**Aalto University School of Science and Technology**
Faculty of Electronics, Communications and Automation
Master's Programme in Electrical Engineering

Lauri Kukkonen

# PROCESSOR EVALUATION FOR LOW POWER FREQUENCY CONVERTER PRODUCT FAMILY

Master's Thesis left to inspection for Master's Degree in Espoo 5 November 2010.

Supervisor Prof. Raimo Sepponen

Instructor MSc Kalle Suomela

**AALTO UNIVERSITY SCHOOL OF SCIENCE AND TECHNOLOGY** **ABSTACT OF THE MASTER'S THESIS**

Aalto University School of Science and Technology

Faculty of Electronics, Communications and Automation

Master's Programme in Electrical Engineering

| | | |
|---|---|---|
| Author: | Lauri Kukkonen | |
| Name of the thesis: | Processor Evaluation for Low Power Frequency Converter Product Family | |
| Date: | 5.11.2010 | Number of pages: 63 |
| Supervisor: | Prof. Raimo Sepponen | |
| Instructor: | MSc Kalle Suomela | |

The aim of this thesis is to study processors to be used in a low power frequency converter. Processors under investigation must be currently or in the near future in the market. The purpose is to examine suitability of a processor to an application in which price is an essential factor. The requirements presented in this study will determine which processor will be reviewed more closely. After a precise review, processor vendors was asked to provide as corresponding device as possible to a test.

Testing was accomplished eventually with five different processors of which two were based on a same core. The aim of the testing was to investigate suitability of the processors to their target task. Suitability was tested by executing code that models frequency converter application. As a result, spent time and clock cycles are presented in certain functions. In addition to performance, the testing included evaluation of the size of the output code the compilers created. Functions under test consisted of a combination of arithmetic and logic operations that was used to interpret the suitability of the processor.

Key words: Frequency converter, microprocessor

# AALTO-YLIOPISTON TEKNILLINEN KORKEAKOULU

# DIPLOMITYÖN TIIVISTELMÄ

Aalto-yliopiston teknillinen korkeakoulu

Elektroniikan, tietoliikenteen ja automaation tiedekunta

Elektroniikka ja sähkötekniikka

| | | |
|---|---|---|
| Tekijä: | Lauri Kukkonen | |
| Työn nimi: | Prosessorien evaluaatio pienitehoisen taajuusmuuttajaperheen komponentiksi | |
| Päiväys: | 5.11.2010 | Sivumäärä: 63 |
| Vastuuopettaja: | Prof. Raimo Sepponen | |
| Työn ohjaaja: | DI Kalle Suomela | |

Tässä työssä tutkitaan markkinoilla olevia tai lähitulevaisuudessa markkinoille saapuvia prosessoreja käytettäväksi pienitehoisissa taajuusmuutajissa. Tutkimuksen tarkoitus on selvittää prosessorin sopivuutta sovellukseen, jossa hinta on merkittävä tekijä. Tutkimuksessa esitettyjen vaatimusten perusteella houkuttelevimmat prosessorit otetaan tarkempaan tutkimukseen. Tarkemman selvityksen jälkeen vaatimuksia teknisesti mahdollisimman tarkasti vastaavat prosessorit pyydettiin valmistajalta testattavaksi.

Testaaminen suoritettiin lopulta viidelle eri prosessorille, joista kaksi perustui samaan ytimeen. Testaamisen tavoitteena on selvittää prosessorin sopivuus käyttökohteeseensa. Sopivuus testattiin suorittamalla prosessoreissa taajuusmuuttajakäyttöä mallintavaa testikoodia. Tuloksina testikoodin ajamisesta saatiin tietyissä aliohjelmissa kulutettu aika sekä kulutetut kellosyklit. Suorituskyvyn lisäksi testaukseen kuului prosessorikohtaisen kääntäjän aikaansaaman koodin koko. Aliohjelmat sisälsivät sekä aritmeettisia, että loogisia operaatioita, joiden kombinaationa mahdollisimman hyvä sopivuus saatiin selvitettyä.

Avainsanat: Taajuusmuuttaja, mikroprosessori

# PREFACE

This Master's thesis has been written to ABB Drives Low Power AC (LAC) profit centre. The results of this study are part of a current and ongoing product development. Similar studies have been written before concerning previous product development projects. The aim of this thesis is to maintain in-house knowhow concerning modern hardware in motor control applications.

I want to thank the leader of electronics team, MSc Jari Mäkilä for the opportunity to take part in this essential segment of product development. I also want to thank MSc Kalle Suomela and MSc Johanna Laukkanen for giving me the right contacts and technical support to pull through this assignment.

Helsinki, 5 November 2010

Lauri Kukkonen

# CONTENTS

# ABBREVIATIONS

| | |
|---|---|
| *ABB* | Asea Brown Boveri |
| *AC* | Alternating Current |
| *ADC* | Analog-to-Digital Converter |
| *ALU* | Arithmetic-Logic Unit |
| *CAN* | Controller Area Network |
| *CLA* | Control Law Accelerator |
| *CISC* | Complex Instruction Set Computer |
| *CSI* | Current Source Inverter |
| *CPU* | Central Processing Unit |
| *DAC* | Digital-to-Analog Converter |
| *DC* | Direct Current |
| *DSP* | Digital Signal Processor |
| *DTC* | Direct Torque Control |
| *DRAM* | Dynamic Random Access Memory |
| *eCAN* | Enhanced Controller Area Network |
| *FPU* | Floating-Point Unit |
| *GPIO* | General Purpose Input/Output |
| *GPTM* | General Purpose Timer Module |
| *HVAC* | Heating, Ventilation, and Air Conditioning |
| *IC* | Integrated Circuit |
| $I^2C$ | Inter-Integrated Circuit |
| *ISR* | Interrupt Service Ruotine |
| *IDE* | Integrated Development Environment |
| *IEEE* | Institute of Electrical and Electronics Engineers |
| *IGBT* | Insulated-Gate Bipolar Transistor |
| *IR* | Instruction Register |
| *LAC* | Low Power AC |
| *LCI* | Load Commutated Inverter |
| *LIN* | Local Interconnect Network |
| *LQFP* | Low-profile Quad Flat Package |
| *LSU* | Load Store Unit |

| | |
|---|---|
| *MAC* | Multiplying Accumulator |
| *MCU* | Microcontroller |
| *MIPS* | Millions Insturctions Per Second |
| *MOSFET* | Metal-Oxide Semiconductor Field-Effect Transistor |
| *NVIC* | Nested Vectored Interrupt Controller |
| *NRZ* | Non-Return-to-Zero |
| *OTP* | One Time Programmable |
| *PC* | Program Counter |
| *PFU* | Prefetch Unit |
| *PWM* | Pulse Width Modulation |
| *RAM* | Random Access Memory |
| *RISC* | Reduced Instruction Set Computer |
| *RMS* | Root-Mean-Square |
| *RPM* | Rounds Per Minute |
| *RSPI* | Renesas Serial Peripheral Interface |
| *ROM* | Read Only Memory |
| *SAR* | Successive Approximation Register |
| *SCI* | Serial Communication Interface |
| *SRAM* | Static Random Access Memory |
| *SVM* | Space-Vector Modulation |
| *SPI* | Serial Peripheral Interface |
| *TQFP* | Thin Quad Flat Package |
| *UART* | Universal Asynchronous Receiver/Transmitter |
| *VSI* | Voltage Source Inverter |

# PRINCIPAL SYMBOLS

| | |
|---|---|
| $B$ | Magnetic field |
| $I$ | Electric current |
| $i_r$ | Rotor current |
| $i_s$ | Stator current |
| $f$ | Frequency |
| $f_s$ | Sampling frequency |
| $L_m$ | Magnetizing inductance |
| $L_r$ | Rotor inductance |
| $l$ | Length |
| $n$ | Motor rotation speed |
| $p$ | Number of pole pairs |
| $R_s$ | Stator resistor |
| $s$ | Slip |
| $u_s$ | Stator voltage |
| $T$ | Torque |
| $T_e$ | Developed torque |
| $T_s$ | Sampling time |
| $\psi_r$ | Rotor flux linkage |
| $\psi_s$ | Stator flux linkage |
| $\Omega_s$ | Stator field geometric angular velocity |
| $\Omega_r$ | Rotor geometric angular velocity |
| $\omega_{cs}$ | Coordinate system electrical angular velocity |
| $\omega_r$ | Rotor electrical angular frequency |
| $\omega_s$ | Stator field electrical angular frequency |

# 1 INTRODUCTION

This Master's thesis has been written for ABB Drives LAC profit centre. Theoretical section of this study focuses on presenting general information on technology assiosiated to motor control and microprocessors. In testing section, a few carefully chosen processors are evaluated in a test bench.

## 1.1 ABB Group in Brief

According to ABB Group, they are a leader in power and automation technologies that enable utility and industry customers to improve performance while lowering environmental impact. The ABB Group operates in around 100 countries and employs about 117 000 people.

ABB businesses are divided in five divisions: power products, power systems, discrete automation and motion, low voltage products, and process automation. LAC profit centre is part of discrete automation and motion which provides products, solutions and related services that increase industrial productivity and energy efficiency. ABB has the leading position in wind generators and a growing offering of solutions in solar energy markets.

The recent global recession has also affected on ABB's revenue. Table 1.1 shows the most critical numbers concerning ABB's business activity.

Table 1.1 ABB's business activity in year 2008 and 2009. **[1]**

| **Total ABB Group** ($ millions) | | |
|---|---|---|
| | 2009 | 2008 |
| Orders | 30 969 | 38 282 |
| Revenues | 31 795 | 34 912 |
| Earnings before interest and taxes (EBIT) | 4 126 | 4 552 |
| EBIT % | 13.0 | 13.0 |
| Net Income | 2 901 | 3 118 |
| Basic Earnings per share ($) | 1.27 | 1.36 |
| Cash flow from operations | 4 027 | 3 958 |
| Return on capital empoyed (ROCE) | 27 | 31 |
| Number of employees | 116 000 | 120 000 |

## 1.2 Background

A microprocessor may be considered the brains and the core of operation of any embedded system. On the other hand, a microprocessor might be one of the most expensive components as a separate expense. Effect on a system cost may be also significant. Processor development, which has lasted already decades, has increased attention to possibilities that processors provide. Following continuously this development has also made possible to improve applications that use microprocessors. On the other hand, application development is even a mandatory process in order to maintain competitiveness.

While designing a new product family, one should make clear what sort of processor in expedinture sense a frequency converter can be based on. Features still must satisfy the need based on custormer feedback. In other words, very high application level problem is in question. The subject of this study is very dynamic, therefore similar studies has been done in the past. Previous studies tries to map contemporary processor supply.

## 1.3 Problem Description

The greatest challenge in this study is accepting compromises constantly. In practice, processor features are always inversely proportional to price. Satisfying device would be easy to choose if expedinture is not an issue. At this point, it is good to keep in mind that procurement process is always influenced by humane factors and previous vendor relations.

Another problems is the question what do we want from the device in development process. Requirements of the processor mirrors directly from desired features of a new frequency converter family.

## 1.4 Study Objective

Study objective briefly is to research the most suitable processor alternative for the next generation low power frequency converter family. The processor must be currently in the market or at latest in the near future.

## 1.5 Study Confining

In the beginning of this study, the basic composition and operation of a frequency converter are explained. These issues create the fundamental requirements to the processor. Chapters 2 and 3 will familiarize the reader with the principles of motor control.

Principles and basic features of a microprocessor are explained in Chapter 4. Processed features are used as selection arguments in latter part of this study. Another purpose is to familiarize the reader with technologies and related terms that are used in future in order to make fluent reading possible.

The most essential part of this study begins in Chapter 5, in which the argumentation is begun in order to solve the final contestants. In section 5.5 , the final contestants are declared to participate in a pedantic analysis. In the analysis in Chapter 6, the factors that influenced on the selection among other features are reviewed.

The testing part of this study begins in Chapter 7. Selected processor go through the same testing procedures which are created to model the operation of a frequency converter. Benchmarking is performed as consistently as possible with different alternatives. Results are compared in section 7.7.

# 2 ELECTRICAL DRIVE

Electric drive is a system that converts electric energy to mechanical energy. The simplest form is a combination of an electric motor and a power source. However, it is recommended to use a controlled drive. This means that a converter is connected between the power source and the motor. At present, a common component of a controlled drive in addition to the power source and the motor is frequency converter. Electric drives have many applications in merchandise industry and process industry as well as in vehicles and housekeeping. Examples of applications are HVAC devices, conveyors, cranes, paper machines, elevators, and robots. Applications can be classified based on power and performance. In this case, performance means wide adjustment range of speed as well as fast and accurate control of rotation speed and position. Pumps may need extremely great power while robots require precise accuracy. **[2] [3]**

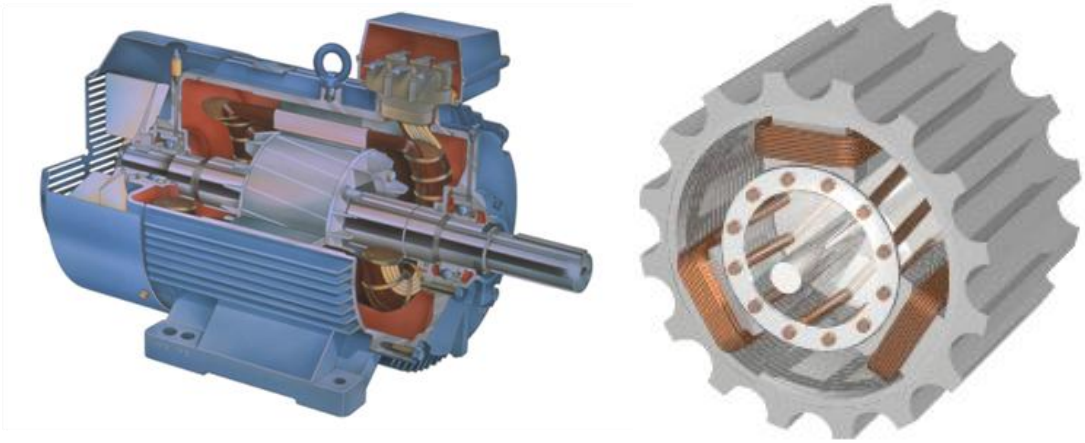
Despite the fact that electric motors consume significant part of all electricity consumption in the world, only a small proportion of them are controlled. Using a controlled drive will decrease the energy consumption substantially. In other words, in addition to improved system performance, energy and money savings can be gained.

## 2.1 Induction Motor

Induction motor is the most common motor type used in industry. It is also called asynchronous motor based on its operation. The structure of an induction motor is very simple and solid. They demand a little service and are cheap to manufacture. Without a frequency converter the rotation speed is tied to the source frequency. Other downsides are the need for parasitic power and the power loss in rotor winding while heavily loaded. Induction motors are used over a range of widely varying power. In industry, the variation may be from a couple hundred watts to several megawatts. Small induction motors can be used for example in domestic appliances such as washing machines and refrigerators. **[4]**

### 2.1.1 Structure

The two main elements in induction motor are a stator and a rotor. The stator consists of three-phase winding which are fed with AC current. The rotor may vary depending on the type of the motor. Alternatives are a squirrel cage rotor and a slip ring rotor of which the squirrel-cage is more common. It is built up of bars that span the length of the rotor and are connected through a conductive ring at each end. The structure is introduced in Figure 2.1. **[4]**



(a) (b)

Figure 2.1 a) Structure of an induction motor. b) Structure of a squirrel cage rotor inside a stator.

### 2.1.2 Operating Principle

The function of induction motor is based on a rotating magnetic field developed by the stator winding. Rotating magnetic field crosses the rotor winding inducing electromotive force impacting on them. Currents in the rotor conductors are in accordance with Lenz's law. In other words, they try to resist the rotating magnetic field. The rotor is affected by a torque which aims to rotate to the same direction as the magnetic field. When the rotation speed of the rotor is reaching the speed of the magnetic field, induced current will decrease. In static state, torque generated by the motor is equal to the counter torque originated by a load. **[4]**

To establish this by formulas, the effecting force has to be considered. When current is flowing in rotor conductors, Lorentz force is impacting on them according to the Formula (1.1). **[5]**

$$F = lI \times B \tag{1.1}$$

where $l$ is the length of the rods, $I$ is the current and $B$ is the magnetic field. The cross product causes directions of the force to be perpendicular to both current and magnetic field. This force causes perpendicular torque at the location of a magnetic pole. The torque can be denoted as

$$T_{1p} = lIB\frac{D}{2} \tag{1.2}$$

where $\frac{D}{2}$ is the radius of the rotor and therefore the lever arm of the force. The same torque is affecting at the other pole at the same time and it rotates the rotor to the same direction. Hence, the total torque is double.

$$T = lIBD \tag{1.3}$$

This equation describes the operation principle in a very simplified manner. In reality the shape and structure of both the stator and rotor have an effect.

### Rotation Speed and Slip

Speed of the motor can be determined from the rotation speed of the magnetic field. While the rotor rotates equal speed to the magnetic field, the rotating magnetic field will not induce any currents to the rotor, hence no torque will be produced. Thus, the rotation speed of the rotor differs slightly from the speed of the magnetic field. This difference is called a slip. Zero slip would be possible only in ideal situation where no torque exists. **[4]**

While the frequency of the grid beint $f$, the synchronous angular velocity of the field in static state is

$$\omega_s = 2\pi f \tag{1.4}$$

When the number of pair of poles is p, the geometrical angular velocity is

$$\Omega_s = \frac{\omega_s}{p} \tag{1.5}$$

The slip is defined as

$$s = \frac{\Omega_s - \Omega_r}{\Omega_s} \tag{1.6}$$

where $\Omega_r$ is the geometrical angular velocity of the rotor and can be restated as

$$\Omega_r = (1 - s)\Omega_s \tag{1.7}$$

Using rounds per time unit as the overall unit, the same can be written as

$$n = \frac{\Omega_r}{2\pi} \tag{1.8}$$

As an example, a motor with a number of pole pairs as 2 and having a slip of 0,03 will rotate at speed of 1455 rounds per minute (rpm).

## 2.2 Frequency Converter

In general, a frequency converter is a device that is coupled between two separated electrical networks. Usually these networks are electrical grid and an electrical motor. In electricity generation, the motor operates as a generator and the energy flow is opposite. Frequency converters are commonly called variable-frequency drives or simply drives.

### 2.2.1 Different Types of Frequency Converter

Frequency converters can be divided into two main types: with or without an intermediate direct-current link. Frequency converters with an intermediate direct-current link can be further divided into constant current drives and constant voltage drives. Existing drives without an intermediate direct-current link are matrix converter and cycloconverter. Only cycloconverter has some commercial importance. **[3]**

Constant current drives can be manufactured using either two thyristor bridges in *load commutated inverter (LCI)* or replacing the bridge with self commutation circuit

on the motor side in *current source inverter (CSI)*. CSI technology has some advantages over LCI such as smoother torque. In addition, *pulse width modulation (PWM)* can be applied. Constant voltage drives also known as *voltage source inverters (VSI)* is presently the most employed type of frequency converters. In the following sections, VSI will be simply called as frequency converter. **[3]**

### 2.2.2 Structure

Structure of a frequency converter in further detail is composed of three main elements: rectifier bridge on the electrical network side, constant DC voltage circuit or a DC bus, and an inverter circuit on the motor side. Structure is pictured in Figure 2.2. Apart of these, also the control unit and possibly a control panel have to be taken into account. The rectifier bridge can be either a diode bridge, a thyristor bridge, or an *IGBT (Insulated-Gate Bipolar Transistor)* bridge. Constant voltage circuit is composed of large capacitors to smooth the voltage alteration and store energy. There are possibly also inductors included to smooth the current. Some smaller frequency converters may be assembled without the inductors. Modern inverters are equipped with IGBT or *MOSFET (Metal-Oxide Semiconductor Field-Effect Transistor)* modules which enable the use of PWM along the whole bandwidth. IGBT modules are suited for as large as 500-1000 kW drives and voltage tolerance is at least 3,3 kV. **[3]**

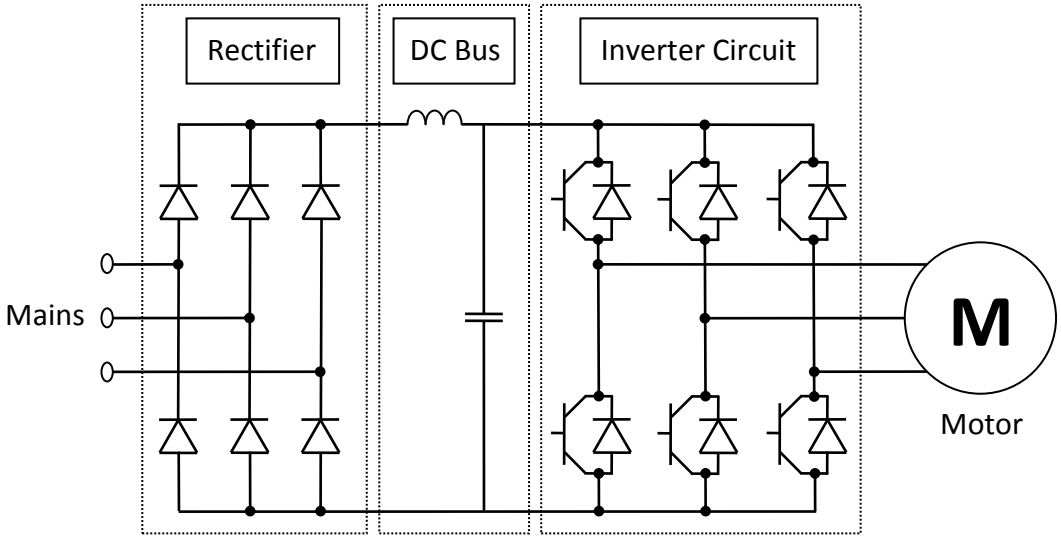



Figure 2.2 Main elements of the structure. Picture is modified from **[3]**.

