VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

IMP – Mikroprocesorové a vestavěné systémy 2020/2021

ARM-FITkit3 či jiný HW: Testování mikrokontrolérů

Vypracoval: Vojtěch Mimochodek
Login: xmimoc01

15. prosince 2020

Obsah

1 Úvod

V této práci je řešen projekt do předmětu Mikroprocesorové a vestavěné systémy. Tématem práce je implementace programu, který bude sloužit k otestování mikrokontroléru. V práci je řešený proces testování paměti RAM. Testování funguje je na základě dvou algoritmů – March X a March C.

2 Návrh a implementace

Aplikace je naimplementována v jednom zdrojovém souboru *main.c*. Ten obsahuje natažení potřebných knihoven, definice konstant, implementaci dílčích funkcí a řádné okomentování. K vývoji aplikace bylo používáno IDE s názvem Kinetis Design Studio (KDS). Jako mikrokontrolér byl použit FITKit3. K připojení na terminál MCU byl použit nástroj PuTTy.

2.2 Main.c

Tento soubor je rozdělen na několik dílčích funkcí.

Hlavní funkcí celého programu je *main()*. Ta na začátku zavolá funkce *initMCU()*, *initPinAndPort()* a *initUART5()*. Ty inicializují MCU, potřebné piny a ostatní věci pro správný chod mikrokontroléru. V další fázi je volána funkce *startTest()*, která jako parametr přijímá typ algoritmu, který se má provést. Výstupem této funkce je výpis výsledků na terminál.

Prvním algoritmem, který byl naimplementován je MarchX. Ten je řešen ve funkci *ramMemoryTestMarchX()*. Tento algoritmus, je průmyslovým standardem pro kontrolu paměťových polí RAM pro „dekodér adres“ a „stuck at“ chyby. Funkce ověří segment paměti z adresy dané parametry *pMemStart*, což je ukazatel na počáteční, nejnižší adresu testované paměti a *pMemEnd*, což je koncový, nejvyšší ukazatel na testovanou adresu. Test je destruktivní a funguje ve čtyřech krocích takto:

- Krok 1: Projde všechny adresy a ve všech polích zapíše 0. Nastaví registry a provede smyčku k vymazání každé adresy paměti.
- Krok 2: Začne na nejnižší zadané adrese a postupně kontroluje zapsané 0. Následně zapíše hodnoty 1, inkrementuje pole a tento proces opakuje.
- Krok 3: Začne naopak na nejvyšší zadané adrese a postupně kontroluje zapsané 1 z předchozího kroku. Ty následně přepíše na 0, dekrementuje pole a tento proces opakuje.
- Krok 4: Na konec zkontroluje všechny 0 z polí.

V případě kdy v některém z kroků nesedí očekávané hodnoty (0 nebo 1) je vyhozena chyba a nalezen chybný bit.

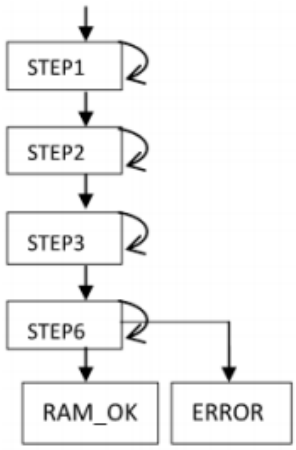
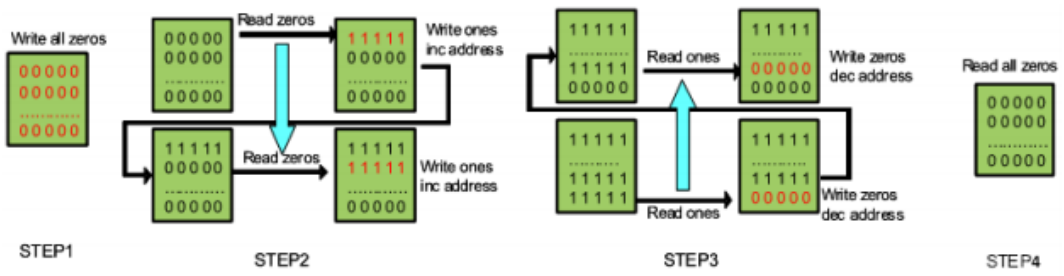


Figure 6. March X RAM memory testing diagram



Obrázek č. 1: Schéma algoritmu MarchX

Druhým algoritmem je MarchC. Algoritmus je řešen ve funkci *ramMemoryTestMarchC()*. Je průmyslovým standardem pro kontrolu paměťových polí RAM pro „d.c.“ poruchy. Funkce testování RAM ověřuje segment paměti z adresy dané parametrem *pMemStart*, což je počáteční ukazatel a *pMemEnd*, což je koncový ukazatel paměti. Test je destruktivní a v šesti krocích funguje takto:

- Krok 1: Projde všechny adresy a ve všech polích zapíše 0.
- Krok 2: Začne na nejnižší zadané adrese. Zkontroluje zapsané 0 z předchozího kroku. Ty následně přepíše na 1. Inkrementuje pole a tento proces opakuje.
- Krok 3: Začne opět na nejnižší adrese. Přečte zapsané 1 z předchozího kroku, které přepíše na 0. Inkrementuje pole a opakuje tento proces.
- Krok 4: V tomto případě se naopak začíná na nejvyšší adrese. Postupně kontroluje 0 zapsané z předchozího bodu. Ty přepíše na 1. Pole se dekrementuje a proces se opakuje.
- Krok 5: Začíná se opět jako v předchozím kroku na nejvyšší adrese. Postupně přečte 1 a přepíše je na 0. Pole se dekrementuje a proces se opakuje
- Krok 6: Finálně se zkontrolují zapsané 0.

