VILNIAUS UNIVERSITETAS
FIZIKOS FAKULTETAS
KIETO KŪNO ELEKTRONIKOS KATEDRA

Mindaugas Kurmauskas

CORTEX R4 MIKROVALDIKLIO ARCHITEKTŪROS TYRIMAS

Pagrindinių studijų kursinis darbas

(studijų programa – TAIKOMOJI FIZIKA)

Studentas Mindaugas Kurmauskas
Darbo vadovas dr. Mindaugas Vilūnas
Katedros vedėjas dr.(HP) Kęstutis Arlauskas

Vilnius 2013

Turinys

Įvadas

Egzistuoja daug užduočių kurioms atlikti reikia didelio patikimumo. Jų atlikimui paprasčiausiu atveju galima naudoti standartinius mikrovaldiklius, kombinuojant su programinėmis gudrybėmis. Tokiomis kaip WDT (watchdog timer), kurios prižiūri pagrindinę programą, kad ji tinkamai atliktu savo darbą. Nuolatiniais testais patikrinti duomenų skaičiavimo patikimumą, pavyzdžiui atlikti algoritma du kartus ir tikrinti ar atsakymai sutampa. Daugeliu atveju visos papildomos priemonės užtikrinti patikimumą prailgina duomenų skaičiavimo laiką.

Sistemose kurioms reikalingas didelis patikimumas ir darbas realeame laike labai praverčia sistemos dubliavimas su tikrinimo funkcijomis. Klaidos tikimybė ne visais atvejais yra tolydi laike, taigi klaidos gali tuo pačiu metu ivykti abiejose sistemose. Tokiu atveju naudinga, kai sistemos dirba pastumtos laike atžvilgiu viena kitos.

Yra daug aplinkų kur gali reikėti tokių sistemų. Pramoninės saugumo reikalaujančios aplinkos, tokios kaip:

- automobilų stabdymo sistemos, tokios kaip ratų anti blokavimo sistema (ABS)
- elektros gaminimo ir paskirstymo sistemos.
- liftai ir eskalatoriai

Medicinoje:

- defibriliatoriai
- radiacijos terapija
- robotizuotos operacijos

Atsižvelgiant į šį poreikį kompanija Teksas Instruments neseniai sukūrė mikrovaldiklių šeimą kodiniu pavadinimu Hercules. Mum kilo klausimas koks yra šių mikrovaldiklių energetinis efektyvumas, ką šiame darba ir bandoma išsiaiškinti.

Tolimesniame darbe būtų idomu patikrinti veikimą esant ekstremaliam poveikui, galimybę panaudoti vieną iš Hercules šeimos atstovų RM48 kosmose, kur jo užduotis galėtų būti navigacija, kur reikia slankaus formato ir skaičių tikslumo.

1 Naudojamos įrangos ir programos

Naudojamas TMDXRM48USB maketas. Jame yra:

- Mikrovaldiklis xRM48L950AZWTT:
  - Du 32-bitu ARM Cortex-R4F procesoriai, veikiantys kartu[1]
  - 3MB flash, 256kB RAM
  - greitis iki 200MHz
  - neslankaus(32bitu) ir dvigubo tikslumo (64bitu) slankaus kablelio aritmetikos modulis
- Integruotas XDS100v2 emuliatorius programavimui per usb
- LED'ai, temperatūros sensorius, šviesos sensorius, akselerometras
- prožektoriukas

Palyginimui naudojamas STM32F4DISCOVERY maketas:

- mikrovaldiklis STM32F407VGT:
  - 32-bitu ARM Cortex-M4F procesorius
  - 1MB flash, 192 kB RAM
  - greitis iki 168MHz
  - neslankaus (32bitu) ir slankaus (32bitu) kablelio aritmetikos modulis
- Integruotas ST-LINK/V2 emuliatorius programavimui per usb
- LED'ai, akselerometras, skaitmeninis mikrofonas
- CS43L22- SAK garsui su integruotu D klases garso stiprinimu

Pasirinktas kompiliatorius - IAR 6.4 kompileris. Darbo pradėjimo metu vienintelis palaikė abu maketus, be papildomos įrangos.