### 2.2.3 Pulse Width Modulation (PWM)

Control of the inverter switches is based on PWM. Many PWM methods may be used in order to create desired AC waveform to each phase. Briefly, PWM is a modulation method with which energy traveling to a load is regulated by changing the duty cycle. Today, computation-intensive and possibly the best PWM technique to control three-phase AC motors is *space-vector modulation (SVM)*. SVM differs from more traditional methods in that there are not separate modulators used for each of the three phases. Instead, the complex reference voltage vector is processed as a whole. SVM is illustrated in more detail next. **[6]**

A three-phase inverter must be controlled so that both switches (of that particular phase) are never turned on at the same time. Otherwise, the DC supply would be shorted. This leads to eight possible switching combination. Different switching positions can be illustrated as vectors according to Figure 2.3. Each vector indicates a certain switch combination. For example, the combination in Figure 2.3a corresponds the vector $V_4$. The target is to consitute a reference signal $\vec{V}^*$, as shown in Figure 2.3b. A convenient way to generate $\vec{V}^*$ is to use adjacent vectors $\vec{V}_1$ and $\vec{V}_2$. From Figure 2.3b can be seen that $\vec{V}^*$ is the sum of vectors $\vec{V}_a$ and $\vec{V}_b$.

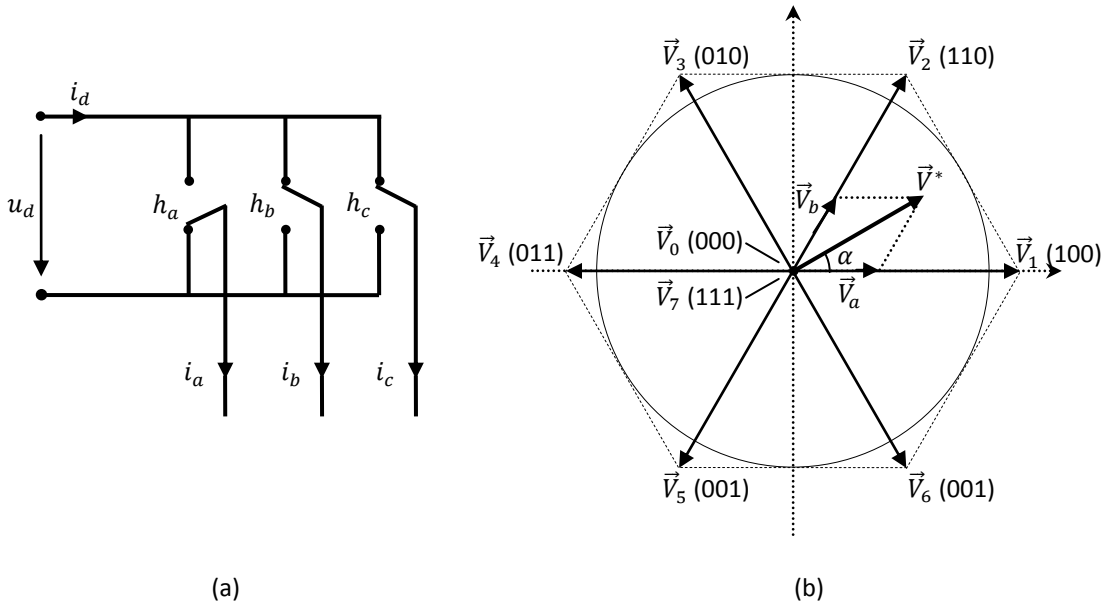



Figure 2.3 (a) Switching state illustrating vector $\vec{V}_4$ (011). (b) Space vectors of three-phase bridge inverter showing reference voltage trajectory and segments of adjacent voltage vectors.

The reference voltage vector $\vec{V}^*$ is sampled at a fixed clock frequency $f_s$, which equals to switching frequency. Sampling time, $T_s$, is the reciprocal of switching frequency, $T_s = 1/f_s$. Time period $T_c$, during which the average output matches the reference, is defined as half of the sampling time $T_s$. Reverence voltage $V^*$ is calculated in Equation (1.9).

$$V^* = V_a + V_b = V_1 \frac{t_a}{T_c} + V_2 \frac{t_b}{T_c} + V_0 \frac{t_0}{T_c} \tag{1.9}$$

where

$$t_a = \frac{V_a}{V_1} T_c \tag{1.10}$$

$$t_b = \frac{V_b}{V_2} T_c \tag{1.11}$$

$$t_0 = T_c - (t_a + t_b) \tag{1.12}$$

Time intervals $t_a$ and $t_b$ satisfy the reference voltage and time $t_0$ fills up the remaining gap for $T_c$ with the zero null vector. Figure 2.4 shows the construction of

the symmetrical pulse pattern for two consecutive $T_c$ intervals that satisfy the reverence voltage vector $\vec{V}^*$ in Figure 2.3b.

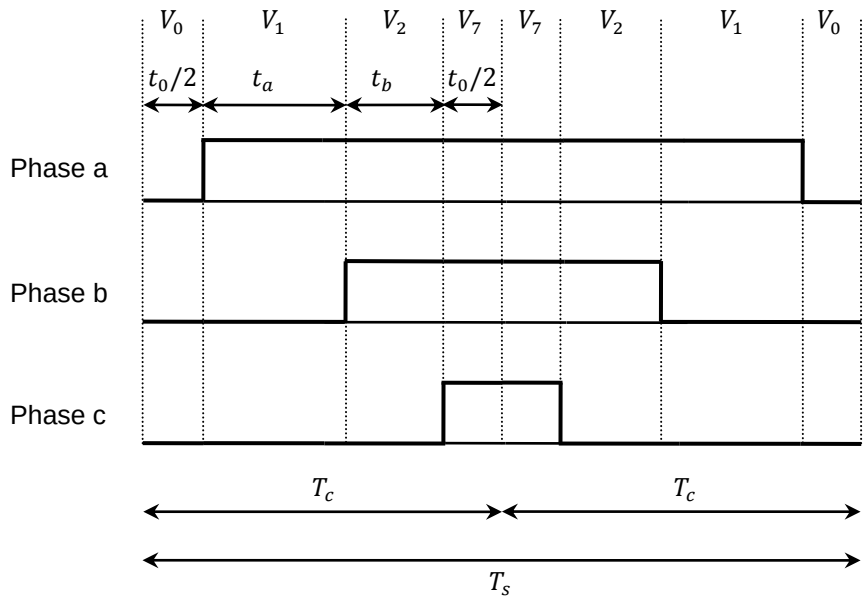



Figure 2.4 Construction of a symmetrical pulse pattern for three phases. **[6]**

## Overmodulation

Combining all switching states appropriately, any voltage vector located within the hexagon in Figure 2.3b can be generated. The circle inside the hexagon determines the limit of linear modulation. Linear modulation will result sinusoidal waveform, while exceeding the circle, the waveform is distorted. Implementing a vector that exceeds the circle, yet takes a position inside the hexacon, is called overmodulation. Overmodulation can be implemented by two alternative methods. The voltage vector must be inside the hexacon, thus an “ideal” voltage vector must be truncated to result either minimum phase error vector or minimum magnitude error vector. These alternatives are described in Figure 2.5.

Overmodulation is used to increase the root-mean-square (rms) value of the voltage. Higher rms value allows greater rotation speeds without stepping into field weakening range. It will also compensate the RMS voltage drop due to losses in frequency converter. Field weakening is explained in more detail in Chapter 3. **[7]**

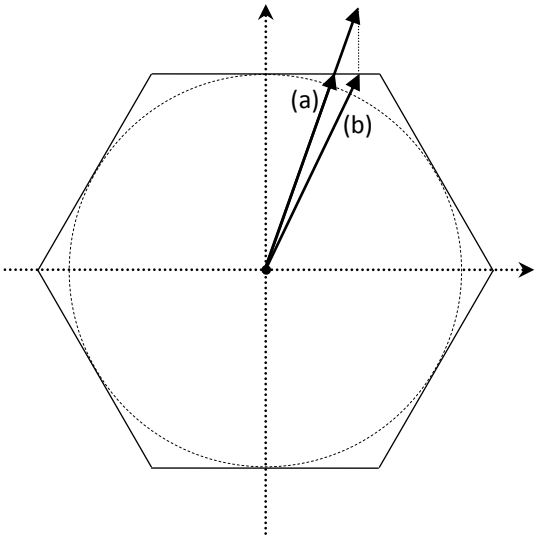



Figure 2.5 Overmodulation resulting (a) minimum phase error vector and (b) minimum magnitude error vector. **[7]**

# 3 CONTROL PRINCIPLES OF INDUCTION MOTOR

In many applications it is not practical to drive the motor by its nominal speed. Considering the interdepence of stator frequency, pole pair number and slip, the following formula can be written.

$$n = (1 - s)\frac{f}{p} \tag{3.1}$$

According to this, speed can be affected by changing input frequency, number of pole pair, or slip. Changing the input frequency is the fundamental technological issue that is the main topic of this chapter. The other means are also mentioned.

Formula (3.1) indicates that increasing the pole pair number will decrease the rotation speed of the motor. This is originated from slower rotating magnetic field in the stator. Stator may be also constructed with two different windings which will result different pole pair numbers. However, this is not very efficient way to exploit the stator, because only another winding is in use simultaneously.

The slip can be adjusted by controlling input voltage. Rotation speed depends on the input voltage and a torque provided by a load. With a lower voltage, a higher current is needed to maintain the power in air gap. With a higher current, the resistive losses increase and torque must be limited to prevent excessive increase of the slip. **[4]**

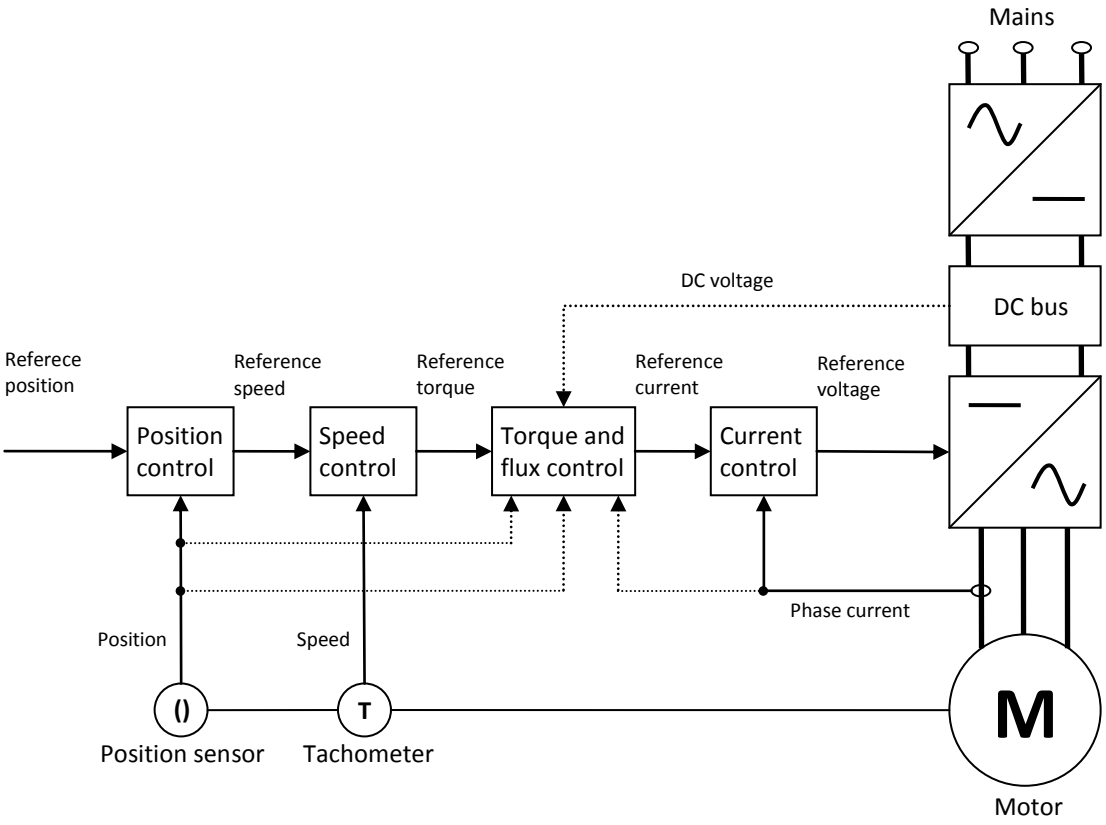



Figure 3.1 Example of a cascade control. Modified and combined picture from **[2]** and **[8]**.

In electric drives, the target is to control speed, torque, or the position of the motor axis. Cascade control is one effective manner to control induction motor. In cascade control, it is possible to divide a complex process to several simpler processes and equip them with own adjustment. Figure 3.1 describes a chain of controls including speed, torque and position control. **[2]**

## 3.1 Open-loop and Closed-loop Control

In the following sections in this chapter is studied how to control rotation speed by changing the output frequency of a voltage source inverter (VSI).

Open-loop and closed-loop control can be distinguished by the means how the target is achieved. Open-loop control is based on reference values and a model of a device. Controlling signals, that will reach a desired phenomenon, are based on the model. In closed-loop control, output is compared to the reference values. The aim of closed-loop control is to minimize this difference by changing the control signals according to the control algorithm. In future, the simple word 'control' will refer to closed-loop control. **[2]**

## 3.2 Scalar Control

Scalar control, in many occasions simply V/f control, is based on controlling only stator voltage and frequency. Dynamic model of the motor is not used which results to a low performance control. Typically, controlling system does not include speed nor torque regulation, which leads to open-loop control. Analyzing only effective value of stator voltage, following estimate may be written for reference value. **[2]**

$$u_{s,ref} = R_s i_s + \omega_{ref} \psi_{s,ref} \tag{3.2}$$

Usually the resistive term is used only when the speed is under 10…15 % of the nominal value. Omitting the resistive term and solving the angular frequency, the formula can be simplified.

$$\psi_s = \frac{u_s}{\omega} \tag{3.3}$$

For simplicity, the subscripts have been shortened. In scalar control, the aim is to maintain the stator flux linkage as constant. This means that magnetization current is also constant. If the current is kept constant, it is obvious that the frequency cannot increase without a limit due to a voltage limit. As a result, the voltage dependence on the frequency is a straight line until the voltage limit is achieved at the nominal frequency. This frequency range is called constant flux range. Increasing the frequency over the nominal value, stator flux linkage has to decrease while the voltage remains constant. This frequency range is called field weakening range. Torque depends on the stator flux linkage. Therefore, increasing frequency is achieved at the expense of torque. Power curve is following the voltage. Flux linkage, voltage, torque, and power are graphically illustrated in Figure 3.2.

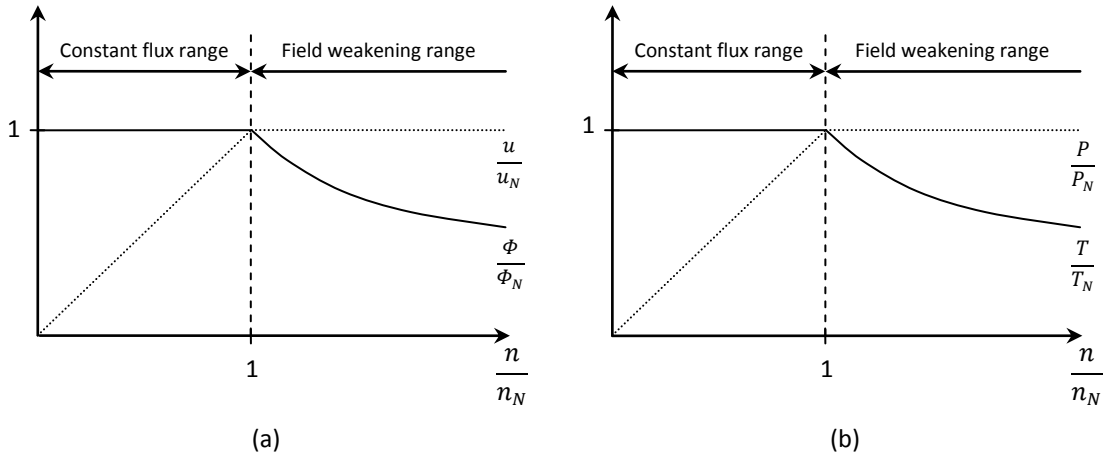



Figure 3.2 a) Flux linkage and voltage. b) Torque and power. All quantities are proportional to their nominal values while motor is fed with its nominal current. **[2]**

## 3.3 Vector Control

Utilizing vector control, significant enhancement of dynamics can be achieved. The aim is to control flux linkage and torque separately. A common means in analysis is to attach the coordinate system to rotor flux linkage. This is called rotor field orientation.

### 3.3.1 Rotor Field Orientation

To analyze the principle of rotor field orientation, formulas of rotor voltage, rotor flux linkage, and developed torque are needed. The formulas can be presented as follows in an arbitrary coordinate system. **[4]**

$$0 = R_r \underline{i}_r + \frac{d\underline{\psi}_r}{dt} + j(\omega_{cs} - \omega_r)\underline{\psi}_r \tag{3.4}$$

$$\underline{\psi}_r = L_m \underline{i}_s + L_r \underline{i}_r \tag{3.5}$$

$$T_e = \frac{3}{2} p \frac{L_m}{L_r} Im\{\underline{\psi}_r^* \underline{i}_s\} \tag{3.6}$$

In Equation (3.4), $\omega_{cs}$ is the angular velocity of a selected coordinate system. The other term $\omega_r$ is the electrical angular velocity of the rotor. If $\omega_{cs}$ is the synchronous angular velocity, $\omega_{cs} - \omega_r$ denotes the slip frequency. Solving vector $\underline{i}_r$ from the Equation (3.5), the rotor voltage Equation (3.4) can be presented as

$$0 = \frac{R_r}{L_r}\psi_r - \frac{R_r}{L_r} L_m (i_{sd} + j i_{sq}) + \frac{d\psi_r}{dt} + j(\omega_{cs} - \omega_r)\psi_r \tag{3.7}$$

where the stator current $\underline{i}_s$ has been separated to a real part $i_{sd}$ (d-axis) and a complex part $i_{sq}$ (q-axis). Rotor flux linkage $\psi_r$ has only real part in rotor field oriented system, thus the vector indicator may be discarded. Solving the real part of the current, the following equation can be obtained.

$$\frac{L_r}{R_r}\frac{d\psi_r}{dt} + \psi_r = L_m i_{sd} \tag{3.8}$$

From this equation can be seen that if the direction of the flux is known, the magnitude of rotor flux linkage may be controlled by regulating $i_{sd}$, in other words the d-component of the stator current. Using separated complex form of the stator current, torque equation can be restated as

$$T_e = \frac{3}{2} p \frac{L_m}{L_r} Im\{\psi_r (i_{sd} + j i_{sq})\} = \frac{3}{2} p \frac{L_m}{L_r} \psi_r i_{sq} \quad (3.9)$$

As an inference, torque can be controlled quickly by regulating the q-component of the stator current which is perpendicular to the rotor flux linkage.

### 3.3.2 Controlling System

An example of a block diagram of a vector control system is illustrated in Figure 3.3. The estimate of rotor flux linkage $\psi_r$ and its angle $\vartheta_k$ can be obtained by calculation from measured stator field orientated stator current $i_s^s$ and stator voltage $u_s^s$. Stator current is then converted to rotor field orientated value by multiplying it by $e^{-\vartheta_k}$. D-component of the reference stator current can be obtained from $\psi_r$ and its reference value, $\psi_{r,ref}$. Combination of $\psi_r$ and reference torque $T_{e,ref}$ will determine the q-component of the reference stator current. $T_{e,ref}$ is offered by the speed regulator. Rotor field orientated $u_{s,ref}$ is the output of the current regulator. PWM module is controlled by the reference voltage $u_{s,ref}^s$, which is gained by multiplying $u_{s,ref}$ by $e^{\vartheta_k}$. Coordinate system conversions between dq-components and phase components are not visible in Figure 3.3, only they are included in other blocks. Neither is pole pair number present in the figure, although calculations are executed using electrical angular velocity $\omega_r$, not mechanical angular velocity $\omega_m$. In other words, speed regulation and rotor flux linkage estimation blocks are equipped with a multiplier equal to pole pair number. **[4]**

## 3.4 Direct Torque Control (DTC)

DTC controls directly the basic quantities of a motor, in other words torque and speed. It is based on measured DC bus voltage and stator current. The aim is to estimate flux magnitude and torque from the measurements and compare them with their reference values. Analysis is done in stator flux oriented coordinate system and thus no coordinate system conversion is needed. In DTC, coupling to the motor is done based on flux and torque without concerning the form of the voltage. In continuous state, the output voltage is sinusoidal. Distinction to PWM inverters, DTC does not feed the motor with sinusoidal voltage in labile state. With DTC, even better dynamics can be achieved compared to DC drives. **[8]**

The operation of DTC can be illustrated with Figure 3.4. Switches $h_a$, $h_b$and $h_c$ are connected between the DC-voltage circuit ($u_{dc}$). The state of the switches can be denoted as zero or one depending whether they are down or up. Voltage vectors $u_1 \dots u_6$ with a length of $\frac{2}{3}u_{dc}$ cover the whole circle and the angle between them is $\frac{\pi}{3}$. In addition, two zero-voltage vectors exists which can be achieved by setting all the switches either up or down. Asserting a certain voltage vector, it is quite straightforward to find the equivalent states of switch. Combination in Figure 3.4a corresponds to the voltage vector $u_4$.

