu modul poskytuje asynchronní komunikaci skrze vodiče URXD a UTXD, u které platí, že nastavená přenosová rychlost je společná pro oba směry komunikace. Možnostem nastavení přenosové rychlosti, formátu přenášených dat a způsobu využití modulu se budu věnovat dále v této části [20].

Formát dat Datový rámec přenášených dat (obrázek 2.8) je tvořen: START bitem (ST), sedmi nebo osmi datovými bity (Dx), volitelně paritním a adresním bitem a jedním nebo dvěma STOP bity [20].



Obrázek 2.8: Datový rámec modulu USART v režimu UART [20]

Přenosová rychlost Asynchronní komunikace se vyznačuje tím, že hodinový signál potřebný pro vzorkování přenosu si příjemce i odesílatel generuje sám. Pro funkční a spolehlivou komunikaci musí být frekvence hodinového signálu u obou stran shodná (určitá chyba může být přípustná). Přenosová rychlost bývá v dokumentaci vyjádřena pomocí jednotky modulační rychlosti (baund). V tomto konkrétním případě je tato hodnota rovna počtu přenesených bitů za sekundu.

Pro generování hodinového signálu o požadované frekvenci je modul vybaven frekvenční děličkou s modulátorem, která dovoluje nastavit i neceločíselný dělící poměr vstupního hodinového signálu. Jako vstupní hodinový signál modulu lze pomocí bitů SSELx z registru UxTCTL vybrat signál SMCLK, ACLK nebo UCLKI[25]. Pro nastavení dělícího poměru slouží registry UxBR0, UxBR1 a UxMCTL. Způsob výpočtu hodnot jednotlivých registrů spolu s obvyklými hodnotami lze nalézt v dokumentaci [20].

Komunikace Modul je vybaven automatickou detekcí chybného formátu datového rámce, poškození datového rámce (pouze při použití paritního bitu), přetečení (nově přijatý bajt přepsal předchozí bez předešlého vyčtení obsahu registru) a přerušení komunikace vyvoláním přerušení jiného, více prioritního zdroje. Každý z detektorů má svůj příznak (FE, PE, OE, BRK), který je po splnění daných podmínek nastaven. V případě nastavení všech příznaků najednou dojde také k nastavení příznaku RXERR. Jednotlivé příznaky jsou automaticky smazány po vyčtení přijatého bajtu z registru. Alternativně je lze vymazat softwarově i bez vyčtení obsahu příslušného registru [20].

Příjem dat : Je povolen nastavením bitu URXEx registru MEx. Po výskytu validního START bitu modul pomocí posuvného registru přijme datový rámec, který je po přijetí STOP bitu uložen do registru UxRXBUF. Volitelně může být kontrolována hodnota

[25] UCLKI (Universal Clock Input)

paritního bitu dle principu, který byl uveden výše. Po vložení přijatého datového rámce je nastaven příznak URXIFGx, který za předpokladu, že je povoleno přerušení pomocí bitu URXIEx, vyvolá žádost o přerušení. Povolovací bit URXEx zůstává nastaven [20].

Odesílání dat : Je povolováno bitem UTXEx registru MEx, přičemž samotné odeslání datového rámce je zahájeno nahráním příslušných dat do registru UxTXBUF. Jeho obsah je odeslán opět s využitím posuvného registru, přičemž po odeslání zvoleného počtu bitů je nastaven příznak UTXIFGx. Ten opět může vyvolat žádost o přerušení (na základě hodnoty bitu UTXIEx). Povolovací bit opět zůstává nastaven, což znamená, že odeslání dalšího datového rámce je automaticky zahájeno po přehrátí obsahu registru UxTXBUF [20].

USART v režimu I^2C

V úvodu této části již zaznělo, že modul USART může v určitých verzích umožňovat také komunikaci pomocí rozhraní I^2C. Konkrétně zařízení MSP430F16x jsou vybaveny dvěma moduly USART, kde modul USART0 podporuje rozhraní UART, SPI a I^2C. Druhý modul, označovaný jako USART1, podporuje pouze rozhraní UART a SPI. Tato část si klade za cíl představit vybrané možnosti modulu USART v režimu I^2C. Samotný popis rozhraní není předmětem této části. Lze ho však nalézt v části 2.3.1.

Modul v tomto režimu umožňuje sedmi i destibitové adresování, přenosovou rychlost až 400 kb/s, využití techniky opakovaného startu, šestnáctibitovou šířku datové sběrnice pro zvýšení propustnosti modulu a generování hodinového signálu o požadované frekvenci [20].

Konfigurace Režim I^2C je zvolen nastavením bitu SYNC registru U0CTL. K povolení činnosti modulu v tomto režimu je součástí registru bit I2CEN, který musí být nastaven. Naopak v době konfigurace modulu musí být tento bit vymazán. V opačném případě je chování modulu opět nepředvídatelné. Vymazání bitu I2CEN mimo vymazání některých bitů řídících registrů způsobí to, že oba vodiče sběrnice I^2C jsou přepnuty do režimu vysoké impendance.

V závislosti na hodnotě bitu MST z registru U0CTL pracuje modul jako zařízení typu *master* nebo *slave*. Směr komunikace je určen pomocí bitu I2CTRX z registru I2CTCTL [20].

Velikost odesílaných/přijímaných dat Modul umožňuje pracovat nad daty o velikosti slova nebo bajtu. Specifikace rozhraní I^2C ovšem definuje komunikaci pouze po blocích dat o velikosti jednoho bajtu. Zápis/čtení jednoho slova tedy vypadá tak, že modul nejprve odešle/uloží nižší bajt a poté vyšší bajt slova. Výhodou tohoto přístupu je redukce počtu paměťových operací mezi modulem a pamětí mikrokontroléru. Pro uložení odesílaného nebo právě přijatého bajtu/slova slouží registr I2CDRB/I2CDRW (I2CDRB označuje spodní polovinu registru I2CDRW). Velikost odesílaných/přijímaných dat je dána hodnotou bitu I2CWORD z registru I2CTCTL [20].

Komunikace v roli master Zařízení v tomto režimu zahajuje a ukončuje komunikaci. To znamená, že je mimo jiné zodpovědné za generování bitů START a STOP. Způsob realizace komunikace je dán bity I2CRM, I2CSTP a I2CSTT z registru I2CTCTL. Celkem je definováno sedm variant komunikace, jejichž popis je v tabulce 2.1.

I2CRM	IC2CSTP	IC2CSTT	Popis činnosti
0/1	0	0	Zařízení je v nečinném stavu.
0	0	1	Komunikace je zahájena neprodleně. Počet přenesených bajtů je dán obsahem registru I2CNDAT. STOP bit je nutné generovat softwarově po ukončení přenosu. Tento režim je například možné využít pro techniku tzv. *opakovaného startu*.
0	1	1	Podobné jako předchozí případ. Rozdíl spočívá pouze v automatickém generování STOP bitu.
1	0	1	Hodnota registru I2CNDAT opět určuje počet přenášených bajtů. Komunikace je zahájena neprodleně. STOP bit je nutné generovat softwarově. Tento režim se doporučuje použít pouze pro větší množství přenášených dat.
0	1	0	STOP bit je automaticky generován po odeslání I2CNDAT bajtů dat.
1	1	0	STOP bit je generován po dokončení aktuálního přenosu. Tento bit není generován v případě, že přenos již byl ukončen jiným STOP bitem.
1	1	1	Na sběrnici neprobíhá žádná aktivita.

Tabulka 2.1: Chování modulu v režimu I^2C v roli *master*

V části 2.3.1 je mimo jiné popsán způsob detekce kolizí, které mohou vzniknout tak, že dvě a více zařízení v jeden okamžik zahájí komunikaci. V tomto případě je programátor informován o ztrátě sběrnice prostřednictvím příznaku přerušení ALIFG z registru I2CIFG. Po nastavení tohoto bitu dojde k vymazání bitů MST a I2CSTP, čímž se ze zařízení typu *master* stává zařízení typu *slave*. Dalším příznakem, který se v tomto režimu využívá, je bit NACKIFG. Ten je nastaven v situaci, kdy nedojde k potvrzení odeslaných dat stranou příjemce. Dalším příznakem přerušení je bit ARDYIFG, který značí, že došlo k odeslání/přijetí všech bajtů a vygenerování STOP bitu (pro I2CRM = 1) [20].

Komunikace v roli slave Veškeré operace jsou řízeny automaticky samotným modulem. S každým taktem hodinového signálu je nasunut do/z posuvného registru právě jeden bit. Zařízení v tomto režimu může po určitý čas tzv. podržet hodnotu hodinového signálu na logické úrovni *nula*. Tím dojde k pozastavení komunikace na určitou dobu, během které může být např. vyčten obsah registru I2CDRB/I2CDRW [20].

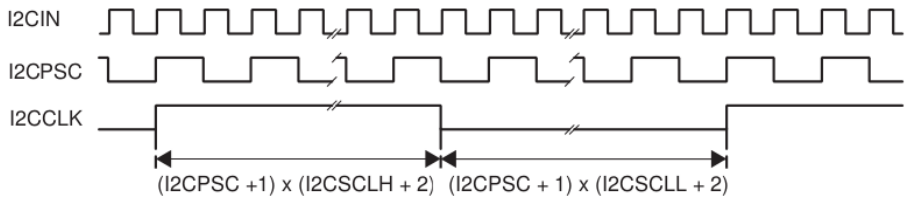
Generování hodinového signálu Pokud je modul v roli *master*, je nutné generovat hodinový signál o dané frekvenci. Samotný hodinový signál opět není generován uvnitř modulu. Namísto toho je zde pouze upravován vstupní hodinový signál (I2CIN) modulu pomocí frekvenční děličky a dalších podpůrných obvodů na výstupní hodinový signál I2CCLK.

Pomocí bitů I2CSSELx z registru I2CTCTL je jako vstupní hodinový signál I2CIN vybrán hodinový signál ACLK nebo SMCLK. Tento signál prochází skrze frekvenční děličku, jejíž dělící poměr je dán obsahem registru I2CPSC. Výstupním hodinovým signálem frekvenční děličky je signál označovaný jako I2CPSC. Délka periody výstupního hodinového

signálu I2CCLK je dána vztahem:

$$(I2CPSC + 1) \times (I2CSCLH + 2) + (I2CPSC + 1) \times (I2CSCLL + 2) \tag{2.1}$$

kdy první část výrazu označuje dobu, kdy je hodinový signál v logické úrovni *jedna*. Druhá část vztahu naopak určuje dobu, po kterou je hodinový signál v logické úrovni *nula*. Výše popsaná závislot jednotlivých hodinových signálů je ukázána na obrázku 2.9.



Obrázek 2.9: Závislost frekvencí hodinových signálů I2CIN a I2CCLK [20]

Přerušení Kromě již zmíněných příznaků přerušení je modul dále vybaven příznaky RXRDYIFG a TXRDYIFG. Oba tyto příznaky jsou spojeny s registrem I2CDRB/I2CDRW. Příznak TXRDYIFG může být nastaven ve dvou konkrétních případech. Za prvé pokud je zařízení v roli *master* připraveno odeslat nová data a za druhé pokud je zařízení v roli *slave* a jsou po něm požadována nová data. Tento příznak je automaticky vymazán po nahrání požadovaných dat do registru I2CDRW. Příznak RXRDYIFG je naopak nastaven pokaždé, když jsou do registru I2CDRW vložena nově přijatá data. Po vyčtení obsahu registru je příznak automaticky vymazán.

Vzhledem k množství příznaků přerušení a faktu, že všechny sdílí jeden vektor přerušení, je modul vybaven speciálním registrem I2CIV. Modul USART v tomto režimu totiž poskytuje mechanismus, který na základě předem daných priorit jednotlivých příznaků přerušení a aktuálního stavu hodnot jednotlivých příznaků určuje zdroj přerušení s nejvyšší prioritou. A právě index tohoto přerušení je ukládán do registru I2CIV. Každý přístup do tohoto registru způsobí vymazání příznaku, jehož index odpovídá hodnotě tohoto registru [20].

2.1.6 Moduly Timer_A a Timer_B

Jedněmi z nejuniverzálnějších modulů, se kterými se lze u mikrokontrolérů MSP430 setkat, jsou časovače Timer_A a Timer_B. Princip činnosti obou modulů je až na pár drobností zcela totožný. Právě proto bude tato část věnována pouze popisu modulu Timer_A s tím, že na konci bude uveden výčet odlišností tohoto modulu oproti modulu Timer_B [20].

Architektura

Základem modulu je šestnáctibitový registr TAR, jehož hodnota je inkrementována/dekrementována s každou náběžnou hranou hodinového signálu modulu. O zdroji a způsobu generování tohoto signálu bude řečeno více v další části.

Hodnota registru TAR může být softwarově vyčtena nebo přepsána. V závislosti na konkrétní verzi modulu má každý časovač několik kanálů, které fungují v režimu *porovnání* nebo *záchytu hrany vstupního signálu* (tzv. compare a capture channel). Pro konfiguraci

modulu slouží registry: TACTL, TAIV a dále registry TACCTLx a TACCRx pro každý z kanálů modulu [20].

Generátor hodinového signálu

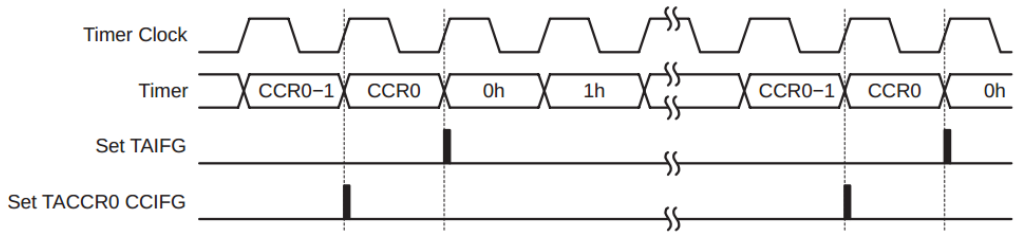
Vnitřní hodinový signál je generován uvnitř modulu z existujícího externího hodinového signálu pomocí frekvenční děličky. Konkrétně lze pomocí bitů IDx registru TACTL vybrat dělící poměr 1:1, 1:2, 1:4 nebo 1:8. Vstupním hodinovým signálem je v závislosti na hodnotách bitů TASSELx registru TASSELx hodinový signál ACLK, SMCLK, TACLK nebo INCLK. První dva zmíněné hodinové signály již známe z části 2.1.2. Zbylé dva jsou externími signály mikrokontroléru [20].

Režimy činnosti

Bity MCx registru TACTL je vybrán jeden ze čtyř následujících režimů činnosti časovače:

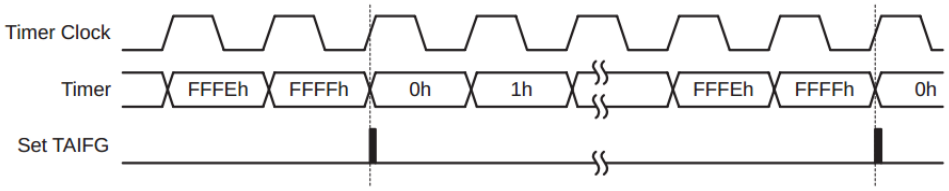
Stop: Činnost časovače je pozastavena.

Čítání nahoru: Po nastavení tohoto režimu je hodnota registru TAR automaticky vynulována. S každým hodinovým taktem je jeho hodnota inkrementována a porovnána s hodnotou registru TACCR0. Při změně hodnoty registru TAR z hodnoty TACCR0 - 1 na hodnotu TACCR0 je nastaven příznak CCIFG kanálu nula. V dalším taktu vnitřního hodinového signálu je hodnota registru TAR vynulována a zároveň je nastaven příznak přerušení TAIFG. Po vynulování registru TAR je automaticky zahájeno další čítání. V případě využití tohoto režimu nelze kanál nula využít k jiným účelům. Graficky je tento režim zobrazen na obrázku 2.10.



Obrázek 2.10: Časový diagram modulu Timer_A v režimu čítání nahoru [20]

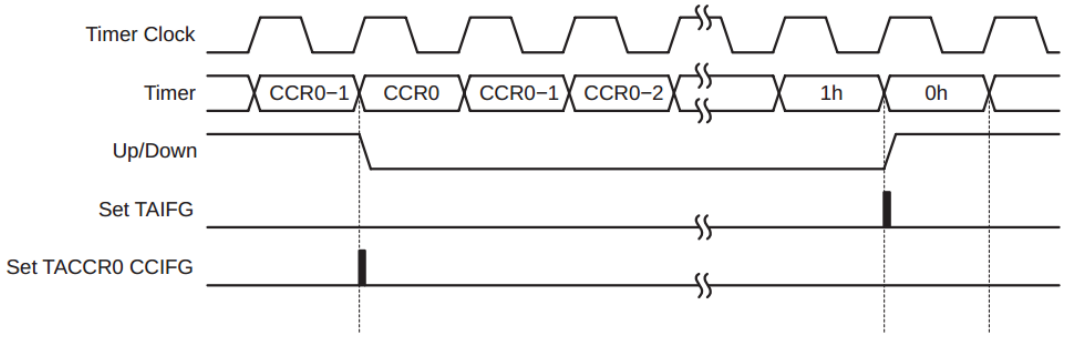
Nepřetržité čítání: Tento režim je velmi podobný tomu předchozímu. Rozdíl spočívá v určení maximální hodnoty registru TAR. Zatímco v předchozím režimu byl určen hodnotou registru TACCR0, zde je určen pouze velikostí registru TAR. To znamená, že je jeho hodnota inkrementována až do hodnoty 0xFFFF. Opět je nastaven příznak TAIFG při změně hodnoty registru TAR z maximální hodnoty na hodnotu nula. Výhodou tohoto režimu je, že kanál nula zůstává nevyužitý. Nevýhoda spočívá v přímé závislosti velikosti periody na frekvenci vstupního hodinového signálu modulu. Odpovídající časový diagram je na obrázku 2.11.



Obrázek 2.11: Časový diagram modulu Timer_A v režimu nepřetržitelného čítání [20]

Čítání nahoru/dolů: Časovač opět využívá kanálu nula pro určení maximální hodnoty registru TAR. Hodnota registru TAR je nejprve inkrementována až na hodnotu registru TACCR0 a poté dekrementována zpět na hodnotu nula. Perioda časovače je tedy dána dvojnásobkem hodnoty registru TACCR0. Příznak CCIFG je nastaven v případě změny hodnoty registru TAR z hodnoty TACCR0 - 1 na hodnotu TACCR0. Druhý příznak TAIFG je naopak nastaven při čítání na hodnotu 0. Toto chování je opět ilustrováno pomocí časového diagramu na obrázku 2.12.

Na rozdíl od předchozích režimů zde neplatí, že při nastavení hodnoty registru TACCR0 a současně běžícím časovači dojde vždy k vynulování obsahu registru TAR. Při dekrementování obsahu registru TAR a současném přepsání hodnoty registru TACCR0 čítač nejprve dokončí dekrementování obsahu registru TAR a až poté zahájí nové čístání s modifikovanou hodnotou registru TACCR0. Při modifikaci obsahu registru TACCR0 v době inkrementování obsahu registru TAR dojde k restartu čítače stejně jako u ostatních režimů [20].



Obrázek 2.12: Časový diagram modulu Timer_A v režimu čítání nahoru/dolů [20]

Kanály časovače

V úvodu této části již zaznělo, že každý kanál časovače může pracovat v jednom ze dvou režimů činnosti. Konkrétní počet takovýchto kanálů vždy závisí na konkrétní verzi mikrokontroléru. Každý kanál disponuje již zmíněnou dvojicí registru TACCTLx a TACCRx, kdy symbol x označuje číslo kanálu (počítáno od nuly). Konkrétní režim činnosti kanálu je dán hodnotou bitu CAP registru TACCTLx [20].

Režim porovnání: Tento režim je použit pro generování signálu s přesně danou šířkou pulzu. Příkladem je generování PWM[26] signálu. Činnost kanálu spočívá v porovnávání hodnot registrů TAR a TACCRx v každém hodinovém taktu, přičemž při shodě hodnot

[26]PWM (Pulse Width Modulation)

obou registrů dojde k: nastavení příslušného příznaku CCIFG, nastavení vnitřního signálu EQUx a namapování signálu CCI na signál SCCI. Význam signálu EQUx bude vysvětlen v samostatné části [20].

Režim záchytu hrany: Režim záchytu hrany je naopak používán pro časové měření výskytů událostí. Událostí rozumíme náběžnou/sestupnou hranu signálu, který je namapován na příslušný kanál modulu. Výběr tohoto signálu je proveden pomocí bitů CCISx registru TACCTLx, přičemž možnými zdroji jsou externí signály CCIxA a CCIxB nebo signály napájecího napětí V_{CC} a GND. Aktuální hodnotu sledovaného signálu lze asynchronně vyčíst ze signálu CCI příslušného kanálu.

Vzhledem k faktu, že hodinový signál mikrokontroléru může být odlišný vůči vnitřnímu hodinovému signálu modulu, je nutné počítat s možností, že výskyt události může být asynchronní vůči hodinovému signálu mikrokontroléru. Přesněji řečeno může dojít k asynchronnímu nastavení určitého příznaku přerušení. Z důvodu bezpečnosti je vhodnější pomocí bitu SCS příslušného registru TACCTLx explicitně synchronizovat nastavení jednotlivých příznaků přerušení.

Samotný princip měření časového úseku mezi jednotlivými událostmi je založen na tom, že při každém výskytu události je automaticky zkopírována aktuální hodnota registru TAR do registru TACCRx. Zároveň je nastaven příslušný příznak CCIFG. Pro určení periody výskytu jednotlivých událostí poté stačí porovnat příslušné hodnoty registrů TACCRx. Při nahrazení obsahu registru TACCRx novou hodnotou registru TAR bez předchozího vyčtení je nastaven příznak přetečení COV [20].

Výstupní jednotka

Každý kanál časovače má svou vlastní rekonfigurovatelnou výstupní jednotku, která pracuje na základě hodnot bitů OUTMODx registru TACCTLx v jednom z osmi režimů činnosti. Výstupem této jednotky je signál OUTx, který je generován na základě hodnoty signálu EQUx. Jedním z dostupných režimů je například tzv. *přepínací režim*. Ten je definován tak, že hodnota signálu OUTx je invertována pokaždé, když dojde k nastavení příznaku CCIFG příslušného kanálu v režimu porovnání. Konkrétní chování jednotlivých režimů lze nalézt v dokumentaci [20].

Přerušení

Pro modul Timer_A jsou vyhrazeny celkem dva vektory přerušení. První vektor slouží výhradně pro příznak CCIFG kanálu *nula*. Tento zdroj přerušení má v rámci modulu nejvyšší prioritu. Druhý vektor je společný pro příznak TAIV a zároveň příznaky všech ostatních kanálů. Pro vyvolání obslužné rutiny přerušení musí být kromě globální masky přerušení nastaven i příslušný bit CCIE registru TACCRx [20].

Odlišnosti modulů Timer_A a Timer_B

Předchozí popis se vztahoval výhradně k modulu Timer_A. Vyjma lehce odlišných názvů registrů a následujících odlišností ho však lze považovat i za popis modulu Timer_B [20].

- Velikost registru TAR (resp. TBR u modulu Timer_B) je volitelně 8, 10, 12 nebo 16 bitů.
- Registry jednotlivých kanálů TBCCRx jsou dvakrát bufferované.