1.1 xRM48L950 aprašas

Mikrovaldiklyje yra du ARM Cortex-R4F veikiantys lockstep režimu, procesoriaus ir atminties vidinių testų loginis modulis (BIST [2]), Flash ir SRAM atminties 1 bito klaidos taisymo ir 2 bitų klaidos atpažinimo galimybės. Įrenginys palaiko little-endian (LE32) formatą. Tai reiškia, kad atmintyje jauniausias bitas yra saugomas pirmas.

1.2 STM32F4 aprašas

[1] lockstep - kartu atlieka tas pačias komandas
[2] BIST - built-in self test logic

2 Naudoti testavimo algoritmai ir jų aprašai

2.1 Matematinių funkcijų skaičiavimo greičių įvertinimo algoritmai

Šiame paragrafe aprašytiems algoritmams buvo naudojamas laiko trūkis, kurio pagalba buvo skačiuojamas algoritmo atlikimo greitis 1ms tikslumu. Abu maketai buvo nustatyti veikti 100MHz greičiu. Algoritmai buvo leisti 100 kartų ir paskaičiuotas laikas suvidurkintas.

2.1.1 Slankaus kablelio algoritmai

Slankaus kablelio Gauss Legendre algoritmas π skaičiavimui:

$$a_0 = 1\ b_0 = \frac{1}{\sqrt{2}}\ t_0 = \frac{1}{4}\ p_0 = 1$$

$$a_{n+1} = \frac{a_n + b_n}{2},$$

$$b_{n+1} = \sqrt{a_n b_n},$$

$$t_{n+1} = t_n - p_n(a_n - a_{n+1})^2,$$

$$p_{n+1} = 2p_n.$$

$$\pi \approx \frac{(a_n + b_n)^2}{4t_n} \tag{1}$$

Naudojant dvigubo tikslumo kintamųjų testą buvo ieškomas 1000 narys, suskaičiuota konstanta nuo pasirinktosios skiriasi $3.55271 * 10^{-15}$.

```
int doubleTest() {
  volatile int i;
  volatile double an,bn,tn,pi;
  volatile double a,b,t,p;
  a = 1.0; b = 1/sqrt(2);
  t = 1/4; p = 1.0;
  for (i = 0; i < 1000; i++) {
    an = (a+b)/2;
    bn = sqrt(a*b);
    tn = t - p*(a-an)*(a-an);
    p *= 2;
    pi = (an+bn)*(an+bn)/(4*tn);
    a = an; b = bn; t = tn;
  }
  if ((pi - 3.14159265358979) <= 3.55271e-15)
    return 1; //testas atliktas sekmingai
  return 0;  //teste ivyko klaida
```

```
  }
```

Naudojant viengubo tikslumo slankaus kablelio kintamuosius buvo ieškomas 120 narys. Konstanta nuo pasirinktosios skiriasi $8.74228 * 10^{-8}$

```
int floatTest() {
  volatile int i;
  volatile float an,bn,tn,pi;
  volatile float a,b,t,p;
  for(volatile int j = 0; j <9; j++) {
    a = 1.0; b = sqrt(0.5);
    t = 0.25; p = 1.0;
    for (i = 0; i < 120; i++) {
      an = (a+b)/2;
      bn = sqrt(a*b);
      tn = t - (p*(a-an)*(a-an));
      p *= 2;
      pi = (an+bn)*(an+bn)/(4*tn);
      a = an; b = bn; t = tn;
    }
  }
  if ((pi - 3.14159265358979) <= 8.74228e-8)
    return 1; //testas atliktas sekmingai
  return 0; //teste ivyko klaida
}
```

2.1.2 Neslankaus kablelio algoritmas

Fiksuoto tikslumo algoritmas:

$$b = \sum_{i=0}^{100000} (i * (-1)^{i+1}) \tag{2}$$

```
void intTest() {
  volatile int a,b;
  a = 1;
  b = 0;
  for(volatile int i = 0; i < 100000; i++) {
    b += i * a;
    a *= -1;
  }
}
```

2.1.3 Loginių funkciję testavimo algoritmas

Mikrovaldikliai atlieka logines AND, NOT, OR, XOR, bei bitų stumdymo funkcijas.

```
short testBits() {
  volatile unsigned sum = 0x55555555;
  // 0101 0101  0101 0101  0101 0101  0101 0101
  for (volatile int i = 0; i < 0x020000; i++) {
    sum = sum << 1;          /// sum = 0xAAAAAAAA
    sum &= 0x0000FFFF;       /// sum = 0x0000AAAA
    sum ^= 0xFFFFFFFF;       /// sum = 0xFFFF5555
    sum = sum << 16;         /// sum = 0x55550000
    sum = sum | (sum >> 16); /// sum = 0X55555555
    sum = ~sum;              /// sum = 0xAAAAAAAA
    sum = sum >> 1;          /// sum = 0x55555555
  }
  if (0x55555555 == sum)
    return 1; //testas ivykditas sekmingai
  return 0; // teste ivyko klaida
}
```