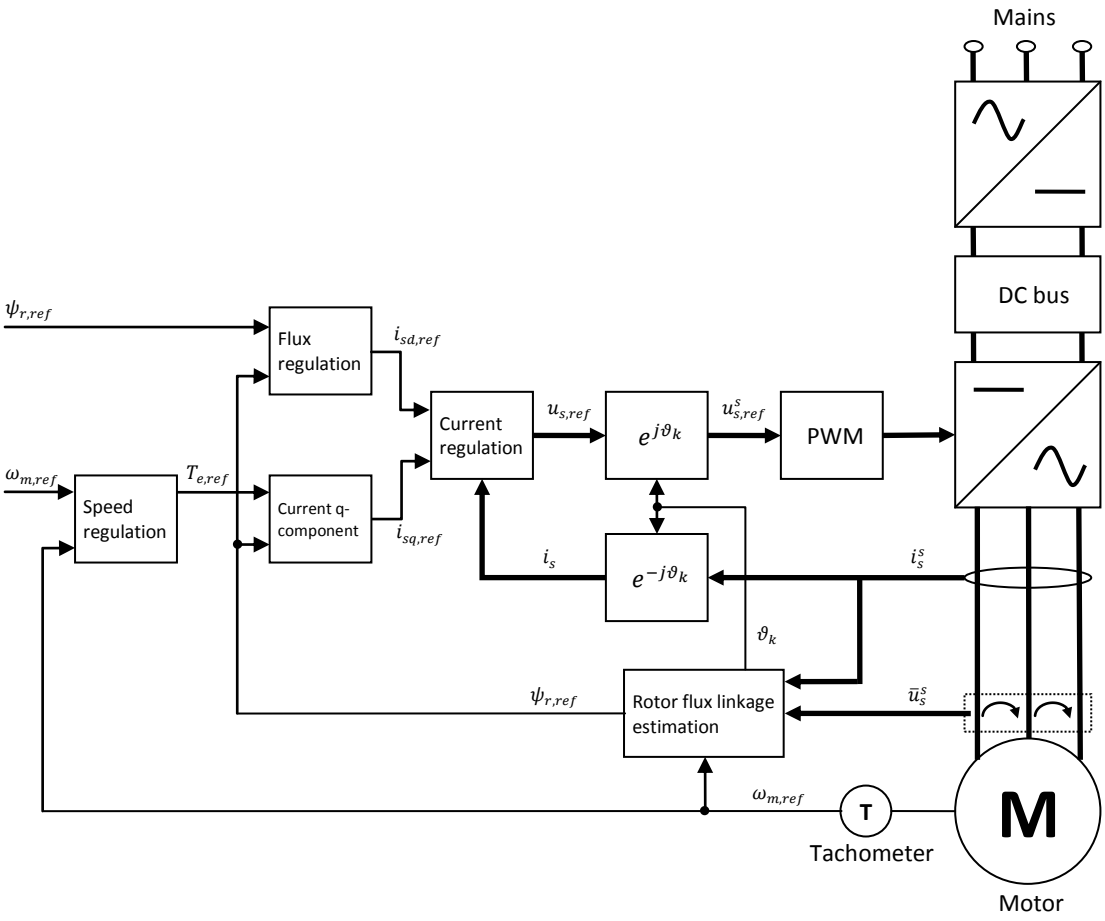
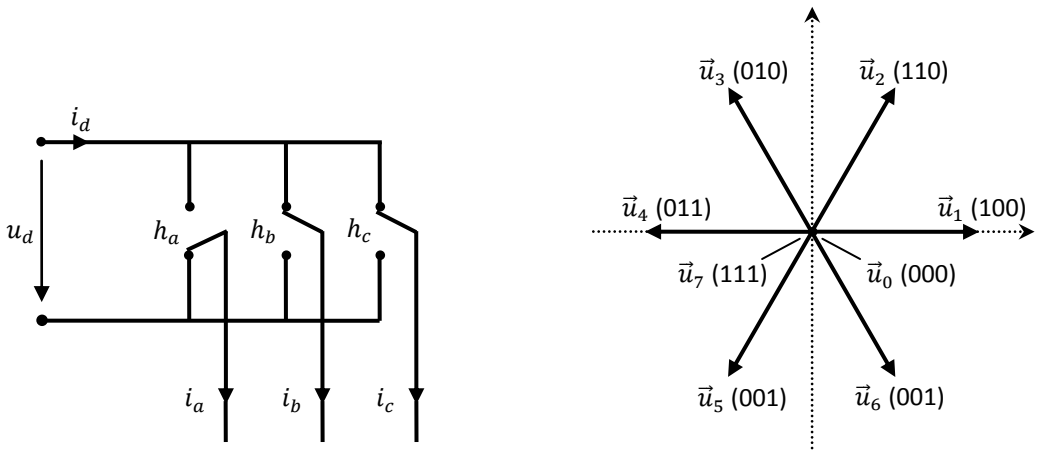



Figure 3.3 A flowchart of a direct rotor flux oriented vector control with a tachometer. Stronger lines indicate quantities in component form. Modified image from **[4]** and **[8]**.




(a) (b)

Figure 3.4 a) Example of an inverter in (011) position. (b) Space vectors of output voltage in stator oriented coordinate system. **[4]**

# 4 MICROPROCESSOR

Microprocessor (or shortly processor) is a device that includes *central processing unit (CPU)* and other functional elements inside an *integrated circuit (IC)*. Microprocessors are intended to solve computational problems in a large variety of applications. The history of microprocessors begins in early 1970s when several manufacturers were developing their project simultaneously. The first commercial single IC processor was Intel 4004, released in 1971 **[9]**.

## 4.1 Different Types of Microprocessors

Microcontroller (MCU) is the core of intelligence in any embedded system. Microcontroller can be considered a simple microprocessor with some support functions, such as an oscillator, timers, serial communication devices, and analog-to-digigal converter (ADC). Also different types of memory are attached inside the same integrated circuit. Incorporating peripherals and memory into the same IC reduces the number of separate ICs, resulting in compact and low-power implementations. The number of peripherals depends on the application to which the microcontroller is designed.

Microcontrollers are used in automatically controlled products and devices. The development of all types of microprocessors has been the key factor of allowing the design of more complex and controllable frequency converters. Actually, the microprocessor in a frequency converter processes so complex signals that it is often referred as *digital signal processor (DSP)*. Digital signal processor is a processor that is highly optimized for processing large amounts of data. In motor control the data would be for example various measurements, such as stator current and rotation speed. DSPs often provide instructions that are central to certain digital signal processing, such as transforming vectors or metrics of data. Frequently used arithmetic functions such as multiply-and-accumulate, are implemented in hardware. Using hardware implementation, faster actions can be achieved than with a software implementation running on a general-purpose processor. DSP may also allow for execution of some functions in parallel. **[10]**

## 4.2 Arithmetic of a Microprocessor

Representing numbers in a processor is based on binary system. In binary system, one number (bit) can be either zero or one. One byte consists of eight bits and a word consists of two or more bits. Processors use two different manners to represent number, fixed point representation and floating point representation. Advantages and disadvantages are explained for both representation in the following section.

### 4.2.1 Fixed-Point Numbers

In fixed-point arithmetic, numbers are represented either as integers or as fractions between -1.0 and +1.0. In practice, most fixed-point DSPs support integer arithmetic and fractional arithmetic. Fractional arithmetic is most useful for signal processing algorithms, while integer arithmetic is useful for control operations, address calculations, and other operations that do not involve signals.

The most common method of representing signed integers on computers is two's complement. Signed integer means, that the most significant bit determines the sign of the number. Two's complement's advantages are the simplicity to add and subtract numbers and a single representation of zero. The representation of eight bit number in two's complement is presented in Figure 4.1. **[11]**

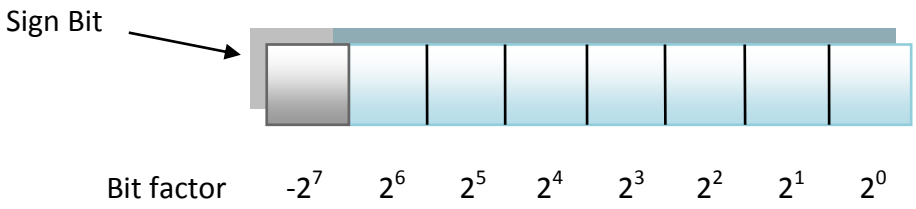



Figure 4.1 8-bit number two's complement representation. **[11]**

### 4.2.2 Floating-Point Numbers

In *floating-point arithmetic*, numbers are represented by the combination of a mantissa and an exponent. The value of the number is represented in the following form:

$$value = mantissa\ x\ 2^{exponent}$$

The mantissa is usually a signed fractional value with a single implied integer bit. The exponent is an integer that represents the number of places that the binary point of the mantissa must be shifted left or right to obtain the original number represented. In general, floating-point processors also support fixed-point data formats. This is necessary to facilitate operations that are inherently integer in nature, such as memory address computations. A simple floating-point data representation is illustrated in Figure 4.2.

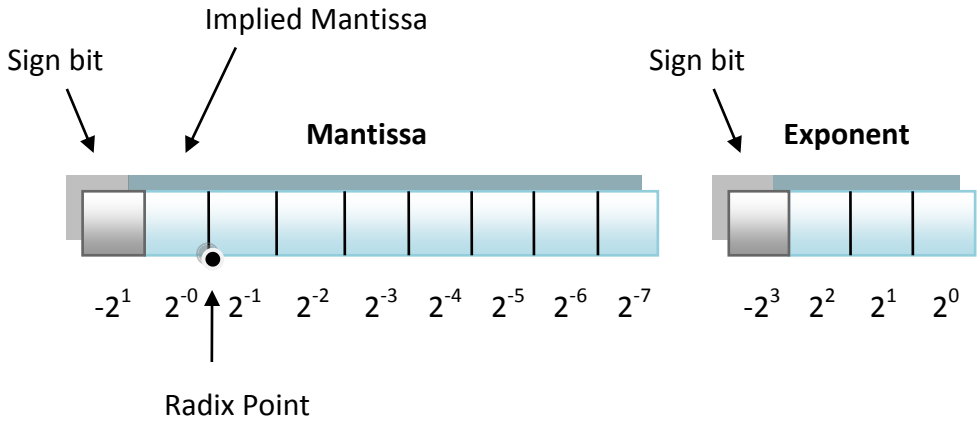



Figure 4.2 Simplified binary floating-point representation. **[11]**

Implied mantissa bit is assumed to always be set to '1', and therefore, the mantissa can have a value only in the ranges of +1,0 to +2,0 and -1,0 to -2,0. **[11]**

Floating-point data representation can be defined in many formats. In 1985, the Institute of Electrical and Electronics Engineers released IEEE Standard 754, which defines standard formats and a set of standard rules for floating-point arithmetic. **[11]**

### 4.2.3 Arithmetic Comparison

Floating-point arithmetic is a more flexible and general mechanism than fixed-point. It provides a wider dynamic range, which means the ratio between the largest and smallest numbers that can be represented. Precision is also better in many cases compared to fixed-point arithmetic. In many cases, the programmer does not have to be concerned about dynamic range and precision with floating-point arithmetic. On the other hand, more complex circuitry is needed, which implies a larger and more expensive chip. **[11]**

## 4.3 Basic Architecture

In this section, a basic architecture of general-purpose processor is briefly covered. CPU consists of a datapath and a control unit, tightly linked with a memory. The basic architecture in a nutshell is described in Figure 4.3. **[10]**

Before going further, the core operation of a microprocessor is stated. Processor executes instructions stored in program memory sequentially. *Program counter (PC)* is a register that is used to hold the information where the processor currently is int the sequence. The operation is presented in more detail in the following sections.

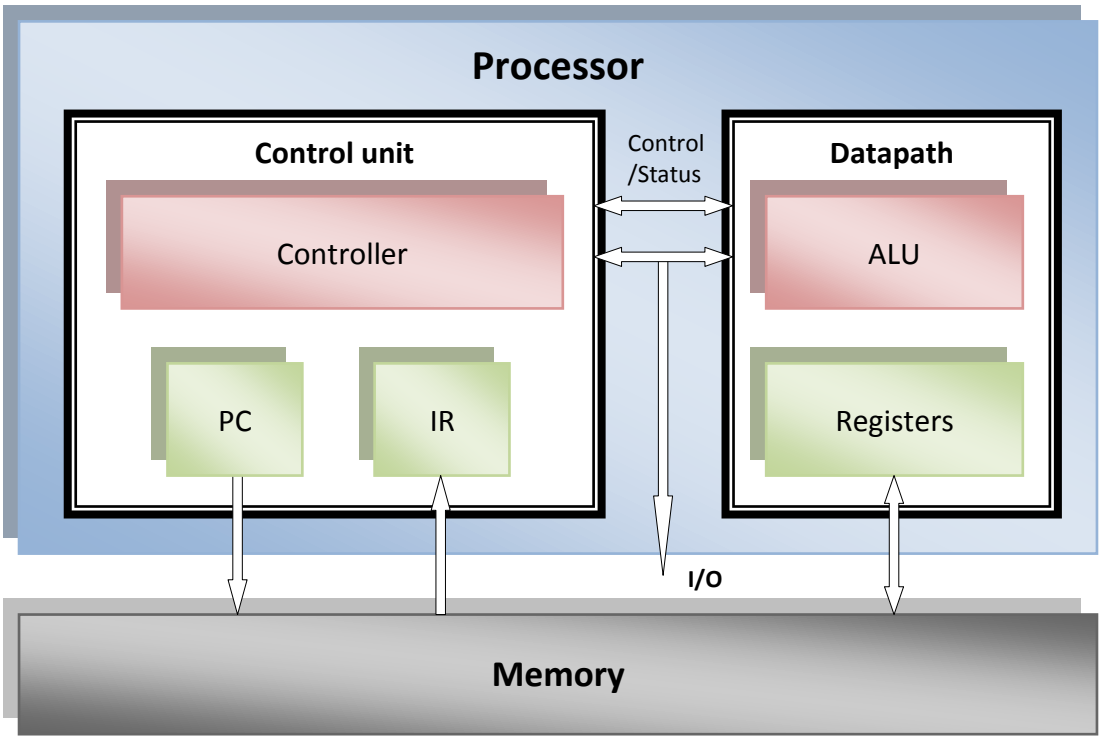



Figure 4.3 Basic architecture of a general purpose processor. **[10]**

In Figure 4.3, the black lines indicate both wires and buses. The term bus refers to a set of wires with a single function within a communication. For example, address

bus and data bus are needed in order to read from or write something to the memory. Commonly the term bus can also indicate an entire collection of wires and buses.

### 4.3.1 Datapath

The datapath consists of the circuitry where the actual processing is done. It contains at least an *arithmetic-logic unit (ALU)* and registers. The main task of the ALU is transforming data through operations such as additions and subtractions, multiplications, logical functions, inverting, and shifting. Registers are used to store temporary data. Temporary data may vary depending on the situation. It may include data brought in from the memory but not yet sent through the ALU or data coming from the ALU. Data from the ALU may be used for later ALU operations or will be sent back to memory. **[10]**

### 4.3.2 Control Unit

The control unit consists of circuitry for retrieving program instructions and for moving data to, from, and trough the datapath according to those instructions. The control unit has a program counter (PC) which was mentioned already in the previous section. PC holds the address of the next program instruction to fetch. The fetched instruction is held in an *instruction register (IR)*. The control unit also has a controller that generates the control signals necessary to read instructions into the IR and control the flow of data in the datapath. **[9]**

### 4.3.3 Memory

Memory is needed to store longer-term information used by the processor. Information can be classified as either program information or data. Program information consists of the sequence of instructions and data represents the values being input, transformed, and output by the program.

Program memory and data memory can have a combined or separated address space. In Von Neumann architecture (also known as Princeton architecture), the memory space is combined and in Harvard architecture, the program memory space is distinct from the data memory space. While Von Neumann architecture may result in a simpler hardware connection to memory, Harvard architecture has a few advantages. Harvard architecture can perform instructions and data fetches simultaneously. Another advantage is that the memories do not need to share same characteristics. In particular, the word width, timing, implementation technology, and memory address structure may differ. Many modern DSPs used in electric drives utilize Harvard architecture. Figure 4.4 illustrates these memory architectures. **[10]**

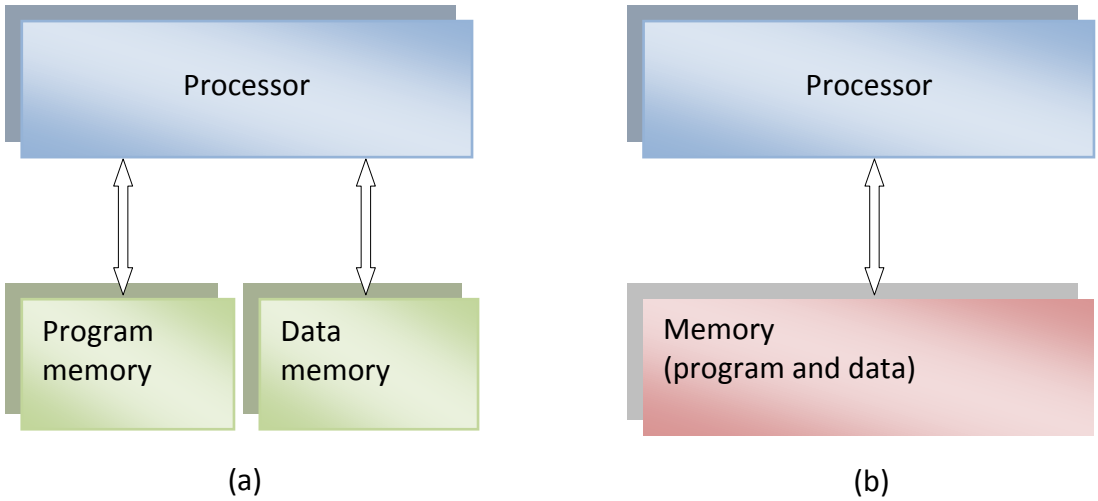



Figure 4.4 Alternative memory architectures: a) Harvard architecture and b) Von Neumann architecture. **[10]**

Memory can be implemented in a separate IC or integrated in the same IC with the other functions of the processor. To reduce the time needed to access memory, a local copy of a portion of memory may be kept in a small but especially fast memory called cache.

Various techniques may be implemented to manufacture on-chip memory. Many microprocessors include fast *Random Access Memory (RAM)* as data memory and *Read Only Memory (ROM)* as program memory. Today, ROM memory is often replaced with re-programmable Flash memory. Information in RAM is lost when the power is switched off from the system while Flash has the ability to hold the information.

### 4.3.4 Instruction Set and Operation

Instruction set is a key element of the processor's architecture. It determines the operations that are possible on the processor. Depending on the instruction set, certain operations are more efficient and natural than others. Technically the instruction set describes the bit configurations allowed in the IR. Each bit sequence in the IR forms an assembly instruction, and a sequence of instructions forms an assembly program. These instructions control how data are sequenced through the processor's data path and how values are read and written to memory. **[10]**

A typical instruction consists of opcode field and oprand fields. An opcode specifies the operation to take place during the instruction. Instruction can be coarsely classified into three categories. Data-transfer instructions move data according to the operands. Depending on the opcode, the data movement can be between registers and memory, registers and input/output channels, or between registers themselves. Arithmetic/logical instructions configure the ALU to carry out a particular function, move data from the registers through the ALU, and move data from the ALU back to a particular register. Branch instructions determine possibly the address of the next program instruction depending on the type of the branch instruction. **[10]**

Microprocessors may be categorized in two large classes by characterizing their instruction set architectures: *complex instruction set computer (CISC)* and *reduced instruction set computer (RISC)*. The instruction set of CISC microprocessors can handle both basic operations and complex functions and typically can take many clock cycles to complete. RISC architecture processors are characterized by large register set and a small instruction set containing frequent simple instructions that can be executed in one clock cycle. **[12]**

### Instruction Execution and Pipelining

Execution of instructions typically consists of four basic stages:

1. Fetch instruction (F) : CPU gets the instruction from the memory
2. Decode instruction (D) : CPU decodes the instruction
3. Fetch operands (R) : CPU reads the operands from the memory
4. Execute operation (E) : CPU executes the instruction

In the microprocessor basic architecture, the instructions are executed in a sequential manner as shown in Figure 4.5.

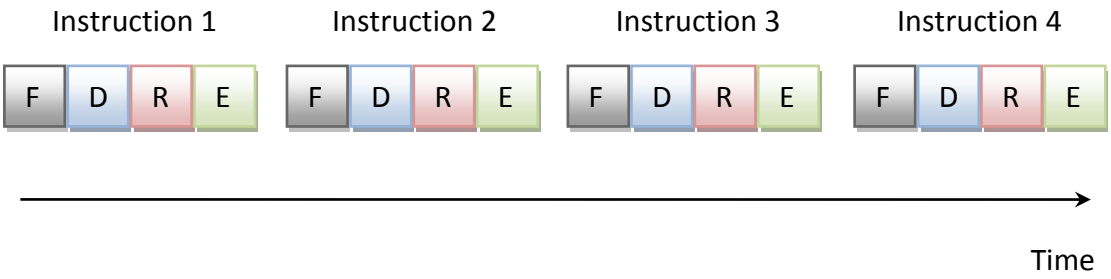



Figure 4.5 Instruction cycles of a basic architecture. **[12]**

In basic architecture, each stage is working only one cycle while being in idle during the rest three cycles. Pipelining is a common means to increase the instruction throughput of a microprocessor. In pipeline architecture basic operations (fetch, decode, read, execute) are allowed to overlap so the microprocessor can handle several instructions at the same time. The execution time can be thus reduced as shown in Figure 4.6.

