Dokumentace k projektu IMP
Aplikace ovládaná pomocí rotačního enkodéru KX/Y-040

Ondřej Koumar, xkouma02

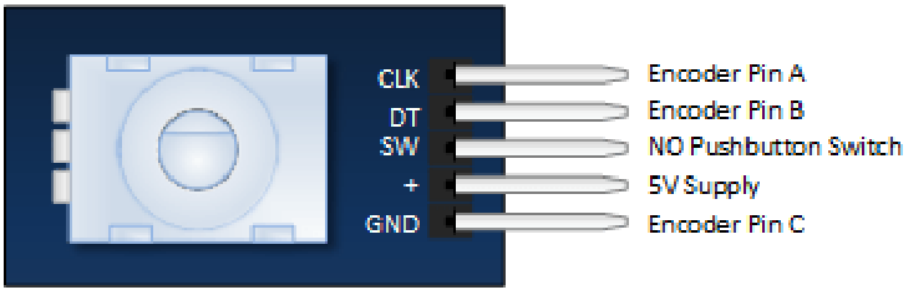
15. prosince 2023

1 Úvod do problematiky

Projekt je řešený na platformě FITkit3, kde je vsazen mikrokontrolér Kinetis MK60DN512VMD10. s použitím dvou externích rotačních enkodérů typu KX/Y-40. Pro interakci s uživatelem je použito sériové komunikační rozhraní UART, přesněji pouze odesílací část. Dále, aby měl uživatel zpětnou vazbu svých akcí, každá interakce s deskou způsobí zaznění tónu na integrovaném bzučáku, délka a výška tónu je dle použitého zařízení, případně směru pohybu daným zařízením (platí pro rotační enkodéry).

1.1 Rotační enkodér KX/Y-040

Rotační enkodér má pět pinů, které se dají zapojit.



Obrázek 1: Piny rotačního enkodéru

Vždy je potřeba mít zapojenou zem a zdroj napětí (piny *GND* a *+*), další piny dle očekávané funkcionality. Aby mohl uživatel na mikrokontroléru správně přečíst otočení libovolným směrem enkodéru, je potřeba zapojit piny *CLK* a *DT*. Pro čtení stisknutí knoflíku je potřeba zapojit pin *SW*. Výchozí hodnota na těchto pinech je log. 1.

1.1.1 Snímání pohybu

Pro snímání pohybu je třeba ještě zjistit, jak rotační enkodér funguje uvnitř. Mějme tři výstupy rotačního enkodéru zobrazeny na obrázku 2a, kde výstup A je spojen s pinem *CLK*, výstup B je spojen s pinem DT a výstup C vede do země.

Přívod napětí je přímo spojen s piny *CLK* a *DT* přes dva pull-up rezistory. Právě díky tomu je výchozí hodnota na těchto pinech log. 1. Na obrázku 2b je vidět mechanismus spínání. V tomto konkrétním případě jsou oba spínače otevřeny a proud jde ze zdroje napětí přímo do země. Pokud by byly spínače zavřeny, a tedy signály A a B byly „mimo zub“ kolečka enkodéru, proud by mohl téct jedině do pinů *CLK* a *DT*.

(a) Vnější konfigurace (b) Vnitřní spínače

Obrázek 2: Výstupy rotačního enkodéru

Díky tomu se dá poznat, zda se rotačním enkodérem pohybuje po směru nebo proti směru hodinových ručiček. Z obrázku lze vidět, že při pootočení knoflíku enkodéru o zhruba čtvrtinu otočky po směru hodinových ručiček se změní signál A a výstup pinu CLK bude log. 1, při otočení na druhou stranu se jako první změní signál B a na výstupu pinu DT bude log. 1. Po postupném otáčení dále se změní i signál druhý.

1.1.2 Postupná změna signálů A a B při plném otočení

Mějme stav, kde oba spínače jsou uzavřeny, tedy oba signály „sedí mezi zuby“. Jako příklad vezměme pouze otočení proti směru hodinových ručiček[1]. Zpočátku bude výstup signálů *CLK (A)* i *DT (B)* na log. 1.

Při otočení o čtvrtinu se signál B spojí se signálem C, tedy se zemí a na výstupu DT bude log. 0. Po dalším kroku se zavře spínač AC, díky čemuž je vygenerována log. 0 i na výstupu CLK. Dalším krokem se dostaneme do stavu, kdy spínač BC je otevřený, AC stále zavřený a ve finále jsou oba spínače opět otevřené a na oba výstupy se znovu generuje log. 1.