- Jednotlivé výstupy kanálů modulu Timer_B mohou být přepnuty do stavu vysoké impendance.
- Funkce signálu SCCI není implementována.

2.2 LCD[27] displeje

Zkratka LCD slouží k označení plochých displejů, jejichž zobrazovací plochu tvoří pole tekutých krystalů. Pojem tekuté krystaly označuje materiál, který vlivem elektrického napětí mění svou molekulární strukturu, přičemž právě tyto struktury přímo ovlivňují množství světla, které prochází jednotlivými krystaly. LCD displeje dělíme na pasivní (STN[28]) a aktivní (TFT[29]).

Pasivní displeje jsou tvořeny mřížkou, ve které se každý obrazový bod (pixel) nachází na průsečíku vodorovných a svislých vodičů. Aktivací jednoho svislého a jednoho vodorovného vodiče dojde k aktivaci právě jednoho pixelu. Aktivní displeje namísto toho využívají složitější síť z tenké vrstvy tranzistorů. Aktivní displeje jsou v porovnání s pasivními díky své složitosti dražší. Naopak mají vyšší obnovovací frekvenci a větší pozorovací úhel. Zároveň dosahují ostřejšího a čistšího obrazu. Z pohledu kvality obrazu tedy aktivní displeje zcela jistě vyhrávají a právě proto je lze nalézt u většiny monitorů, televizí apod. Pasivní displeje se naopak hodí tam, kde není vyžadována vysoká kvalita obrazu [5, 36].

2.2.1 Paralelní rozhraní 6800

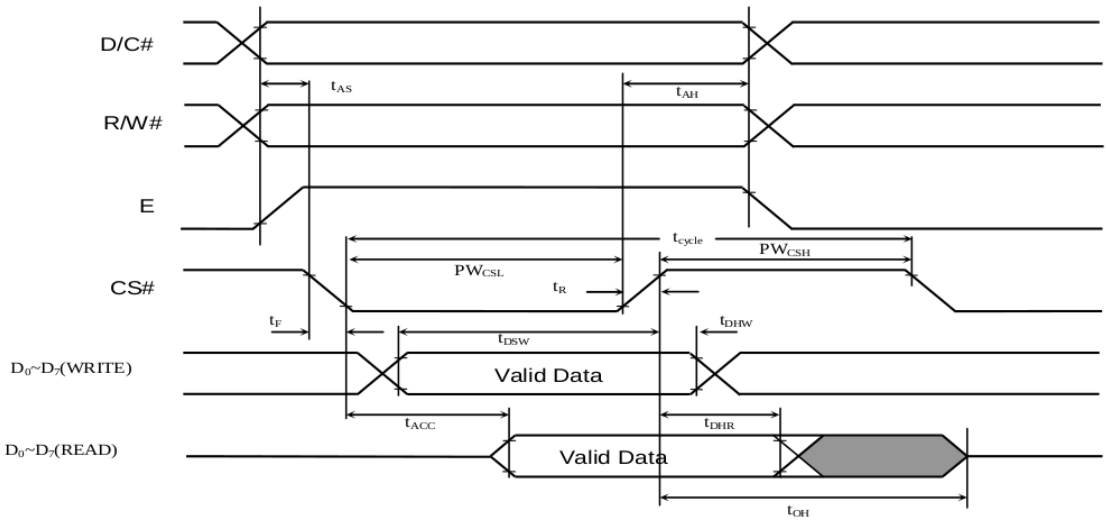
LCD displej (přesněji řečeno řadič displeje) je možné připojit ke zbytku obvodu pomocí sériového nebo paralelního rozhraní. Konkrétní rozhraní vždy závisí na použitém řadiči LCD displeje. Například řadit ST7565P od firmy Sitronix podporuje oba typy rozhraní, přičemž v rámci tohoto textu se budu nadále věnovat pouze paralelnímu rozhraní typu *6800* [18].

To je tvořeno osmi obousměrnými datovými vodiči (D7 - D0) a řídícími vodiči R/W, E, D/C a CS. Řídící signál R/W určuje směr komunikace s řadičem, přičemž logická hodnota *nula* označuje zápis a logická hodnota *jedna* označuje čtení z vnitřní paměti nebo z registrů řadiče. Signál D/C určuje, zda má být obsah pinů D7-D0 interpretován jako data nebo příkaz. Hodnota tohoto pinu je brána v potaz pouze při zápisu do displeje. Signál E slouží k zahájení činnosti hodinového obvodu displeje, přičemž aktivní je v logické hodnotě jedna. Poslední řídící signál CS, který je aktivní v logické hodnotě nula, povoluje komunikaci s řadičem. Pro funkční komunikaci je velmi důležité dodržet kromě úrovní také časování jednotlivých signálů. To lze nalézt v dokumentaci ke konkrému řadiči LCD displeje. Příklad časového diagramu paralelního rozhraní typu 6800 je na obrázku 2.13.

2.2.2 Batron BTHQ128064AVD1-COG-STF-12-LED02YG

Jedná se o pasivní monochromatický LCD displej s rozlišením 128x64 pixelů. Displej je vybaven již zmíněným řadičem ST7565P od firmy Sitronix a žlutozeleným podsvícením obrazovky. Displej je možný připojit pomocí paralelního rozhraní typu 6800 nebo 8080, přičemž obě rozhraní sdílí stejné piny. Typ rozhraní je určen logickou úrovní na jednom

[27] LCD (Liquid Crystal Display)
[28] STN (Supertwist Nematic)
[29] TFT (Thin Film Transistor)



Obrázek 2.13: Časový diagram paralelního rozhraní typu 6800 [17]

ze vstupních pinů displeje. Řadič displeje obsahuje mimo jiné obvody pro generování hodinového signálu a napětí, paměť RAM a několik registrů. Výsledný obraz je dán obsahem vnitřní paměti řadiče a jeho konkrétním nastavením. Řadič je tedy zodpovědný za interpretaci obsahu vnitřní paměti RAM a komunikaci s okolím [1].

Nastavení modulu

Veškeré nastavení displeje je realizováno pomocí předdefinovaných příkazů řadiče, které jsou zasílány skrze paralelní rozhraní řadiči. Pomocí těchto příkazů je například možné nastavit úroveň jasu, způsob adresování vnitřní paměti, způsob interpretace obsahu paměti, kurzor, režim displeje nebo vyčíst stav zařízení (aktuální nastavení) [1].

Nastavení obrazu

Obrazovka, která má rozlišení 128x64 pixelů, je rozdělena do osmi vodorovně orientovaných částí, kterým říkáme *stránky*. Každá *stránka* má tedy rozměry 128x8 pixelů. Stav každého pixelu je dán právě jedním bitem vnitřní paměti RAM a konkrétním nastavením řadiče. Pro aktivní pixel platí, že nesmí být vystaven stejnosměrnému napětí po dobu delší než třicet minut. Při nesplnění této podmínky hrozí poškození displeje. Vnitřní paměť RAM je možné adresovat pouze po bajtech, kdy každý bajt tvoří jeden sloupec jedné *stránky*.

K adresování vnitřní paměti slouží již zmíněné příkazy řadiče. Ty dovolují nastavit *řádek*, *sloupec* a *stránku*. Dále je možné nastavit automatické inkrementování čísla *sloupce* při zápisu dat do vnitřní paměti. Zápis jedné *stránky* poté může vypadat tak, že je na začátku nastavena počáteční adresa *sloupce* a *řádku*. Tuto sérii příkazů následuje 128 bajtů dat. Pro nastavení celého obsahu displeje je nutné tento krok provést osmkrát (pro každou *stránku* zvlášť).

Veškerý zápis do vnitřní paměti je realizován skrze buffery (pomocná paměť), díky čemuž nedojde ke změně obsahu vnitřní paměti v době vykreslování obrazu řadičem [18, 1].

2.3 Obvody reálného času

Obvody reálného času (RTC[30]) dovolují systémům, ve kterých jsou obsaženy, pracovat s aktuálním časem. Tyto obvody jsou obyčejně napájeny z baterií nebo superkondenzátorů, což znamená, že kromě malé velikosti pouzdra je kladen velký důraz i na spotřebu elektrické energie. V závislosti na konkrétním výrobci RTC obvodu se lze setkat s různými druhy rozhraní. V tomto textu se budu nadále soustředit pouze na sériové rozhraní I^2C, se kterým se lze velmi často setkat [34].

2.3.1 Rozhraní I^2C

Rozhraní I^2C pracuje na principu *master/slave*. Termínem *master* označujeme zařízení, které na dané sběrnici zahajuje a ukončuje komunikaci, určuje příjemce (nastaví adresu *slave* zařízení) a generuje hodinový signál. Termínem *slave* označujeme příjemce, tedy zařízení, jehož adresa odpovídá vystavené adrese.

K adresování jednotlivých zařízení na sběrnici se používá sedm (alternativně deset) bitů, přičemž šestnáct adres je rezervováno. Pro větší přehlednost textu bude dále brána v potaz pouze varianta se sedmi adresovacími bity. U té platí, že je možné připojit na sběrnici až 112 zařízení současně. V praxi je ovšem maximální možný počet zařízení většinou menší, protože je zde současně podmínka, které stanovuje maximální možnou kapacitu sběrnice (400 pF) [10].

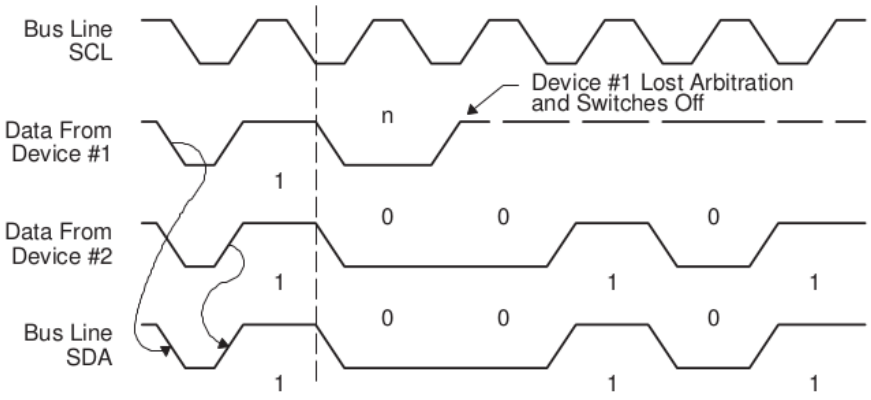
Zapojení

Jednotlivé stanice jsou připojeny ke sdílené sběrnici, která je tvořena vodiči SDA[31] a SCL[32]. Vodič SDA slouží pro přenos dat a vodič SCL pro přenos hodinového signálu, který je generován zařízením typu *master*. V tzv. *klidovém režimu* musí být na obou vodičích logická úroveň *jedna*. V době přenosu slouží vodič SDA pro přenos jednotlivých bitů, přičemž jeho úroveň může být změněna pouze, pokud je logická úroveň signálu SCL *nula*. Tato zakázaná kombinace je totiž rezervována pro zahájení a ukončení komunikace, kdy sestupná hrana signálu SDA značí tzv. START bit (zahájení komunikace) a náběžná hrana signálu značí tzv. STOP bit (ukončení komunikace). Definované přenosové rychlosti jsou 100 kb/s, 400 kb/s, 1 Mb/s a 3.4 Mb/s [15].

Komunikace

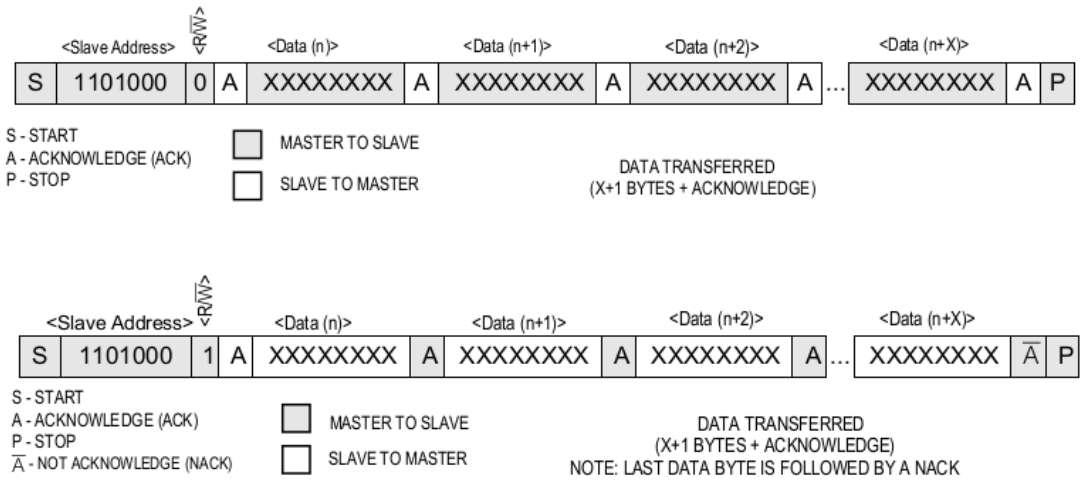
Zahájit komunikaci může kterékoliv zařízení, pokud je sběrnice v tzv. *klidovém režimu*. Ten je definován tak, že oba vodiče sběrnice mají logickou úroveň *jedna*. Každé zařízení, které je v režimu *master* (tedy probíhá komunikace), kontroluje hodnotu signálu SDA a porovnává ji s jím vysílanými daty. Díky tomu lze detekovat některé kolize, které vzniknou v případě, že se na sběrnici současně vyskytuje více zařízení typu *master* (zařízení zahájila komunikaci ve stejný čas). Zařízení typu *master*, které detekuje odlišnou hodnotu signálu SDA od očekávané hodnoty, se vzdá sběrnice a přejde do režimu *slave*. Kolizi, kdy se vyskytuje na sběrnici více zařízení typu *master*, lze detekovat pouze pokud budou jednotlivá zařízení typu *master* vysílat odlišná data. V opačném případě nebude kolize detekována. Příklad úspěšné detekce kolize můžeme vidět na obrázku 2.14.

[30] RTC (Real Time Clock)
[31] SDA (Serial Data Line)
[32] SCL (Serial Clock Line)



Obrázek 2.14: Vznik a detekce kolize na rozhraní I^{2}C [20]

Samotná komunikace, kterou iniciuje zařízení typu *master*, se vždy skládá ze START bitu a sedmi bitů adresy *slave* zařízení. Bity adresy doplňuje na velikost bajtu řídící bit $R/\overline{W}$, který určuje směr komunikace. *Slave* zařízení poté, co rozpozná svou adresu, odpoví odesláním potrzovacího bitu ACK[33] na vodič SDA. V závislosti na směru komunikace jsou následně odesílány jednotlivé bajty, přičemž každý bajt je potvrzován ze strany příjemce bitem ACK nebo NACK[34]. V obou případech je komunikace ukončena zařízením typu *master*. Způsob ukončení však závisí na směru komunikace. Při odesílání dat ze zařízení *master* je komunikace ukončena ve chvíli, kdy je vystaven STOP bit. V opačném směru je komunikace ukončena odesláním bitu NACK namísto bitu ACK za posledním bajtem, který měl být zařízením *master* přijat. Tento bit opět následuje STOP bit, který opět generuje *master*. Časové průběhy obou směrů komunikace jsou znázorněny na obrázku 2.15.

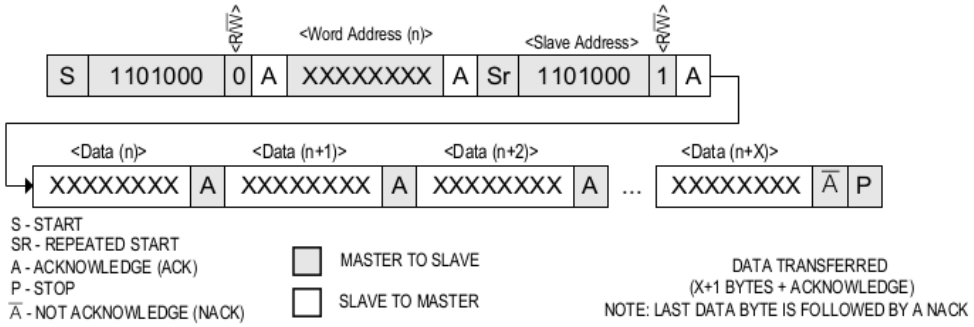


Obrázek 2.15: Ukázka komunikace skrze rozhraní I^{2}C [12]

Rozhraní I^{2}C dále poskytuje třetí typ komunikace, který vznikl kombinací obou směrů. Ten umožňuje změnu směru komunikace bez nutnosti ukončení právě probíhající komunikace. Příkladem využití může být situace, kdy zařízení typu *master* chce odeslat nějaká

[33] ACK (Acknowledged)
[34] NACK (Not Acknowledged)

data a ihned poté jiná data přijmout. Tuto situaci lze vyřešit buď pomocí dvou nezávislých komunikací, nebo pomocí techniky tzv. *opakovaného startu*. Tato technika funguje tak, že zařízení typu *master*, které odesílá nějaká data, odešle namísto STOP bitu znovu START bit. Po něm odešle opět bajt, který bude tvořen sedmi bity adresy *slave* zařízení a jedním bitem, který určí nový směr komunikace. Zbytek komunikace probíhá zcela standardně. Výhodou tohoto přístupu je to, že zařízení typu *master* nemusí znovu zabírat sběrnici. Ta mu zůstane přidělena až do doby, než komunikaci ukončí pomocí STOP bitu. Ukázka využití této technika je na obrázku 2.16.



Obrázek 2.16: Ukázka využití techniky opakovaného startu u rozhraní I^2C [12]

2.3.2 Maxim DS1338Z-33+

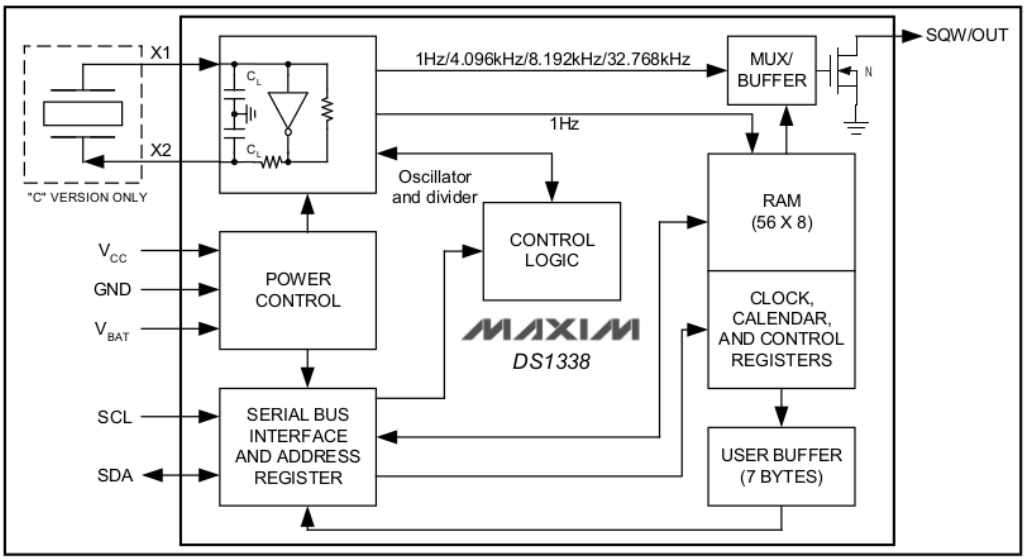
Jedná se o RTC obvod, který disponuje vnitřní pamětí RAM o velikosti 64 bajtů. Pro komunikaci s okolím je vybaven sériovým rozhraním I^2C, které podporuje dva režimy. Ve standardním režimu může být frekvence hodinového signálu na vodiči SCL v rozmezí 0 - 100 kHz a v rychlém režimu v rozmezí 100 - 400 kHz. Obvod je dále vybaven dvěma napájecími vstupy, mezi kterými umí samovolně přepínat. Napájecí napětí se pohybuje v rozmezí 3.0 - 5.5 V (V_{CC}) nebo 1.3 - 3.7 V (V_{BAT}). Pro oba způsoby napájení platí, že je zachován obsah vnitřní paměti RAM. Obvod dále potřebuje připojení externího krystalu o frekvenci 32 kHz na piny X1 a X2. Výstupem obvodu (pin SQW/OUT) může být obdélníkový signál o frekvenci 1 Hz, 4 kHz, 8 kHz nebo 32 kHz [12]. Blokové schéma obvodu je na obrázku 2.17.

Organizace paměti

Vnitřní paměť, jejíž velikost je 56 bajtů, je možné adresovat výhradně po bajtech. Prvních sedm bajtů slouží pro uložení času a data v několika možných formátech. Osmý bajt paměti je vyhrazen pro řídící bity, pomocí kterých je nastavován hodinový obvod modulu. Zbylých 48 bajtů je možné využít zcela libovolně. Ukazatel do paměti, jehož hodnota je automaticky inkrementována při čtení nebo zápisu do paměti, je uložen v jednom z registrů modulu [12].

Komunikace pomocí I^2C

Obvod se vždy chová jako zařízení typu *slave* (adresa zařízení je pevně stanovena), kdy na základě rozpoznané adresy a hodnoty bitu $R/\overline{W}$ (0 pro zápis, 1 pro čtení) přechází do jednoho ze dvou režimů:



Obrázek 2.17: Blokové schéma obvodu Maxim DS1338Z-33+ [12]

Režim zápisu: První přijatý bajt, který následuje za bajtem s adresou a řídícím bitem slouží pro nastavení hodnoty ukazatele do paměti. Zařízení odpoví odesláním bitu ACK. Následně může přijmout *nula* až *n* bajtů dat. Ty budou ukládány do vnitřní paměti od adresy, která byla přijata. Zbytek komunikace probíhá dle principů popsaných v části 2.3.1.

Režim čtení: Zařízení v tomto režimu odesílá jednotlivé bajty od adresy dané hodnotou ukazatele do paměti. Po odeslání každého bajtu je hodnota ukazatele do paměti automaticky inkrementována. Zbytek komunikace opět probíhá dle principů komunikace skrze rozhraní I^2C [12].

Kromě výše uvedených režimů je zároveň podporována technika tzv. *opakovaného startu* (více v části 2.3.1). Tu lze v tomto případě použít pro vyčtení konkrétní hodnoty z vnitřní paměti bez ohledu na aktuální hodnotu ukazatele do paměti. Komunikace poté bude vypadat tak, že zařízení typu *master* zahájí komunikaci s modulem v režimu zápisu, kdy odešle pouze jediný bajt s adresou paměti. Zařízení *master* poté odešle znovu START bit spolu s adresou modulu a řídícím bitem, který přepne modul do režimu čtení. V důsledku toho začne modul odesílat jednotlivé bajty z paměti od adresy, kterou právě přijal. Množství dat, které bude odesláno je určeno zařízením typu *master* pomocí bitu NACK [12].

2.4 Framework Qt

Qt je označení multiplatformního frameworku určeného primárně pro vývoj aplikačního softwaru. Podporovány jsou například platformy Android, Windows, iOS a Linux [30].