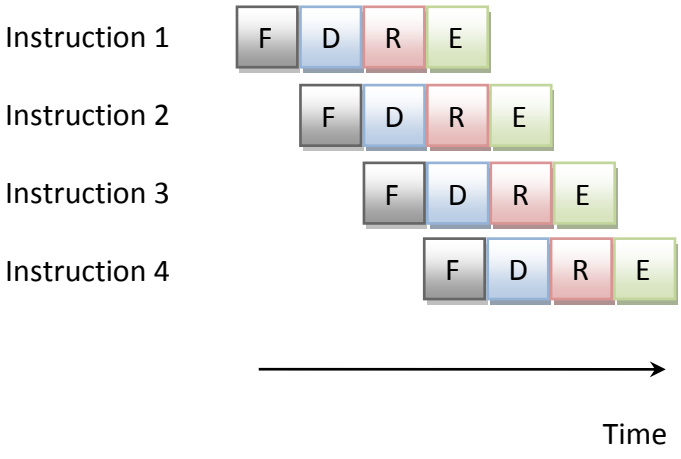



Figure 4.6 Executing instructions using pipeline. **[12]**

### 4.3.5 Cache

In order to avoid using the slower *dynamic random access memory (DRAM)* technology, numerous microprocessors are equipped with fast cache that is implemented with static *random access memory (SRAM)* technology. Typically the size of the cache is very modest compared to other elements in memory hierarchy. The aim is to keep only the most essential data in the cache as possible at each time.

The operation of the whole memory system is hierarchical. When some memory address is accessed, first the system checks for a copy of that location in cache. If the copy is in the cache, called a cache hit, then the access is quick. If the copy is not there, called a cache miss, the address must be first read into the cache. There are several cache design choices which can have a significant impact on system cost, performance, as well as power. **[10]**

### 4.3.6 Interrupt Controller

Interrupt controller is usually a built-in element in a processor that forwards an interrupt from an external souce to the central processing unit. External interrupt may be provided for example by some peripheral. An interrupt will cause the processor to stop executing its current program and branch to a special block of code called an *interrupt service routine (ISR)*. At this time, some particular subroutine will be executed. After handling the interrupt, the processor will resume execution where it left off. Interrupt can come from a variety of sources, such as on-chip peripherals, external interrupt lines, or software interrupts. Interrupt controller is equipped with a certain amount of interrupt lines. The interrupt lines may have unequal priority that assures the urgent interrupt to be handled first. **[2]**

## 4.4 Peripherals

In order to function effectively, a microprocessor in embedded system must provide a good selection of on-chip peripherals and peripheral interfaces. This minimizes the demand for external hardware to support its operation and interface it to the outside world. In this section, the essential peripherals for motor control microprocessors are briefly discussed.

### 4.4.1 Communication Interface

Transfer of data between a microprocessor and peripherals can be accomplished by serial or parallel transmission. A serial interface transmits and receives data one bit at a time. In contrast, parallel ports send and receive data in parallel, typically 8, 16, or 32 bits at a time. Serial interface is more compact form, requiring as few as three or four pins for a complete interface. On the other hand, serial interface do not function as fast as parallel interface.

Serial interface can be categorized into two classes: synchronous and asynchronous. A synchronous serial port transmits a bit clock signal in addition to the serial data bits. The receiver uses this clock to decide when to sample the received data. Figure 4.7a shows the waveforms of a typical synchronous peripheral interface. In asynchronous communication, clock is not contained within the data stream. The transmitter sends data at a programmed frequency and the receiver operates at the

same nominal frequency. The receiver clock is required to resynchronize on each character. Specialized data communication unit, which are commonly known as *universal asynchronous receiver transmitter (UART)*, are needed to interface the microprocessors and the communication channel. The efficiency of asynchronous transmission is lower than that of synchronous transmission due to the control bits required for each data character. A typical asynchronous serial communication interface is shown in Figure 4.7b. **[12]**

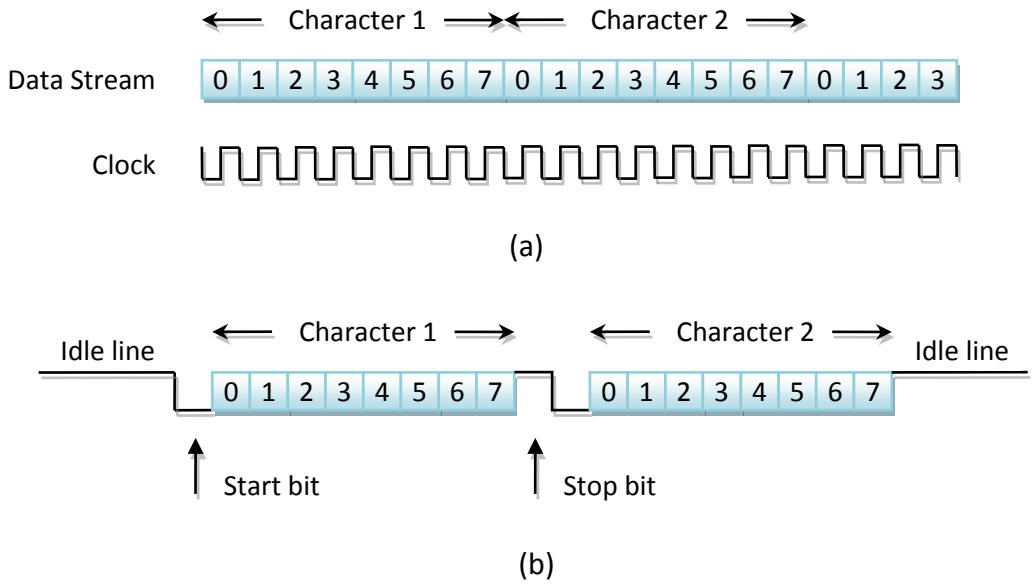



Figure 4.7 Serial communication: a) synchronous mode and b) asynchronous mode. **[12]**

Examples of commonly used serial communication standards are *serial peripheral interface (SPI)* and RS-232. SPI is a synchronous serial data link while RS-232 is asynchronous. RS-232 is commonly known as serial port in personal computer. In the following chapters, *serial communication interface (SCI)* stands for generally other serial interfaces than SPI.

Parallel communication is used when high-speed data transfer is needed. Disadvantage is the requirement of multiple-conductor cables and connectors.

### 4.4.2 Timers

A timer is an extremely common peripheral device that can measure time intervals. Fundamentally, a timer is much like a serial port bit clock generator: it consists of a clock source, a prescaler, and a counter, as shown in Figure 4.8.

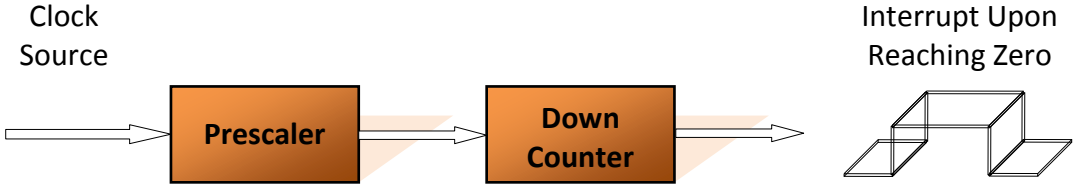



Figure 4.8 A typical DSP processor timer peripheral. **[10]**

The purpose of the prescaler is to reduce the frequency of the clock source so that the counter can count longer periods of time. Lower frequency is achieved by dividing the source clock frequency by some selectable value. Counter then uses this prescaled signal as a clock source, typically counting down from a preloaded value on every rising clock edge. Counter is often programmed to interrupt the processor upon reaching zero.

A special type of timer is a watchdog timer. In normal function, a signal is generated for the watchdog timer every certain time interval. If the watchdog timer does not succeed to receive this signal, the timer "times out" and will indicate a failure. One common use of a watchdog timer is to enable an embedded system to restart itself in case of a failure. **[10]**

Digital implementatation of PWM is also realized by using timers. Figure 2.4 can be used here as an example. The first switching takes place at $t_0/2$ when a PWM timer runs out. At this moment, the timer is reloaded with a value $t_a$. After the time period $t_a$, a new switching combination is coupled and timer is reloaded with a value $t_b$. When time $t_b$ is spent, a new switching combination is again coupled to correspond zero vector and timer is reloaded with $t_0/2$.

### 4.4.3 A/D Converter

In order to process analog signals from the environment, microprocessor needs to convert the signal to suitable binary data. This conversion is executed by the A/D converter (or simply ADC). A/D converters can be divided into two main classes: indirect converters and direct converters. Indirect converter implies that the conversion is first made to the ratio of voltage and frequency and then obtained a numeric value with a counter. Direct converter includes a Flash converter and various types of closed-loop converters. The most common type is describes in this section. **[2]**

Present standard is a method called *successive approximation register (SAR)*. First the analog voltage is stored in a sample and hold circuit in order to maintain a constant value during the conversion. SAR is initialized so that the most significant bit is equal to a digital "one". This code the converted by the *digital-to-analog converter (DAC)* to obtain a reference value to the analog input. Now the comparator compares the reference value to the input voltage. If the input voltage is higher than the reference value, the most significant bit will retain its value in the SAR. Otherwise, the most significant bit will be set to zero. The same operation is performed to all bits sequentially until the final result is achieved. Functional blocks are shown in Figure 4.9. **[2]**

Advantages of SAR converter are relatively high speed and adaptability. The conversion takes one clock-cycle for each bit of resolution desired. One converter can also be used to digitize multiple analog signals. Then each analog signal is stored to a specific sample and hold circuit to which the converter is coupled in turn with a multiplexer.

The quality of the ADC can be determined by its capabilities. The most fundamental features are resolution, accuracy, and sample rate. Resolution indicates how many quantization level the converter possesses. Accuracy describes how the converter

behaves with different types of errors. The sample rate simply defines the amount of samples the converter produces in a second.

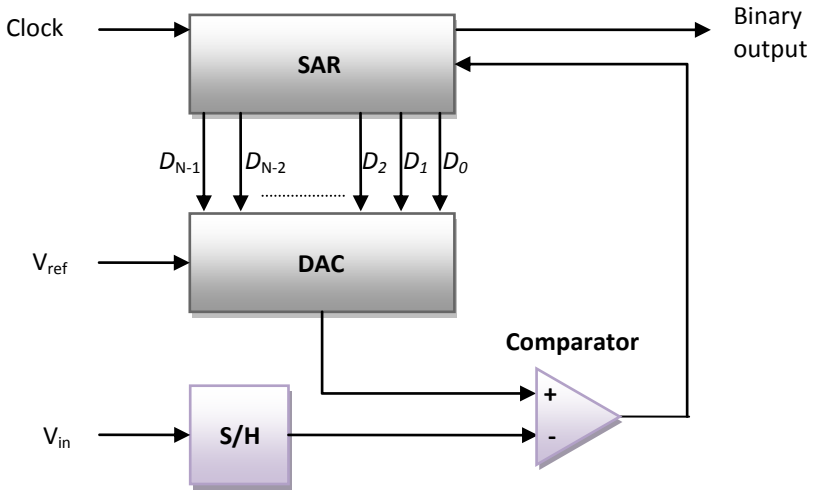



Figure 4.9 Structure of SAR converter. Black/white boxes indicate digital blocks while purple boxes indicate components that do not handle binary data. Picture is modified picture from **[2].**

# 5 PROCESSOR REQUIREMENTS AND ALTERNATIVES

The scope of this chapter is to determine requirements of the processor to be used in low-power frequency converter. The aim is to find a processor that fulfills the requirements in a reasonable price. Low-power products have great volumes and thus, the price has a significant role in decision making. This may lead to certain manufacturers ABB has good relations with and prior contracts. In addition to the technical requirements, the availability is a major issue. Selected MCU must be in right point of its life cycle in order to maintain availability. Also support from the manufacturer must be taken into account.

In the following sections, requirements are discussed in itemized form. Basically, topics in the fourth chapter are analyzed in a more practical manner. This analysis is then used making exclusive decision between processor alternatives.

Facts in the following sections are not strictly based on science. On the contrary, they are based on sophisticated conjectures made mostly by experienced professionals. Information from previous product families is used and cognizable modification to the following family is taken into account.

## 5.1 Performance and memory

Evaluating processor's performance is not always a straightforward task. The easiest means are to compare the clock frequency of the processor. Due to versatile processor architectures, clock frequency is not comprehensively comparative quantity. Often used measure to determine the speed of a processor is the amount of instructions executed per second. Common unit is *million instructions per second (MIPS).* It is still essential to keep in mind that the amount of instructions executed in a time unit depends highly on the applications and instruction sets. Some artificial instruction sequences may lead to greatly higher values than a sequence of a practical application. In addition, it is not meaningful to compare RISC and CISC processors in this manner. To ease the comparison, computing benchmark programs has been developed.

The scope in this research are processors having the speed of 60 – 100 MIPS. This was the first fundamental requirement for the manufacturers to suggest some of their models.

Defining the minimum internal memory is extremely difficult task at the present juncture of product development. Determining precise requirements for future product family is fairly impossible. To provide some boundary conditions to the manufacturers, memory requirements were kept somewhat identical to the present processor. In more detail, minimum amount of flash memory was quantified to 256 kilobytes and the amount of RAM memory to 32 kilobytes. In addition, an option of expanding to external memory is mandatory.

## 5.2 Peripherals and communication

The third baseline is the peripherals. Certain amount of features and communication are required in order to create a good quality motor control. The most fundamental of them are reviewed in this section.

According to the ABB specialists, A/D-converter inside the processor has to fulfill at least the same resolution as the present processor does. Consequently, the requirements were set to 12-bit resolution and 12 input channels. In many occasions, processors under investigation provide more than 12 channels with several separate A/D-converters.

Minimum requirement for PWM outputs was designated to be six. IGBT-modules can be driven with these six channels and additional brake chopper can be implemented with a universal timer if necessary. In addition to PWM timer, processor must include at least three general purpose timers.

Communication interfaces are fundamental to external peripherals and memory. SPI is especially essential in order to have external memory and SCI is used for example to attach a control panel or even a PC to the drive. Field bus is also a relevant feature of the following product family, and therefore *controller area network (CAN)* was kept mandatory. Minimum requirements were one CAN, one SPI and two SCIs. Numerous modern processors fulfill these requirements even extravagantly. Minimum technical requirements sent to manufacturers are shown in Table 5.1.

Table 5.1 Technical requirements.

| Feature | Requirement |
|---|---|
| **Performance:** | 60 MIPS |
| **RAM:** | 32 kB |
| **Flash Memory:** | 256 kB |
| **A/D Converter:** | 12-bit x 12 channels |
| **SCI:** | 2 |
| **SPI:** | 1 |
| **CAN:** | 1 |
| **PWM:** | 6 channels |

## 5.3 Availability

In order to maintain reliable production, availability of the components has to be taken into account. Attention has to be paid especially to processors due to their laborious replacement project. Requirements, sent to manufacturers concerning availability, are described next. Processor is recently added to product road map, release date to market is not later than during the third quarter year in 2011, and the processor has to be in production for at least several following 10 years. Road map means the selection of present products and available products in the near future already having specifications.

## 5.4 Security

One topic of investigation is the possibility to encrypt the code loaded to the processor. At a later point of the evaluation, this option was set mandatory. Ability to encrypt the code will prevent other parties to abuse ABB's knowhow. Encryption will make backward engineering and code uploading impossible or at least complicate it drastically. Different manufacturers have their own means to implement security technologies which they call by their individual names.

## 5.5 Processor alternatives

After a comprehensive evaluations based on technical data and prices, a few processors pulled through to the testing phase. Due to time limitations concerning this thesis and sample availabilities, all processors could not be tested. Processors under investigation in the last phase include two processors from Texas Instruments. Other processors are from different vendors. The processors are:

Texas Instruments TMS320F28057PNT

Texas Instruments LM3S9B96

STMicroelectronics STM32F215VET6

Freescale MC56F8441

Unfortunately, exactly same models were not available and every manufacturer offered some close alternative. Texas Instruments provided their TMS320F28035 model to the test. TMS320F28035 does not fulfill the requirements in Table 5.1, nevertheless directional results could be achieved. STMicroelectornics offered STM32F217IGT6 which has additionally camera and Ethernet interfaces. Otherwise 217 model is identical to 215 model. The two letters after STM32F21x indicate pin count and Flash memory size. LM3S9B96 was taken along to the evaluation to be compared to STM32F215VET6 that is based on the same processor core. Freescale offered MC56F8257 which differs from 8441 model by having slightly newer core, lower clock frequency, and smaller memory size. According to Freescale experts, 8441 model has a few additional instruction compared to 8257 model.

In addition, one processor with a separate *floating-point unit (FPU)* was provided to the test from Renesas Electronics after the other contestants had been already tested. Renesas SH7216 group device was tested equally among other devices and added to the results. Device model is actually R5F7216FAU, yet the group name will be used in future. SH7216 is an excellent enhancement to this evaluation and depicted the effectiveness of the FPU in calculations.

# 6 PROCESSOR COMPARISON

In this chapter, processors under investigation and testing results are presented in detail. Primary targets listed below are now disregarded and discussion is only about the ones that was provided to author's test bench. Test bench contents are described superficially without explaining the entire test code.

## 6.1 Texas Instruments TMS320F28035

TMS320F28035 (in future shortly F28035) is a 60 MHz processor from Texas Instruments with an additional *Contol Law Accelerator (CLA)* core besides the main C28x core. F28035 is designated as a 32-bit Real-time MCU by the manufacturer and is a member of the C2000™ family. Inside the C2000™ family, the F28035 belongs to the Piccolo™ series. C28x-based processors has been used in many previous ABB product families, such as ACS355, and ACS550. Testing was began with F28035 due to vast amount of technical support inside ABB and previous testing data as a comparison. Functional overview is presented in Figure 6.1. **[13]**

### 6.1.1 Architectural Overview

**Central Processing Unit**

As mentioned in previous section, F28035 has two separated cores: C28x and CLA. According to the manufacturer, C28x is a very efficient C/C++ engine at math tasks and system control tasks. C28x has a 32-bit fixed-point architecture and the 32 x 32-bit *multiplying accumulator (MAC)* 64-bit processing capabilities enable the controller to handle higher numerical resolution problems efficiently. The device has an 8-level-deep protected pipeline with pipelined memory accesses. **[13]**

The CLA is an independent single-precision 32-bit floating-point unit that extends the capabilities of the C28x CPU by adding parallel processing. It has own bus structure, fetch mechnanism, and pipeline. Eight individual CLA tasks, or routines, can be specified. Each task is started by software or a peripheral such as the A/D converter, a pulse width modulator, or a specific CPU timer. CLA was not used due to challenges in code modification. **[13]**

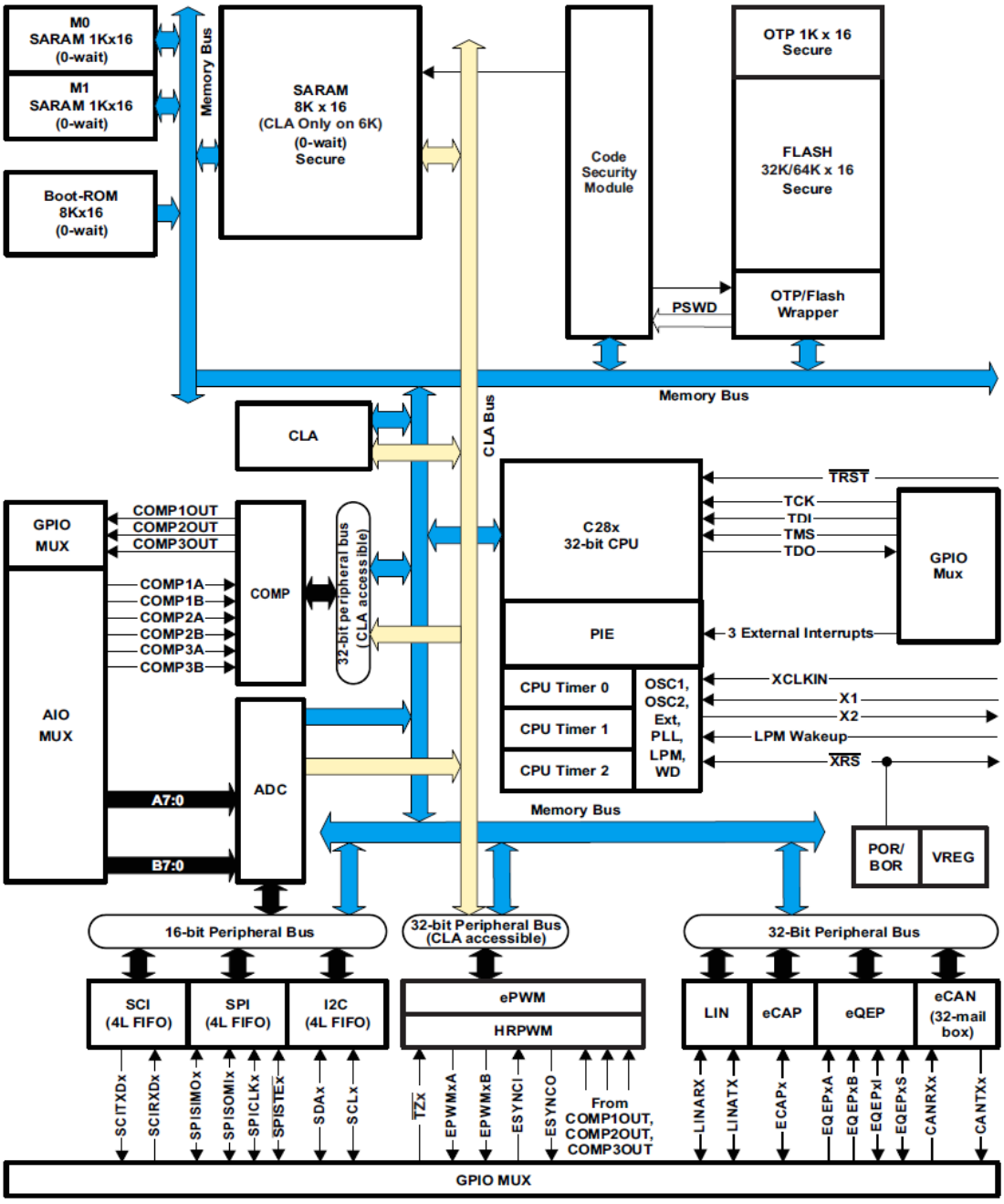



Figure 6.1 Block diagram of F28035. **[13]**

## Memory

The F28035 has the Harvard memory bus architecture containing a program read bus, data read bus, and data write bus. The program read bus consists of 22 address lines and 32 data lines. The data read and write busses consist of 32 address lines and 32 data lines each. The Harvard architecture enables the C28x to fetch an instruction, read a data value and write a data value in a single cycle.