2.1.4 Greitas diskretinis Furjė eilutės transformavimo testas

Naudotos arm dsp[3] bibliotekos, atskirai optimizuotos R4 bei M4 mikrovaldikliams. Naudota 1024 kompleksinių taškų, kurie masyve išdėstyti vienas po kito ir bendras masyvo ilgis 2048. Slankaus kablelio testui masyvas užpildomas funkcija:

$$Re(z) = (sin(v * i) + cos(v * 7)),\ t = 0, 2, 4...2046$$

$$Im(z) = 0 \tag{3}$$

```
#define TEST_LENGTH_SAMPLES 2048
static float32_t testInput[TEST_LENGTH_SAMPLES];
static float32_t testOutput[TEST_LENGTH_SAMPLES/2];
uint32_t fftSize = 1024;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;

void prepareTestFloat(void) {
  for(volatile int t = 0; t < 2048; t+=2) {
    testInput[i] = sin(t*3) + cos(t*7);
    testInput[i+1] = 0;
```

[3]Dsp - digital signal processing (skaitmeninis signalų apdorojimas)

```
  }
}

int32_t testCfftFloat(void) {
  arm_status status;
  arm_cfft_radix4_instance_f32 S;
  float32_t maxValue;

  arm_cfft_radix4_init_f32(&S, fftSize, ifftFlag, doBitReverse);

  arm_cfft_radix4_f32(&S, testInput);

  arm_cmplx_mag_f32(testInput, testOutput, fftSize);

  arm_max_f32(testOutput, fftSize, &maxValue, &testIndex);

  if(testIndex == refIndex) {
    return 1;
  }
  return 0;
}
```

Fiksuoto tikslumo Furijė transformacijai naudoti ARM Q31 fiksuoto kablelio kintamieji. Iš slankaus kablelio paversti į Q31 naudota funkcija:

$$Q31(x) = ((int)\ ((x) * (float)(1 << 31))\ ) \tag{4}$$

Masyvo užpildo funkcija:

$$Re(z) = Q31(1000 * (sin(v * i) + cos(v * 7)))),\ t = 0, 2, 4...2046$$

$$Im(z) = 0 \tag{5}$$

```
#define TEST_LENGTH_SAMPLES 2048
uint32_t fftSize = 1024;
uint32_t ifftFlag = 0;
uint32_t doBitReverse = 1;
static q31_t testInputQ[TEST_LENGTH_SAMPLES];
static q31_t testOutputQ[TEST_LENGTH_SAMPLES/2];
uint32_t refQIndex = 0;

void prepareTestQ31(void) {
  double skaicius=0;
```

```
  int nulis = (int) ((0 * (float)(1<<31)));
  for(volatile int32_t i = 0; i < 2048; i+=2) {
    skaicius = 1000*sin(i*3) + 1000*cos(i*7);
    testInputQ[i] = ((int)(skaicius * (float)(1<<31)));
    testInputQ[i+1] = nulis;
  }
}

int32_t testCfftQ31(void) {
  arm_status status;
  arm_cfft_radix4_instance_q31 S;
  q31_t maxValue;

  status = arm_cfft_radix4_init_q31(&S, fftSize, ifftFlag, doBitReverse);

  arm_cfft_radix4_q31(&S, testInputQ);

  arm_cmplx_mag_q31(testInputQ, testOutputQ, fftSize);

  arm_max_q31(testOutputQ, fftSize, &maxValue, &testIndex);

  if(testIndex == refIndex) {
    return 1;
  }
  return 0;
}
```

2.2 Įėjimo ir išėjimo į trūkį algoritmo aprašymas

Naudojami maketai buvo paleisti veikti 100MHz greičiu. Abu maketai keisdavo išėjimo porto vienos kojos loginį lygį, bei buvo įjungtas nuo laiko priklausantis tuščias trūkis. Osciloscopu matuota portų išvadų lygmenų kitimo laikas.