Od verze 5.4 je Qt k dispozici pod několika licencemi. První, placená licence, je určena především pro vývoj komerčních nebo právně chráněných aplikací, kde není možné zveřejnit zdrojové kódy. Opakem je verze, která je licencována pomocí licencí GPL[35] a LGPL[36] verze

[35]GPL (GNU General Public License)
[36]LGPL (GNU Lesser General Public License)

3 a verze 2.1. Tyto verze frameworku jsou vhodné pro vývoj tzv. *volného softwaru*, který splňuje podmínky použitých licencí. Nevýhodou oproti placené verzi frameworku je to, že některé jeho části zde nejsou dostupné [31].

Kromě knihoven v jazyce C++ jsou součástí frameworku i aplikace Qt Designer a Qt Linquist pro tvorbu a pro lokalizaci GUI[37]. Rozšířením Qt jazyka C++ jsou *signály* a *sloty*. Slotem rozumíme speciální metodu, která je volána v případě výskytu signálu. Signálem je pak mechanismus, který slouží k informování okolí o vzniku vybrané události. Samotné propojení příslušného signálu a slotu je realizováno pomocí příkazu *connect*. Od verze 4.7 je součástí frameworku i modul Qt Quick, jehož stručný popis lze nalézt v části 2.7.3 [29].

2.5 Protokoly transportní vrstvy

Transportní vrstva slouží k přenosu dat mezi aplikacemi na zdrojovém a cílovém počítači. Jedná se tedy o jakýsi druh logického spojení dvou procesů, kdy každý proces běží na jiném počítači. Jednotkou transportní vrstvy je *paket*, který slouží k přenosu dat po síti.

Tato vrstva v závislosti na konkrétním protokolu nejčastěji zajišťuje segmentaci dat do paketů, přenos paketů mezi koncovými zařízeními, ustavení spojení, řízení toku a spolehlivý přenos. Základními protokoly této vrstvy jsou protokoly UDP[38] a TCP[39]. Zásadním rozdílem mezi nimi je to, zda zajišťují (TCP) nebo nezajišťují (UDP) spolehlivý přenos dat skrz síť. Oba tyto protokoly totiž pracují nad protokolem IP, tedy protokolem síťové vrstvy, který sám o sobě nezaručuje spolehlivý přenos. Tato vlastnost vcelku jasně vymezuje použití jednoltivých protokolů v konkrétních aplikacích. Typickým příkladem využití protokolu TCP je elektronická pošta nebo zabezpečené přihlašování. Naopak pro UDP je typickým využitím přenos audio a video záznamu. [11].

2.5.1 Protokol TCP

TCP je spojově orientovaný, plně duplexní protokol, který garantuje spolehlivé doručení ve správném pořadí. Vstupní datový proud je rozdělen do několika paketů, které nemusí být doručeny ve správném pořadí. Z tohoto důvodu je každý paket opatřen sekvenčním číslem, na základě kterého jsou jednotlivé pakety na straně příjemce přeskládány do správného pořadí. K zajištění spolehlivého doručení se využívá technika *pozitivního potvrzování*. Ta pracuje na principu odesílání potvrzovacího paketu ze strany příjemce zpět odesílateli po úspěšném přijetí datového paketu. Na straně odesílatele tento mechanismus funguje tak, že každý paket, který nebyl do určité doby potvrzen, je odeslán znovu. Kontrola obsahu paketu je realizována pomocí kontrolního součtu, kdy každý paket s odlišným kontrolním součtem na straně příjemce není potvrzován.

Další službou, kterou protokol nabízí, je *řízení toku*. Již bylo řečeno, že každý v pořádku přijatý paket je stranou příjemce potvrzován. Až na základě této skutečnosti je možné paket na straně odesílatele označit za doručený. Selektivní potvrzování každého paketu by však příliš zvyšovalo režii přenosu. TCP proto využívá vyrovnávací paměť, kterou nazýváme *okno*. Její velikost označuje počet paketů, které mohou být odeslány bez přijetí potvrzení. Důležitou skutečností je, že velikost *okna* odesílatele určuje příjemce. Ten může v reakci na stav sítě zrychlit (zvětšit velikost) nebo naopak zpomalit komunikaci (zmenšit velikost).

[37] GUI (Graphical User Interface)
[38] UDP (User Datagram Protocol)
[39] TCP (Transmission Control Protocol)

V extrémním případě může příjemce komunikaci zcela zastavit a to tak, že nastaví velikost *okna* na nula položek.

Samotné spojení mezi klientem a serverem je vytvořeno pomocí mechanismu *3-way handshake(třífázová synchronizace)* a je aktivní až do doby, než dojde k jeho ukončení. To je obvykle iniciováno také klientem, přičemž je realizováno pomocí výměny čtyř paketů [16, 11].

2.5.2 Programování nad TCP pomocí frameworku Qt

Komunikace skrze prokol TCP probíhá v Qt pomocí třídy *QTcpSocket*, která je potomkem třídy *QAbstractSocket*. Tato třída dále dědí od třídy *QIODevice*. To znamená, že je možné použít pro čtení a zápis dat standardní metody *read()/readAll()* a *write()*. K rozpoznání nově příchozích dat slouží signál *readyRead()*, který je emitován pokaždé, když jsou úspěšně přijata nová data.

Práce s TCP sockety je tedy z pohledu čtení a zápisu totožná jako práce s jakýmkoliv vstupně/výstupním zařízením. Třída *QAbstractSocket* dále přidává i signály *connected()* a *disconnected()*, které jsou emitovány při připojení nebo odpojení socketu [26].

TCP server

V předchozí části již bylo řečeno, že komunikace skrze protokol TCP probíhá na principu *klient-server*. Pro vytvoření serveru je v Qt připravena třída *QTcpServer*. Po vytvoření objektu této třídy je nutné zavolat metodu *listen(const QHostAddress & address, quint16 port)*, která zajistí, že server bude naslouchat na definované adrese a portu. Při použití parametru *QHostAddress::Any* jako adresy bude server naslouchat na všech síťových rozhraní.

Třída *QTcpServer* dále obsahuje signál *newConnection()*, který je vyvolán po připojení nového klienta. Socket, na kterém je nově připojený klient, lze získat pomocí metody *nextPendingConnection()*. Ta vrací objekt třídy *QTcpSocket*, který může být použit pouze z vlákna ve kterém běží server [27].

TCP klient

Klientská část je v Qt tvořena objektem již zmíněné třídy *QTcpSocket*, kdy pro připojení klienta k serveru slouží metoda *connectToHost(const QHostAddress & address, quint16 port, OpenMode openMode)*. Jejími parametry jsou: adresa a port na kterém server naslouchá a režim činnosti. Zápis a čtení dat nad socketem probíhá pomocí již zmíněných funkcí *read()/readAll()* a *write()* [27].

2.6 Bezdrátová komunikace

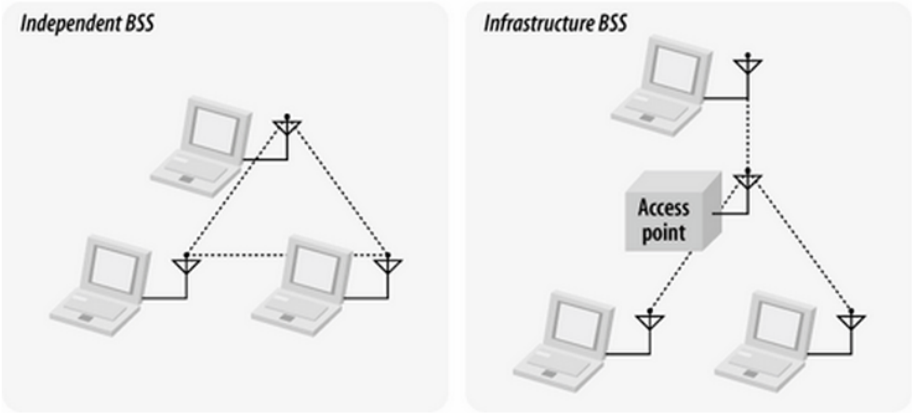
Standardů pro bezdrátovou komunikaci existuje celá řada. Vhodnými příklady jsou standardy IEEE 802.15.1 a IEEE 802.11, kdy první zmíněný je spíše znám jako technologie Bluetooth. Druhý standard, přesněji řečeno rodina standardů, definuje technologii Wi-Fi[40]. Výhodou těchto technologií je především jejich dostupnost. Můžeme je nalézt téměř v každém chytrém telefonu, tabletu či notebooku. Zásadním rozdílem obou technologií je maximální možná vzdálenost, na jakou lze komunikovat. U technologie Bluetooth se v závislosti

[40] Wi-Fi (Wireless Fidelity)

na třídě jedná o jeden, deset nebo sto metrů [24]. U standardu 802.11b, který je v současné době jedním z nejrozšířenějších, se maximální dosah pohybuje okolo 300 metrů v otevřeném prostoru. Tato vzdálenost je mimo jiné značně ovlivněna i přenosovou rychlostí, kdy tento údaj odpovídá v praxi běžné hodnotě 5,5 Mbit/s [7].

2.6.1 Wi-Fi

Již na začátku této části zaznělo, že pojem Wi-Fi označuje technologii bezdrátových sítí dle standardů IEEE 802.11x. Parametry dané sítě tedy definuje konkrétní standard, kterému daná síť odpovídá. Z pohledu architektury těchto sítí je důležitým parametrem identifikátor dané sítě, tzv. SSID[41]. Bez znalosti tohoto textového řetězce se nelze do dané sítě připojit. Wi-Fi sítě dělíme dle architektury na *ad-hoc (Independent BSS[42])* a *infrastrukturní sítě (Infrastructure BSS)*. Graficky jsou oba typy ukázány na obrázku 2.18 [7].



Obrázek 2.18: Architektura ad-hoc (vlevo) a infrastrukturní (vpravo) sítě [7]

Ad-hoc sítě

Jedná se o nezávislé sítě bez tzv. *přístupových bodů* (AP[43]). To znamená, že zde není žádný prvek sítě, který by zastával roli *arbitra*. Typicky se tyto sítě využívají pro menší lokální sítě, kdy jednotlivé stanice nejsou od sebe vzdáleny více než pár metrů (jednotlivé stanice musí být v rádiovém dozsahu) [7].

Infrastrukturní sítě

Na rozdíl od *ad-hoc* sítě je zde přítomen jeden nebo více *přístupových bodů*. Jednotlivé stanice, které chtějí být členy dané sítě se musí nejprve asociovat vůči příslušnému AP. Ten na základě nastavení přijme nebo zamítne danou žádost o připojení. Každému AP v dané síti je přidělen identifikátor SSID, který nemusí být unikátní [7].

2.6.2 Hi-Link HLK-RM04

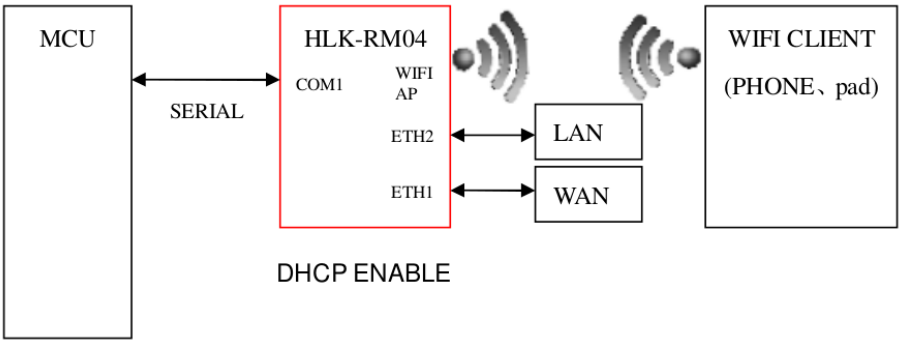
Jedná se o modul umožňující připojení libovolného zařízení prostřednictvím sériového rozhraní UART (COM1) k bezdrátové síti Wi-Fi(WIFI) nebo drátové síti Ethernet (ETH1,

[41] SSID (Service Set Identification)
[42] BSS (Basic Service Set)
[43] AP (Access Point)

ETH2). Modul podporuje standardy IEEE 802.11n/g/b pro bezdrátovou síť a standardy IEEE 802.3 a IEEE 802.3u pro připojení k síti Ethernet. Sériová linka je plně konfigurovatelná a umožňuje přenosové rychlosti v rozmezí 50 až 230 400 bitů za sekundu. Modul podporuje protokoly TCP/UDP, filtrování MAC adres, zabezpečení WEP[44] (64, 128 a 152bitový klíč) a WPA[45]/WPA2[46]-AES[47]/TKIP[48]. Modul je vybaven pasivní anténou s možností připojení externí antény pomocí konektoru U-FL. Napájecí napětí modulu je 5 V [8].

Princip činnosti

Na základě aktuálního nastavení modul pracuje v jednom ze čtyř níže uvedených režimů činnosti. Funkční schéma modulu, kde jsou zobrazeny všechny rozhraní, je na obrázku 2.19. Konfiguraci modulu je možné provést buď skrze rozhraní COM1 pomocí tzv. AT instrukcí, nebo skze webovou stránku. Aktuální nastavení je možné uložit do konfiguračního souboru, který se poté kdykoliv může zpětně použít pro konfiguraci modulu.



Obrázek 2.19: Funkční schéma modulu Hi-Link HLK-RM04 [12]

UART - Ethernet: Data jsou přeposílána mezi aktivními rozhraními COM1 a ETH1. Zbylá rozhraní jsou neaktivní. IP adresa modulu může být zadána staticky nebo získána z DHCP[49].

UART - Wi-Fi klient: V tomto režimu jsou povoleny rozhraní COM1 a WIFI. Modul se chová jako TCP/UDP klient, kdy jeho IP adresa může být opět statická nebo dynamická. Veškerá data, která jsou přijata od serveru jsou postupně odeslána skrze rozhraní COM1. Pro opačný směr komunikace platí totéž.

UART - Wi-Fi AP: Modul vytváří přístupový bod (AP). Maximální počet připojených klientů v jeden okamžik je dvacet. Z pohledu síťové komunikace funguje modul jako TCP/UDP server, kdy veškerá síťová data jsou posílána dále skrze rozhraní COM1. Naopak veškerá data, která jsou přijata na rozhraní COM1 jsou dále rozesílána pomocí broadcastu.

[44] WEP (Wired Equivalent Privacy)
[45] WPA (Wi-Fi Protected Access)
[46] WPA2 (Wi-Fi Protected Access II)
[47] AES (Advanced Encryption Standard)
[48] TKIP (Temporal Key Integrity Protocol)
[49] DHCP (Dynamic Host Configuration Protocol)

Výchozí: V tomto režimu jsou povolena všechna rozhraní, která byla zmíněna v této části. Modul opět vytváří přístupový bod, přičemž rozhraní ETH1 slouží k připojení do sítě WAN a rozhraní ETH2 naopak k připojení do sítě LAN [8].

2.7 Relační databáze

Základem relačních databází jsou tabulky, které slouží jako fyzická reprezentace jednotlivých relací. Každá tabulka se skládá z hlavičky, které říkáme *schéma relace* a těla, kterému říkáme *tělo relace*. Jednotlivé sloupce tabulky označujeme názvem *atribut*, přičemž každý atribut má definován datový typ a obor hodnot. Takto pojmenované množině skalárních hodnot říkáme *doména*. Počet sloupců dané tabulky udává *stupeň relace*. Pro práci s relačními databázovými systémy je používán standardizovaný strukturovaný dotazovací jazyk SQL[50].

Tabulky relační databáze mají tyto vlastnosti: jednoznačný identifikátor, každá buňka dané tabulky nabívá právě jedné hodnoty, jména sloupců v rámci jedné tabulky jsou unikátní, hodnoty v jednom sloupci náleží právě do jedné domény, nezáleží na pořadí sloupců a řádků a zároveň tabulka neobsahuje redundantní data [3].

2.7.1 Normální formy

V souvislosti s relačními databázemi se lze velmi často setkat s pojmem *normalizace*. Ta označuje proces převodu tabulek relační databáze do tzv. *normálních forem*. Tyto formy lze chápat jako soubor pravidel, která musí všechny tabulky splňovat. Cílem normalizace je obecně zjednodušit a zefektivnit manipulaci s daty (např. zamezením redundance). Nejpoužívanějšími normálními formami jsou první normální forma (1NF), druhá normální forma (2NF) a třetí normální forma (3NF) [3].

1NF

Ke splnění této základní formy musí platit to, že všechny atributy jsou dále nedělitelné. To znamená, že každá položka tabulky by měla obsahovat právě jednu hodnotu. Lze si všimnout, že popis této normální formy se shoduje s obecnou charakteristikou tabulek relační databáze [3].

2NF

Pro tuto normální formu musí platit, že tabulka splňuje 1NF a zároveň platí, že každý atribut (vyjma primárního klíče) je úplně závislý na primárním klíči. Jinými slovy žádný atribut nesmí být závislý pouze na části primárního klíče (pro jednoduché primární klíče je tato podmínka automaticky splněna). Obecně platí, že při převodu tabulky do této normální formy dojde k rozpadu původní tabulky, která nebyla v 2NF, na více menších tabulek, které již budou v druhé normální formě. Převod tabulky do této normální formy často vede k odstranění redundantních dat [3].

3NF

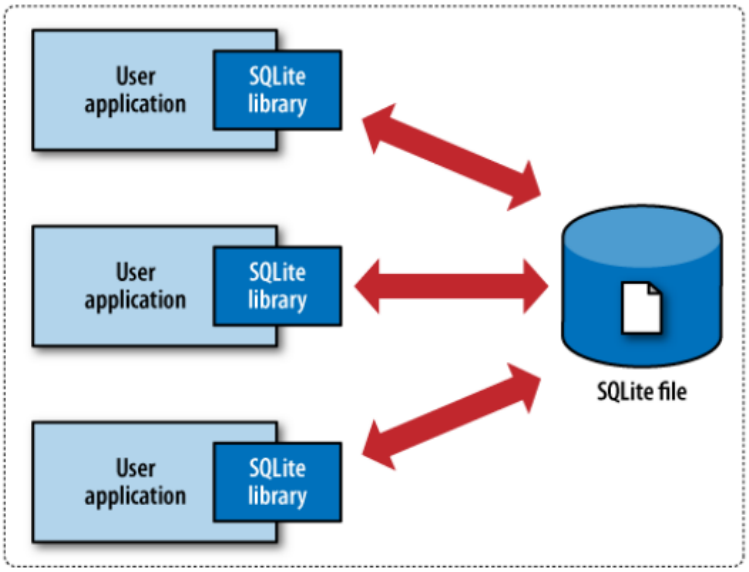
Kromě splnění předchozích dvou normálních forem musí zároveň platit podmínka, že každý neklíčový atribut není závislý na žádném jiném neklíčovém atributu. Jinými slovy zde není

[50] SQL (Structured Query Language)

tranzitivní závislost mezi primárním klíčem a neklíčovým atributem [3].

2.7.2 SQLite

Jedná se *public domain*[51] knihovnu, která slouží pro realizaci relační databáze. Na rozdíl od klasické SQL databáze nemá SQLite architekturu *klient-server*. To znamená, že na cílovém systému nemusí být žádný databázový server, ke kterému bychom se připojovali. Díky tomu odpadají starosti spojené s přenositelností, závislostí aplikace na dalších knihovnách, nutností provést konfiguraci apod. SQLite je koncipována tak, že sama obsahuje vše potřebné pro realizaci relační databáze. Jednotlivá data dané databáze jsou poté uložena ve speciálním souboru na lokálním datovém úložišti. Tento soubor obsahuje jak data, tak i schéma dané databáze. Formát souboru je nezávislý na konkrétní platformě. To znamená, že lze danou databázi velmi jednoduše fyzicky přenést na jiný systém. Architektura SQLite je naznačena na obrázku 2.20.



Obrázek 2.20: Architektura SQLite [9]

Použití této knihovny je velmi snadné. Pouze ji stačí přilinkovat k dané aplikaci, přičemž knihovna se stará o komunikaci se zmíněným souborem prostřednictvím jazyka SQL. Tento způsob realizace datového úložistě má kromě výhod (některé byly již zmíněny výše) také i několik nevýhod. Absence serveru zcela jasně značí, že tento druh databáze není vhodný pro implementaci sdíleného datového úložiště. Naopak je tato knihovna velmi vhodná jako lokální databáze pro ukládání historie, nastavení apod. jednotlivých aplikací [9].

2.7.3 Využití SQLlite pomocí frameworku Qt

Součástí frameworku Qt je od verze 4.7 modul Qt Quick, který umožňuje vytvářet pomocí základních primitiv uživatelské rozhraní s dynamickými efekty. Současně s představením této platformy se objevil i nový jazyk QML[52]. Jedná se o vysokoúrovňový deklarativní

[51] druh licence opravňující k volnému šíření, použití a editaci
[52] QML (Qt Meta-Object Language)

skriptovací jazyk, který slouží pro popis chování i vzhledu samotné aplikace. Jednou z výhod Qt Quicku je možnost interakce mezi C++ a QML v obou směrech. Zároveň lze velmi snadno využít JavaScriptu uvnitř QML [37].

Součástí platformy Qt Quick je i modul s názvem *Local Storage*, jehož chování odpovídá návrhovému vzoru *jedináček*. Tento modul poskytuje prostřednictvím JavaScriptu rozhraní pro SQLite databáze. Modul poskytuje metodu: *object openDatabaseSync(string name, string version, string description, int estimated_size, jsobject callback(db))*, která zpřístupní nebo vytvoří a zpřístupní SQL databázi dle zadaných parametrů. Těmi jsou: název, verze, popisný název a velikost databáze v bajtech. Volitelně může být přidán i tzv. *callback parametr*. Tím je ukazatel na metodu, která je volána v případě, že nebylo možné vytvořit databázi. Spojení aplikace s databází je uzavíráno automaticky. Modulu *Local Storage* zároveň definuje API[53] pro práci s databází prostřednictvím JavaScriptu [33, 32].

2.8 Dostupná zařízení na trhu

Na trhu je celá řada zařízení, která umožnují záznam provozních údajů letadel. Tato zařízení lze rozdělit na vestavěné palubní přístroje a univerzální přenosná zařízení. Zamýšlené zařízení, které je předmětem této práce, spadá do první z těchto kategorií. A právě proto se budu nadále soustředit výhradně na zařízení spadající do kategorie vestavěných palubních přístrojů. Cílem této části je poskytnout stručný popis několika vybraných a v praxi používaných přístrojů.

2.8.1 Winter Instruments FSZMD

Nejjednodušší zařízení z tohoto přehledu. Vstupem tohoto přístroje je kromě napájení i výstup tlakového senzoru (pitotovy trubice). Ten je použit pro stanovení rychlosti letadla, přičemž při překročení rychlosti 60 km/h dojde k automatické detekci začátku letu. Přístroj umí pouze zobrazovat celkový počet nalétaných hodin na malém LCD displeji. Automatická detekce přistání nebo možnost ukládání jednotlivých letů zde není. Napájecí napětí je 12 až 24 V. Zařízení lze přímo zabudovat do otvoru s průměrem 57 mm v palubní desce. Cena se pohybuje okolo 10 000 Kč vč. DPH. Vzhled zařízení je na obrázku 2.21 [25].

Obrázek 2.21: Winter Instruments FSZMD [25]

[53] API (Application Programming Interface)

2.8.2 Vtec Electronics GmbH Flisys80P

Podstatně složitější zařízení, které nabízí automatickou detekci startu i přistání s možností ukládání jednotlivých letových záznamů a motohodin. Je vybaveno dvouřádkovým LCD displejem, čtyřmi tlačítky, slotem na SD kartu a USB konektorem, který slouží výhradně pro aktualizaci vnitřního programu zařízení (firmwaru). Čelní strana přístroje je na obrázku 2.22.