The device contains 64K x 16 of embedded flash memory, segregated into eight 8K x 16 sectors. Also a single 1K x 16 one time programmable (OTP) memory is included. Appendix X shows a memory map of the F28035. **[13]**

### 6.1.2 Peripherals

#### Timers

The F28035 has three identical 32-bit timers. CPU-Timers 0, 1, and 2 have presettable periods and 16-bit clock prescaling. The timers have a 32-bit count-down register, which generates an interrupt when the counter reaches zero. When the timer reaches zero, it is automatically reloaded with a 32-bit value. The counter is decremented at the CPU clock speed divided by the prescale value setting. Either 12 or 14 Enhanced Pulse Width Modulation outputs are included depending on the package type. The smaller package type is 64-pin PAG TQFP while the larger is 80-pin PN LQFP package. **[13]**

#### Communication

Several communication interfaces are included in F28035 such as two SPI modules. SPI module takes four pins and of the processor and if not used, all four pins can be used as *General Purpose Input/Output (GPIO)*. Another alternative is one Serial Communication Interface (SCI). The SCI module supports digital communications between the CPU and other asynchronous peripherals that use the standard *non-return-to-zero (NRZ)* format. *Local Interconnect Network (LIN)* included in F28035 can be used also as another SCI module. In addition, *enhanced Controller Area Network (eCAN)* and *Inter-Integrated Circuit (I2C)* are included in the chip. **[13]**

#### A/D Converter

The F28035 is equipped with a 12-bit ADC. The ADC takes 4.6 million samples in a second while conversion time is 216.67 ns. Number of channels is 14 in the smaller package while 16 in the larger package. The ADC block contains two sample-and-hold units for simultaneous sampling. **[13]**

### 6.1.3 Feature Summary

The most essential features in summarized form are presented in Table 6.1.

Table 6.1 TMS320F28035 main features. Variations depend on the package. **[13]**

| Feature | TMS320F28035 |
|---|---|
| **Clock Frequency:** | 60 MHz |
| **RAM:** | 20 kB |
| **Flash Memory:** | 128 kB |
| **A/D Converter:** | 12-bit x 14-16 channels |
| **SCI:** | 1 (LIN excluded) |
| **SPI:** | 2 |
| **CAN:** | 1 |
| **PWM:** | 12-14 channels |
| **Package** | 64 TQFP, 80 LQFP |

## 6.2 Texas Instruments LM3S9B96

The first two letters 'LM' stands for Luminary Micro which was the first company that manufactured microcontrollers based on the ARM Cortex-M3 core. Luminary Micro was acquired by Texas Instruments in May of 2009. ARM Cortex-M3 devices comprises the commercially known Stellaris® product family. High-level block diagram is presented in Figure 6.2. **[14]**

### 6.2.1 Architectural Overview

#### Central Processing Unit

LM3S9B96 has the ARM® Cortex™-M3 processor core (shortly ARM Cortex-M3) with a clock speed of 80 Mhz. According to the manufacturer, the ARM Cortex-M3 device provides the core for a high-performance, low-cost platform that meets the need of minimal memory impelemtation, reduced pin count, and a low power consumption, while delivering outstanding computation performance and exceptional system response to interrupts. **[15]**

As an ARM Cortex-M3 processor, LM3S9B96 core contains 16 registers of which 13 are general-purpose. The processor supports 32-bit, 16-bit, and 8-bit data types. Instructions are both 16-bit and 32-bit long. Pipeline has three stages and hardware division block is included. **[16]**

*Nested Vectored Interrupt Controller (NVIC)* is closely integrated with the processor core. According to Cortex™-M3 Technical Reference Manual, this faciliates low latency exception processing. **[16]**

#### Memory

LM3S9B96 has the Harvard memory to enable simultaneous instruction fetches with data load and store. Memory accesses are controlled by a separate Load Store Unit (LSU) and a 3-word entry Prefetch Unit (PFU). LSU decouples load and store operations from the ALU and PFU ensures that a stall cycle is only necessary for one particular type of instruction fetched. At system clock speeds of 50 MHz and below, the Flash memory is read in a single cycle. Above 50 MHz, a prefech buffer of the Flash memory controller is used automatically and the Flash operates at half of the system clock. **[15]**

The device provides 96 kB of single-cycle on-chip SRAM. In order to reduce the number of slow read-modify-write operations, ARM has introduced *bit-banding* technology in the Cortex-M3 processor. With this technology, certain regions in the memory map can use address aliases to access individual bits in a single operation. **[15]**

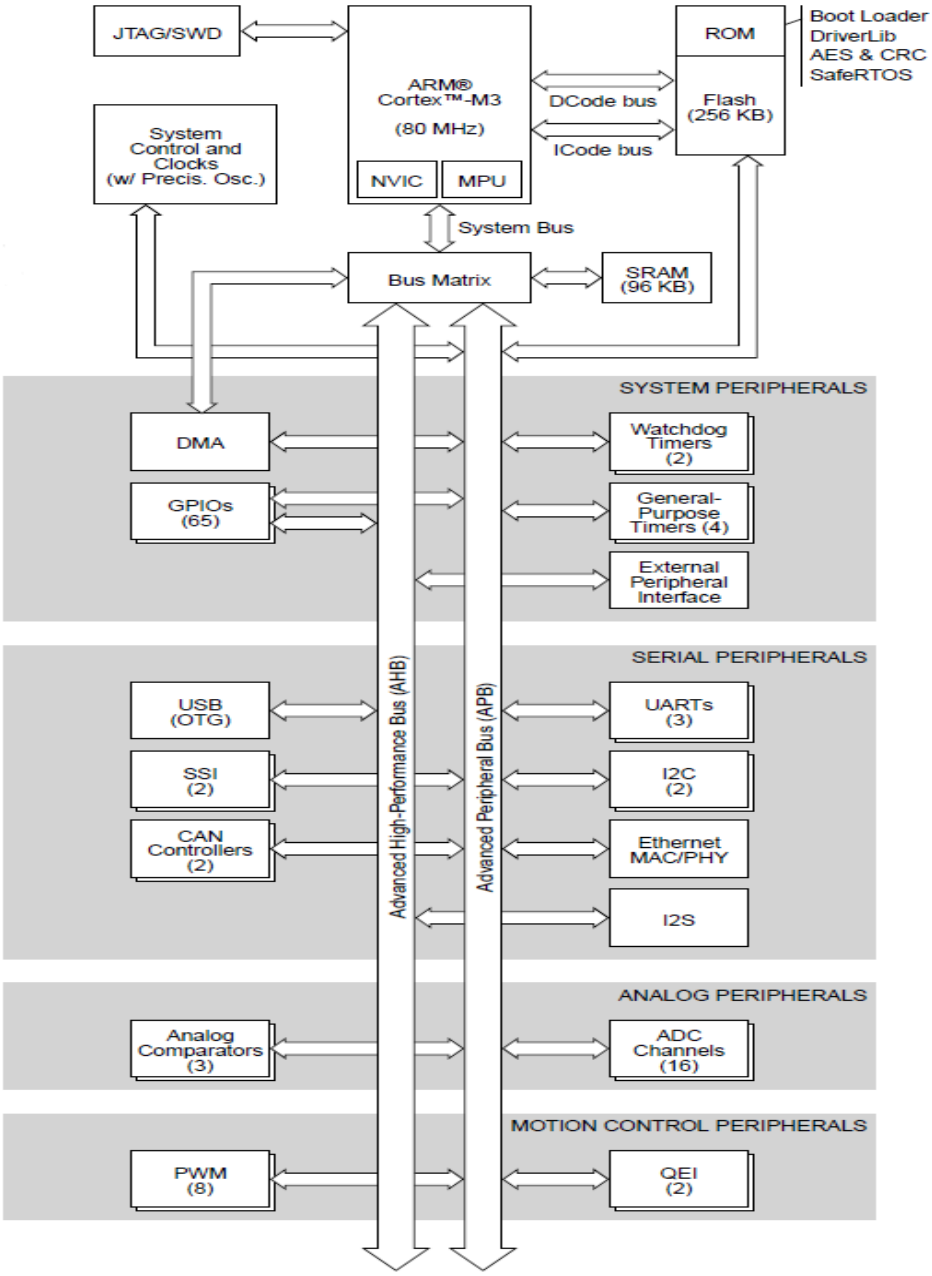



Figure 6.2 LM3S9B96 block diagram. **[15]**

## 6.2.2 Peripherals

### Timers

LM3S9B96 has four 32-bit timers and two Watchdog timers. The Stellaris® General-Purpose Timer Module (GPTM) contains four GPTM blocks (Timer 0, Timer 1, Timer 2, and Timer 3). Each GPTM block provided two 16-bit timers or counters (Timer A, Timer B) that can be configured to operate independently as timers or event counters, or configured to operate as one 32-bit timer or one 32-bit Real Time Clock). Timers can be used to trigger A/D conversions. In addition to GPTM, other timer resoureces include the System Timer (SysTick) and the PWM timer in the PWM module. PWM module contains eight PWM outputs. **[15]**

**Communication**

The most essential communication interfaces included in LM3S9B96 are three SCIs, two SPIs and two CAN interfaces. Also Ethernet, Inter-Integrated Circuit ($I^2C$), and USB interfaces are included. **[15]**

**A/D Conveter**

The Stellaris® ADC module features 10-bit conversion resolution and supports sixteen input channels. Input source, trigger events, interrupt generation, and sequencer priority are fully configurable Maximum sample rate is one million samples per second. **[15]**

### 6.2.3 Feature Summary

The most essential features in summarized form are presented in Table 6.2.

Table 6.2 LM3S9B96 main features. **[15]**

| Feature | LM3S9B96 |
|---|---|
| **Clock Frequency:** | 80 MHz |
| **RAM:** | 96 kB |
| **Flash Memory:** | 256 kB |
| **A/D Converter:** | 10-bit x 16 channels |
| **SCI:** | 3 |
| **SPI:** | 2 |
| **CAN:** | 2 |
| **PWM:** | 8 channels |
| **Package** | 100 LQFP, 80 LQFP |

## 6.3 STMicroelectronics STM320F207IG

STM32F207IG is the another ARM® Cortex™-M3 based product participating in this evaluation. Naturally, many properties are quite similar with Texas Instruments LM3S9B96 due to the same core architecture. While writing this thesis, STM32F207IG is not yet in production, hence device related issues are confidential, and therefore not discussed. Nevertheless, Cortex™-M3 Technical Reference Manual is public and may be considered an information source.

## 6.4 Freescale MC56F8257

The MC56F8257 is classified as digital signal controller by Freescale and appears often by simpler name: 56F8257. The product is relatively fresh and available technical data is quite scarce. 56F8257 is based on the 56800E core with 60 MHz operation frequency and according to the manufacturer, it combines DSP processing power and microcontroller functionality with a flexible set of peripherals on a single chip creating a cost-effective solution. **[17]**

### 6.4.1 Architectural Overview

**Central Processing Unit**

The 56800E core consists of three execution units operating in parallel, allowing as many as six operations per instruction cycle. The core includes three internal address buses and four internal data buses. Main functional elements of the core are single-cycle 16 x 16-bit MAC, four 36-bit accumulators and a 32-bit arithmetic and logic multi-bit shifter. DSP and controller functions are supported with 155 basic instructions. A simplified block diagram of the MC56F825x series is illustrated in Figure 6.3. **[17]**

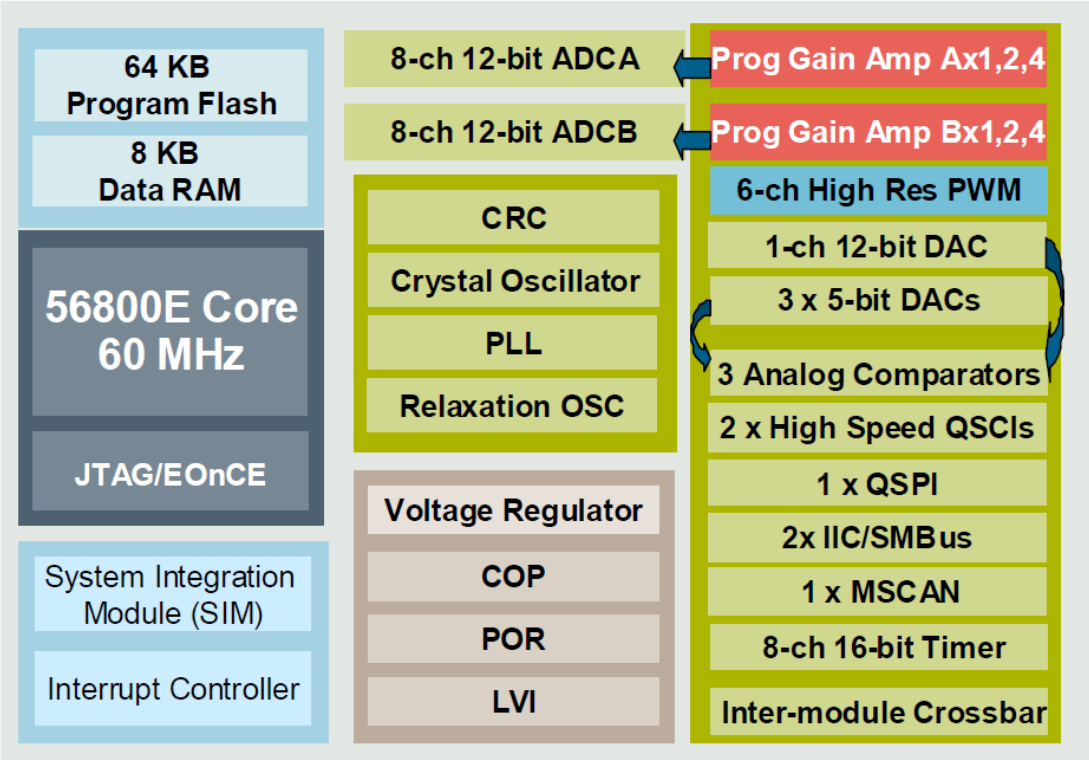



Figure 6.3 Simplified block diagram of 56F8257. **[17]**

**Memory**

Memory is implemented as dual Harvard architecture that permits three simultaneous accesses to program and data memory. The 56F8257 provides 64 kB of flash memory and 8 kB of RAM memory. Word length is 16 bits. Both flash and RAM supports 60 MHz program execution. Flash security and protection prevents unauthorized users from gaining access to the internal flash. As can be seen from Table 5.1, in which the processor requirements are presented, the 56F8257 does not fulfill these requirements. Nevertheless, directional information of the potential of the Freescale core may still be achieved. **[17]**

### 6.4.2 Peripherals

**Timers**

The 56F8257 has two four-channel 16-bit multi-purpose timer modules that are up to 120 Mhz operating clock. Each timer has capture and compare and quadrature decoder capability. Pulse width modulation is handled by Enhanced Flex Pulse

Width Modulator (eFlexPWM) module. The module has nine output channels with 16-bit resolution. **[17]**

**Communication**

The 56F8257 provides two queued serial communication interfaces (QSCI) modules. Main features of the modules are four bytes deep transmit and receive buffers, programmable 8- or 9-bit data format, and LIN slave functionality. In addition, one queued serial peripheral interface (QSPI) is included. Other communication interfaces are two $I^2C$ ports and one fully CAN 2.0 A/B compliant interface. **[17]**

**A/D Converter**

Two independent 12-bit ADCs are included in the device. Both ADCs has 8-channel exernal inputs. Maximum ADC clock frequency is 10 MHz with a total conversion time of 1450 ns. ADC conversion can be synchronized by eFlexPWM and timer modules via so called internal crossbar module. **[17]**

### 6.4.3 Feature Summary

The most essential features of the 56F8257 are presented in Table 6.3.

Table 6.3 56F8257 main features. **[17]**

| Feature | TMS320F28035 |
|---|---|
| **Clock Frequency:** | 60 MHz |
| **RAM:** | 8 kB |
| **Flash Memory:** | 64 kB |
| **A/D Converter:** | 12-bit x 16 channels |
| **SCI:** | 2 |
| **SPI:** | 1 |
| **CAN:** | 1 |
| **PWM:** | 9 channels |
| **Package** | 64 LQFP |

## 6.5 Renesas SH7216

Renesas Electronics is quite obvious participant to this evaluation after all. According to the manufacturer, Renesas Electronics became the world's largest microcontroller supplier through the merger of NEC Electronics and Renesas Technology. SH7216 belongs to SuperH series in the manufacturer's road map and is the fastest device in this evaluation with a clock frequency of 200 Mhz. Block diagram of SH7216 is illustrated in Figure 6.4. **[18]**

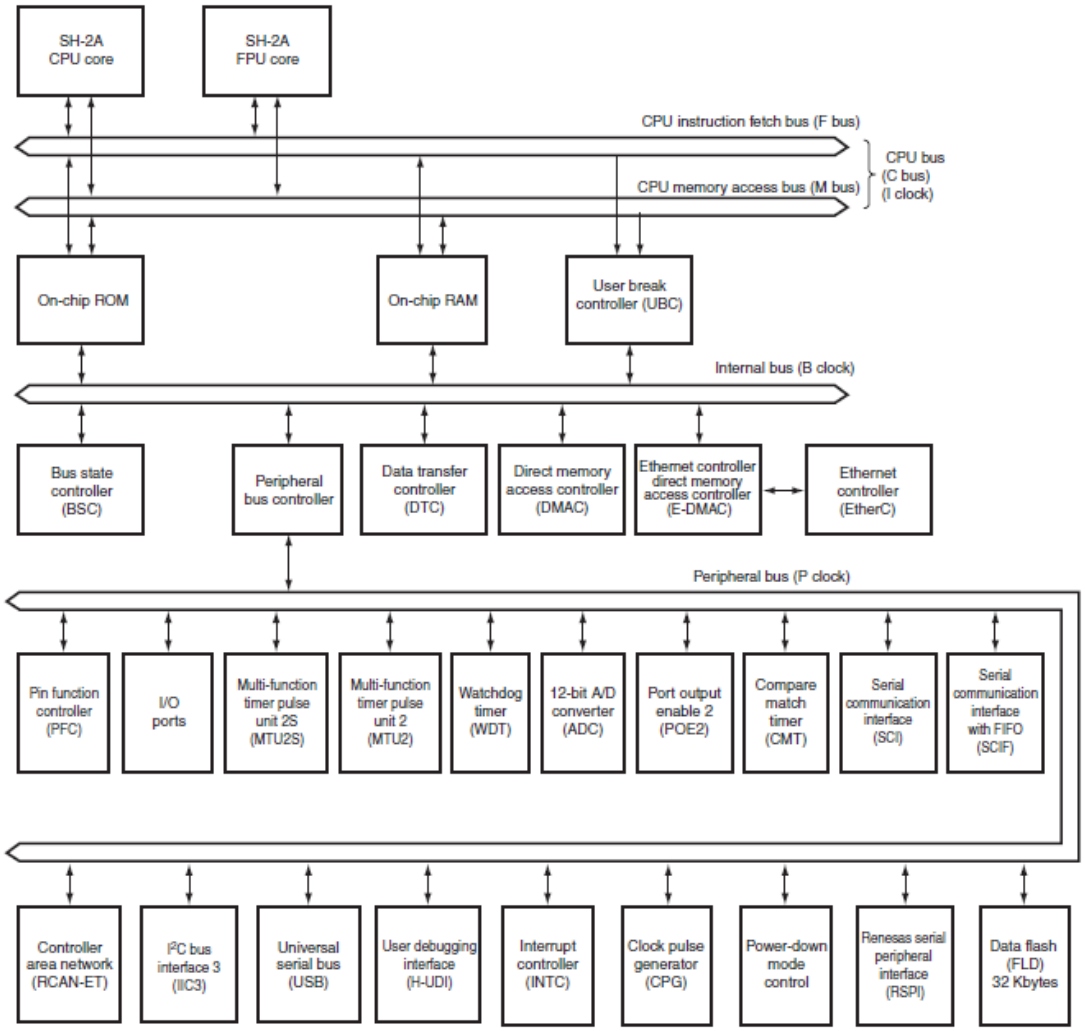



Figure 6.4 Block diagram of SH7216. **[18]**

## 6.5.1 Architectural Overview

### Central Processing Unit

SH7216 has a RISC-type instruction set and uses superscalar and Harvard architectures. Superscalar architecture allows the device to execute two instructions at one time. SH7216 has a 32-bit internal-bus architecture, sixteen 32-bit registers, four 32-bit control registers, and four 32-bit system registers. The instruction set consists of 16-bit basic instructions and 32-bit instructions for high performance. Pipeline has five stages. **[19]**

SH7216 is equipped with a double-precision (64-bits) FPU, which supports IEEE 754-compliant data types. The FPU has sixteen 32-bit floating-point registers and two 32-bit system registers. Instruction set supports directly multiplying and accumulation, division, and square root. Also instructions that load constant 0 or 1 is included in the instruction set. **[19]**

### Memory

SH7216 has a 4 Gbyte address space and incorporates 1 MB flash memory, 32 kB data flash memory, and 128 kB RAM. Flash memory is designed for the storage of instruction code while data flash is for storing data. RAM can be accessed in 8, 16,

or 32 bits and can be effectively used as program area or stack for access at high speed. **[19]**

## 6.5.2 Peripherals

### Timers

SH7216 has two multi-function timer pulse units (MTU2 and MTU2S). MTU2 comprises six 16-bit timer channels and a selection of eight counter input clocks for each channel except channel 5, which has only four input clocks. MTU2S comprises three 16-bit timer channels. MTU2 can operate at 50 MHz max. while MTU2S can operate at 100 Mhz max. for complementary PWM output functions. For other peripherals, the maximum operation speed of MTU2S is 50 Mhz. **[19]**

### Communication

SH7216 is quite abundant with its connectivity features. The device comprises five SCI channels, $I^2C$ bus interface 3, a modified Controller Area Network (RCAN-ET) module, USB function module, Ethernet Controller, and *Renesas Serial Peripheral Interface (RSPI)*. The SCI can handle both asynchronous and clock synchronous serial communication. The RSPI is capable of full-duplex synchronous, high-speed serial communications with multiple processors and peripheral devices.