2.3 Srovės matavimai

STM32F4DISCOVERY makete srovė matuota tiesiogiai multimetru tarp mikrovaldiklio ir srovės stabilizatoriaus. . TMDXRM48USB maketas neturėjo trumpiklių tarp srovės stabilizatoriaus ir mikrovaldiklio. Tarp stabilizatoriau 3.3V išėjimo ir mikrovaldiklio išvadų maitinimo bei stabilizatoriaus 1.2V išėjimo ir mikrovaldiklio logikos maitinimo buvo įterptos 0.12Ω varžos. Visų matavimų metu išvadai buvo nustatyti išvesties režime ir nustatyti nuliniame loginiame lygije. Buvo atlikti matavimai esant šiom konfigūracijomis:

- įjungtas "stanby" režimas:
  - Cortex-M4:
    * vidinis 1.2V įtampos reguliatorius išjungtas
    * užrakintos fazės ciklo (pll), vidinis 16MHz ir išorinis 8MHz taktiniai osciliatoriai atjungti. Veikia vidinis 32kHz taktinis osciliatorius
    * procesorius sustabdytas
    * mikrovaldiklis minimalioje srovės suvartojimo būsenoje.
    * režimą galima išeiti:
      · gavus išorinį impulsą į NRST arba WKUP išvadus
      · realaus laiko laikrodžiui (RTC) sugeneravus trūkį
  –

- įjungtas negilaus miego režimas esant įvairiems taktavimo greičiams:
  - Cortex-M4:
    * sistema veikia užrakintos fazės ciklo taktavimo dažniu
    * procesorius sustabdytas (CPU)
    * iš režimo galima išeiti betkokio trūkio pagalba

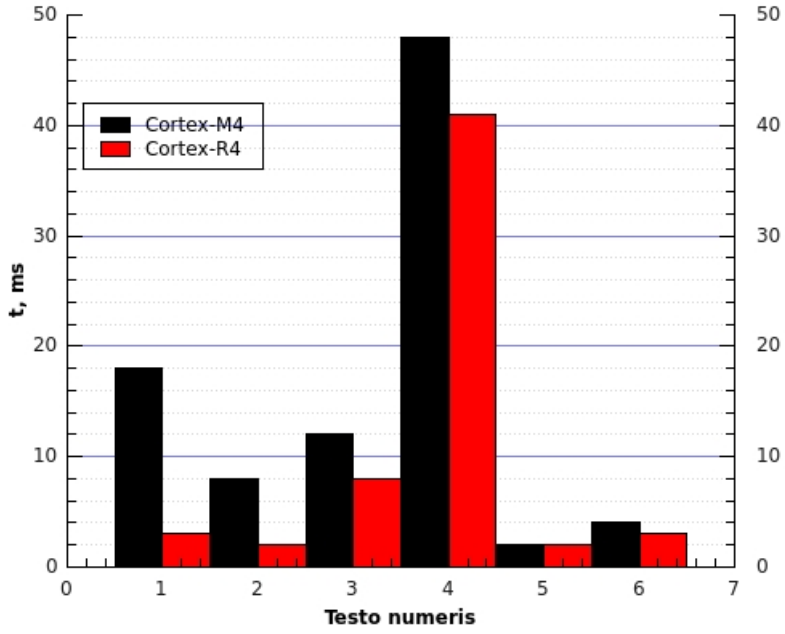
- mikrovaldikliai tuščiame cikle

```
( while(1){;} )
```

- mikrovaldikliai atlikinėjo loginį testavimo algoritmą be trūkio esant įvairiems taktavimo greičiams

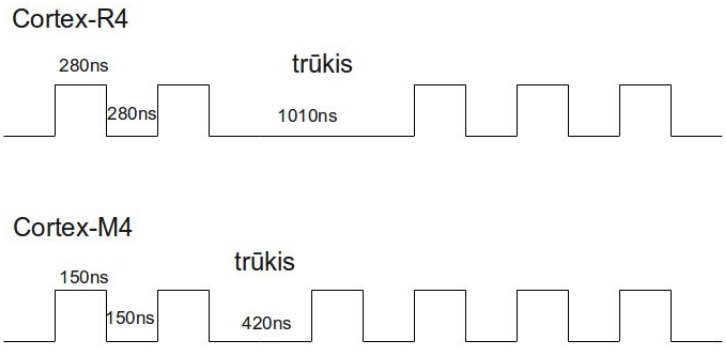
3 Rezultatai

Cortex-R4 mikrovaldiklis beveik visus skaičiavimo testus atliko greičiau nei cortex-M4 mikrovaldiklį.