Stejně jako v předchozím případě je k zařízení nutné připojit výstup z Pitotovy trubice. Vstupní dynamický tlak je poté uvnitř zařízení transformován na rychlost. K zařízení dále mohou být připojeny dva vstupní signály, které slouží pro ovládání počítadla motohodin a jako alternativní vstup pro detekci letu. Zařízení dále disponuje výstupním signálem, jehož hodnota odpovídá vnitřnímu stavu zařízení (letový/pozemní režim). Zařízení lze zabudovat do standardního 80 mm otvoru v palubní desce. Napájecí napětí je 9 až 32 V.

Start a přistání jsou detekovány buď na základě rychlosti letadla, nebo volitelně pomocí hodnoty příslušného vstupního signálu. Pro první případ platí, že uživatel může definovat vlastní prahové hodnoty rychlosti pro přistání a start. Aktuální čas a datum jsou získávány z RTC obvodu napájeného z vnitřní lithiové baterie. Paměť zařízení umožňuje uložit až 4000 letových záznamů. Cena zařízení je okolo 13 000 Kč vč. DPH [23].

Obrázek 2.22: Vtec Electronics GmbH Flisys80P [23]

2.8.3 MGL Avionics Flight-2

Toto zařízení je z této trojice nejuniverzálnější. Může například plnit roli výškoměru, rychloměru, palivoměru, palubního deníku a stopek. Zařízení je osazeno LCD displejem, třemi tlačítky a LED diodou pro indikování překročení vybraných hraničních hodnot. Na vnitřní paměť zařízení lze uložit pouze 24 letových záznamů. Ty jsou vytvářeny buď zcela automaticky, nebo manuálně. Podmínkou pro automatické zahájení letového záznamu je překročení definované rychlosti po dobu delší než 60 sekund. Naopak při poklesu rychlosti pod tuto definovanou hodnotu po 30 a více sekund dojde k ukončení letového záznamu.

Vzhledem ke složitosti celého zařízení je nutná celá řada vstupů. Kromě napájecího napětí o velikosti 12 až 24 V je zde opět vstup pro Pitotovu trubici, palivoměr, otáčkoměr atd. Konkrétní zapojení lze opět nalézt v dokumentaci. Cena zařízení je okolo 10 000 Kč vč. DPH. Vzhled přístroje lze vidět na obrázku 2.23 [13].

Obrázek 2.23: MGL Avionics Flight-2 [13]

Kapitola 3

Analýza současného stavu a požadavků

V úvodu této práce jsem již zmínil, že navrhované zařízení využívá technického návrhu zařízení Flytimer. Autory návrhu jsou MUDr. Zdeněk Moravec, Prof. Dr. Ing. Pavel Zemčík a Ing. Antonín Hegar. Iniciátorem vzniku zařízení Flytimer je MUDR. Zdeněk Moravec, který zároveň působí jako letecký instruktor ultralehkých letadel na letišti v Kotvrdovicích. Zařízení Flytimer mělo primárně sloužit pro potřeby místního aeroklubu. Já jsem tento tento projekt převzal v roce 2013 ve stavu, kdy byl hotový technický návrh zařízení a bylo zhotoveno několik desek plošných spojů. Během posledních pár let však došlo k částečné obměně letecké techniky místního aeroklubu. Tyto změny měly za následek drobné změny požadavků na dané zařízení.

Mým cílem je dokončit a zrealizovat návrh zařízení Flytimer. Na konci realizace by mělo být funkční zařízení, splňující požadavky členů aeroklubu z Kotvrdovic, které by sloužilo jako elektronický palubní deník. Toto zařízení bychom poté rádi prostřednictvím aeroklubu nabídli dalším provozovatelům letadel. Osobně jsem na toto téma hovořil s předsedou Svazu ultralehkého létání, panem Michalem Seifertem. Navrhované zařízení ho zaujalo a ve své podstatě se požadavky na funkcionalitu, které byly vzneseny panem Seifertem a členy aeroklubu příliš neliší.

3.1 Stanovení požadavků

Na základě informací, které jsem získal od kolektivu autorů byly stanoveny následující požadavky. Přístroj musí být možné napájet napětím v rozsahu 12 až 20 V. Zařízení musí umožnovat záznam jednotlivých letů, včetně jejich délky, data a času zahájení. Délka letu musí být ukládána s přesností na minuty. U každého letu musí být dále uveden pilot. V přístroji musí být možné uložit alespoň deset posádek, které se před zahájením každého letu přihlásí prostřednictvím uživatelského rozhraní.

Zařízení by mělo umožnovat manuální, popřípadě automatické zahájení počítání doby letu. Let, jehož začátek byl automaticky detekován systémem, musí být uložen i v případě, kdy není přihlášen žádný uživatel. Takovýto let musí být později rozlišitelný. Při ztrátě napájení systému nesmí dojít ke ztrátě informací o jakémkoliv letu. Ukončení letu není nutné provádět automaticky a za vyhovující se považuje možnost ukončení letu stiskem tlačítka nebo vypnutím zařízení. Zařízení by dále mělo uchovávat informaci o aktuálním času a datu a to i v případě ztráty napájecího napětí. Přístroj musí být dále vybaven

stopkami, které je možné používat i v době letu. Po přihlášení uživatele musí být možné zobrazit aktuální dobu letu, celkový počet nalétaných hodin přihlášené posádky, stopky, aktuální čas a datum.

Oprávněným osobám musí být dále umožněno prostřednictvím snadno ovladatelné aplikace přidat/odebrat/přejmenovat jednotlivé posádky, nastavit provozní údaje zařízení a vymazat/modifikovat záznamy jednotlivých letů. Dále musí být možné vhodným způsobem ukládat letové záznamy i mimo palubní přístroj a to tak, aby bylo možné tyto záznamy kdykoliv nahrát zpět do zařízení. Aplikace musí být spustitelná na operačních systémech Windows 7 a 8.1.

3.2 Technický návrh zařízení Flytimer

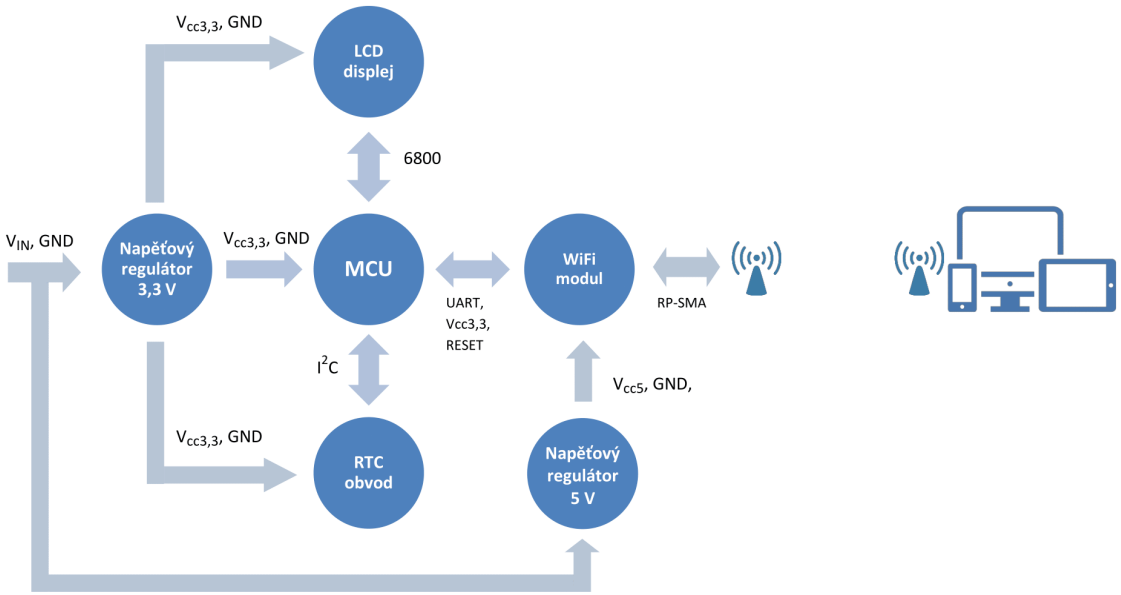
Deska plošného spoje (DSP) zařízení Flytimer obsahuje mikrokontrolér *MSP430F1611*. K němu je pomocí paralelního rozhraní typu *6800* možné připojit LCD displej *Batron BTHQ128064AVD1-COG-STF-12-LED02YG*, který byl představen spolu s rozhraním *6800* v části 2.2. Dále je k mikrokontroléru připojen pomocí rozhraní I^2C obvod reálného času *Maxim DS1338Z-33+*. Popis tohoto obvodu a rozhraní I^2C lze nalézt v části 2.3. Na DSP je dále umístěn napětový regulátor *LM1117DT-3.3* od firmy National Semiconductor. Funkcí tohoto obvodu je převod vstupního napájecího napětí (V_{IN}) v rozsahu *4,6* až *20 V* na výstupní napájecí napětí ($V_{CC3,3}$) *3,3 V*. Toto napájecí napětí je poté využito všemi ostatními integrovanými obvody na DSP. Dalším obvodem, který tvoří zařízení Flytimer, je integrovaný obvod *TPS3825-33DBV* firmy *Texas Instruments*. Tento obvod je zde pouze pro kontrolu hodnoty generovaného napájecího napětí. V případě poklesu pod hodnotu *2,93 V* bude tímto obvodem způsoben reset mikrokontroléru [22]. Posledním integrovaným obvodem, který lze na desce plošného spoje nalézt, je obvod *MAX3232CSE* od firmy *Maxim Integrated*. Ten slouží pro převod mezi rozhraními *UART* a *RS-232*, jenž se od sebe liší pouze napětovými úrovněmi [21]. Většina zbylých pinů mikrokontroléru je vyvedena skrze ochranné obvody na několik konektorů. Ovládací prvky jsou na zařízení Flytimer zastoupeny čtyřmi tlačítky.

3.3 Návrh požadavků

Navrhovaný systém, jehož schéma je na obrázku 3.1, lze tedy rozdělit na přístroj, který bude umístěn v letadle a aplikaci, která budou sloužit pro správu zařízení. Návrh přístroje je z velké části dán technickým návrhem zařízení Flytimer. Jistou modifikací tohoto návrhu je například způsob propojení přístroje a aplikace, která slouží pro správu dat.

Originální návrh počítal s propojením pomocí sériové linky *RS-232*. K tomuto způsobu propojení by bylo však nutné zhotovit propojovací kabel, který by obsahoval převodník sériové linky *RS-232* na rozhraní *USB*[1]. Cena takovéhoto propojovacího kabelu by se pohybovala zhruba okolo tří set korun. Za velmi podobnou cenu lze však pořídit moduly umožnující připojení sériové linku *RS-232/UART* do *Wi-Fi* sítě. Toto řešení lze při současném technickém vybavení notebooků, tabletů a chytrých telefonů označit přinejmenším za pohodlnější a cenově srovnatelné. Právě z tohoto důvodu bude namísto tohoto řešení použito modulu *Hi-Link HLK-RM04* (popis v části 2.6.2), který bude připojen skrze rozhraní *UART* k modulu *USART* mikrokontroléru *MSP430F1611*. Pro realizaci reset obvodu bude WiFi modul dále připojen k mikrokontoléru pomocí dalších dvou vodičů, kdy jeden slouží

[1] USB (Universal Serial Bus)



Obrázek 3.1: Blokové schéma návrhu zařízení

jako zdroj napájecího napětí 3,3 V a druhý jako povolovací vstup. Drobnou nevýhodou použitého WiFi modulu je hodnota napájecího napětí, která činí *5 V*. To znamená, že je nutné rozšířit zařízení Flytimer ještě o jeden napětový regulátor *LM1117DT-5.0* od firmy National Semiconductor.

K ovládání palubního přístroje slouží čtyři tlačítka a LCD displej. Běžnému uživateli bude přístupné pouze tohoto rozhraní, prostřednictvím kterého mu bude umožněno přihlášení do systému, prohlížení jednotlivých časových údajů, obsluhu stopek a zahájení/ukončení letu. Palubní přístroj také bude umožnovat zapnutí v tzv. *administrátorském režimu*. Ten bude umožňovat prohlížení celkového stavu zařízení, přidání nového uživatele, nastavení základních parametrů zařízení a vymazání exportovaných dat z datového úložiště. Oprávněným osobám bude dále umožněno komunikovat s daným zařízením skrze *Wi-Fi* pomocí podpůrné aplikace. Ta bude provozovateli poskytovat prostředek pro správu dat a přizpůsobení palubního přístroje. Pro ukládání dat bude tato aplikace obsahovat databázi, jejíž obsah bude možné prohlížet a spravovat přímo z aplikace. Jako vhodné řešení se jeví využití platformy *Qt Quick* a modulu *Local Storage* (viz. část 2.7.3). Síťovou část aplikace je naopak vhodné vytvořit pomocí existujících tříd knihovny *Qt*. Stručný přehled možností lze nalézt v části 2.5.2.

Při každém připojení palubního přístroje k aplikaci bude vytvořen logovací soubor, který bude obsahovat obsah vnitřní paměti přístroje spolu s časovou známkou. Obsah vnitřní paměti palubního přístroje bude možné kdykoliv obnovit ze zvoleného nebo prostřednictvím aplikace vytvořeného logovacího souboru.

Letová data budou v palubním přístroji ukládána do vnitřní paměti typu *Flash* mikrokontroléru *MSP430F1611*. Její velikost je *48 kB*, přičemž pro ukládání historie jednotlivých letů lze využít zhruba polovinu z celkové velikosti vnitřní paměti. Toto omezení znamená, že je nutné použít velmi efektivní způsob ukládání letových záznamů. Ve vnitřní paměti bude dále uložen seznam posádek a provozní informace, které slouží pro zajištění správné činnosti zařízení.

Pro zjištění časových údajů je využito obvodu reálného času. Ten je napájen buď pomocí

generovaného napájecího napětí 3,3 V nebo z externí 3 V lithiové baterie. Komunikaci s tímto obvodem bude na straně mikrokontroléru zajišťovat modul *USART0* v režimu I^2C.

LCD displej bude připojen k mikrokontroléru pomocí rozhraní *6800*. Mikrokontrolér však nedisponuje žádným hardwarovým modulem, který by umožnoval tento druh komunikace, což znamená, že ji bude nutné realizovat čistě softwarově. Jako vhodné se jeví využití modulu *DMA*, kterým byl představen v části 2.1.3.

3.4 Správa dat

V předchozí části již zaznamělo, že se navrhované řešení musí potýkat se značně omezenou kapacitou vnitřní paměti. Dopady této skutečnosti je možné minimilizovat zavedením určitého druhu kódování dat, který by snížil pamětovou náročnost. Zároveň se jedná o způsob, který nenavyšuje obvodovou složitost a cenu zařízení.

Dalším problém, který musí být řešen je možná ztráta napájení zařízení bez předchozího ukončení a uložení letového záznamu. To znamená, že je nutné dobu letu průběžně ukládat tak, aby v případě ztráty napájení zůstal v paměti uložen poslední časový údaj. Z pohledu implementace to znamená, že je nutné na začátku letu vytvořit v paměti příslušný letový záznam, u kterého bude periodicky inkrementována doba letu.

K ukládání letových dat je ovšem vyhrazena paměť typu *Flash*, u které je životnost omezena počtem zápisů. V tomto konkrétním případě se jedná o zhruba sto tisíc zápisů [35]. To znamená, že při frekvenci zápisu jeden hertz by došlo k poškození paměti již po zhruba dvaceti osmi hodinách. Tento druh paměti tedy není příliš vhodný pro ukládání dat s velmi vysokou frekvencí. Vhodným řešením je využití paměti typu *RAM* v obvodu *DS1338Z-33+*, jejíž obsah zůstává zachován i v případě ztráty primárního napájení (platí pouze pokud je zapojena záložní baterie) a zároveň není její životnost omezena počtem zápisů. Obsah vnitřní paměti obvodu *DS1338Z-33+* tedy bude vždy odpovídat délce právě probíhajícího/posledního letu. Ke spolehlivému uložení daného letového záznamu tedy stačí při každém zapnutí zařízení vyčíst obsah této paměti a uložit jej do vnitřní paměti typu *Flash*.

Kapitola 4

Popis implementace

Mou prací bylo vytvořit vestavěný systém pro ultralehká letadla a vrtulníky, který by sloužil pro záznam provozních údajů. Tento systém je postaven na základě technického návrhu zařízení Flytimer. Ten jsem převzal ve stavu, kdy bylo zhotoveno několik desek plošných spojů. Dále bylo třeba nalézt a vytvořit vhodný nástroj pro administraci celého systému. Budoucí uživatel by tak měl ve výsledku dostat komplexní řešení, které by mu umožnovalo elektronický záznam potřebných provozních údajů letounu.

Navrhovaný systém lze rozdělit na palubní přístroj a podpůrnou aplikaci. Tomuto dělení odpovídá i struktura této kapitoly. Ta je rozdělena na části, které se zabývají popisem implementace softwaru pro zařízení Flytimer a podpůrné aplikace, popisem komunikačního protokolu, který je použit pro komunikaci mezi oběma hlavními částmi systému a několika podkapitolami, jenž popisují přizpůsobení, možnost využití a způsob testování.

Celý systém, jehož součástí je palubní přístroj je koncipován tak, že palubní přístroj slouží pro identifikaci posádek a ukládání několika posledních letových záznamů. Za ideálních podmínek by měla být v místě, kde je dané letadlo uskladněno, dostupná infrastrukturní WiFi síť. K ní by se po zapnutí automaticky snažil připojit palubní přístroj a v případě, že je na hostitelském počítači spuštěna podpůrná aplikace, by přístroj provedl export dat do dané aplikace. Prostřednictvím té je poté možné zpracovat takto importovaná data. Systém by tedy měl být schopný samostatného provozu alespoň do doby, než dojde k zaplnění vnitřní paměti mikrokontroléru.

4.1 Palubní přístroj

V době realizace návrhu palubního přístroje bylo klíčové dosáhnout vhodného kompromisu mezi dobami trvání datových přenosů mezi MCU a LCD řadičem a času, kdy MCU vykonává jinou užitečnou činnost. Nevýhodou, která pramení z technického návrhu zařízení Flytimer, je nutnost využití tzv. *poolingu* pro detekci stisku jednotlivých ovládacích prvků přístroje. V případě, že by většinu času strávil mikrokontrolér obnovou obsahu LCD displeje, by došlo ke zhoršení odezvy na pokyn uživatele. Příliš nízka frekvence obnovy obsahu displeje by však vedla ke stejným potížím. Obdobný problém nastává i v případě realizace datového přenosu mezi přístrojem a aplikací pro správu dat.

4.1.1 Fyzická podoba

Fyzická podoba zařízení, která odpovídá návrhu z části 3.3, je na obrázku 4.1. Ve finální podobě se přístroj skládá z již dříve navrhnuté desky plošného spoje zařízení Flytimer, nové

základní desky pro WiFi modul, montážní krabičky, konektorů, antény a drobného spojovacího materiálu. Schéma obvodu základní desky, která obsahuje reset obvod a napájecí obvod pro WiFi modul, je v příloze na obrázku B.1. Na zadní straně přístoje se nachází dvojice konektorů pro napájení (souosý konektor, vnitřní průměr 5 mm) a připojení WiFi antény (RP-SMA samice).

Obrázek 4.1: Fyzická podoba vytvořeného přístroje

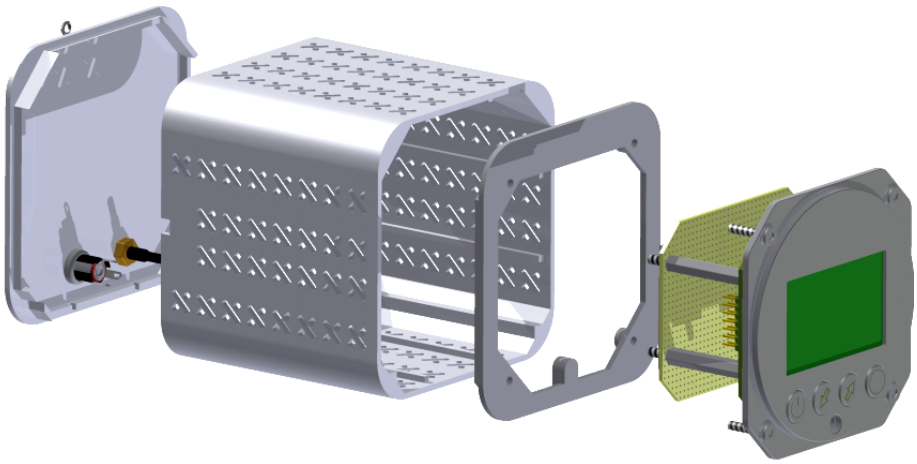
Zapojení

Napájecí napětí je z konektoru na zadní straně krabičky přivedeno jak na desku plošného spoje zařízení Flytimer (piny 5, 7 konektoru X10), tak i na základní desku pro WiFi modul (piny 4, 3 konektoru P1). Součástí obou desek jsou mimo jiné i regulátory napětí, které upravují vstupní napájecí napětí z rozsahu 10 - 14 V na napětí 3,3 V (Flytimer) a 5 V (WiFi modul). Obě desky jsou dále propojeny dvěma vodiči rozhraní UART. Konkrétně jde o propojení pinů 5, 6 konektoru P1 a pinů 4,5 portu P3 mikrokontoléru MSP430F1611 z desky Flytimer. Oba piny na straně zařízení Flytimer jsou vyvedeny přímo z pouzdra mikrokontoléru. Dále jsou pro potřeby reset obvodu WiFi modulu propojeny piny 1, 2 konektoru P1 a piny 9, 11 konektoru X9 na DSP zařízení Flytimer. Konektor RP-SMA pro připojení externí WiFi antény je připojen ke konektoru U-FL modulu HLK-RM04.

Montážní krabička

Pro snadnější manipulaci a ochranu jednotlivých komponent přístroje jsem se rozhodl vytvořit prototyp montážní krabičky pomocí 3D tisku. Model krabičky jsem zhotovil pomocí studentské verze nástroje Autodesk Inventor Professional 2015. Výsledné modely jsem v tomto nástroji exportoval do formátu STL. Bohužel se v době tisku zjistla nepříjemnost, která spočívá v ukládání desetkrát zmenšených modelů. Pro jakékoliv budoucí zpracování modelů, které jsou na přiloženém CD, je tedy nutné mít tuto skutečnost na paměti a provádět tisk v měřítku 10 : 1. Samotná tvorba modelu nečinila žádné obtíže. Použitý nástroj je velmi intuitivní, stabilní a nemá žádné přehnané hardwarové nároky.

Poněkud složitější situace nastala v době, kdy došlo na samotnou realizaci 3D tisku. Tu prováděli vedoucí této diplomové práce pan profesor Pavel Zemčík a Filip Kotouček z firmy *ChciTo3D*. Značná velikost jednotlivých součástí (vnější rozměry krabičky 90 x 90 x 90 mm) a zvolený materiál ABS plast značně ztížily celý proces. Po několika experimentech s různými verzemi modelů se však povedlo docílit požadové kvality a získat tak prototyp montážní krabičky na navržený přístroj. Celkově je krabička tvořena tělem a dvěma víčky. Nákres jednotlivých součástí, včetně elektroniky je na obrázku 4.2.