### A/D Converter

Successive approximation type A/D converter has eight input channels and a 12-bit resolution. Maximum clock frequency is 50 MHz and with with 50 conversion states, the minimum conversion time is 1,0 µs per channel. Conversion can be started by software, with timers, or with an external trigger.

## 6.5.3 Feature Summary

The most essential features of the SH7216 are presented in Table 6.4.

Table 6.4 SH7216 main features. **[19]**

| Feature | TMS320F28035 |
|---|---|
| **Clock Frequency:** | 200 MHz |
| **RAM:** | 128 kB |
| **Flash Memory:** | 1 MB |
| **A/D Converter:** | 12-bit x 8 channels |
| **SCI:** | 5 |
| **SPI:** | 1 (RSPI) |
| **CAN:** | 1 (RCAN-ET) |
| **PWM:** | 12 channels |
| **Package** | 176 LQFP |

# 7 TEST RESULTS

Testing was performed with a code that has been used in processor benchmarking before in ABB. The code consists of various arithmetic operations and logic operations. By these means, a trustworthy evaluation can be achieved. Results can be also compared to the tests performed in the past.

## 7.1 Benchmarking

### 7.1.1 Code Contents

As mentioned in previous section, the target was to evaluate the performance of the processors in arithmetic and logic functions. The task of the arithmetic function was to evaluate performance by calculating the basic arithmetic operations, in other words addition, subtraction, and multiplication. In addition, testing was performed again separately both with division and with trigonometric functions. In logic function, various logic operations were executed. Several variable statements were used to confirm that the code is executed in correct order and the results are reliable. Both arithmetic and logic function were executed once in the first stage, and as a loop of ten cycles in the second stage. Nevertheless, time values in result tables correspond only one complete execution of a function. Looping of ten cycles is used only to confirm reliability of the result in the first stage, and therefore is not documented.

Two different arithmetics were used to evaluate arithmetic performance: floating-point arithmetic and IQ-arithmetic. IQ-arithmetic is based on high precision mathematical function library collection by Texas Instruments. While evaluating TMS320F28035, a special TMS320C28x IQmath Library was used. Otherwise, calculations were made using substitutive IQ definitions created by an ABB expert. The importance of IQ-arithmetic evaluation is to compare the performance to the control concept used in current product families. The aim is to test devices using both arithmetic type. Unfortunately, division and trigonometric functions are defined only to F28035 while using IQ-arithmetic.

Due to various processor architectures, varying main functions had to written in order to execute the benchmark code. Main functions including definition statements are presented in Appendix A – C. Depending on the Integrated Development Environment (IDE), varying methods was used to obtain the results. SH 7216 was tested by observing directly the profile clock tool provided by the IDE, while with other processors, calculation was done additionally by a processor timer.

### 7.1.2 Memory Utilization

Another target was to examine the operation speed performed from various storage alternatives. By choosing certain settings in the benchmark code or in the development environment, execution code was stored either in RAM memory or in Flash memory. External memory devices was not used in this evaluation.

### 7.1.3 Optimization

In order to achieve as fast results as possible, code optimization was used during compilation. Optimization tries to minimize either the size of the compiled code or

the time needed to execute some process. In this evaluation, minimum execution time is the main target and level 3 optimization offered by Codegen Tools, RealView MDK-ARM and Metrowerks M56800E C Compiler. F28035 was evaluated with an additional Optimize for Speed (-mf) value set to level 5.

## 7.2 Texas Instruments TMS320F28035 Evaluation

### 7.2.1 Development Environment

The TMSF28035 evaluation was made with Code Composer Studio v3.3. As code generation tools were used Codegen Tools v5.2.1. Evalution board used was F28035 Piccolo Experiment's Kit (TMDXDOCK28035) equipped with F28035 Piccolo controlCARD (TMDXCNCD28035). Evaluation hardware is illustrated in Figure 7.1.

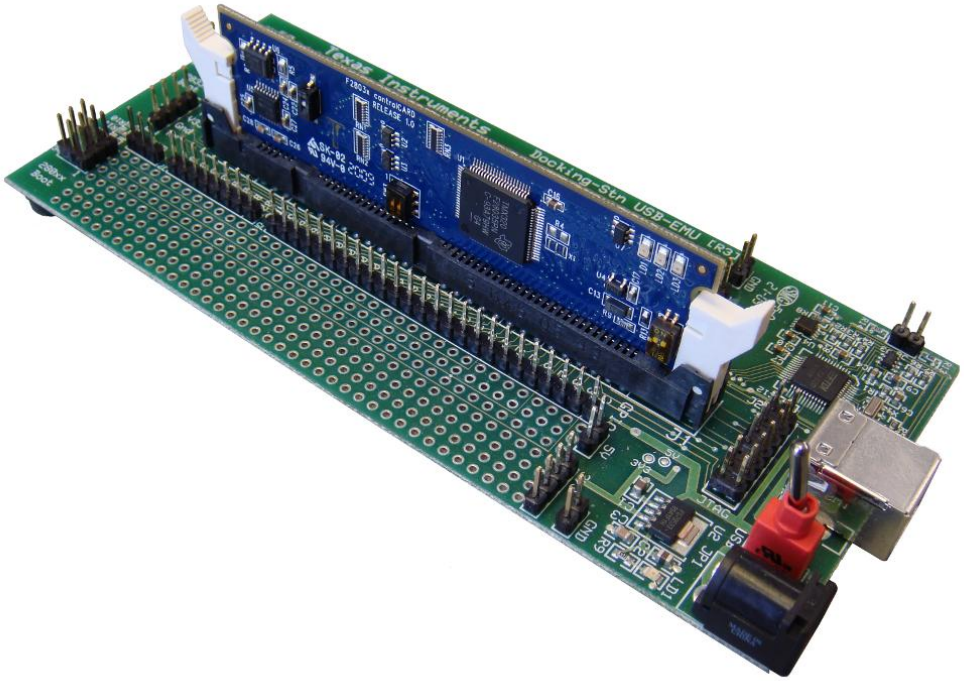



Figure 7.1 F28035 Piccolo Experiment's Kit equipped with F28035 Piccolo controlCARD.

Although manufacturer promises remarkable benefit by using the independent CLA co-processor in floatin-point arithmetic, it was not used due to need for relatively complex assembly language coding. The F28035 has no alternative floating point unit, thus floating-point arithmetic operations are calculated purely by the main CPU. Program code were stored in, and run from both RAM memory and flash memory. Target memory was assigned in a linker command file included in the development project. Texas Instruments offers their own library in order to calculate effectively IQ-arithmetic with a C28x-based device.

### 7.2.2 Results

As the following devices, the F28035 was tested with floating-point operations and with IQ-operations. In addition, results from logic function are listed. All results are documented using both storage alternatives. Math libraries indicate that

trigonometric functions were included in the arithmetic function. All results are listed in Table 7.1.

The test itself was performed using both processor timer and Profile Clock tool. Profile Clock tool provided thoroughly slightly faster results, in other words, lower clock cycle expedinture. Due to difficulties using Profile Clock tool while running the code from flash memory, results using the processor timer are considered final. The results in Table 7.1 are converted to microseconds by dividing the number of cycles by the clock frequency of the device. Looping resulted ten times longer execution times, thus the test may be considered successful. Code content of the main function including necessary definitions is presented in Appendix A. Implementation of the loops is excluded due to its trivial nature. Naturally additional code is needed to set up the processor. Code content of the arithmetic and logic operations executed during the test is not presented.

Table 7.1 F28035 execution times with and without division and math libraries are illustrated in tables (a) - (c).

| **No division nor math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 27.37 | 5.07 | 17.32 |
| **Flash** | 33.75 | 5.30 | 19.33 |

**(a)**

| **Division included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 31.30 | 6.20 | 17.40 |
| **Flash** | 38.55 | 6.57 | 19.35 |

**(b)**

| **Math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 141.42 | 8.88 | 17.40 |
| **Flash** | 158.72 | 9.43 | 19.32 |

**(c)**

Similar IQ-results was obtained also by TI experts. Cooperation was performed due to extremely weak performance in Flash execution during the first testing round. In other words, run optimization and device configuration may be considered similar compared to the settings made by the professionals.

## 7.3 Texas Instruments LMS3S9B96 Evaluation

### 7.3.1 Development Environment

LM3S9B96 is an ARM Cortex-M3 based device, thus Keil µVision4 was used as development environment. Version of µVision was V4.03 and used toolchain was RealView MDK-ARM Version: 4.10. Evaluation board was DK-LM3S9B96. In addition, ULINK*pro* Debug and Trace Unit was used in debugging. All hardware is shown in Figure 7.2.


Figure 7.2 LM3S9B96 evaluation hardware setup.

### 7.3.2 Results

Testing was accomplished based on a running cycle counter and a profile clock. The profile clock provided directly microseconds as a result. Results were confirmed by initializing a core timer and storing the timer value to a variable just before and immediately after the execution of a function under investigation. Substitutive IQ definitions do not include division nor trigonometric function, thus in Table 7.2 (b) – (c), IQ-math column is empty.

Table 7.2 LM3S9B96 execution times with and without division and math libraries are illustrated in tables (a) – (c).

| **No division or math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 24.63 | 4.74 | 17.80 |
| **Flash** | 16.51 | 4.05 | 11.69 |

**(a)**

| Division included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 25.78 | - | 17.80 |
| **Flash** | 17.41 | - | 11.64 |

**(b)**

| Math libraries included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 40.76 | - | 17.80 |
| **Flash** | 32.04 | - | 11.64 |

**(c)**

Results obtained with LM3S9B96 was not verified by TI experts due to somewhat expected and consistent values in Table 7.2. Execution times from Flash was achieved by default settings in µVision 4. RAM execution offered alternative manners how memory allocation was performed.

Main function and assiociated definitions are presented in Appendix B.

## 7.4 STMicroelectronics STM320F207IG Evaluation

### 7.4.1 Development Environment

The same Keil µVision4 was used as development environment while debugging STM32F207IG. Development board is not publicly available, thus no picture is shown. Besides the development board, all debugging software, equipment, and manners are identical to LM3S9B96.

### 7.4.2 Results

Tools and test manners were identical to LM3S9B96 while evaluating STM32F207IG. Test results are presented in Table 7.3.

Table 7.3 STM320F207IG execution times with and without division and math libraries are illustrated in tables (a) – (c).

| No division or math libraries included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 19.03 | 3.11 | 11.54 |
| **Flash** | 19.83 | 3.13 | 12.16 |

**(a)**

| Division included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 20.07 | - | 11.54 |
| **Flash** | 21.49 | - | 12.16 |

**(b)**

| Math libraries included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 36.09 | - | 11.54 |
| **Flash** | 37.98 | - | 12.16 |

**(c)**

## 7.5 Freescale MC56F8257 Evaluation

### 7.5.1 Development Environment

The development environment offerd by Freescale was CodeWarrior IDE v5.9.0. Freescale offers also FreeMASTER application as a separated software to allow control of an embedded application from a graphical environment running on a PC. According to Freescale, the application was initially created for developers of hard real-time motor control applications. Nevertheless, FreeMASTER was not used during this evaluation. The CodeWarrior included also a tool called Processor Expert. Briefly, Processor Expert offered a component library of the device inside the project. For example, registers of a peripheral could be controlled in a graphical environment with Processor Expert tool.

The compiler used was Metrowerks M56800E C Compiler and linker M56800E Linker. Freescale provided their TWR-56F8257 evaluation board that has an in-circuit debugging option. Due to complex usability, an external debugger CodeWarrior USB TAP was used. Complete hardware environment is presented in Figure 7.3.

Similar code than with F28035 was created while testing MC56F8257. A timer was initialized and reset before a function call and the value of the counter register was stored right after the function.


Figure 7.3 Freescale TWR-56F8257 with CodeWarrior USB TAP emulator.

### 7.5.2 Results

For some unknown reason, program execution did not take place from the RAM memory. Also, compilation of IQ-math code caused serious problems. Consequently, the results in Table 7.4 include only Float Math and Logic columns.

Table 7.4 MC56F8257 execution times in floating-point arithmetic and logic operations.

| **No division or math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | - | - | - |
| **Flash** | 150,72 | - | 38,22 |

**(a)**

| **Division included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | - | - | - |
| **Flash** | 156,22 | - | 38,22 |

**(b)**

| Math libraries included | | | (µs) |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | - | - | - |
| **Flash** | 413,75 | - | 38,22 |

**(c)**

Main function and assiociated definitions are presented in Appendix C. Timer initialization is implemented as a separate function before the main function.

## 7.6 Renesas SH7216 Evaluation

### 7.6.1 Development Environment

Renesas Electronics provides its own devepment environment High-performance Embedded Workshop (HEW). Alternatively IAR Embedded Workbench can be used. HEW v.4.07.00.007 was used in this evaluation. The compiler was C/C++ Compiler Package V.9.01. for SuperH Family. Evaluation board was Renesas Starter Kit2+ and debugging device was E10A. Hardware is shown in Figure 7.4.

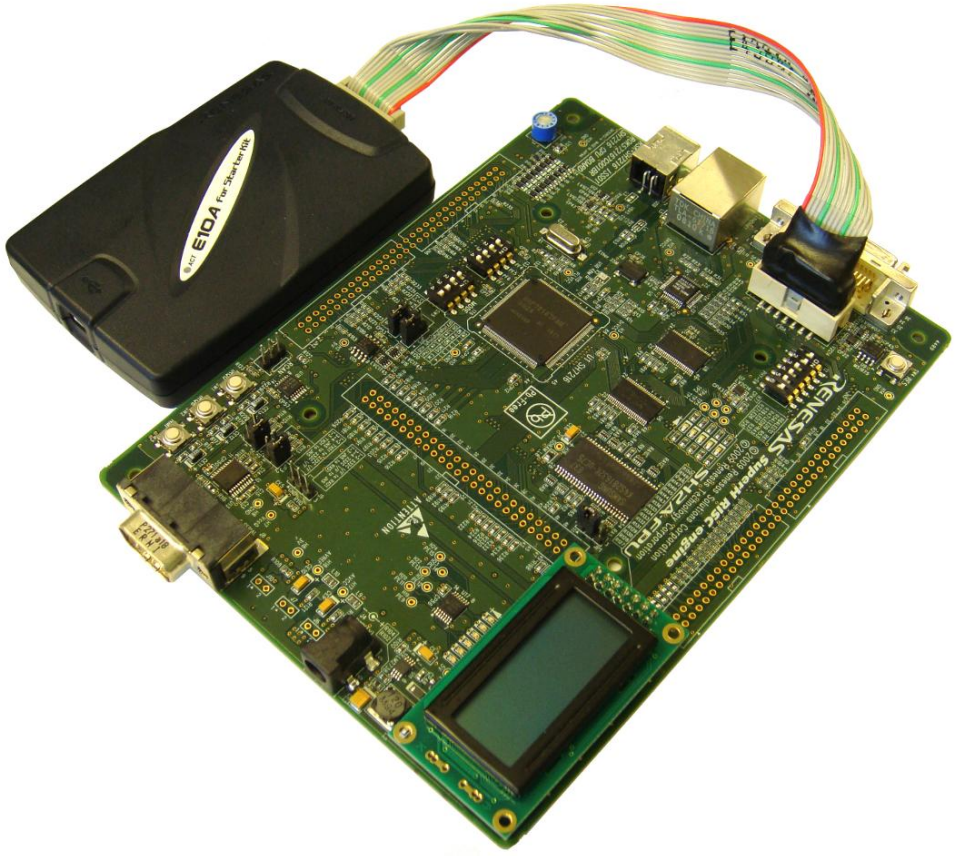



Figure 7.4 Renesas Starter Kit2+ with E10A-USB emulator.

### 7.6.2 Results

Timers could not be used to evaluate SH7216 due to relatively weak time resolution. Instead, Performance Analysis tool of the IDE was used to calculate spent clock cycles. Results are presented in Table 7.5.

Table 7.5 SH7216 execution times with and without division and math libraries are illustrated in tables (a) – (c).

| **No division or math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 1.39 | 4.89 | 6.99 |
| **Flash** | 1.24 | 4.83 | 7.02 |

**(a)**

| **Division included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 1.47 | - | 6.99 |
| **Flash** | 1.32 | - | 7.01 |

**(b)**

| **Math libraries included** | | | **(µs)** |
|---|---|---|---|
| | Float Math | IQ-Math | Logic |
| **RAM** | 2.59 | - | 6.99 |
| **Flash** | 2.76 | - | 7.15 |

**(c)**

Peripheral timer was not used in evaluation, and therefore, presentation of a main function does not provide any added value to this study. Consequently, code content is not presented as an appendix.

## 7.7 Comparison

In this section, the results obtained above will be compared and analyzed in more detailed manner. Judging realibility issues and discussing the reasons that led to these results will be also a significant part of this section. Results will be illustrated in a graphical form to maximize readability.

### 7.7.1 Comparison by Time

In order to define, which processor has the highest performance, measurement results are given in microseconds. Time comparison is an effective means to combine the core competency with the benchmark code and the nominal speed of the processor.

First, all RAM results will be shown maintaining the same colors as the Tables 7.1 – 7.5 have as an indicator of used code contents. As a reminder, blue indicates that no division nor trigonometric functions are included. Red color indicates division and green color indicates trigonometric functions. Logic will be illustrated with a purple bar.

Result charts begin with floating-point arithmetic comparison run from RAM in Figure 7.5. Floating-point comparison is followed by IQ-math chart in Figure 7.6. Logic performance is included in both charts to provide some reference. Reader should keep in mind, that time spent in logic operations differs minimally between arithmetic content. The same pattern will continue throughout this section and the following section.

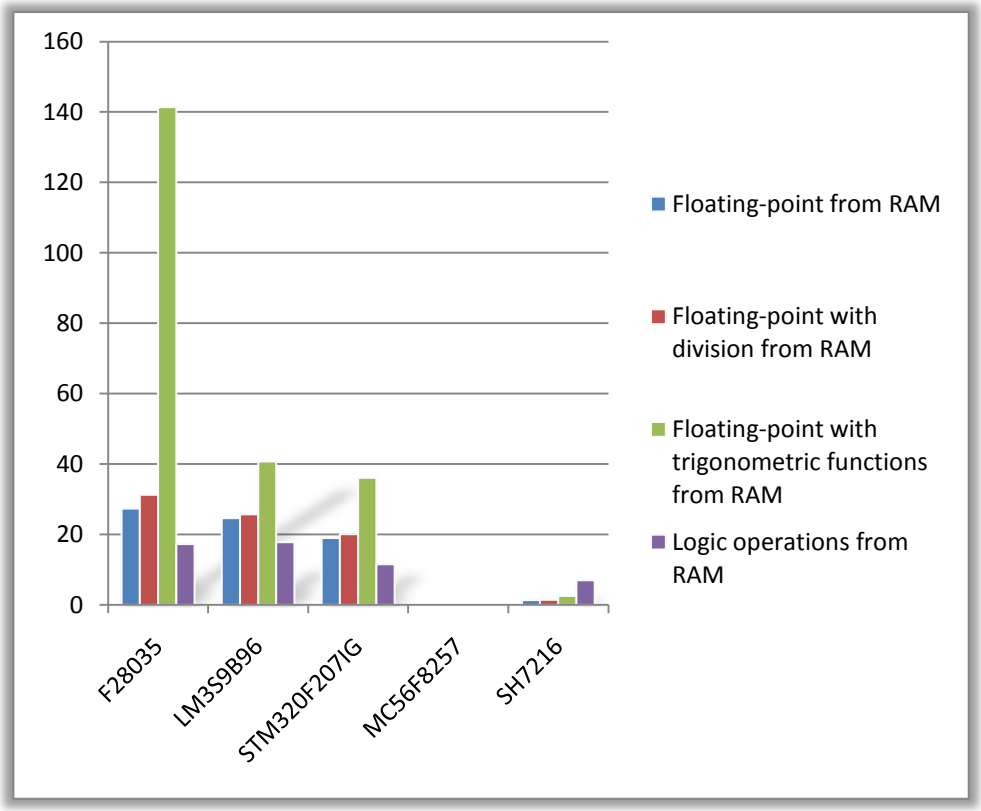



Figure 7.5 Execution times of floating-point arithmetic and logic using RAM. No results for MC56F8257 due to unsuccessful execution from RAM.