1.2 UART (Universal Asynchronous Reciever Transmitter)

UART je rozhraní pro asynchronní sériovou komunikaci mezi dvěma zařízeními. Má konfigurovatelné parametry typu formát datového slova a přenosová rychlost. Pro tento projekt stačí, abychom si definovali tři důležité principy.

- **Baud rate** – pro dnešní počítačové systémy ekvivalent k *bit rate*. Není tomu ale vždycky tak; jedná se o počet změn signálu za sekundu, kdy změna signálu může být popsána i více bity.
- **Parita** – technika používaná pro kontrolu integrity dat v datovém přenosu. Může být sudá nebo lichá; posledním (paritním) bitem je datové slovo doplněno tak, aby počet jedniček ve slově byl buď sudý nebo lichý[2].

[1] Po směru hodinových ručiček by sekvence vypadala stejně, jen by se prohodilo pořadí, ve kterém se hodnoty signálů mění.

[2] Berme v potaz pouze jeden paritní bit.

- **Synchronizace** – je potřeba, aby generátor hodinových pulsů přijímače běžel na co „nejstejnější“ frekvenci jako generátor vysílače. Synchronizace se provádí předem dohodnutou změnou na datovém vodiči, v případě UARTu se jedná o přechod z log. 1 do log. 0.

1.2.1 Komunikace mezi dvěma zařízeními

Vysílač zahájí přenos datového slova tzv. *start bitem*, který je právě již zmiňovanou dohodnutou změnou na datovém vodiči z klidové úrovně do opačné. Další bit je už datový, stejně jako 7 dalších[3]. Za posledním vyslaným datovým, případně paritním bitem je vyslán minimálně jeden stop bit, který má vždy hodnotu klidového stavu. Ten slouží k tomu, aby byla jasně oddělena datová slova, zároveň tím ale přijímač získává dostatek času ke zpracování přijímaných dat.

2 Implementovaná aplikace

Pro implementaci byla jako aplikace vybrána jednoduchá kalkulačka ovládaná pomocí dvou rotačních enkodérů a tlačítka SW6 na desce.

2.1 Návod na použití a uživatelské interakce

Otáčením knoflíku jednoho rotačního enkodéru se vybírá první operand. Každým otočením po směru hodinových ručiček se zvýší hodnota operandu o 1, proti směru se o 1 sníží. Stisknutím tlačítka prvního enkodéru se výběr operandu potvrdí.

Po potvrzení se zobrazí výzva pro výběr operátoru. V základu je jich šest – výčtem sčítání, odčítání, násobení, dělení, druhá odmocnina a obecná mocnina. Stisknutím knoflíku se zvolí operátor. V případě výběru druhé odmocniny se rovnou spočítá výsledek, jinak se pokračuje na výběr druhého operandu, který probíhá stejně jako výběr prvního.

Po výběru druhého operandu následuje výpočet a uživateli se vypíše výsledek, případně chybová hláška, pokud uživatel zadal nedefinovanou dvojici operandů pro danou funkci.

2.1.1 Pole předešlých výsledků

Alternativně lze vybírat z pole posledních výsledků. K tomu se uživatel dostane stisknutím knoflíku na druhém enkodéru, otáčením si pak může vybrat z 5 posledních vypočtených výsledků[4]. Opětovným stisknutím knoflíku je vybrán operand z pole předešlých výsledků.

2.1.2 Floating-point čísla

Díky operacím, jako je `sqrt()`, `pow()` nebo dělení, často se ve výsledku vyskytnou desetinná čísla. Kalkulačka má plnou podporu desetinných čísel, ale musí se vybrat z předešlých výsledků, protože při výběru operandů otáčením knoflíku se pouze inkrementuje nebo dekrementuje celé číslo o 1.

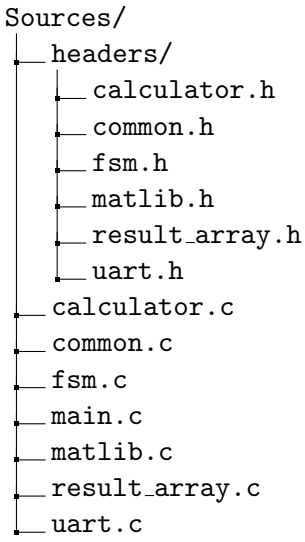
Výpis desetinných čísel je podporován do osmi desetinných míst.