Obrázek 4.2: Vizualizace komponent přístroje

4.1.2 Princip činnosti

Činnost přístroje je téměř výhradně řízena pomocí několika kanálů časovačů mikrokontroléru. Ty řídí chod systému s využitím série přerušení, které periodicky zahajují obnovu zobrazovaných údajů, délky letu apod. Typ momentálně zobrazovaných údajů je dán vnitřním stavem zařízení. Ten si volí uživatel prostřednictvím ovládacích prvků přístroje. Množina všech dostupných stavů, včetně popisu zobrazované informace, bude popsána v následující části. Na základě změny stavu zařízení uživatelem však nedochází k přímé změně zobrazovaných dat, nýbrž pouze k lokální změně obsahu příslušného datového bufferu. Způsobu přenosu informace z tohoto bufferu do LCD displeje je věnována samostatná část 4.1.6.

V tuto chvíli zařízení nedisponuje prostředky pro automatickou detekci letu. Tato činnost je momentálně na uživateli, který musí manuálně zahájit let prostřednictvím uživatelského rozhraní přístroje. Záznam doby letu je možné ukončit buď manuálně prostřednictvím stejného uživatelského rozhraní, nebo pouhým vypnutím celého přístroje. Způsob stanovení a ukládání délky letu je blíže popsán v části 4.1.7.

Při realizaci komunikace přístroje s aplikací hraje opět zásadní roli přerušení. V rámci obslužné rutiny dochází dle aktuálního nastavení komunikačního *kontextu* ke zpracování nově příchozích znaků. Komunikace je na straně přístroje povolena pokaždé, když je zařízení zapnuto a aktuálně neprobíhá záznam doby letu. Popisu způsobu realizace komunikace na straně přístoroje bude opět věnován prostor v jedné z následujících částí.

4.1.3 Režimy činnosti

Malá velikost displeje nedovoluje zobrazení většího množství informací v jeden okamžik. Proto jsem se rozhodl tento problém vyřešit rozdělením zobrazovaných informací na několik

obrazovek, mezi kterými lze za běhu přepínat. Každá obrazovka sdružuje příbuzné údaje a v některých případech poskytuje rozhraní pro změnu nastavení přístroje.

Přístroj lze zapnout v jednom ze dvou režimů činnosti. První je tzv. *standardní*, který slouží pro potřeby běžných uživatelů. Umožňuje autentizaci uživatele, záznam doby letu, použití stopek a možnost prohlédnutí uživatelského profilu. Druhým režimem je tzv. *administrátorský*. Ten primárně slouží provozovateli letadla pro nastavení a prohlížení vnitřního stavu přístroje.

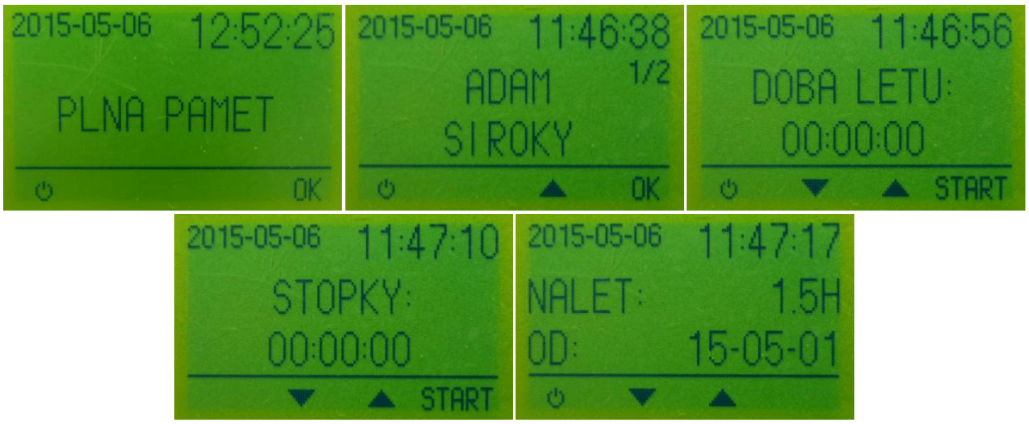
Pomocí dlouhého stisku zapínacího/vypínacího tlačítka je přístroj spuštěn ve *standardním* režimu. Pro spuštění v *administrátorském* režimu je ještě nutné současně držet stisknuté tlačítko *OK*.

Standardní režim

Ihned po zapnutí zařízení je provedena obnova obsahu vnitřního datového LCD bufferu, inicializace všech potřebných modulů, kontrola obsahu vnitřní paměti RTC obvodu a případné zpracování zde uloženého letového záznamu. Bližší popis těchto procedur bude uveden níže v textu. Dále je provedena kontrola stavu zaplnění vnitřní paměti mikrokontroléru. Při plném zaplnění paměti zařízení se zobrazí příslušné varování na obrazovce přístroje. Po potvrzení tohoto varování přechází zařízení do stavu, kdy je zobrazena nabídka jednotlivých uživatelských profilů. V případě dostatku volného místa je tato nabídka zobrazena ihned po zapnutí. Po vybrání a potvrzení jednoho z uživatelů dojde k přihlášení daného uživatele. Tomu jsou poté k dispozici tři obrazovky, mezi kterými může libovolně přepínat pomocí tlačítek se šipkami.

Součástí každé obrazovky je také tzv. *stavový řádek* v dolní části displeje. Ten ukazuje v závislosti na stavu zařízení význam jednotlivých tlačítek. Například ve stavu, kdy jsou aktivní stopky neslouží zapínací/vypínací tlačítko k vypnutí zařízení, ale naopak je použito k resetu stopek. Každá obrazovka dále obsahuje v horní části aktuální datum a čas.

Celkem má uživatel po přihlášení do systému k dispozici obrazovky, které slouží pro zahájení, ukončení a sledování délky aktuálního letu. Dále jsou zde obrazovky se stopkami a provozními údaji uživatele. Těmito údaji jsou celkový počet nalétaných hodin a datum prvního letu. Diagram přechodu mezi jednotlivými obrazovkami/stavy je na obrázku C.1 v příloze. Layouty jednotlivých obrazovek v tomto režimu jsou na obrázku 4.3.



Obrázek 4.3: Layouty LCD displeje ve standardním režimu - (zleva nahoře) stavy: plná paměť, přihlášení uživatele, aktuální let, stopky, uživatelský profil

Administrátorský režim

Zahajovací procedura po zapnutí zařízení v tomto režimu je shodná s předchozím popisem. Opět je provedena inicializace a kontrola stavu zaplnění paměti. Uživatel má v tomto případě o něco více možností a tím pádem i samotných obrazovek. Každá opět obsahuj ve spodní části popis významu jednotlivých tlačítek v závislosti na stavu přístroje.

Celkově v tomto režimu rozlišujeme sedm stavů. V prvním jsou zobrazeny informace o letadle. Konkrétně se jedná o imatrikulaci, celkový počet nalétaných hodin a datum prvního letu. Další obrazovka obsahuje aktuální datum, čas a aktuálně nastavený jazyk. Uživatel si může vybrat češtinu nebo angličtinu. Třetí obrazovka nabízí přehled o množství uživatelských profilů. Kromě zobrazení již existujících uživatelů je zde možné přidávat i nové. Jejich jméno je implicitně tvořeno unikátním řetězcem „NEWx“, kde x nabývá hodnot z rozsahu <1,15>. U každého uživatelského profilu je možné zobrazit počet nalétaných hodin a období, ke kterému se tento údaj vztahuje. Dále je možné zobrazit stav zaplnění vnitřní paměti jednotlivých letových záznamů a případně i odstranit již exportované letové záznamy. Pomocí poslední obrazovky je možné provés obnovu továrního nastavení WiFi modulu, který je součástí přístroje. Diagram přechodů mezi jednotlivými obrazovkami lze opět nalézt v příloze na obrázku C.2. Příklady zobrazovaných dat v tomto režimu jsou na obrázku 4.4.

Obrázek 4.4: Layouty LCD displeje v administrátorském režimu - (zleva nahoře) stavy: plná paměť, informace o letadle, datum - čas - jazyk, uživatelé, uživatelský profil, stav paměti, odstranění exportovaných dat, obnova nastavení WiFi

4.1.4 Konfigurace hodinových signálů mikrokontroléru

Vnitřní hodinový modul BCM je konfigurován tak, že zdrojem hodinového signálu MCLK je krystalový oscilátor LFXT1. Ten využívá externího krystalu o frekvenci 7,3728 MHz, který je součástí DPS zařízení Flytimer. Hodinový signál ACLK, který je použit například

pro časovače, pochází ze stejného zdroje. Jeho frekvence je pouze uvnitř modulu BCM upravována na jednu osminu pomocí frekvenční děličky. Výsledná frekvence je tedy 921,6 kHz. Posledním hodinovým signálem je signál SMCLK. Jeho zdrojem je vnitřní RC oscilátor mikrokontoléru DCO. Ten je kalibrován na frekvenci z rozsahu 4,4 až 5,4 MHz v závislosti na okolních podmínkách.

Smyslem tohoto nastavení je použití přesných a stabilních oscilátorů tam, kde je nutné nebo velmi žádoucí dosáhnout vysoké přesnosti. Typickým příkladem jsou časovače, které v jsou tomto konkrétním případě taktovány hodinovým signálem ACLK. Drobnou nevýhodou je vyšší doba ustálení frekvence po zapnutí daného oscilátoru. V tomto případě však tato skutečnost nečiní žádné problémy, jelikož nejsou využity žádné úsporné režimy MCU, které by deaktivovaly dané oscilátory za chodu. Využití RC oscilátoru se značným frekvenčním rozsahem pro signál SMCLK, který je, jak bude uvedeno dále v textu použit pro taktování modulu USART v režimu I^2C, se může na první pohled zdát podivné. Z principu činnosti komunikace však vyplývá, že není nutné mít stabilní zdroj hodinového signálu. Zároveň platí, že všechny hodnoty z daného frekvenčního rozsahu jsou přípustné, a tím pádem nehrozí selhání komunikace. Odlišné je to ovšem v případě komunikace pomocí modulu USART v režimu UART. Na rozdíl od rozhraní I^2C, kde je hodinový signál přenášen spolu s daty, je u rozhraní UART nutné mít stabilní a přesný zdroj hodinového signálu. Právě proto je modul USART v režimu UART taktován hodinovým signálem ACLK namísto signálu SMCLK.

4.1.5 Konfigurace časovačů mikrokontroléru

Klíčovou roli v celém systému hrají časovače Timer_A a Timer_B. Konkrétní využití jednotlivých kanálů bude vždy uvedeno v související části. Z pohledu konfigurace modulů časovačů je důležité, že *kanál 0* časovače Timer_B je využit jako zdroj periodického přerušení s periodou 177 ms. *Kanál 1* časovače Timer_1 je opět zdrojem periodického přerušení, tentokrát však s periodou jedna sekunda.

Časovač Timer_B je nastaven do režimu *čítání nahoru*. Šestnáctibitový registr TBCCR0, jehož hodnota určuje periodu, je inicializován na hodnotu 20483. Zdrojem hodinového signálu je hodinový signál mikrokontroléru ACLK o frekvenci 921,6 kHz. Ten je uvnitř časovače dále dělen osmi. Časovač pracuje v šestnáctibitovém režimu, přičemž činnost kanálu je povolována, příp. zakazována prostřednictvím povolení nebo zakázání přerušení.

Druhý časovač Timer_A pracuje v režimu čítání nahoru/dolů. Ten je specifický tím, že pro svou činnost implicitně využívá registru TACCR0 *kanálu 0*. To znamená, že tento kanál časovače nemůže být jinak využit. Z těchto důvodů je využito *kanálu 1*, kdy perioda výskytu přerušení je dána dvojnásobkem hodnoty registru TACCR0. Při vstupního hodinovém signálu ACLK o frekvenci 921,6 kHz, který je opět dělen osmi uvnitř modulu, je nutné incializovat registr TACCR0 na hodnotu 57600. Důvodem využití hodinového signálu ACLK je skutečnost, že jeho zdrojem je oscilátor LFXT1, který v porovnání s RC oscilátorem DCO dostahuje podstatně větší přesnosti a stability.

4.1.6 Komunikace s LCD displejem

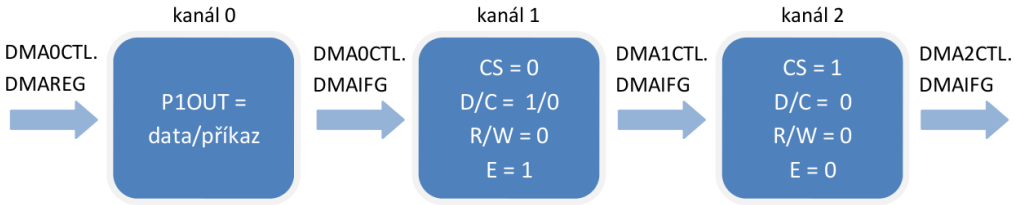
Komunikace s řadičem displeje je realizována pomocí tří vzájemně spolupracujících kanálu modulu DMA. Konkrétní zapojení obvodu bylo již zmíněno v předchozím textu. Pouze by bylo vhodné si připomenout, že LCD displej je připojen pomocí paralelního rozhraní typu *6800*, které se skládá z osmi datových (port P1, registr P1OUT), čtyř řídících (port P2, registr P2OUT) a jednoho reset signálu. Princip zobrazení určitého vzoru na displeji spočívá

v modifikaci lokálního LCD datového bufferu a v jeho následném odeslání do vnitřní paměti displeje. Způsob interpretace obsahu této paměti je poté čistě v režii řadiče příslušného displeje. Tato část si klade za cíl popsat způsob odeslání obsahu lokálního LCD bufferu do vnitřní paměti displeje pomocí modulu DMA.

Princip komunikace

Z pohledu komunikace s LCD řadičem jsou rozlišovány dva typy přenosů - datové, které slouží pro změnu obsahu vnitřní paměti displeje, a řídící, které slouží k nastavení řadiče a adresování vnitřní paměti. Typ přenosu je dán hodnotou řídícího signálu D/C. Konkrétní typ příkazu je poté dán aktuální hodnotou datových signálů. Z pohledu realizace komunikace je tedy způsob odeslání datového a řídícího příkazu odlišný pouze v hodnotě signálu D/C.

Odeslání jednoho datového nebo řídícího bajtu do LCD displeje je realizováno ve třech krocích. V prvním kroku dojde k vystavení příslušných dat nebo příkazu o velikosti jeden bajt na piny portu P1 pomocí *kanálu 0*. *Kanál 1* je nastaven tak, že je spouštěn automaticky po dokončení přenosu *kanálu 0* (dojde k nastavení příznaku přerušení DMAIFG). Tento kanál přenese na výstupní piny portu P2 kombinaci řídících bitů, která zajistí navzrokování obsahu datových signálů řadičem displeje. Po nastavení příznaku přerušení (po dokončení přenosu) *kanálu 1* dojde opět k automatickému zahájení přenosu *kanálu 2*, který vystaví na port P2 kombinaci řídících signálů, jenž odpovídá klidovému režimu. Po dokončení přenosu *kanálu 2* je vyvolána žádost o přerušení. Popisu obslužné rutiny přerušení se bude věnovat samostatná část v textu níže. Graficky je výše uvedený princip spolupráce jednotlivých kanálů modulu DMA ukázán na obrázku 4.5.



Obrázek 4.5: Diagram přenosu jednoho bajtu pomocí DMA kanálů

Výhodou tohoto přístupu je vysoká rychlost a skutečnost, že celý přenos je realizován výhradně modulem DMA. Zároveň platí, že doba jednoho datového přenosu kanálu DMA je větší, než minimální požadovaná doba, po kterou dochází ke zpracování datových signálů řadičem. To znamená, že není nutné přidávat žádné extra zpoždění po dokončení datového přenosu *kanálu 1*.

Zdrojem dat *kanálu 0* je již zmíněný vnitřní datový buffer (lcdData). Ten obsahuje kromě samotných dat i několik příkazů pro adresování vnitřní paměti displeje. Konkrétně je vnitřní datový buffer o velikosti 1048 B logicky rozdělen na osm částí po 131 bajtech. Každá část slouží k popisu právě jedné stránky vnitřní paměti displeje. Hodnota prvních tří bajtů příslušné stránky v tomto bufferu je použita k adresování stránky a nastavení počátečního čísla sloupce. Adresa sloupce je poté s každými přijatými daty uvnitř řadiče automaticky inkrementována. Zbylých 128 bajtů tvoří datovou část stránky. Zdrojem pro datový přenos *kanálu 1* je tzv. řídící buffer (lcdCommand) o velikosti 131 bajtů. Ten obsahuje posloupnost hodnot pro řídící signály tak, aby došlo ke správné interpretaci datových

signálů. Pro potřeby *kanálu 2* je definována konstanta *lcdSteady*, jejíž hodnota odpovídá klidovému stavu řídících signálů portu P2.

Konfigurace modulu DMA

Všechny tři kanály modulu DMA pracují v adresovém režimu *pevná adresa na pevnou adresu* a přenosovém režimu *jeden opakovaný přenos*. Velikost přenášených dat je ve všech případech jeden bajt. Význam jednotlivých atributů byl popsán v části 2.1.3. Zbylé atribut jednotlivých kanálů jsou v tabulce 4.1.

Kanál	DMASA	DMADA	DMAxTSELx	DMAIE
0	&lcdData[x][1]	P1OUT	DMA0CTL.DMAREG	0
1	&lcdCommand[y][2]	P2OUT	DMA0CTL.DMAIFG	0
2	&lcdSteady[3]	P2OUT	DMA1CTL.DMAIFG	1

[1] Adresa prvku x lokálního datového bufferu.
[2] Adresa prvku y řídícího bufferu.
[3] Adresa konstanty s hodnotou, jenž odpovídá klidovému stavu řídících signálů.

Tabulka 4.1: Konfigurace kanálů modulu DMA

Zahájení přenosu a ochrana displeje

Přenos obsahu vnitřního bufferu do paměti displeje je periodicky iniciován přibližně šestkrát za sekundu pomocí jednoho z kanálu časovače TimerB. Zahájení přenosu je tedy provedeno v rámci obslužné rutiny přerušení *kanálu 0* modulu TimerB nastavením bitu DMAREG registru DMA0CTL. Na základě toho dojde k zahájení činnosti prvního kanálu modulu DMA, přičemž po odeslání celého obsahu vnitřního datového bufferu je provedena reinicializace modulu DMA a přechod do výchozího stavu. Samotný přenos obsahu vnitřního bufferu je řízen výhradně modulem DMA dle výše uvedeného principu.

Pro použitý displej platí, že na něm nesmí být zobrazen déle než 30 minut stejný vzor. V opačném případě hrozí nevratné mechanické poškození samotného displeje. Z tohoto důvodu je v rámci obslužné rutiny přerušení modulu TimerB provedena ještě před zahájením přenosu dvojí inverze obsahu LCD displeje. Přesněji řečeno je pomocí dvou příkazů dvakrát provedena změna způsobu interpretace obsahu vnitřní paměti displeje řadičem. Odeslání těchto dvou příkazů je provedeno bez využití modulu DMA. Pouhým okem není tato dvojí inverze postřehnutelná.

Obslužná rutina přerušení modulu DMA

Obslužná rutina přerušení modulu DMA je vyvolána pokaždé, když dojde k dokončení přenosu *kanálu 2*. V rámci obslužné rutiny je provedeno počítání odeslaných bajtů a stránek, přičemž pro odeslání jedné stránky je potřeba odeslat 131 bajtů (3 B příkazy a 128 B data). Po odeslání každého bajtu je proveden test, zda bylo odesláno všech 131 bajtů dané stránky. V opačném případě je inkrementována zdrojová adresa prvních dvou kanálů modulu DMA a zároveň je pomocí bitu DMAREG registru DMACTL0 zahájen další přenos *kanálu 0*. V případě, že byly odeslány všechny bajty dané stránky, je nastavena zdrojová adresa *kanálu 1* na adresu první položky bufferu *lcdCommand*, zdrojová adresa *kanálu 0* je inkrementována a čítač odeslaných znaků nulován. Po odeslání všech stránek se provede

reinicializace modulu DMA a přechod do výchozího/nečinného stavu. Zdrojová adresa *kanálu 2* je neměnná. Funkční schéma obslužné rutiny lze nalézt na obrázku v příloze této práce.

Znaková sada

Znaková sada LCD displeje je tvořena dvěma skupinami znaků, přičemž každá se skládá ze znaků abecedy, číslic, pomlčky, mezery, tečky, lomítka a dvojtečky. Hlavním rozdílem obou skupin je velikost symbolů. První obsahuje znaky o velikosti (šířka x výška) 5x8 (písmena, číslice, lomítko) a 3x8 pixelů (dvojtečka, tečka, mezera, pomlčka) a je primárně použita pro zobrazení navigačních nebo méně významných údajů. Oproti tomu druhá sada, která obsahuje znaky o velikosti 8x12, 5x12, 4x12 a 3x12, je použita pro zobrazení hlavního obsahu.

Jednotlivé symboly jsou uloženy v hlavní paměti jako konstanty. Pro obě znakové sady platí, že nepodporují diakritiku. Textové řetězce z hlavní paměti (jména uživatelů), které obsahují diakritiku, jsou vypisovány na displeji bez diakritiky. Výhodou tohoto přístupu je, že při opakovaném importu a exportu dat nedochází ke ztrátě informace. Pro správnou funkcionalitu však musí platit, že tyto řetězce jsou uloženy pomocí znakové sady *ISO/IEC 8852-2*.

Návrh jednotlivých znaků byl vytvořen v nástroji LibreOffice Calc. Masky příslušných symbolů byly po dokončení návrhu vygenerovány pomocí vytvořeného makra. Jednotlivé symboly obou znakových sad můžeme najít na obrázku v příloze na konci práce.

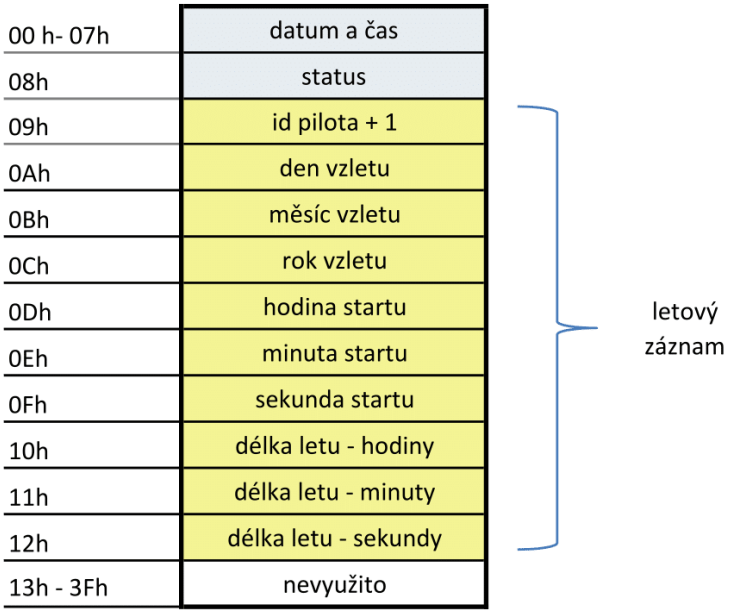
4.1.7 Komunikace s RTC

Komunikace s použitým RTC obvodem skrze rozhraní I^2C je realizována pomocí *kanálu 0* modulu USART. Ten je konfigurován tak, že zdrojem hodinového signálu modulu je signál SMCLK o frekveci 4,4 až 5,4 Mhz (frekvenční rozsah oscilátoru DCO). Perioda vnitřního hodinového signálu modulu a tedy i signálu SCL se střídou 50:50 je pomocí příslušných registrů nastavena na 14 taktů. Výsledná frekence signálu SCL se tedy pohybuje v rozmezí 314 až 385 kHz. To znamená, že je komunikace realizována v tzv. *rychlém režimu* (100 až 400 kHz). Do příslušného registru modulu USART je dála nastavena sedmibitová adresa příslušného RTC obvodu (lze nalézt v dokumentaci). Zvolený adresový režim modulu odpovídá délce této adresy.