As can be seen form Figure 7.5, MC56F8257 does not have any result. This is due to difficulties to execute instructions from RAM. With MC56F8257, the framework of the code, including timer initialization etc., seemed to run correctly, however after including arithmetic code, execution began to behave inconsistently. Logic code could not even be loaded to RAM due to scarce memory size. According to the Freescale experts, they have a tendency to guide customers to run code from Flash.

As can be seen, scaling makes almost the result bars of SH7216 disappear. Nevertheless, the results can be seen in Table 7.5 in numeric form.

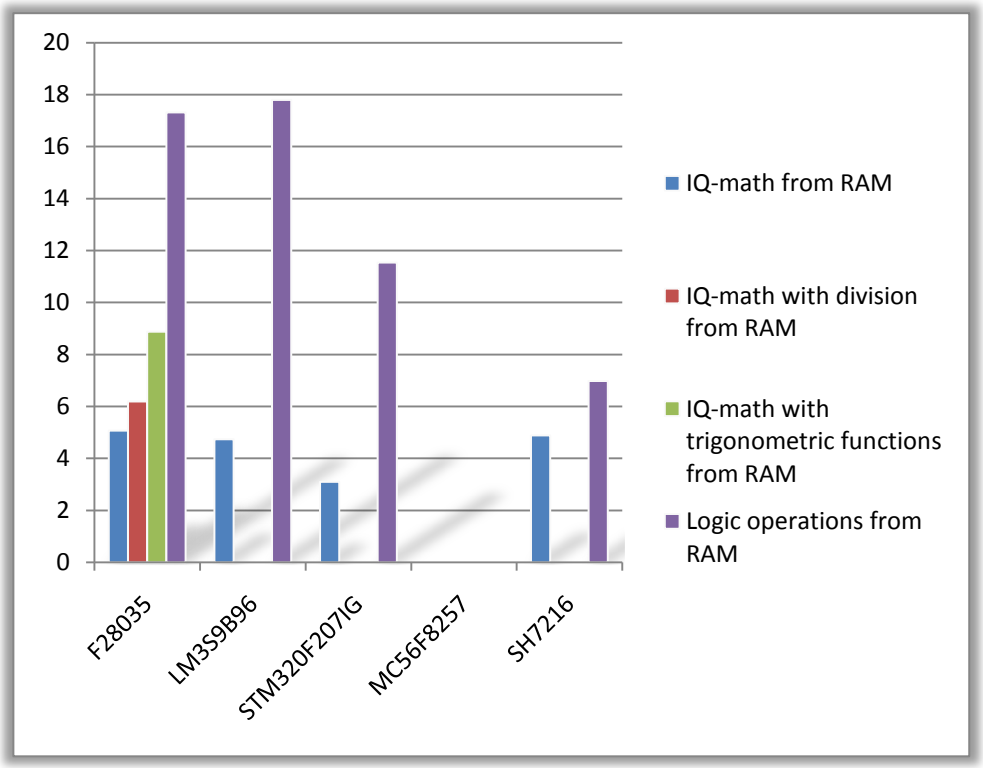



Figure 7.6 Execution times of IQ-math and logic using RAM. Again, no results for MC56F8257.

Flash execution provided interesting changes to the results. Corresponding charts are presented in Figure 7.7 and in Figure 7.8.

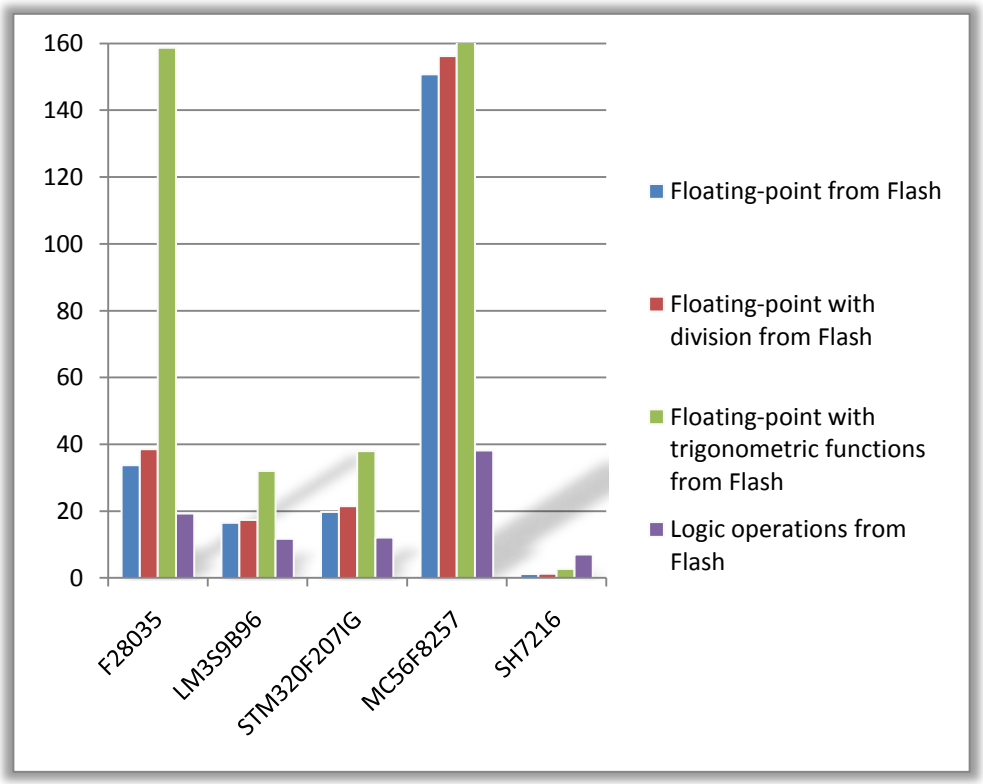



Figure 7.7 Execution times of floating-point arithmetic and logic using Flash. MC56F8257 results are partially cut off due to scaling.

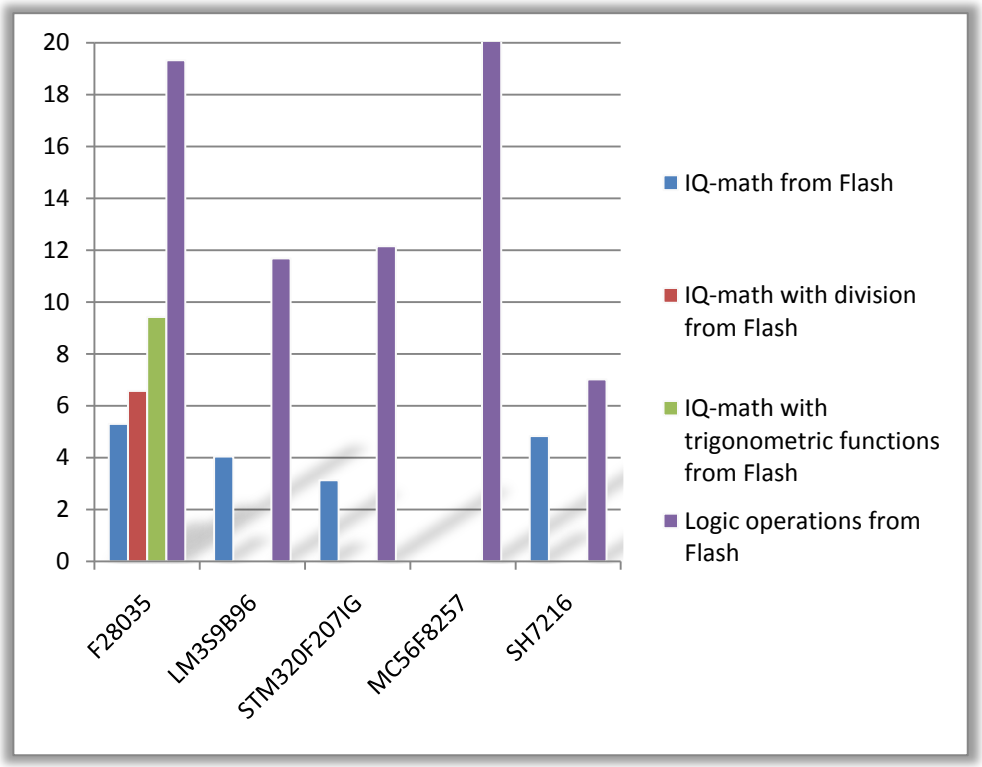



Figure 7.8 Execution times of IQ-math and logic using Flash. MC56F8257 results are partially cut off due to scaling.

Figures 7.7 and 7.8 indicate that while F28035 and STM320F207IG became somewhat slower, LM3S9B96 accomplished the tasks much faster. SH7216 achieved relatively similar results from both memories.

**Discussion**

From Figure 6.9 can be easily seen that the execution time increases consistently when arithmetic burden is increased. This trend continues throughout all the measurements.

While running floating-point arithmetic, SH7216 was clearly the fastest device due to built-in floating point unit. Otherwise the execution speed of floating-point arithmetic is quite consistent with the clock frequency. One curious observation is extremely slow performance of F28035 while calculating trigonometric functions with floating-point arithmetic.

As one might guess, F28035 would be clearly the fastest device in IQ-arithmetic due to the built-in IQ-architecture. Despite this fact, the ARM devices reach lower time values than F28035. SH7216 is relatively quite poor in IQ-arithmetic. One reason for this fact is the means how multiplication is performed using IQ-arithmetic. While the final result is achieved with ARM devices by shifting the product by 24 bits, SH7216 has to labor more to gain the desired result. Another important observation is lack of results including division and trigonometric functions with non-IQ processors. The code used in this study defines division and trigonometric functions only for F28035.

Logic operations was executed most quickly from RAM by SH7216. This is mostly due to the highest clock frequency.

When attention is turned into Flash execution, the results reached another track in some extent. While F28035 and STM320F207IG got slightly slower, LM3S9B96 turned out to be even faster. According to TI experts, this phenomenon roots to the fact that while executing from RAM, only one bus is used. The standard 12 cycle interrupt latency is increased when data access interrupts code fetch. LM3S9B96 runs with 1 wait state from Flash at speed of 80 MHz, only separate data bus and code bus can be utilized.

Another reason was discovered just before the testing was ended. Contrary to supposition, LM3S9B96 was configured to 50 MHz instead of 80 MHz during the whole testing due to a Flash error. According to the manufacturer, this problem exists only in evaluation boards with LM3S9B96. Anyhow, this leads to the fact that the Flash memory was always operating with zero wait-state. As mentioned in section 6.2.1, if the clock frequency was over 50 MHz, the Flash would operate at half of the clock frequency and this would affect on the clock cycles with real 80 MHz clock. Reader must remember now, that the microseconds are calculated by scaling the spent clock cycles using erroneously 80 MHz clock speed. In other words, the results of LM3S9B96 are not reliable from Flash.

On the other hand, at speeds over 50 MHz, the prefetch buffer is turned on allowing some branches to be executed with zero wait-states. Still, if the processor would have operated 80 MHz, more clock cycles and more time would have been spent. Results using RAM on the other hand are reliable, and therefore new calculations using 50 MHz clock speed are not done.

MC56F8257 obtained generally quite poor results. The difference from other products is so obvious, that the possibility of unsuccessful processor configuration must be considered. In figure 7.7, these results are partly presented. In order to maintain maximal readability, the vertical axis has the same scaling as in Figure 7.5. This will cause the results of MC56F8257 to be cut of. Numerical data is presented in Table 7.4.

Figure 7.8 is indicating that F28035 is losing its relative advantage in IQ-arithmetic over other contestants when the code is executed from Flash. MC56F8257 results are again cut off to maintain comparativeness to results in Figure 7.6.

### 7.7.2 Comparison by Cycle Count

To compare the efficiency of the core to accomplish the tested tasks, the following charts will illustrate the consumed clock cycles in arithmetic and logic functions. While counting only clock cycles, the effect of clock frequency will be disregarded and core suitability will have all importance. Order of the figures will be exactly the same as in section 7.7.1. Figures 7.9 and 7.10 present the cycle count using RAM.

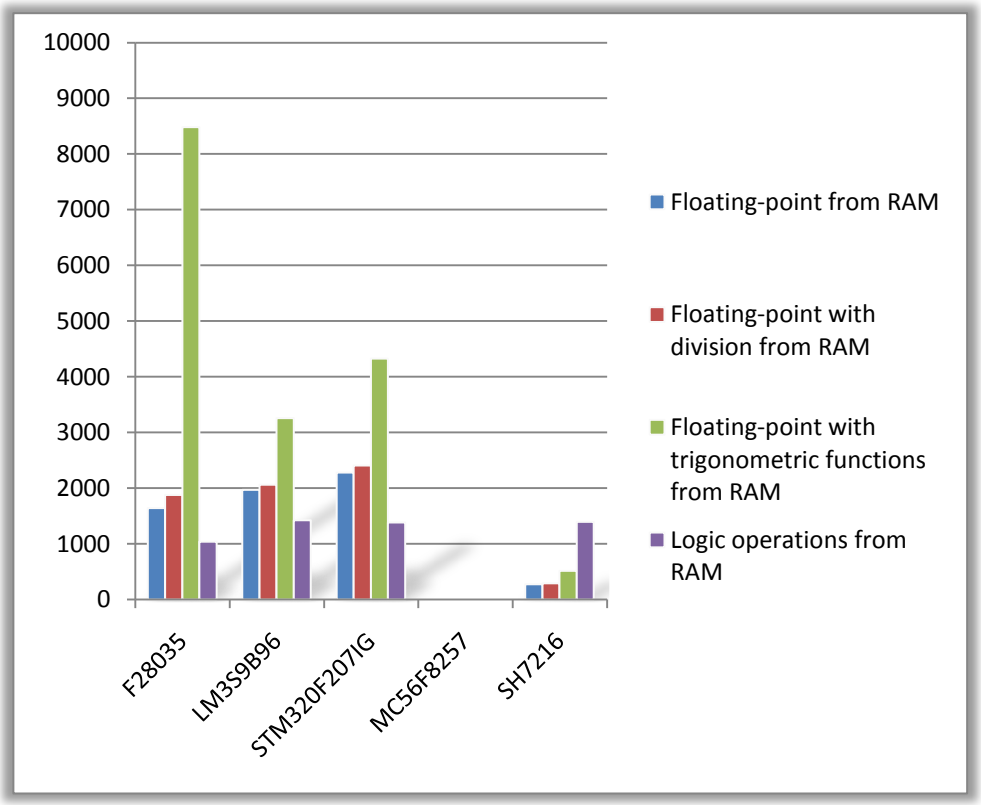



Figure 7.9 Execution cycles of floating-point arithmetic using RAM.

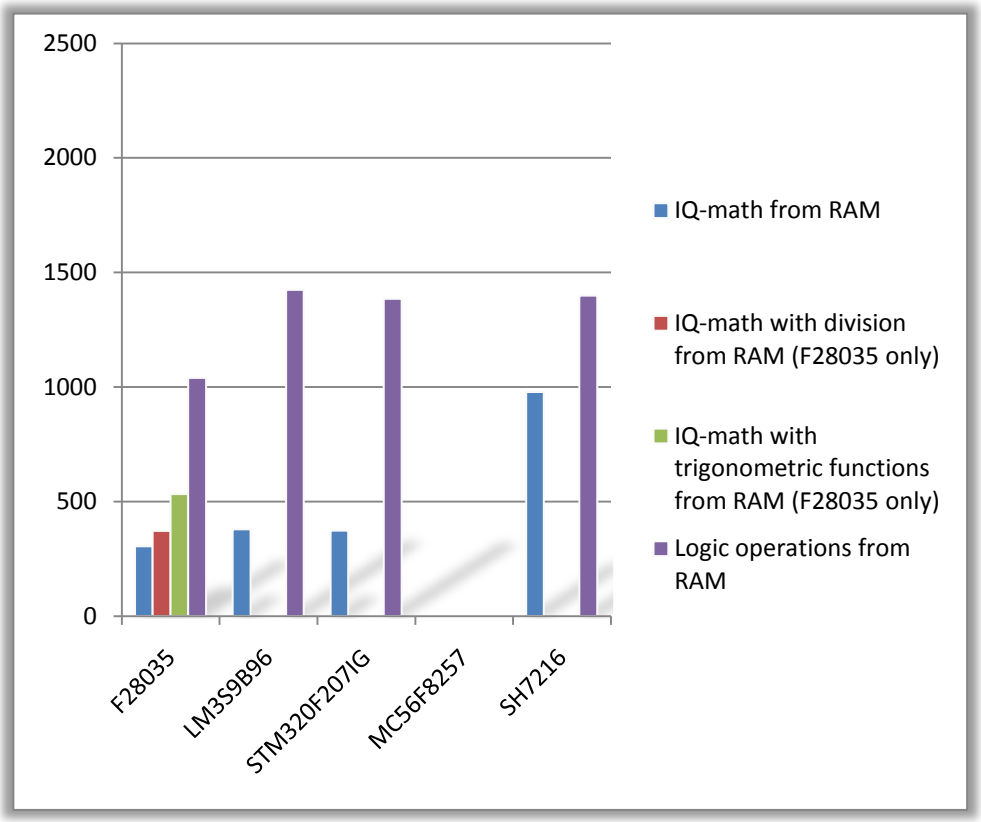



Figure 7.10 Execution cycles of IQ-arithmetic using RAM.

Figures 7.11 and 7.12 present the same charts when the code is executed from Flash memory.

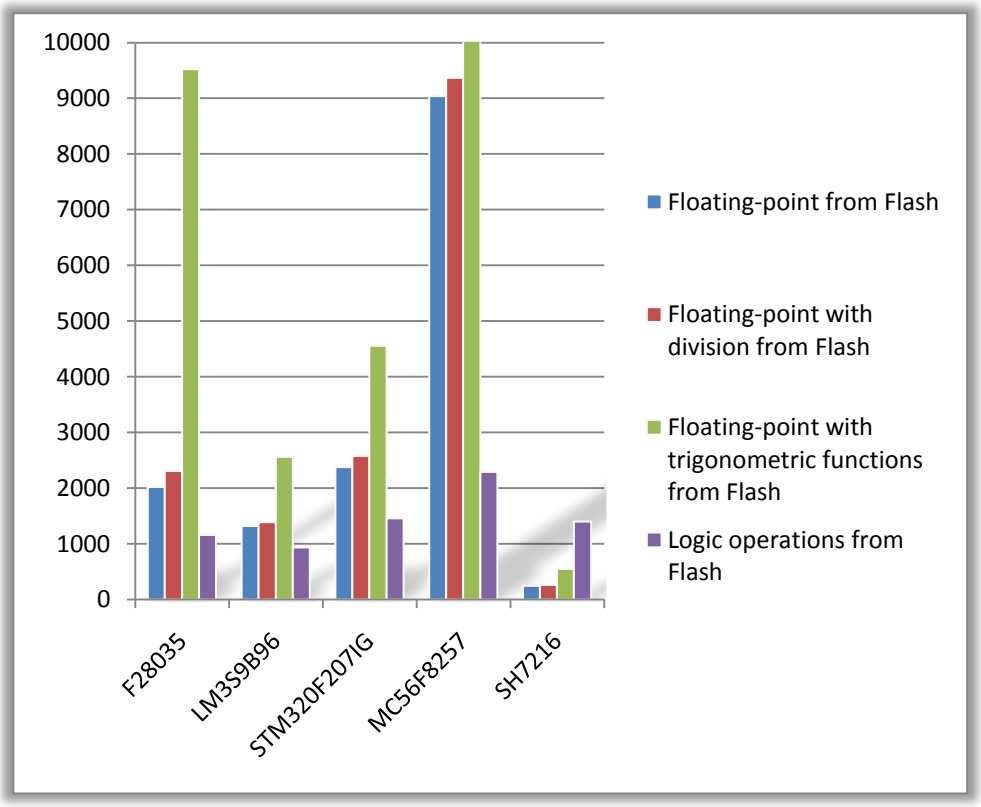



Figure 7.11 Cycle count of floating-point arithmetic using Flash. MC56F8257 results are partially cut off due to scaling.