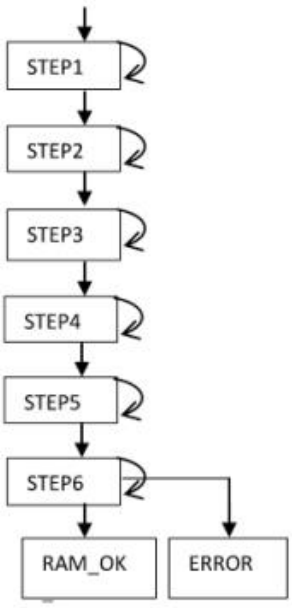
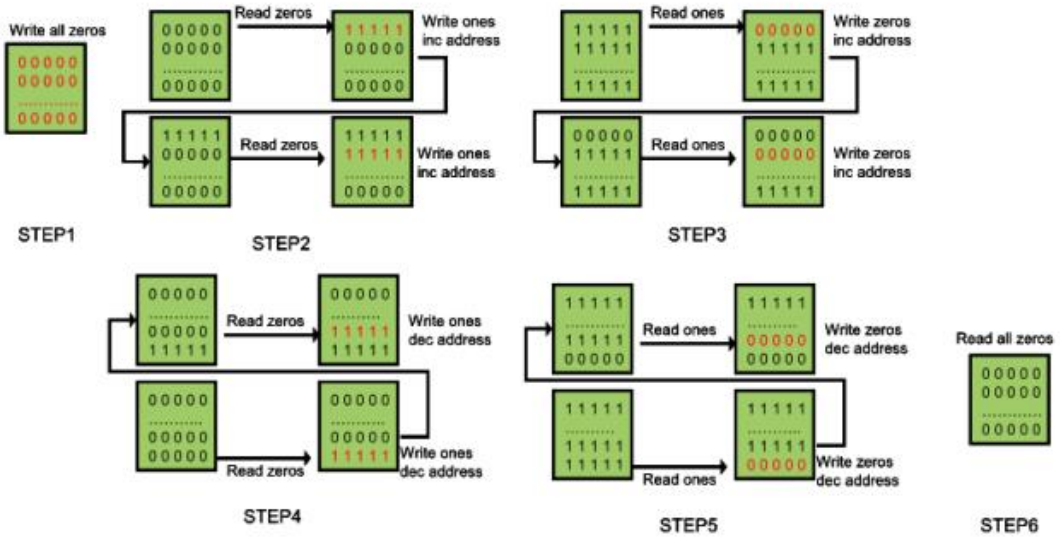


Figure 8. March C RAM memory testing diagram



March C (van der Goor, 1991)

Obrázek č. 2: Schéma algoritmu MarchC

Dalšími pomocnými funkcemi k funkcím algoritmů jsou: *isBitZero()*, *isBitOne()*. Ty zkontrolují, zdali aktuálně zpracovávaný bit z pole je buď 0 nebo 1. Pro přenastavení aktuálně zpracovávaného bitu na jinou hodnotu jsou naimplementovány funkce *setBitToZero()* a *setBitToOne()*. Všechny tyto funkce využívají definovanou funkci *BITBAND_REGION_BIT()*. Ta pomocí bitových posunů dokáže pracovat s aktuálně zpracovávaným bitem z pole.

Funkcemi *sendChar()* a *sendString()* je zajišťován výpis na terminál.

Pojmenování funkcí je v coding standardu CamelCase. Umístění složených závorek při cyklech, if podmínkách a funkcích jsou pevně určeny. Před každou novou funkcí jsou dva řádky volné. Uvnitř funkce k oddělení částí vždy jeden řádek volný

3 Závěr

Podařilo se mi naimplementovat vybrané algoritmy pro otestování paměti RAM.

Tento projekt mi pomohl ke zdokonalení se v jazyce C/C++ a především mi rozšířil znalosti ohledně práce s mikrokontroléry.

4 Zdroje

- https://www.nxp.com/docs/en/application-note/AN4873.pdf?fbclid=IwAR03dsRtyhNjK3exlkGKjWbWJg5W6PsxhDUF66kIXUglbFC0YkRMpQ4hIOE

- https://community.arm.com/developer/ip-products/processors/f/cortex-m-forum/6679/bit-banding-in-sram-region-cortex-m4

- Laboratorní cvičení z předmětu IMP