Vnitřní pamětový prostor

Část vnitřní paměti RAM RTC obvodu se využívá pro stanovení délky letu. Na začátku každého letu je ve vnitřní paměti obvodu vytvořen letový záznam, který je tvořen hodnotou identifikátoru posádky plus jedna (hodnota 0 je vyhrazena pro stav prázdná paměť), dobou letu a aktuálním datem a časem. Časový údaj je před vytvořením letového záznamu vyčten z RTC obvodu. Doba letu je při vytvoření nastavena na nulu. Organizace paměťového prostoru RTC obvodu je na obrázku 4.6.

Po vytvoření letového záznamu v paměti RTC obvodu dále dochází k přepnutí vnitřního stavu zařízení. Na základě toho je poté prostřednictvím *kanálu 1* modulu Timer_A každou sekundu spuštěn inkrement doby lety ve vnitřní paměti RTC obvodu. Výhodou tohoto přístupu je, že obsah vnitřní paměti obvodu je zachován i v případě ztráty napájecího napětí palubního přístroje (za předpokladu, že je v obvodu zapojena baterie s dostatečnou kapacitou). Zařízení si tedy poradí i s „nekorektním“ ukončením letového záznamu. Zároveň



Obrázek 4.6: Organizace paměťového prostoru RTC obvodu

však nedochází k poškození samotné paměti při opakovaném zápisu jedné buňky tak, jako by tomu bylo u paměti typu Flash.

Po zapnutí palubního přístroje je vždy vyčten a vymazán vymezený prostor vnitřní paměti RTC obvodu. Případný letový záznam je poté uložen na příslušnou adresu do vyhrazené části vnitřní paměti mikrokontroléru.

4.1.8 Komunikace s aplikací

Již bylo zmíněno, že komunikace mezi aplikací a palubním přístrojem je vytvořena pomocí WiFi modulu HLK-RM04, který je k mikrokontroléru připojen pomocí rozhraní UART. Na straně mikrokontroléru je komunikace s tímto modulem realizována opět pomocí *kanálu 0* vnitřního modulu USART. Dvojí využití jednoho kanálu tohoto modulu značí, že je nutné provádět rekonfiguraci za běhu dle vnitřního stavu zařízení. Ve výchozím stavu je tento kanál v režimu UART. Pouze pro potřeby komunikace s RTC obvodem je dočasně provedena rekonfigurace (viz 4.1.7).

Pro tyto účely je modul USART konfigurován na přenosovou rychlost 115200 baudů. Datový rámec je tvořen osmi datovými bity, žádným paritním a jedním stop bitem. Zdrojem hodinového signálu pro modul je vnitřní hodinový signál mikrokontroléru ACLK.

Princip komunikace

Komunikace je realizována pomocí protokolu, jehož podrobný popis byde uveden v části 4.2. Jednou z informací, kterou se v této části později čtenář dozví je to, že po dobu komunikace je udržován jistý *kontext*. Zároveň platí, že veškerá komunikace je iniciována ze strany aplikace. Na straně mikrokontroléru jsou nově příchozí data zpracovávána pomocí přerušení *kanálu 0* modulu USART. Možnost výskytu víceprioritních žádostí o přerušení v době běhu aplikace však může zapříčinit, že dojde ke ztrátě určitých znaků z příchozího datového proudu. Právě proto je výše uvedený protokol navržený tak, že každý datový

proud obsahuje hlavičku pro ustanovení spojení a nastavení *kontextu*. Ten se skládá z typu, směru a množství přenášené informace.

Komunikace je zahájena přijetím bajtu s hodnotou z rozsahu <0, 15>. Hodnota tohoto bajtu udává *kontext* komunikace. Po jeho nastavení je dále zakázáno přerušení časovače Timer_B a případně i pozastaven právě probíhající přenos obsahu vnitřního datového LCD bufferu modulem DMA. Taktéž je povolena činnost kontrolního časovače, který využívá periodického výskytu přerušení (jedna sekunda) *kanálu 1* modulu *Timer_A*. Po této bezpečnostní proceduře, která zajišťuje spolehlivost komunikace, je odeslána příslušná odpověď a zahájeno čekání na definovaný počet bajtů. Po přijetí příslušného počtu bajtů je provedeno zpracování přijatých dat, povolení činnosti modulu DMA, povolení přerušení časovače Timer_B a vynulování *kontextu*. Pro vybrané druhy komunikace je část přijetí a zpracování přijatých dat vynechána.

Příznak kontrolního časovače je resetován pokaždé, když dojde k přijetí nového bajtu z datového proudu. Naopak je nastaven pokaždé, když je očekáván příjem ještě alespoň jednoho bajtu od WiFi modulu. V případě výskytu přerušení časovače Timer_A a současném nastavení příznaku dojde k resetu komunikačního kontextu, povolení činnosti modulu DMA a povolení přerušení časovače Timer_B. Funkční schéma realizace komunikace z pohledu mikrokontroléru je na obrázku F.1 v příloze této práce.

Konfigurace modulu HLK-RM04

Předpokladem pro funkční komunikaci mezi přístrojem a aplikací je existence infrastrukturní WiFi sítě. K ní dále musí být připojen klient, na kterém běží podpůrná aplikace. Ta je poté dostupná na portu 26500.

Modul HLK-RM04 je konfigurován tak, že se automaticky snaží připojit pomocí protokolu TCP ke službě na portu 26500 klienta s předem definovanou IP adresou v rámci konkrétní sítě. Režim činnosti modulu je *UART - Wi-Fi klient*. Parametry dané sítě včetně IP adresy klienta je nutné nakonfigurovat pomocí webového rozhraní. K tomu je nutné být připojen ve stejné síti a znát IP adresu samotného WiFi modulu. Pro případ, kdy se není možné připojit do stejné sítě s WiFi modulem a tím pádem ani nelze provést rekonfiguraci pomocí webového rozhraní, je v palubním přístroji v administrátorském režimu možné provést obnovu továrního nastavení WiFi modulu. Ten se poté nachází v režimu *UART - WiFi AP*, kdy modul tvoří přístupový bod do sítě s parametry: SSID - *HI-LINK_243D*, heslo - *12345678*, zabezpečení - *WPA2 AES*. Po přihlášení do této sítě je poté na adrese *192.168.16.254* dostupné webové rozhraní pro konfiguraci modulu. Přihlašovací údaje do nastavení jsou *admin/admin*. Kromě parametrů dané sítě je po obnově výchozího nastavení ještě nutné zkontrolovat a případně nastavit množinu hodnot, která je v tabulce 4.2.

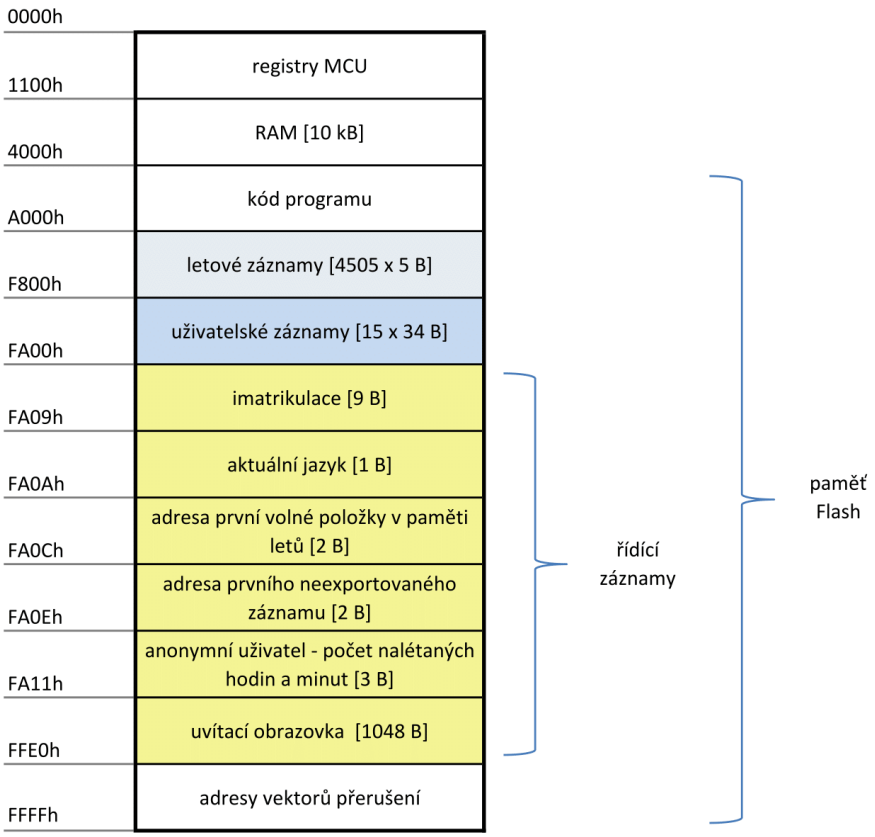
Parametr	**Hodnota**
serial configure	115200,8,n,1
serial framing lenth	160
serial framing timeout	0
network protocol	TCP
network timeout	0
TCP auto connect	disable
TCP client auto check	enable

Tabulka 4.2: Konfigurace WiFi modulu HLK-RM04

4.1.9 Organizace pamětového prostoru mikrokontroléru

Při realizaci návrhu zařízení bylo nutné zohlednit omezenou kapacitu vnitřní paměti a zvolit vhodný způsob ukládání dat. Zvolený formát, kterému se věnuje tato část, nakonec zajišťuje více než dostatečnou kapacitu datového úložiště. Konkrétně se jedná o 4 505 letových záznamů. Na první pohled se může zdát, že je tento údaj natolik velký, že je zbytečné přistupovat k níže popsanému způsobu komprese dat. Na druhou stranu dává tento způsob dostatek prostoru pro případné navýšení počtu ukládaných dat bez nutnosti zasahovat do stávajícího formátu dat. Drobná omezení, která pramení z omezeného prostoru pro ukládání vybraných atributů jednotlivých záznamů, lze s přihlédnutím k zamýšlenému způsobu využití systému bez výhrad akceptovat.

Do virtuálního adresového prostoru mikrokontroléru MSP430F1611 o velikosti 65 535 B jsou namapovány registry modulů mikrokontroléru, vnitřní paměť RAM (10 kB) a vnitřní paměť Flash (48 kB). Ta je rozdělena do tří částí, kdy první část (adresy 4000h - 9FFFh) slouží pro uložení kódu programu, druhá část (adresy A000h - FFDFh) jako datové úložiště a třetí část (adresy FFE0h - FFFFh) je použita pro uložení adres přerušovacích vektorů. Graficky je organizace paměti zobrazena na obrázku 4.7.



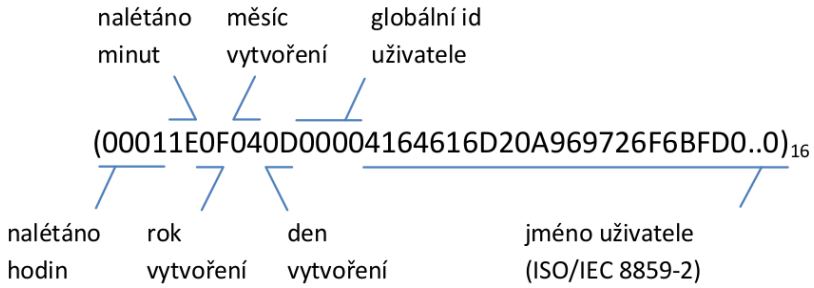
Obrázek 4.7: Organizace virtuálního adresového prostoru mikrokontroléru

Datové úložiště je využito pro uložení letových, řídících a uživatelských záznamů. Dále je zde uložen snímek uvítací obrazovky, kterým je při zapnutí zařízení inicializován vnitřní datový LCD buffer.

Uživatelské záznamy

Zařízení umožnuje uložení až patnácti uživatelských záznamů (účtů) zároveň. Každý účet je tvořen strukturou o velikosti 34 B. Ta se zkládá z: celkového počtu nalétaných hodin (dva bajty) a minut (jeden bajt), datem vytvoření uživatele (tři bajty), globálním identifikátorem (dva bajty) a jménem (26 bajtů). Jméno uživatele je explicitně kódováno do znakové sady *ISO/IEC 8859-2*. Důvody použití této sady byly vysvětleny v části 4.1.6. Globální identifikátor daného uživatele odpovídá hodnotě interního identifikátoru v databázi podpůrné aplikace. Tato hodnota je zde pouze pro potřeby importu dat do databáze ze zařízení. Z pohledu činnosti palubního přístroje je tato hodnota nezajímavá.

Na obrázku 4.8 je příklad uložení uživatelského účtu s atributy: počet nalétaných hodin - *1.5*, datum vytvoření - *13.04.2015*, globální identifikátor - *0*, jméno - *Adam Široký*.

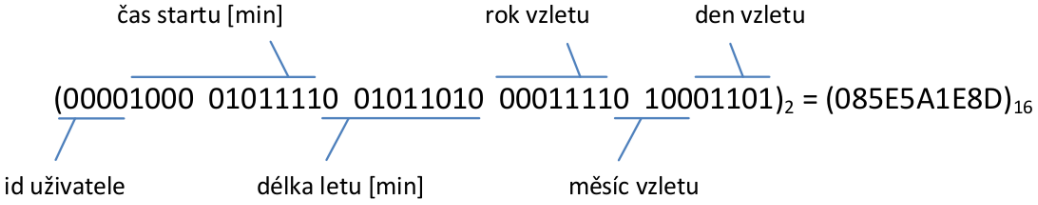


Obrázek 4.8: Ukázka uložení jednoho uživatelského záznamu v paměti

Letové záznamy

Každý letový záznam je tvořen identifikátorem (id) uživatele (čtyři bity), délkou letu v minutách (devět bitů), dnem (pět bitů), měsícem (čtyři bity) a kalendářním rokem (sedm bitů), kdy byl let zahájen. Dále je zde čas zahájení letu v minutách (devět bitů), který se počítá od půlnoci daného dne. Velikost jednoho záznamu je tedy pět bajtů (40 bitů). Díky tomuto způsobu ukládání dat lze do vyhrazeného prostoru vnitřní paměti Flash uložit až 4 505 takovýchto záznamů.

Omezením, které pramení ze zvoleného způsobu ukládání dat, je maximální možná délka letu - osm a půl hodiny. V oblasti sportovního létání je však tato hodnota zcela dostatečná. Pro určení uživatele, který daný let vykonal, slouží již zmíněné čtyři bity na začátku každého záznamu. Hodnota *15* je rezervována pro anonymního/nepřihlášeného uživatele. Zbývající hodnoty slouží k identifikaci aktuálně přihlášeného uživatele na daném zařízení (id uživatele je dáno pořadím uživatelských účtů v paměti). Příklad uložení letového záznamu s atributy: id uživatele - *0*, čas - *17:51*, datum - *13.04.2015* a délka letu - *90 minut* je na obrázku 4.9.



Obrázek 4.9: Ukázka uložení jednoho letového záznamu v paměti

Řídící záznamy

Pro správný chod palubního přístroje je nutné ukládat některé konfigurační informace do nevolatilní paměti tak, aby nedošlo ke ztrátě aktuálního nastavení při ztrátě napájení. Pro tyto účely je v hlavní paměti vyhrazeno 17 bajtů, které obsahují: imatrikulaci letadla (devět bajtů), jazykové nastavení (jeden bajt), adresu první volné položku v paměti letových záznamů (dva bajty), koncovou adresu naposledy exportovaných dat (dva bajty) a celkový počet nalétaných hodin (dva bajty) a minut (jeden bajt) anonymní posádky.

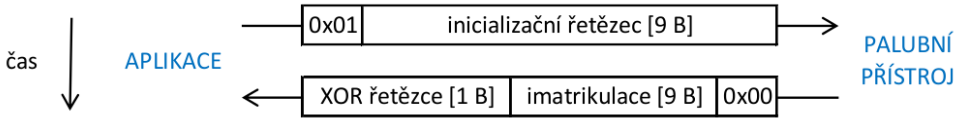
4.2 Komunikační protokol

Pro potřeby realizace systému bylo nutné navrhnout vlastní komunikační protokol. Použitý protokol TCP transportní vrstvy sám o sobě zajišťuje spolehlivé doručení jednotlivých paketů ve správném pořadí. Tento popis se však stahuje pouze na komunikaci mezi aplikací, přesněji řečeno hostitelským počítačem, a použitým WiFi modulem, který je součástí palubního přístroje. Roli tohoto WiFi modulu lze ve zkratce nalézt v kapitole 3.3 podrobněji v části 4.1.8. Důležitou skutečností je však to, že z pohledu aplikace není žádným způsobem zaručena dostupnost palubního přístroje na síťovém rozhraní v daný okamžik. Zároveň není na straně palubního přístroje žádná hardwarová podpora ve formě bufferu. Tato skutečnost znamená, že bylo nutné nalézt vhodný způsob, který zaručuje, že komunikace bude zahájena pouze ve chvíli, kdy je zajištěna dostupnost obou komunikujících stran.

Komunikace mezi aplikací a přístrojem funguje na principu *žádost - odpověď*. V roli iniciátora je vždy aplikace. Ta na začátku každé komunikace vysílá datový rámec, který je kromě jednoho speciálního případu tvořen právě jedním bajtem (typem komunikace). Na základě hodnoty tohoto bajtu dojde k nastavení komunikačního *kontextu* na straně přístroje. Tento *kontext* je udržován až do doby, než proběhne předem definovaná výměna dat nebo naopak dojde k vypršení definovaného časového prostoru (viz část 4.1.8). V obou případech dojde k resetu *kontextu* na straně palubního přístroje. Celkově jsou rozlišovány čtyři typy komunikace, jejichž popisu je věnován zbytek této části.

4.2.1 Autentizace

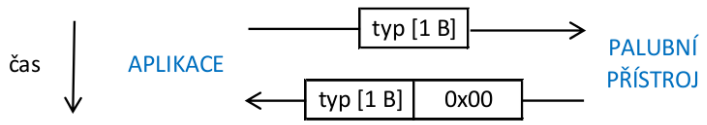
Autentizační datový rámec, pomocí kterého probíhá autentizace palubního přístroje vůči aplikaci, je výjimkou oproti všem ostatním typům komunikace. Velikost datového rámce, prostřednictvím kterého aplikace nastavuje *kontext*, je v tomto případě zvětšena o inicializační řetězec. Ten je zaslán spolu s jedním bajtem z aplikace do přístroje. Délka inicializačního řetězce je pevně stanovena na devět znaků (bajtů). Za validní odpověď je považován datový rámec, který se skládá z: hodnoty, která vznikne jako XOR jednotlivých znaků přijatého inicializačního řetězce, imatrikulací letadla uložené ve vnitřní paměti zařízení a jedním bajtem s hodnotou 0x00 (tzv. *stop* bajt). Graficky je průběh autentizace na úrovni datových rámců ukázán na obrázku 4.10.



Obrázek 4.10: Průběh autentizace na úrovni datových rámců

4.2.2 Řídící komunikace

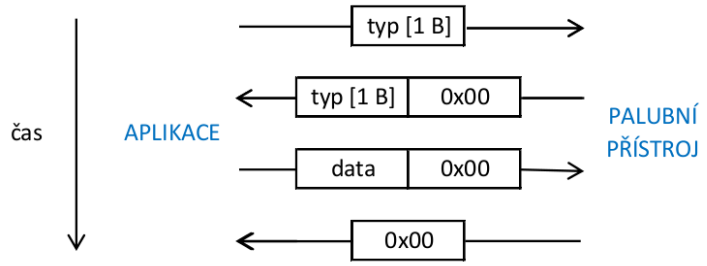
Tento typ komunikace se používá pro vymazání obsahu vnitřní paměti palubního přístroje a pro kontrolu stavu připojení (tzv. ping rámce). Odpovědí na datový rámec, který obsahuje pouze pouze řídící bajt, je ze strany palubního přístroje datový rámec tvořený přijatým bajtem a *stop* bajtem. Graficky je tento průběh ukázán na obrázku 4.11.



Obrázek 4.11: Průběh řídící komunikace na úrovni datových rámců

4.2.3 Import dat do přístroje

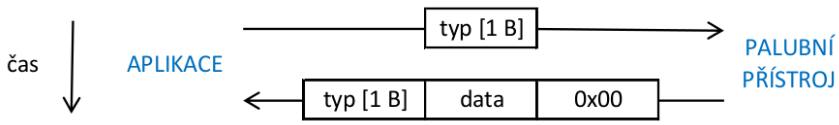
O něco složitější průběh komunikace nastává v případě, kdy je nutné importovat data do zařízení. První polovina komunikace je stejná jako v předchozím případě. Aplikace pomocí řídící sekvence nastaví komunikační *kontext*. Ta svou připravenost přijmout data potvrdí odesláním již známého datového rámce, který mimo jiné obsahuje přijatý bajt z prvního rámce. Následující rámec, který je odeslán z aplikace, obsahuje definovaný počet datových bajtů a jeden *stop* bajt. Přijetí předem známého počtu datových bajtů potvrdí zařízení odesláním rámce obsahující *stop* bajt. Ilustrace průběhu komunikace je na obrázku 4.12.



Obrázek 4.12: Průběh importu dat do přístroje na úrovni datových rámců

4.2.4 Export dat z přístroje

Poslední typ komunikace slouží pro export dat ze zařízení do aplikace. Aplikace si vyžádá pomocí příslušné hodnoty inicializačního datového rámce příslušní data. Odpovědí na tuto žádost je rámec tvořený typem dat, samotnými daty a *stop* bajtem. Průběh komunikace je zobrazen na obrázku 4.13.