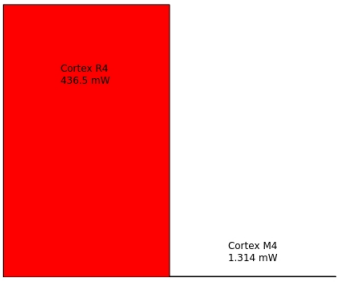
1 pav. *Skaičiavimo algoritmų laikai. 1 - dvigubo tikslumo slankaus kablelio testas. 2 - slankaus kablelio testas. 3 - neslankaus kablelio testas. 4 - loginių funkcijų testas. 5 - Furijė slankaus kablelio. 6 - Furijė neslankaus kablelio.*

Cortex-R4 generavo 280ns ilgio impulsus. Įėjimo į trūkį ir išėjimo laikas 730ns. Cortex-M4 generavo 150ns ilgio impulsus. Įėjimo į trūkį ir išėjimo laikas 200ns.

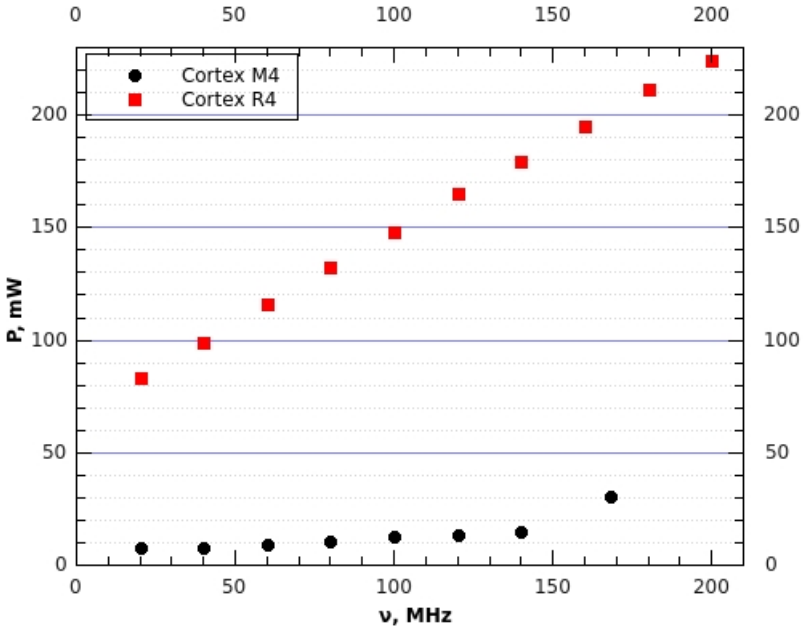


2 pav. *Generuotų impulsų laikinė diagrama.*

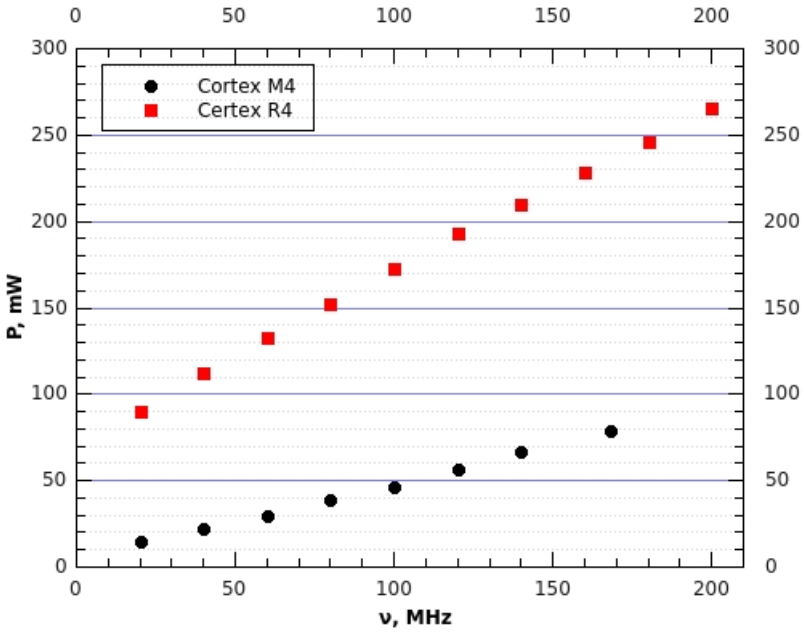
Mikrovaldiklio Cortex-R4 šerdis maitinama 1.2V, periferijos 3.3V.Išjungtos periferijos naudojo 14.6mA srovę, tad prie suvartojamos galios buvo pridėta 48.18mW dedamoji, kuri nekito visų matavimų metu .Cortex-M4 maitinamas 2.39V.