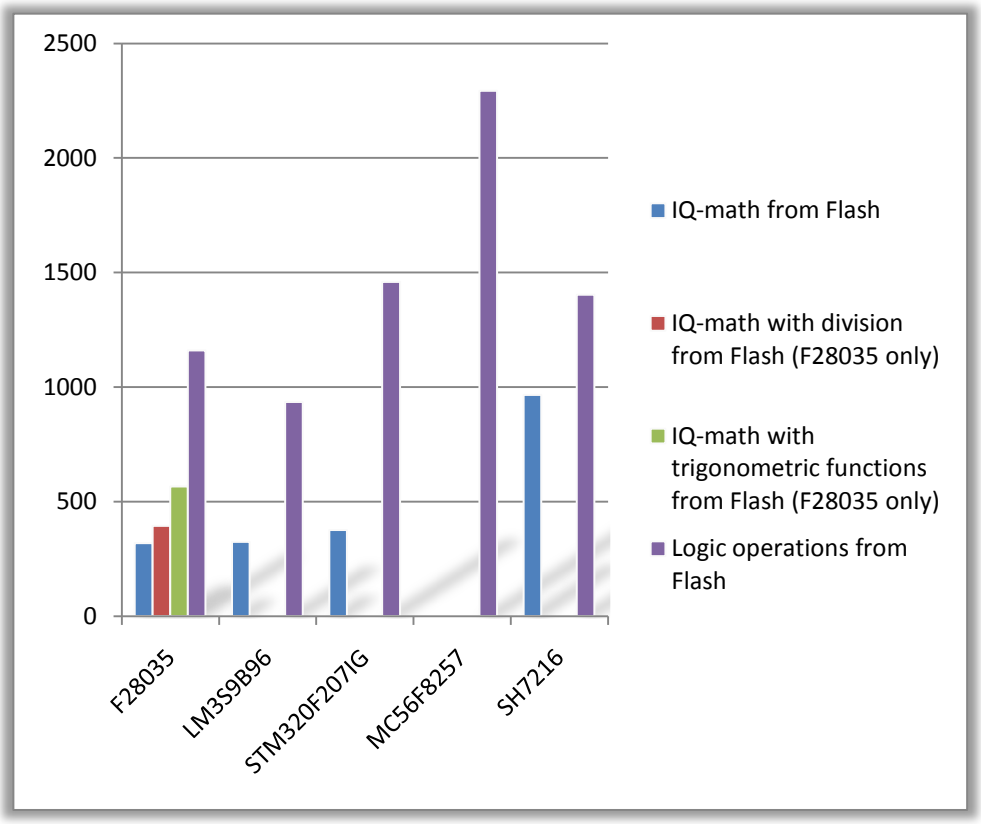



Figure 7.12 Cycle count of IQ-arithmetic using Flash.

The results presented in Figures 7.9 – 7.12 may be scaled to any clock frequency as long as operation speed of Flash memory is taken into account. This evaluation does not include two identical processors that basically would differ only by clock frequency. However, it is justifiable to assume that clock cycle count remains somewhat equal if other features do not change drastically.

**Discussion**

Figures 7.9 and 7.10 present the effectiveness of F28035 executing code from RAM. Naturally, SH7216 is much more efficient in floating-point arithmetic. In logic operations, F28035 is surprisingly competent. On the other hand, according to ABB experts, logic code is always executed from Flash. In cycle count comparison, built-in IQ-architecture shows its strength leaving both ARM devices behind.

Comparing cycles while executing code from Flash, the efficiency of LM3S9B96 is conspicuous. The reason for this is the clock speed of 50 MHz as mentioned in last discussions. This phenomenon is especially prominent in floating-point arithmetic and in logic operations. Using 80 MHz instead, the author has a reason to assume, that the results would be closer to STM320F207IG. Microseconds are calculated based on these cycle counts and therefore, the error mirrors to Figures 7.7 and 7.8 as well.

From Figure 7.11 can be seen again that MC56F8257 is substantially slower compared to its competitors. The result with trigonometric functions is not fully presented due to the scaling.

With IQ-arithmetic, F28035 was slightly more efficient than LM3S9B96. The result of LM3S9B96 is still very likely to increase with a clock speed of 80 MHz. Figure 7.12 presents quite clearly how SH7216 is significantly less efficient with IQ-arithmetic. This is due to more complex implementation of IQ-multiplication.

### 7.7.3 Comparison by Code Size

One valuable comparison addition to speed is code size that the compilers result. In this section, four different compilation results are achieved because the ARM based devices are built with the same compiler RealView MDK-ARM v4.10. Results are interpreted from the map files compilers create and they are illustrated in two separate manner. First, only the precise code included in the executed functions are taken into account. Secondly, all the code included in corresponding modules is presented. The difference in these manners is not massive, however, the logic module consists of a few additional functions besides the principal function `logic_bm()`. Additional entity of code besides the principal function `arithm_bm()` also appears to the arithmetic module when IQ-arithmetic is chosen with ARM processors and with SH7216.

Taking only the principle functions into account is quite a straightforward task. However, including the fragmental pieces from the map file to form the corresponding module is notably more laborious task. Reader should notice that calculations including the code module do not include every single piece of code they are consisted of, nevertheless the error is negligible. This is due to the fact, that the code is divided in sections and only the most significant section is included in calculation. One inconsistency is also the word length. In F28035 and MC56F8257

case, the word length is 16 bits while others have 8 bit word length. Results are shown in Figure 7.13 and 7.14.

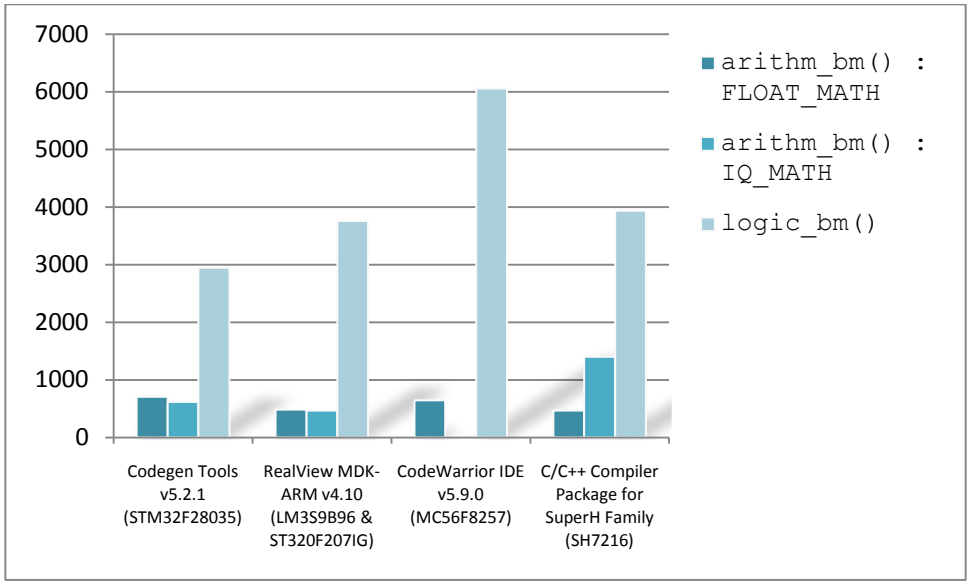



Figure 7.13 Code size comparison taking only the principle functions into account. Units in the figures are bytes.

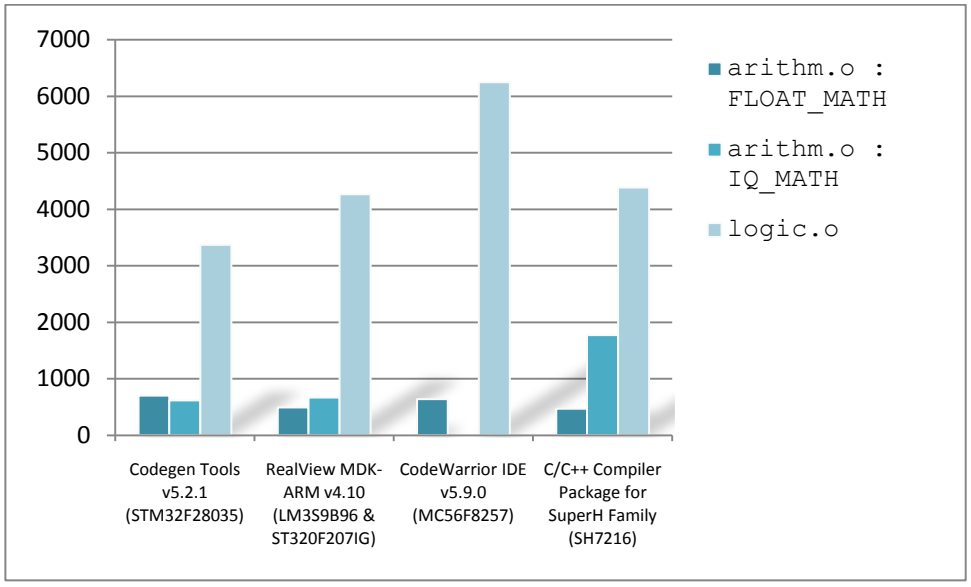



Figure 7.14 Code size results including the smaller fragments of the code in the modules. Units in the figures are bytes.

# 8 CONCLUSION

The meaning of the results in this study is to survey possible processor alternatives in a decent price range. Succesful study will lead to fully functional component with financial benefits. Unfortunately, this study will not finally determine the target processor, only guides to the best solution in following surveys. Results in this study yet give precious new information about the processors on the market today. Exact prices are not known by either party at this stage and therefore, only price ranges are used as reference.

Based on the results achieved in Chapter 7, may be stated that if IQ-arithmetic is utilized, some TI device similar to F28035 is a fine choice. As mentioned before, F28035 can not be chosen due to lack of memory. Unfortunately, similar device with larger memory is not available at present. On the other hand, also ARM devices handled IQ-arithmetic pretty good as far as IQ calculations were performed. However, this issue would require deeper study including division and trigonometric functions to state a final recommendation. Otherwise ARM devices offer a great package of connectivity and memory. Naturally this means also placing to a higher price range.

On the other hand, operating with floating-point arithmetic would be less laborious to programmers in the long run. From this point of view, SH7216 would be an excellent choice. Without any precise offer, it is quite rational to assume, that SH7216 belongs to a higher price range compared to other processors in this study. In other words, efforts after this study might concern mosty finding a suitable FPU device. For example, only RX and SuperH families from Renesas Electronics present a large collection of processors with a FPU. In addition, an upcoming ARM Cortex™-M4 is an attractive choice with a FPU. Cortex-M4 device was not tested, because the first samples are available not before 2011.

One fundamental conclusion after the testing is consideration to use floating-point arithmetic with a FPU processor. Previously, corresponding product families has not been based on a FPU device, thus the results of these study might be epochal.

# BIBLIOGRAPHY

[1] ABB Group. The ABB Group Annual Report 2009. 2009. 154 p. [WWW-document]. <http://www02.abb.com/global/seitp/seitp255.nsf/bf177942f19f4a98c1257148003b7a0a/4adfda5dea9d9091c12576e10034d9ec/$FILE/ABB+Group+annual+report+2009.pdf>.

[2] Niiranen, J. Sähkömoottorikäytön digitaalinen ohjaus. Helsinki, Finland: Valopaino, 2000. 381 p. ISBN 951-672-300-4.

[3] Luomi, J & Niiranen, J & Niemenmaa, A. Sähkömekaniikka ja sähkökäytöt Part 1. Espoo, Finland: Helsinki University of Technology, 2006. 166 p.

[4] Luomi, J & Niiranen, J & Niemenmaa, A. Sähkömekaniikka ja sähkökäytöt Part 2. Espoo, Finland: Helsinki University of Technology, 2006. 146 p.

[5] Young, H. D. & Freedman, R. A. University Physics 11th ed. San Francisco, USA: Addison Wesley, 2004. 1714 p. ISBN 0-8053-8684-X

[6] Bose, B. K. Modern Power Electronics and AC Drives. Upper Saddle River, NJ, USA: Prentice Hall, Inc, 2002. 711 p. ISBN 0-13-016743-6.

[7] Harnefors, L. Conrtol of Variable-Speed Drives. Västerås, Sweden: Department of Electronics, Mälardalen University, 2003. 234 p.

[8] ABB Oy, Technical guide book. 2008 [WWW-document]. <http://www05.abb.com/global/scot/scot201.nsf/veritydisplay/5108024cea1cb6f7c125748e004681c5/$File/TechnicalGuideBook_1_9_EN_revD.pdf>.

[9] Bellis, M. Intel 4004 – The World's First Single Chip Microprocessor. [WWW-document]. <http://inventors.about.com/od/mstartinventions/a/microprocessor.htm>. (Referenced 7.9.2010).

[10] Vahid, F. & Givargis, T. Embedded System Design: A Unified Hardware/Software Introduction. New York, NY, USA: John Wiley & Sons, 2002. 352p. ISBN 978-0-471-38678-0.

[11] Lapsley, P., Bier, J., Shoham, A. & Lee, E. A., DSP Processor Fundamentals: Architectures and Features. New York, NY, USA: Institute of Electrical and Electronics Engineering, Inc, 1997. 210 p. ISBN 0-7803-3405-1.

[12] Bose, B. K. Power Electronics and Variable Frequency Drives. New York, NY, USA: Institute of Electrical and Electronics Engineers, Inc, 1997. 640 p. ISBN 0-7803-1084-5.

[13] Texas Instuments, Piccolo Microcontrollers. 2009. 143 p. [WWW-document]. <http://focus.ti.com/lit/ds/sprs584d/sprs584d.pdf>.

[14] Luminary Micro. Luminary Micro - Stellaris® - the industry's first Cortex-M3 MCUs. [WWW-document]. <http://www.luminarymicro.com/>. (Referenced 7.9.2010).

[15] Texas Instruments, Stellaris® LM3S9B96 Microcontroller Data Sheet. 2010. 1282 p. [WWW-document]. <http://focus.ti.com/lit/ds/spms182g/spms182g.pdf>

[16] ARM, Cortex™-M3 Technical Refence Manual. ARM Limited, 2006. 384 p. [WWW-document]. <http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf>.

[17] Freescale Semiconductors. MC56F825x/MC56F824x Product Brief. 2010. 14 p. [WWW-dokumentti]. <http://cache.freescale.com/files/microcontrollers/doc/prod_brief/MC56F825XPB.pdf?fpsp=1&WT_TYPE=Product%20Briefs&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation>.

[18] Renesas Electronics. Company Message. 2010. 5 p. [WWW-document]. <http://www.renesas.eu/comp/data/renesas_en.pdf>

[19] Renesas Electronics. SH7214 Group, SH7216 Group User's Manual: Hardware. 2010. 1920 p. [WWW-document]. <http://documentation.renesas.com/eng/products/mpumcu/rej09b0543_sh7216hm.pdf>

# APPENDIX A: CODE CONTENT WITH F28035

```
//---------------------------------------------------------------------
// DEFINITIONS
//---------------------------------------------------------------------
#define READ_TIMER           CpuTimer0Regs.TIM.all
#define START_COUNT()        start_count = READ_TIMER;
#define STOP_COUNT()         final_count = start_count - READ_TIMER;

//---------------------------------------------------------------------
// Data types
//---------------------------------------------------------------------
#define UWORD32              unsigned long // 32 bits unsigned

//---------------------------------------------------------------------
// VARIABLES
//---------------------------------------------------------------------
UWORD32 start_count = 123;
UWORD32 final_count = 456;
UWORD32 bm_count_arithm = 0;        // Cycle count of arithm_bm
UWORD32 bm_count_logic = 0;         // Cycle count of logic_bm

//---------------------------------------------------------------------
// EXTERNAL FUNCTIONS
//---------------------------------------------------------------------
extern void arithm_bm(void);
extern "C" {
extern void logic_bm(void);
}

//---------------------------------------------------------------------
// MAIN FUNCTIONS
//---------------------------------------------------------------------
int main(void)
{

   // Step 4. Initialize the Device Peripheral. This function can be
   // found in DSP2803x_CpuTimers.c
   InitCpuTimers();   // Only initialize the Cpu Timers

   // Configure CPU-Timer 0, 1, and 2 to interrupt every second:
   // 60MHz CPU Freq, 1 second Period (in uSeconds)
   ConfigCpuTimer(&CpuTimer0, 60, 1000000);
   ConfigCpuTimer(&CpuTimer1, 60, 1000000);
   ConfigCpuTimer(&CpuTimer2, 60, 1000000);

   // To ensure precise timing, use write-only instructions to write
   // to the entire register. Therefore, if any
   // of the configuration bits are changed in ConfigCpuTimer and
   // InitCpuTimers (in DSP2803x_CpuTimers.h), the
   // below settings must also be updated.
   CpuTimer0Regs.TCR.all = 0x4001;
   CpuTimer1Regs.TCR.all = 0x4001;
   CpuTimer2Regs.TCR.all = 0x4001;

   #if (MEM_TYPE == USE_FLASH)
   // Flash setup and other specified code to RAM
   // The  RamfuncsLoadStart, RamfuncsLoadEnd, and RamfuncsRunStart
   // symbols are created by the linker. Refer to the F28035.cmd
   // file.
   MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
```

```
  // Call Flash Initialization to setup flash waitstates
  // This function must reside in RAM so the memory copy from flash
  // to RAM must be before this function call.
  InitFlash();
  #endif

  // Step 6. IDLE loop. Just sit and loop forever (optional):
  while(1)
  {
    //-------------------------------------------------------------
    // Benchmark arithmetics: measurement when called once
    //-------------------------------------------------------------
    START_COUNT()
    arithm_bm();
    STOP_COUNT()
    bm_count_arithm = final_count;

    //-------------------------------------------------------------
    // Benchmark logic: measurement when called once
    //-------------------------------------------------------------
    START_COUNT()
    logic_bm();
    STOP_COUNT()
    bm_count_logic = final_count;
  }
}
```

# APPENDIX B: CODE CONTENT WITH ARM-DEVICES

```
//-------------------------------------------------------------------
// DEFINITIONS
//-------------------------------------------------------------------
//-------------------------------------------------------------------
// Cycle counting macros
//-------------------------------------------------------------------
#define    START_COUNT()         systick_before = *STCVR;
#define    STOP_COUNT()          systick_after = *STCVR;

//-------------------------------------------------------------------
// Data types
//-------------------------------------------------------------------
#define    UWORD32               unsigned long // 32 bits unsigned

//-------------------------------------------------------------------
// VARIABLES
//-------------------------------------------------------------------
UWORD32 bm_count_arithm;         // Cycle count of arithm_bm()
UWORD32 bm_count_logic;          // Cycle count of logic_bm
UWORD32 systick_before;          // Timer value before counting
UWORD32 systick_after;           // Timer value after counting

//-------------------------------------------------------------------
// EXTERNAL FUNCTIONS
//-------------------------------------------------------------------
extern void arithm_bm(void);
extern "C" {extern void logic_bm(void);}

//-------------------------------------------------------------------
// MAIN FUNCTION
//-------------------------------------------------------------------
int main(void)
{
  // Initialize SysTick core timer to run free
  int *STCSR = (int *)0xE000E010;
  int *STRVR = (int *)0xE000E014;
  int *STCVR = (int *)0xE000E018;

  *STRVR = 0xFFFFFF; // max count
  *STCVR = 0;        // force a re-load of the counter value register
  *STCSR = 5;        // enable FCLK count without interrupt

  //
  // Loop forever while the timer run.
  //
  while (1)
  {
    // CALCULATE bm_count_arithm
    START_COUNT()
    arithm_bm();
    STOP_COUNT()
    bm_count_arithm = systick_before - systick_after;

    // CALCULATE bm_count_logic
    START_COUNT()
    logic_bm();
    STOP_COUNT()
    bm_count_logic = systick_before - systick_after;
  }
}
```

# APPENDIX C: CODE CONTENT WITH MC56F8257

```
//-------------------------------------------------------------------
// DEFINITIONS
//-------------------------------------------------------------------
//-------------------------------------------------------------------
// Cycle counting macros
//-------------------------------------------------------------------
#define READ_TIMER()  timer_value = TMRA1_CNTR;
#define START_COUNT() setReg(TMRA1_CNTR, 0x00);
#define STOP_COUNT()  READ_TIMER(); final_count = timer_value;

//-------------------------------------------------------------------
// Data types
//-------------------------------------------------------------------
#define UWORD32 unsigned long   // 32 bits unsigned

//-------------------------------------------------------------------
// VARIABLES
//-------------------------------------------------------------------
UWORD32 final_count = 456;
UWORD32 timer_value = 666;
UWORD32 bm_count_arithm = 0;    // Cycle count of arithm_bm function
UWORD32 bm_count_logic = 0;     // Cycle count of logic_bm function

//-------------------------------------------------------------------
// EXTERNAL FUNCTIONS
//-------------------------------------------------------------------
extern void arithm_bm(void);
extern void logic_bm(void);

//-------------------------------------------------------------------
// FUNCTIONS
//-------------------------------------------------------------------
void Timer_Init(void)
{
// TMRA1_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,Co_INIT=0,OM=0
setReg(TMRA1_CTRL,0x1000);
// TMRA1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,
// INPUT=0,Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0
setReg(TMRA1_SCTRL,0x0000);
setReg(TMRA1_CNTR,0x0000);
setReg(TMRA1_CMPLD1,0x0000);
setRegBitGroup(TMRA1_CTRL,CM,1);           /* Run counter */
}

//-------------------------------------------------------------------
// MAIN FUNCTION
//-------------------------------------------------------------------
void main(void)
{
  // Initialize Timer
  Timer_Init();

  while (1)
  {
    //---------------------------------------------------------------
    // Benchmark arithmetics: measurement when called once
    //---------------------------------------------------------------
    START_COUNT()               // In reality resets the counter
    arithm_bm();
    STOP_COUNT()
    bm_count_arithm = final_count;
```

```
    //-------------------------------------------------------------
    // Benchmark logic: measurement when called once
    //-------------------------------------------------------------
    START_COUNT()
    logic_bm();
    STOP_COUNT()
    bm_count_logic = final_count;
  }
}
```