Obrázek 4.13: Průběh exportu dat z přístroje na úrovni datových rámců

4.3 Podpůrná aplikace

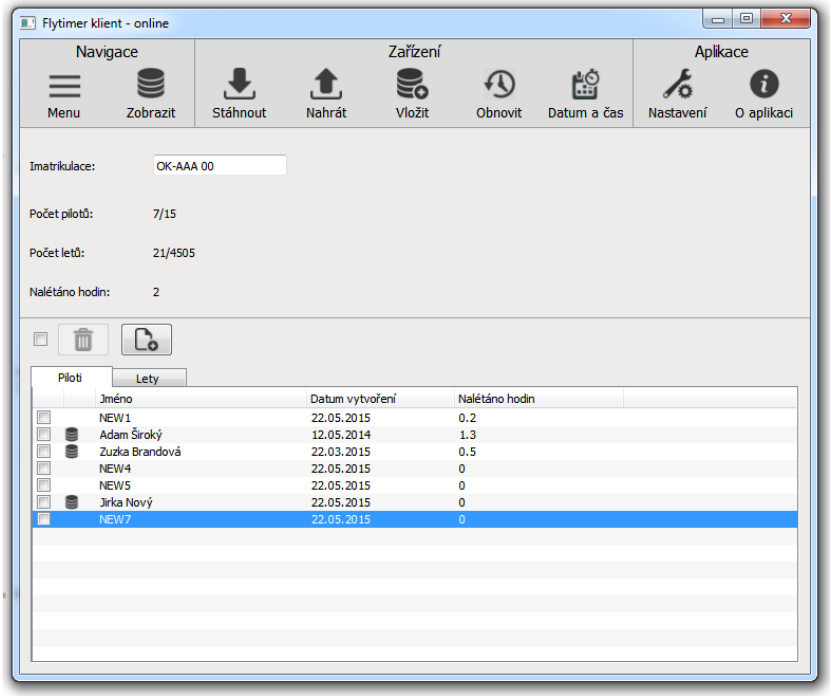
Nedílnou součástí této práce byla tvorba vhodné aplikace, která by sloužila pro správu přístroje a dat. Tato část je rozdělena na dvě menší, kdy první se zabývá popisem využití a možností dané aplikace. Druhá část poté popisuje způsobu realizace vybraných částí aplikace.

4.3.1 Popis využití

Z uživatelského pohledu lze aplikaci rozdělit na dvě hlavní části, mezi kterými lze za chodu přepínat. První část je výhradně určena pro správu palubního přístroje. Druhá naopak slouží výhradně pro správu dat v centrální databázi. Většina dat je vizualizována pomocí tabulek, které poskytují možnost přímé editace vybraných hodnot. Například nelze měnit přímo počet nalétaných hodin daného uživatele. Lze však editovat letové záznamy a nepřímo tak měnit hodinový nálet jednotlivých pilotů, popřípadě letadel v databázi.

Přístrojová část

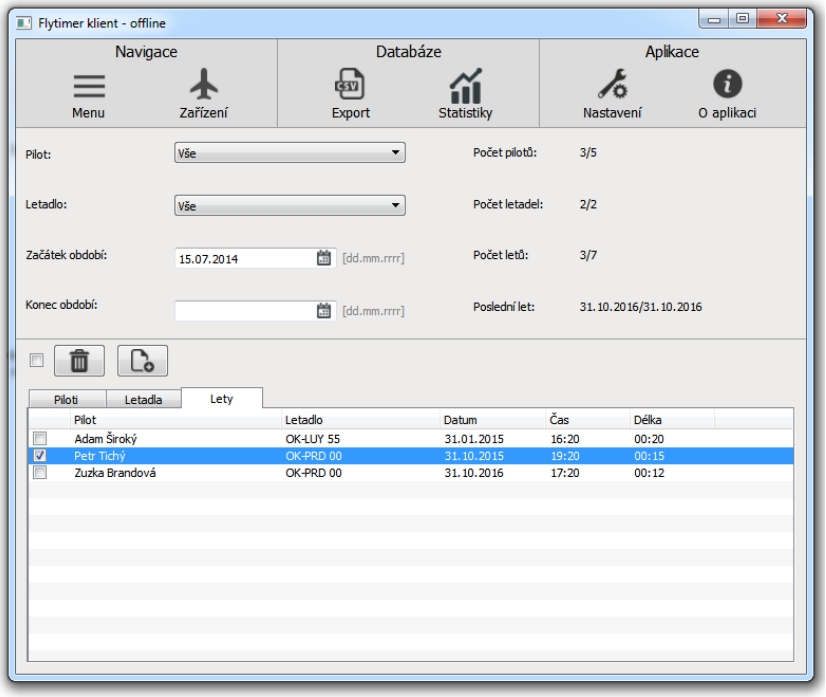
Tento režim slouží pro inicializaci časových a provozních údajů zařízení, importu, export, editaci uživatelských a letových záznamů a importu stažených údajů do centrální databáze. Většina těchto voleb je dostupná pouze ve stavu, kdy je detekováno aktivní spojení mezi aplikací a přístrojem. Snímek aplikace v tomto režimu je na obrázku 4.14. Důležitou vlastností aplikace je, že importovaná data z přístroje nejsou automaticky odstraňována z jeho vnitřní paměti. Důvod, proč je tomu tak, je vcelku jednoduchý. Kapacita vnitřní paměti palubního přístroje je více než dostatečná. Palubní přístroj zároveň umožňuje přímé odstranění exportovaných dat, takže je možné uvolnit určitou část paměti i mimo dosah příslušné sítě. Naopak výhodou tohoto přístupu je určitý druh zálohy dat.



Obrázek 4.14: Náhled aplikace - správa přístroje

Databázová část

Naopak tato část slouží pouze pro správu databázových dat. Ta je možné filtrovat pomocí imatrikulace letadla, jména pilota, počátečního nebo koncového data, popřípadě vzájemné kombinace. Pro snažší orientaci je pro zadaná kritéria zobrazován počet vyfiltrovaných dat. Uživatel má zároveň k dispozici grafy vývoje počtu nalétaných hodin v závislosti na čase. Aktuální obsah databáze je možné kdykoliv exportovat do souboru formátu CSV[1]. Vždy je vykreslen časový úsek, který je dán aktuálním datem a datem prvního letu daného uživatele v databázi. Snímek aplikace v tomto režimu je na obrázku 4.15.



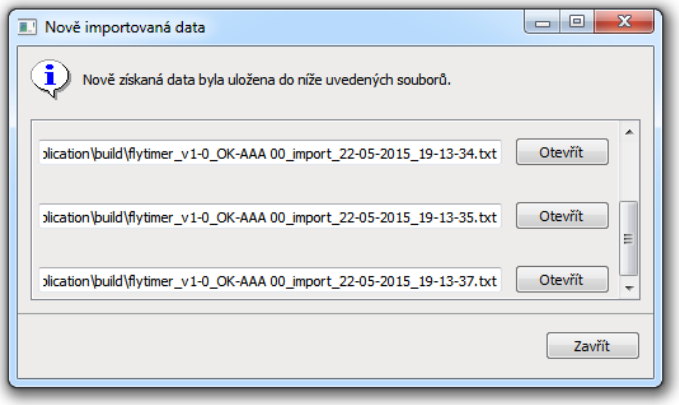
Obrázek 4.15: Náhled aplikace - správa dat

Logování

Prostřednictvím menu lze povolit nebo zakázat logování dat, která jsou přenášena mezi aplikací a palubním přístrojem. Každý *log* je uložen jako textový soubor, který v názvu souboru obsahuje směr komunikace, datum a čas. Adresář, do kterého jsou tyto soubory ukládány je nastavován v menu aplikace. Takto uložená data lze kdykoliv načíst prostřednictvím aplikace a použít je například pro obnovu dat v palubním přístroji. Smyslem této funkce je tedy primárně záloha dat. Zároveň však slouží k ukládání automaticky importovaných dat. Palubní přístroj je totiž konfigurován tak, že se snaží samočinně importovat data do aplikace. Ta jsou v případě, kdy je povoleno logování a je spuštěna aplikace, automaticky ukládána do jednotlivých logovacích souborů. Uživatel je o nově importovaných datech informován prostřednictvím vyskakovacího okna (obrázek 4.16), které obsahuje seznam všech takto vytvořených souborů. Výhodou tohoto mechanismu je, že nevyžaduje interakci s uživatelem. Pouze je nutné mít správně nastavenou a spuštěnou aplikaci. Uživatel poté může

[1] CSV (Comma-separated Values)

prostřednictvím vytvořených záznamů lehce načíst poslední známý stav zařízení a například provést import dat do databáze.



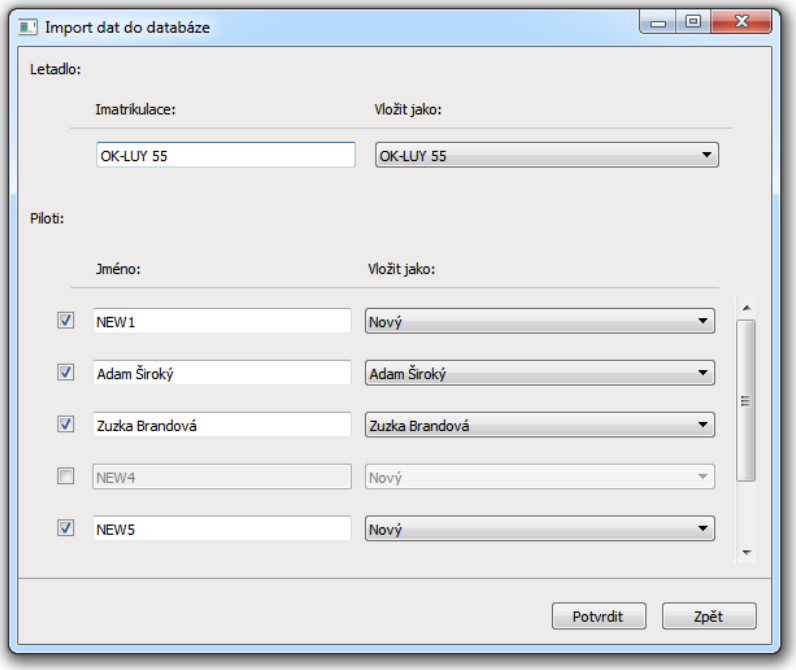
Obrázek 4.16: Náhled aplikace - automaticky importovaná data

Import dat do databáze

Aktuálně zobrazená data lze pomocí dialogového okna z obrázku 4.17 importovat do centrální databáze. Z pohledu importu letových záznamů je však nejprve nutné provést mapování letadla a jednotlivých posádek na již existující nebo nové databázové záznamy. Mou snahou bylo vytvářet příslušné mapování automaticky tak, aby v ideálním případě uživatel prováděl pouze kontrolu zadaných údajů.

U mapování letadla je to velmi snadné. Integritním omezením databáze, jejíž schéma je uvedeno v části 4.3.2, je to, že hodnota imatrikulace musí být unikátní. To znamená, že dle aktuální hodnoty imatrikulace a stavu databáze je uživateli nabídnuto dostupné mapování, případně je mu umožněno přidání nového letadla do databáze. U uživatele je situace poněkud složitější. V části 4.1.9 bylo zmíněno, že každý uživatelský záznam obsahuje položku *globální identifikátor*. Tento údaj je číselně roven primárnímu klíči v tabulce *piloti* databáze, který je vytvářen pomocí autoinkrementu s tím, že počáteční hodnota je jedna. Hodnota nula v položce *globální identifikátor* v uživatelském záznamu značí, že zadaný pilot buď v databázi není, nebo prozatím nedošlo k uložení příslušného mapování do paměti přístroje. Uživateli je v tomto případě primárně umožněno přidat daného pilota do databáze jako nového. Může však ručně pomocí rozbalovacího menu provést mapování na jeden z existujících záznamů v dané tabulce. Po dokončení mapování každého z pilotů je nastavena položka *globální identifikátor* u každého uživatelského záznamu. V případě změny mapování nebo vložení nové položky do tabulky *piloti* je doporučené provést export dat zpět do palubního přístroje. Tato procedura bude mít za následek to, že při všech následujících importech bude možné provést mapování pilotů na existující položky v databázi zcela automaticky.

Po dokončení mapování je proveden import jednotlivých letových a případně i uživatelských záznamů. Všechny správně importované záznamy budou po dokončení importu označeny ikonou databáze.



Obrázek 4.17: Náhled aplikace - export dat do databáze

4.3.2 Způsob implementace

Aplikace je napsána pomocí frameworku Qt verze 5.4. Grafické uživatelské rozhraní je vytvořeno výhradně pomocí jazyka QML. Aplikace podporuje češtinu a angličtinu. Konkrétní jazyk je vybrán automaticky po spuštění aplikace dle systémového nastavení, přičemž pro všechna systémová nastavení kromě češtiny je vybrána angličtina. Překlady jsou vytvořeny pomocí nástroje Qt Linguist.

Samotnou aplikaci není nutné na cílovém počítači instalovat. Archiv, ve kterém je umístěna obsajuje přeloženou verzi aplikace spolu se všemi potřebnými knihovnami. Aplikaci lze spusti na operačních systémech Windows 7 a 8.1. Minimálním hardwarovým nárokem odvíjejícím se od použité platformy je nutná podpora OpenGL verze 2.1 a vyšší. Vytvořená aplikace spadá pod licenci GPL.

Síťová komunikace

Pro zajištění plynulého chodu celé aplikace veškerá bezdrátová komunikace probíhá ve vlastním vlákně. Pro její realizaci byly vytvořeny třídy *Wifi* a *MyServer*. Třída *Wifi* je potomkem třídy *QThread*. Tato třída obsahuje několik metod, jejíž definice jsou opatřeny makrem *Q_INVOKABLE*. Takto definované metody lze poté volat přímo z QML. Druhý směr komunikace je realizován pomocí signálů. Členem třídy *Wifi* je objekt třídy *MyServer*. Jejím členem je objekt třídy *QTcpServer* (viz část 2.5.2). Pomocí ní třída *MyServer* implemetuje TCP server, ke kterému se poté připojuje palubní přístroj.

Třída *Wifi* tedy slouží jako rozhraní mezi třídou *MyServer* a QML. Tato C++ třída je registrována do QML pomocí funkce *qmlRegisterType*. V rámci definice třídy je reimplementována metoda *run* třídy *QThread*. Tato metoda je volána automaticky po vytvoření objektu třídy *Wifi*. V jejím těle je vytvořen objekt třídy *MyServer*, provedeno napojení jednotlivých signálu a slotů tříd *MyServer, Wifi* a spuštění TCP serveru. Vykonávání těla

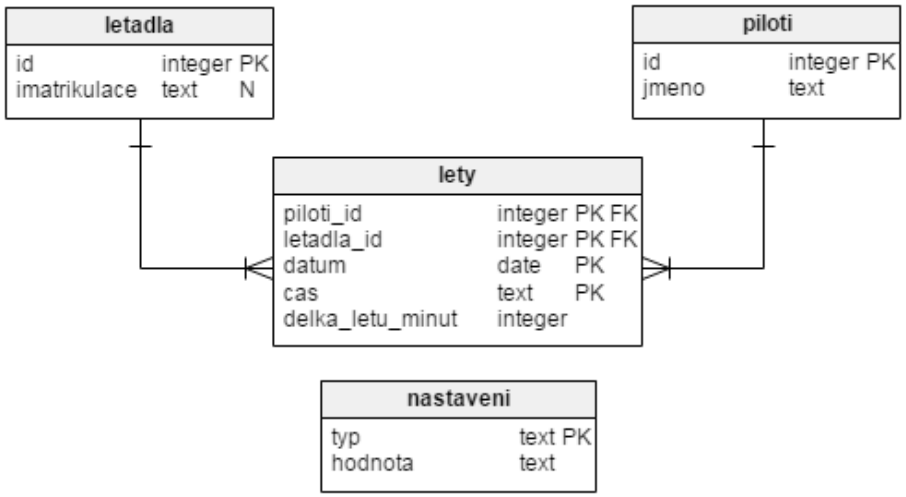
funkce je skončeno až po ukončení celé aplikace, kdy je přijat signál *quit*. Maximální možný počet přihlášených klientů k serveru v jeden okamžik je omezen na jednoho klienta.

Veškerý právě probíhající přenos je pro větší přehlednost signalizován pomocí *progress baru*. Uživatelské a letové záznamy jsou přenášeny pomocí definovaného protokolu (viz 4.2) ve formátu, který je použit pro uložení těchto dat do vnitřní paměti mikrokontoléru. Detekce aktivního stavu spojení mezi přístrojem a serverem je realizována pomocí periodického zasílání tzv. ping rámců.

Navržený protokol pracuje na principu *žádost - odpověď*, přičemž žadatelem je vždy aplikace. Odpověď od přístroje musí být vždy doručena do vypršení určitého časového rámce. V opačném případě dojde k ukončení právě probíhajícího přenosu a zobrazení příslušného chybového hlášení.

Databáze

Na obrázku 4.18 je schéma jednoduché databáze, která byla vytvořena pro potřeby správy dat a nastavení aplikace. Databáze je v druhé normální formě. Na základě dohody s členy již zmíněného aeroklubu jsem se prozatím rozhodl do datbáza zahrnout pouze ty údaje, které jsou nutné pro chod celého systému. Je však velmi snadné a rychlé v budoucnu přidat atributy k jednotlivým entitám. Na straně aplikace je pro implementaci využito modulu *Local Storage*, který byl popsán v části 2.7.3. Databáze momentálně obsahuje tabulky pilotů, letadel, letových záznamů a tabulku pro uložení nastavení aplikace.



Obrázek 4.18: Schéma databáze

4.4 Přizpůsobení systému

Stejně jako u většiny zařízení je i v tomto případě nutné před nasazením systému provést určitou sekvenci inicializačních kroků. Konkrétně se jedné o tuto posloupnost:

naprogramování a inicializace MCU : Součástí přiloženého CD je projekt v nástroji IAR Embedded Workbench verze 6.10.7. Současně daný adresář obsahuje soubory *memory.txt* a *msp430f1611.ddf*, které slouží pro inicializaci obsahu vnitřní paměti Flash mikrokontoléru. Tu lze provést v již zmíněném vývojovém prostředí pomocí

konfiguračního souboru *memory.txt*. Ještě před tím je ovšem nutné nahradit soubor *\430\config\debugger\msp430f1611.ddf* v adresáři, kde je nainstalované vývojové prostředí, přiloženým konfiguračním souborem *msp430f1611.ddf*. Tento soubor se liší od původního pouze řádkem, který určuje zda lze z paměti Flash mikrokontoléru pouze číst (výchozí nastavení) nebo do ní lze i zapisovat z vývojového prostředí. Samotná inicializace vnitřní paměti se skládá z nastavení výchozích hodnot provozních a letových údajů, ukazatelů do paměti a vytvoření virtuálního vnitřního datového bufferu, který slouží pro komunikaci s LCD displejem. Ten kromě výchozích hodnot datových bajtů obsahuje také několik řídících hodnot pro adresování vnitřní paměti LCD displeje.

inicializace WiFi modulu : WiFi modul, který je součástí palubního přístroje, je nutné nakonfigurovat tak, aby se samočinně připojoval do příslušné WiFi sítě. Popis konfigurace, kterou je vhodné zahájit obnovou továrního nastavení pomocí uživatelského rozhraní palubního přístroje, byl uveden v části 4.1.8.

inicializace provozních údaju přístroje : Posledním krokem je inicializace všech provozních údajů v palubním přístroji prostřednictvím podpůrné aplikace. Konkrétně je nutné nastavit datum a čas, imatrikulaci letadla a uživatelské účty. Doporučený postup je takový, že je nutné vytvořit v dané aplikaci příslušné uživatele, vložit je do databáze, popřípadě vytvořit dané uživatele z existujících záznamů v databázi a poté provést export dat do palubního přístroje. Cílem tohoto postupu je zjednodušit pozdější proces importu letových dat z paměti přístroje do databáze.

4.5 Využitelnost systému

Popisovaný systém je primárně určen pro potřeby aeroklubu v Kotvrdovicích. V současném stavu ho lze vzít a zcela určitě nasadit do reálného provozu. Nevýhodou tohoto systému je, že neumí automaticky detekovat počátek a konec letu. V případě detekce konce letu se nejedná o příliš velký problém a za uspokojivý lze označit současný stav, kdy konec letu je dán buď ruční volbou nebo vypnutím celého přístroje, včetně případu, kdy dojde ke ztrátě napájecího napětí. Počátek letu je momentálně nutné manuálně označit prostřednictvím uživatelského rozhraní palubního přístroje. Zvažovaných možností, jak by tento problém šlo vyřešit, bylo několik. Konkrétně by bylo možné prostřednictvím impulzů ze zapalování určit aktuální otáčky motoru s tím, že za start lze považovat situaci, kdy dojde k překročení určité hodnoty otáček motoru po předem definovaný časový interval. Časový úsek by zde byl pro vyloučení motorové zkoušky, během které může po velmi krátkou dobu dojít k dosažení maximálních otáček motoru. Typicky je však tato doba v porovnání s délkou startovací procedury výrazně kratší.

Další možností, jak lze určit počátek letu, je měřením polohy pomocí GPS přijímače. Ideální variantou by bylo rozšířit navržený přístroj o modul, který by pracoval jako rádiový výškoměr. Jeho výstup by poté bylo možné použít jak pro detekci počátku i konce letu, tak i jako přistávacího asistenta. Ten by byl přínosem především pro začínající piloty a provozovatele leteckých škol. Možnosti využití radaru pro stanovení výšky pohybujícího se objektu nad terénem se momentálně věnuje student Marek Absolon v rámci bakalářské práce na zdejší fakultě. Dle předběžných výsledků se zdá, že jeho řešení lze s úspěchem v budoucnu využít, a právě proto jsem se po dohodě s vedoucím této diplomové práce v tuto chvíli rozhodl automatickou detekci letu vynechat s tím, že je do budoucna počítáno s rozšířením navrženého řešení o tento modul. Ten poskytuje ze všech zvažovaných možností

největší přidanou hodnotu. Takto rozšířený systém by poté dle slov pana Michala Seiferta nejspíše našel uplatnění i u ostatních provozovatelů ultralehkých letadel.

4.6 Testování systému

Součástí řešení bylo provedení softwarových testů, které měly za úkol ověřit správnost implementace všech částí systému. Pokročilé testy nebyly náplní této práce. Z pohledu budoucího využití systému by však zcela jistě bylo vhodné provést důkladné testy, které by ověřily například mechanickou odolnost tlačítek a elektrickou odolnost vstupů palubního přístroje. Zároveň by bylo vhodné provést testování i v extrémních teplotních podmínkách a ověřit tak výrobci udávané parametry. Dále by bylo možné zaměřit se na kvalitu navrženého uživatelského rozhraní palubního přístroje a aplikace. Samotné testování by mohlo probíhat formou úkolů, které by byly zadávány uživatelům. Sledovány by byly jejich reakce a časy, za které jsou schopni dané úkoly splnit.

Kapitola 5

Závěr

Cílem práce bylo vytvořit vestavěný systém, který využívá technického návrhu zařízení Flytimer pro záznam provozních údajů ultralehkých letadel a vrtulníků. Zároveň bylo nutné navrhnout a implementovat aplikaci, která by sloužila pro administraci celého systému. Veškeré tyto cíle byly splněny.

V rámci zadání jsem musel nastudovat problematiku mikropočítačů, principy vytváření vestavěných systémů a v neposlední řadě také způsob vytváření síťových a databázových aplikací. Na základě těchto znalostí jsem poté mohl provést určité změny v existujícím návrhu. Ty se týkaly výhradně způsobu správy dat a komunikace se zařízením, které slouží pro záznam provozních údajů letounu. Současně jsem vytvořil návrh aplikace pro správu dat prostřednictvím vestavěné databáze a pro komunikaci s palubním přístrojem pomocí bezdrátové sítě WiFi. Tyto návrhy jsem posléze zrealizoval pomocí běžně používaných komponent a technologií. Dosažené výsledky jsem průběžně konzultoval s budoucími uživateli z řad zástupců aeroklubu v Kotvrdovicích. Jejich připomínky a náměty byly promítnuty do výsledného systému. Jeho funkcionalitu jsem názorně demonstroval vedoucímu diplomové práce, který je současně aktivním pilotem. S ním jsem poté diskutoval dosažené výsledky a možnosti vytvořeného systému.