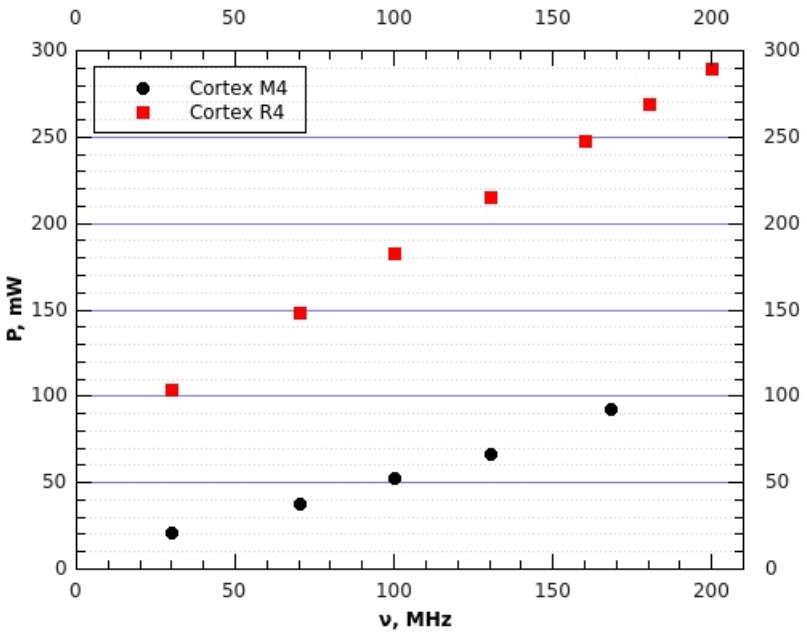
3 pav. *Mikrovaldikliai "stanby" režime.*



4 pav. *Mikrovaldiklių galios vartojimo priklausomybė nuo taktinio greičio sleep režime.*



5 pav. *Mikrovaldiklių galios sunaudojimo priklausomybė nuo taktinio greičio tuščiaje eigoje.*



6 pav. *Mikrovaldiklių galios sunaudojimo priklausomybė nuo greičio atliekant logines funkcijas*

4 Išvados

1. Atliekant skaičiavimus Cortex-R4 yra pranašesnis
2. Cortex-R4 turi tik du trūkio režimus ir įėjimo bei išėjimo laikai stipriai atsilieka nuo M4, todėl jis labiau tinkamas naudoti ten kur trūkių yra nedaug, o juose reikia atlikti daug skaičiavimų.
3. Du procesoriai esantys R4 stipriai padidina mikrovaldiklio naudojamą galią

Literatūros sąrašas

1. Q formatai ir konvertavimas.

   http://infocenter.arm.com/help/topic/com.arm.doc.dai0033a/DAI0033A_fixedpoint_appsnote.pdf

2. TMDXRM48USB trumpas aprašas.

   http://processors.wiki.ti.com/images/5/5c/RM48_USB_QUICK_START.pdf

3. Detalus Cortex-R4 aprašas.

   http://www.ti.com/lit/ds/spns177a/spns177a.pdf

4. Cortex-R4 energijos vartojimo optimizavimo aprašas.

   http://www.ti.com/lit/an/spna172a/spna172a.pdf

5. Cortex-R4 slankaus kablelio modulio savybių aprašas.

   http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dai0240a/index.html

6. STM32F4-Discovery trumpas aprašas.

   http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATA_BRIEF/DM00037955.pdf

7. Detalus Cortex-M4 mikrovaldiklio aprašas.

   http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/DM00037051.pdf

8. Cortex-M4 slankaus kablelio modulio sovybių aprašas.

   http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/DM00037051.pdf

9. π skaičiavimo algoritmas.

   http://en.wikipedia.org/wiki/Gauss-Legendre_algorithm