2.2 Implementační detaily

2.2.1 Souborová struktura

Struktura zdrojových a hlavičkových souborů je na obrázku 3. Hlavní funkcionalita aplikace je v souborech *calculator.c*, *matlib.c* a *fsm.c*. V souboru *result_array.c* je pak definováno pole předešlých výsledků, se kterým se pracuje v hlavním API aplikace. *uart.c* obsahuje funkce pro práci s UARTem, včetně jeho

[3] Může být i jiný počet, obě strany se ale musí předem dohodnout.
[4] V kódu lze jednoduše toto číslo změnit.



Obrázek 3: Zdrojové a hlavičkové soubory

inicializace, *common.c* pak definuje funkce a datové struktury společné pro celý program. V *main.c* je pak hlavní smyčka programu.

2.2.2 API

Hlavní programové API je definováno v souboru *calculator.c*. Zde se nachází například funkce `PickNumber()`, `PickFromPreviousResults()`, `CalculateResult()`, `DisplayOperand()` a další pomocné funkce. Z pohledu hlavní programové smyčky jsou nejpodstatnější čtyři funkce:

- `PickNumber()`: Přečte uživatelskou interakci s deskou[5] a průběžně ukazuje uživateli, co má zrovna vybráno a co může potvrdit. Po potvrzení operandu zmáčknutím knoflíku funkce vrací hodnotu operandu.
- `PickOperator()`: Taktéž čeká na interakci s deskou, průběžně ukazuje aktuální operátor a potvrdí se stisknutím knoflíku.
- `CalculateResult()`: Po vybrání obou operandů a operátoru se zavolá příslušná funkce z *matlib.c* a vypočte se výsledek operace, případně uloží typ chyby.
- `DisplayResult()`: Zobrazí uživateli do terminálu buď výsledek operace, nebo chybovou hlášku.

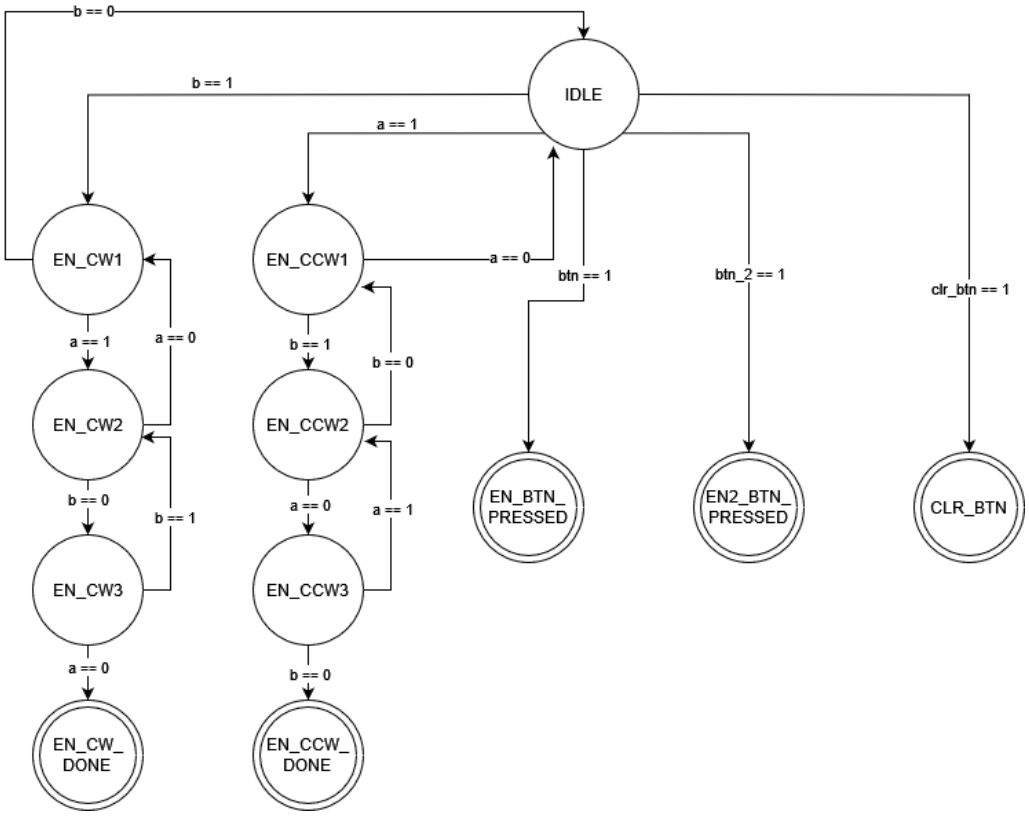
2.2.3 Datové struktury

Hlavní datové struktury jsou definovány v *common.h*. Struktura, která slouží pro předávání výsledků a operandů mezi funkcemi, se nazývá `CalcData`.