Vytvořený systém slouží jako elektronický palubní deník, který umožnuje autentizaci uživatelů a ukládání jednotlivých letových záznamů. Systém lze rozdělit na palubní přístroj a podpůrnou aplikaci. Přístroj slouží jednotlivým uživatelům daného letadla primárně k identifikaci a záznamu letových údajů. Provozovateli daného letadla je zároveň určena aplikace, která slouží pro celkovou správu palubního přístroje. Ta poskytuje prvek pro komunikaci s přístrojem pomocí bezdrátové WiFi sítě, jeho inicializaci, přizpůsobení a v neposlední řadě i samotnou databázi pro správu dat. Navržený systém tedy lze označit za plnohodnotný prostředek pro monitorování určité množiny provozních údajů letounu.

Největším přínosem pro mě bylo seznámení se s pokročilým návrhem a možnostmi využití vestavěných systémů. Zejména bych rád zmínil množství rad a poznatků z oblasti hardware, které mně předal vedoucí diplomové práce pan profesor Pavel Zemčík. Jako další výhodu bych také označil komplexnost celého systému.

Na vývoji systému plánuji pokračovat i nadále. Mým cílem je rozšířit navržený systém o rádiový výškoměr formou externího modulu tak, aby výsledné zařízení umožňovalo automatickou detekci letu a zároveň mohlo sloužit jako výukový prostředek pro začínající piloty. Rád bych tak oslovil širší spektrum odborné veřejnosti a případně jim dle jejich individuálních prostředků přizpůsobil daný systém na míru. Současné řešení tuto možnost s ohledem na vnější komunikační prostředky, výpočetní výkon nebo kapacitu paměti umožňuje.

Literatura

[1] Batron: *Specification - BT45229 - BTHQ128064AVD1-STF-12-LED02YG-COG*. October 2010, [Online], [cit. 2013-12-24].
URL http://www.datasheetlib.com/datasheet/1223583/bthq128064avd1-cog-stf-12-led02yg_batron.html

[2] BIERL, L.: *MSP430 Family Mixed-Signal Microcontroller*. January 2000, [Online], [cit. 2013-04-01].
URL http://www.ti.com/lit/an/slaa024/slaa024.pdf

[3] Conolly, T.; Begg, C.; Holowczak, R.: *Mistrovství - databáze: Profesionální průvodce tvorbou efektivních databází*. Computer Press, první vydání, 2009, ISBN 978-80-251-2328-7, 584 s.

[4] Davies, J. H.: *MSP430 Microcontroller Basics*. Elsevier Science, 2008, ISBN 978-0-7506-8276-3.

[5] Fujitsu: *Fundamentals of Liquid Crystal Displays - How They Work and What They Do*. 2006, [Online], [cit. 2013-12-25].
URL
http://www.fujitsu.com/downloads/MICRO/fma/pdf/LCD_Backgrounder.pdf

[6] Gaspar, P. D.; Santo, A. E.; Ribeiro, B.; aj.: *MSP430 Architecture*. 2009, [Online], [cit. 2014-12-22].
URL
http://transfer.cizgi.com.tr/nsaral/mcu-turkey/MSP430_egitim/chapt4.pdf

[7] Gast, M. S.: *802.11 Wireless Networks: The Definitive Guide*. O'Reilly Media, Inc., druhé vydání, 2005, ISBN 978-0-596-10052-0, 656 s.

[8] Hi-Link: *HLK-RM04 User Manual*. Čtvrté vydání, November 2013, [Online], [cit. 2015-03-29].
URL http://www.hlktech.net/inc/lib/download/download.php?DId=21

[9] Kreibich, J. A.: *Using SQLite*. O'Reilly Media, Inc., první vydání, 2010, ISBN 978-0-596-52118-9, 503 s.

[10] Kugelstadt, T.: *Designing an isolated I^2C Bus© interface by using digital isolators*. 2011, [Online], [cit. 2013-12-25].
URL http://www.ti.com/lit/an/slyt403/slyt403.pdf

[11] Matoušek, P.: *Síťové aplikace a jejich architektura*. Vutium, první vydání, 2014, ISBN 978-80-214-3766-1.

[12] Maxim: *DS1338 I^2C RTC with 56-Byte NV RAM*. Třetí vydání, March 2012, [Online], [cit. 2013-12-26].
URL http://datasheets.maximintegrated.com/en/ds/DS1338-DS1338Z.pdf

[13] MGL Avionics: FLIGHT-2 Operating Manual. [Online], [cit. 2015-04-04].
URL http://www.mglavionics.com/FLIGHT2.pdf

[14] Muehlhofer, A.: *MSP430 Flash Self-Programming Technique*. November 2000, [Online], [cit. 2014-12-22].
URL http://www.gaw.ru/pdf/TI/app/msp430/slaa103.pdf

[15] NXP Semiconductors: *I^2C-bus specification and user manual*. 6 vydání, April 2014, [Online], [cit. 2013-12-29].
URL http://www.nxp.com/documents/user_manual/UM10204.pdf

[16] Postel, J.: *Transmission Control Protocol*. RFC 793, September 1981.

[17] Remigio, S.: *SPI, Intel 8080, and Motorola 6800 communication protocol between MCU and OLED driver*. May 2005, [Online], [cit. 2013-12-24].
URL http://www.docdatabase.net/download_pdf.php?did=616950

[18] Sitronix: *ST7565P - 65 x 132 Dot Matrix LCD Controller/Driver*. November 2007, [Online], [cit. 2013-12-24].
URL http://www.alldatasheet.com/datasheet-pdf/pdf/326240/SITRONIX/ST7565.html

[19] Texas Instruments: *Basic Clock Module, Oscillator and Clock Generator*. September 1998, [Online], [cit. 2014-12-22].
URL http://www.datasheetarchive.com/SLAU023-datasheet.html

[20] Texas Instruments: *MSP430x1xx Family*. 2006, [Online], [cit. 2014-12-22].
URL http://www.ti.com/lit/ug/slau049f/slau049f.pdf

[21] Texas Instruments: *MAX3222 DS*. 7 vydání, January 2007, [Online], [cit. 2013-12-29].
URL http://www.mouser.com/ds/2/256/MAX3222-MAX3241-103207.pdf

[22] Texas Instruments: *Processor Supervisory Circuits*. April 2013, [Online], [cit. 2013-12-29].
URL http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=tps3825-33&fileType=pdf

[23] Vtec Electronics GmbH: Flisys80P / 57P. May 2014, [Online], [cit. 2015-04-04].
URL http://vtec-avionics.ch/Flisys80P%2057P%20ins_1.pdf

[24] WWW stránky: Bluetooth Basics - A Look at the Basics of Bluetooth Technology. [Online], [cit. 2014-12-21].
URL http://www.bluetooth.com/Pages/Basics.aspx

[25] WWW stránky: Flying hours counter. [Online], [cit. 2015-04-04].
URL http://www.winter-instruments.de/english/produkte/flugstunden.html

[26] WWW stránky: Grefické programy v Qt 4-8 (TCP klient). [Online], [cit. 2015-03-29].
URL http://www.abclinuxu.cz/clanky/programovani/graficke-programy-v-qt-4-8-tcp-klient

[27] WWW stránky: Network Programming with Qt. [Online], [cit. 2015-03-29].
URL http://doc.qt.io/qt-5/qtnetwork-programming.html

[28] WWW stránky: Overview for Ultra-low Power. [Online], [cit. 2014-12-22].
URL http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/ultra-low_power/overview.page

[29] WWW stránky: Qt 5.4. [Online], [cit. 2015-05-16].
URL http://doc.qt.io/qt-5/index.html

[30] WWW stránky: Qt Licensing. [Online], [cit. 2015-05-16].
URL http://www.qt.io/qt-framework/

[31] WWW stránky: Qt Licensing. [Online], [cit. 2015-05-16].
URL http://doc.qt.io/qt-5/licensing.html

[32] WWW stránky: Qt Quick Local Storage QML Types. [Online], [cit. 2015-03-30].
URL http://doc.qt.io/qt-5/qtquick-localstorage-qmlmodule.html#opendatabasesync

[33] WWW stránky: Qt Quick (Qt User Interface Creation Kit). [Online], [cit. 2015-03-30].
URL https://qt-project.org/wiki/Qt_Quick

[34] WWW stránky: REAL-TIME CLOCKS (RTC) ICS. [Online], [cit. 2013-12-25].
URL http://www.maximintegrated.com/en/products/digital/real-time-clocks.html

[35] WWW stránky: What is FRAM. [Online], [cit. 2014-12-22].
URL http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/ultra-low_power/msp430frxx_fram/what_is_fram.page#under

[36] Šimková, D.: *Hardware pro začátečníky - průvodce nitrem počítačů na první pokus.* Grada, první vydání, 2007, ISBN 80-247-2029-9.

[37] Řezník, J.: QML - moderní uživatelská rozhraní v Qt (2). March 2012, [Online], [cit. 2015-03-30].
URL
http://www.abclinuxu.cz/clanky/qml-moderni-uzivatelska-rozhrani-v-qt-2

Seznam příloh

Příloha A

Obsah CD

dokumentace: Zdrojové soubory tohoto textu psaného v LaTeX.

projektIAR: Projekt pro vývojové prostředí IAR Embedded Workbench verze 6.10.7 se zdrojovými a konfiguračními soubory přístroje.

projektQt: Projekt pro vývojové prostředí Qt Creator se zdrojovými soubory.

projektInventor: Model montážní krabičky v nástroji Autodesk Inventor Proffesional 2015.

aplikace: Spustitelná verze aplikace.

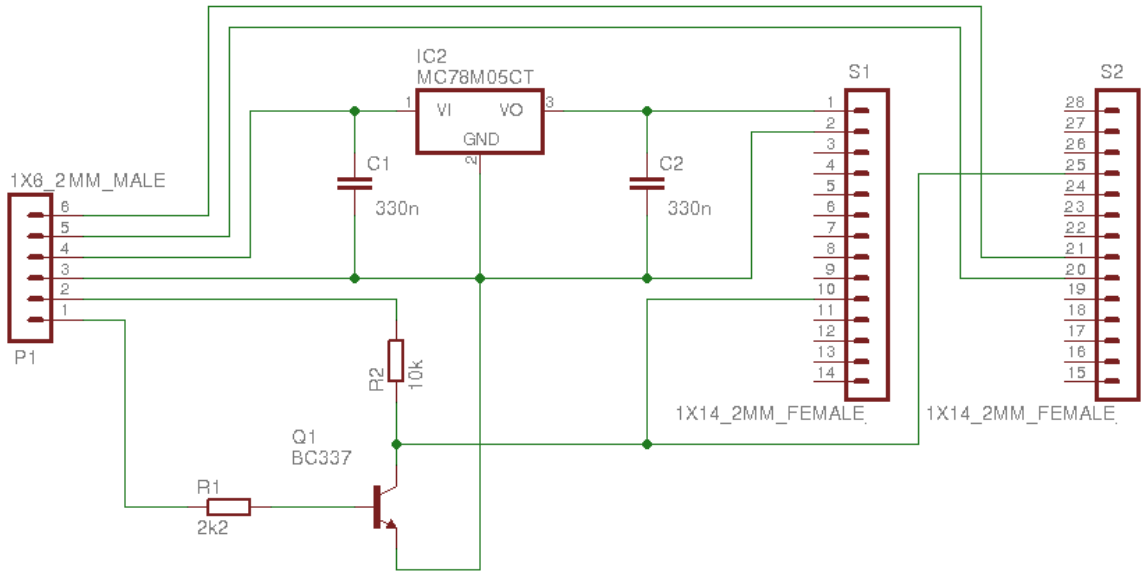
projekt.pdf: Technická zpráva.

fonty.ods: Znaková sada LCD displeje.

readme.txt: Obsah CD.

Příloha B

Základní deska pro modul HLK-RM04

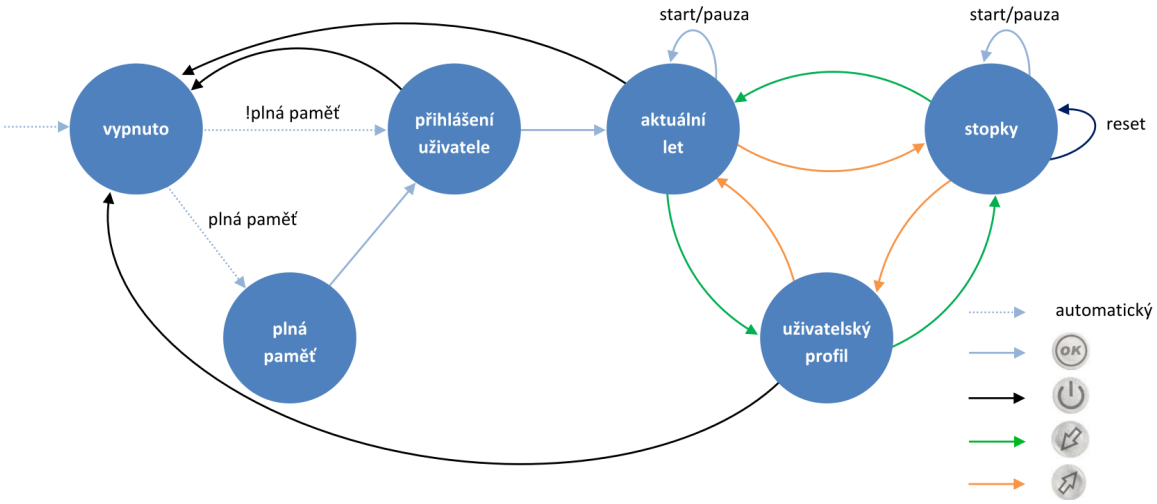


Obrázek B.1: Schéma základní desky pro WiFi modul HLK-RM04

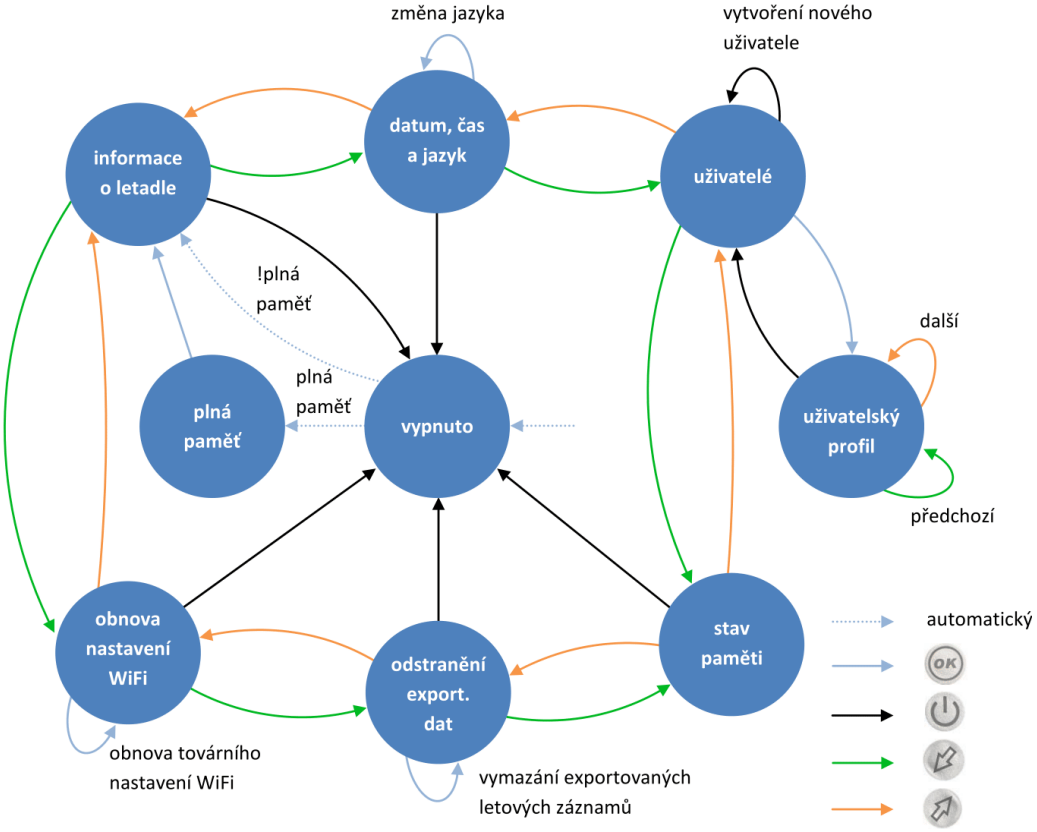
Příloha C

Režimy činnosti palubního přístroje

Níže uvedené diagramy přechodu v obou režimech činnosti palubního přístroje slouží jako doplnění textu z kapitoly 4.1.3.



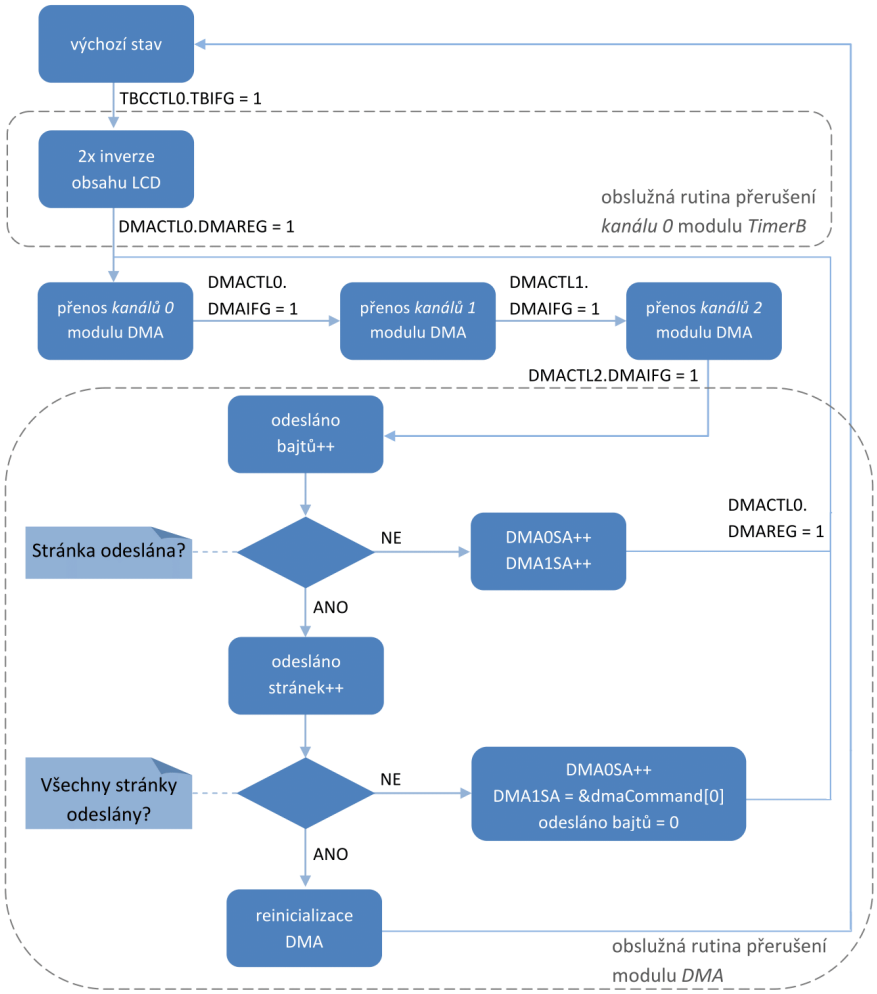
Obrázek C.1: Diagram přechodu mezi obrazovkami ve standardním režimu



Obrázek C.2: Diagram přechodu mezi obrazovkami v administrátorském režimu

Příloha D

Přenos dat do paměti LCD displeje



Obrázek D.1: Funkční schéma přenosu vnitřního datového bufferu do paměti LCD displeje (doplnění kapitoly 4.1.6)

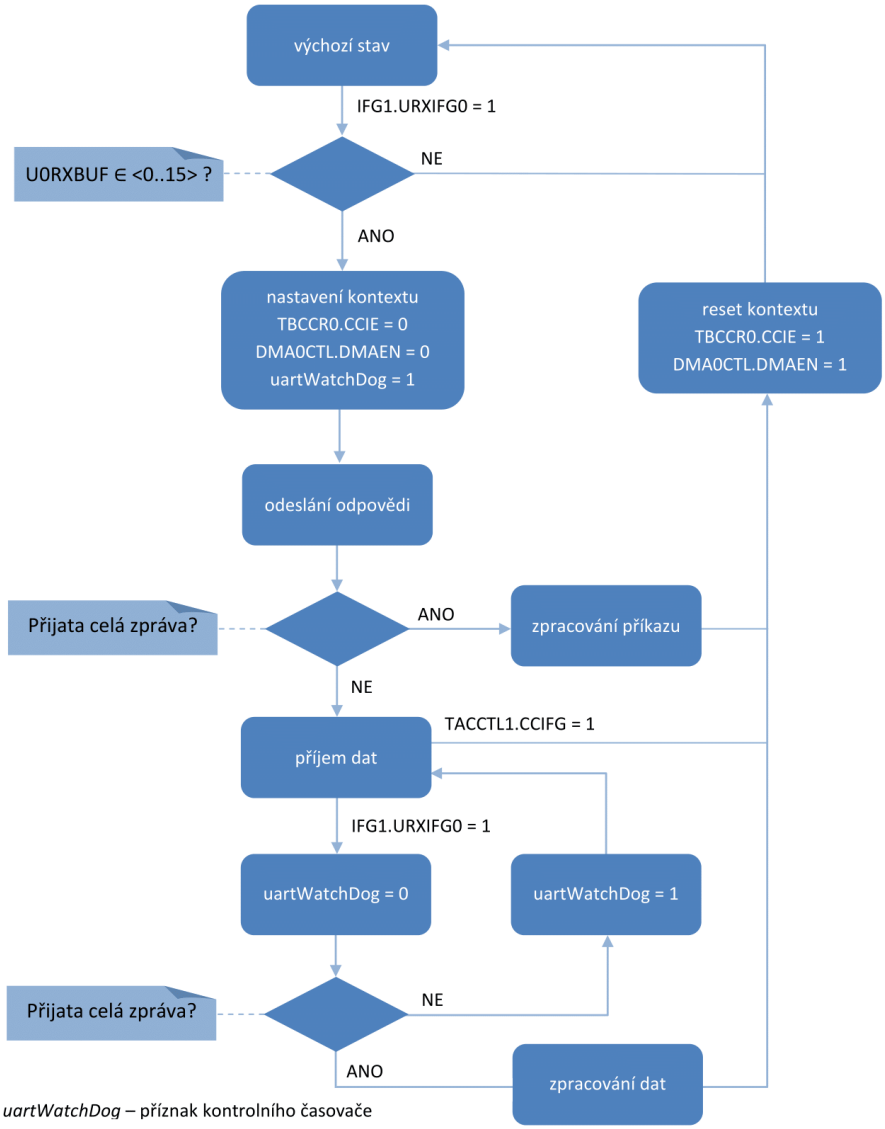
Příloha E

Znaková sada LCD displeje

Obrázek E.1: Symboly znakové sady LCD displeje

Příloha F

Komunikace MCU s HLK-RM04



Obrázek F.1: Funkční schéma realizace komunikace mikrokontroléru s modulem HLK-RM04 (doplnění kapitoly 4.1.8)