Je to datová struktura obsahující:

- informace o datovém typu výsledku (enum `Type`),
- výsledek samotný – int xor float, a tedy je použita unie `Value`,
- informace o chybě, která při výpočtu nastala, enum `FaultType`, výchozí hodnota je `NONE`.

[5] Provolává funkci `ReadBoardInteraction()`, která aktivně čeká.



Obrázek 4: Stavový automat čtení vstupu od uživatele

Složitost této datové struktury je dána podporou floating-point čísel a operací s nimi. Je třeba mít informaci o typu výsledku, aby se dalo přistoupit do výsledkové unie.

Díky enumerátoru `FaultType` je pak možné při zobrazování výsledku uživateli místo číselného výsledku zobrazit chybovou hlášku, aniž by nás zajímala hodnota v rámci výsledkové unie.

2.2.4 Konečný automat čtení vstupu od uživatele

Implementaci konečného automatu lze nalézt v souboru *fsm.c* ve funkci `ReadBoardInteraction()`. Tato funkce bere jeden parametr, a to `mode`, který indikuje, zda se vybírá operand inkrementací/dekrementací (mód 1) celého čísla, nebo se vybírá z pole posledních výsledků (mód 2).

Konečný automat zobrazený na obrázku 4 platí pouze pro mód 1. V případě módu 2 je automat skoro stejný, nicméně v něm chybí větev vedoucí do koncového stavu `EN2_BTN_PRESSED`. Důvod je ten, že v módu 1 je možné provádět všechny operace s enkodérem 1 a stisknout knoflík enkodéru 2, zatímco v módu 2 se dá pouze interagovat s enkodérem 2 v jeho plném rozsahu.

V rámci konečného automatu jsou naimplementovány stavy `EN_CW1`, `EN_CW2`, `EN_CW3`, `EN_CW_DONE`, stejně tak pro CCW. Značí to postupné otáčení rotačního enkodéru po směru hodinových ručiček (CW) a proti směru hodinových ručiček (CCW), jak je popsáno v kapitole 1.1.1. Pokud je pootočeno jedním nebo druhým směrem, pak stisknutí knoflíků, případně tlačítka na desce, není dostupné. Ty se dají stisknout ve stavu `IDLE`, ve kterém automat zůstává, pokud není žádná aktivní interakce od uživatele.

3 Zhodnocení

Implementací výše uvedených vlastností byla vytvořena jednoduchá kalkulačka, která umí spočítat několik základních operací a spolehlivě zvládne zpracovat chyby. Při implementaci projektu bylo dbáno na rozšiřitelnost – dají se jednoduše přidat matematické operace, zpracování chyb a další funkcionalita.

Implementace je dohromady zhruba na 800 – 1000 řádků kódu (můj odhad) a celková doba práce je zhruba 40 hodin, včetně studování funkčnosti rotačního enkodéru a bez psaní dokumentace.

3.1 Nejistoty programu

Kalkulačka samozřejmě není dokonalá, vždy by se dala více dotáhnout do konce. Nicméně smyslem tohoto projektu nebylo udělat perfektní program. Věci, které by se daly zlepšit, jsou například:

- uživatelská přívětivost,
- převádění floatů na celá čísla, pokud výsledkem např. funkce `pow()` je celé číslo,
- propracovanější chybové hlášky při počítání funkce `pow()`.

3.2 Očekávané hodnocení

Projekt splnil všechny body zadání. Aplikace se ovládá oběma rotačními enkodéry, využívá další I/O prvky jako tlačítko nebo bzučák. Je relativně uživatelsky přívětivá, má ošetřené chybové stavy a vypisuje chybové hlášky. Složitost projektu je podle názoru autora projektu adekvátní.

V rámci implementace bylo dbáno na rozložení na podproblémy a přehlednost zdrojového kódu. V hlavičkových souborech můžeme nalézt stručné popisy funkcí ve formátu *Doxygen* komentářů. Rozšiřitelnost projektu je jednoduchá.

V dokumentaci byly popsány všechny technické prvky, které se v projektu vyskytly a stručně popsána implementace. Při prezentaci projektu nenastaly žádné komplikace a aplikace byla vedoucím projektu schválena.

Proto očekávané hodnocení autora projektu je (13 – 14)/14.