freescale™

Freescale Motor Control Software Library

JNK-IND-T0998

Luke Chun

Luke.chun@freescale.com

July 19, 2013

How to implement Motor Control Project?

Agenda

1. Select Target Motor

PMSM, IM, BLDC, SRM, DC…………….

2. Select Control Algorithm and Target Performance

Sensorless, Sensored, Target Control Response…….

3. Select Target MCU

Core, Core Speed, ADC performance, Vcc Level….

Implement MCU!!!

0. Read & Understand Datasheet, Reference Manual and Errata
1. Peripheral Setting : ADC, PWM, Timer……
2. Implementation Motor Control Algorithm : Vector Control, Encoder Interface…..
3. Tuning Motor performance : PI Gain Tuning..
4. Application Implementation : Washing Machine, Robot….
5. Application Tuning
6. TEST and Debugging
7. TEST and Debugging
8. TEST and Debugging

Implement MCU!!!

1. Peripheral Setting : ADC, PWM, Timer……

ProcessorExpert, *Quick Start(GCT)*

2. Implementation Motor Control Algorithm : Vector Control, Encoder Interface…..

FSL Library, ProcessorExpert

3. Tuning Motor performance : PI Gain Tuning..

freemaster

4. Application Implementation : Washing Machine, Robot….

FSL Library, ProcessorExpert

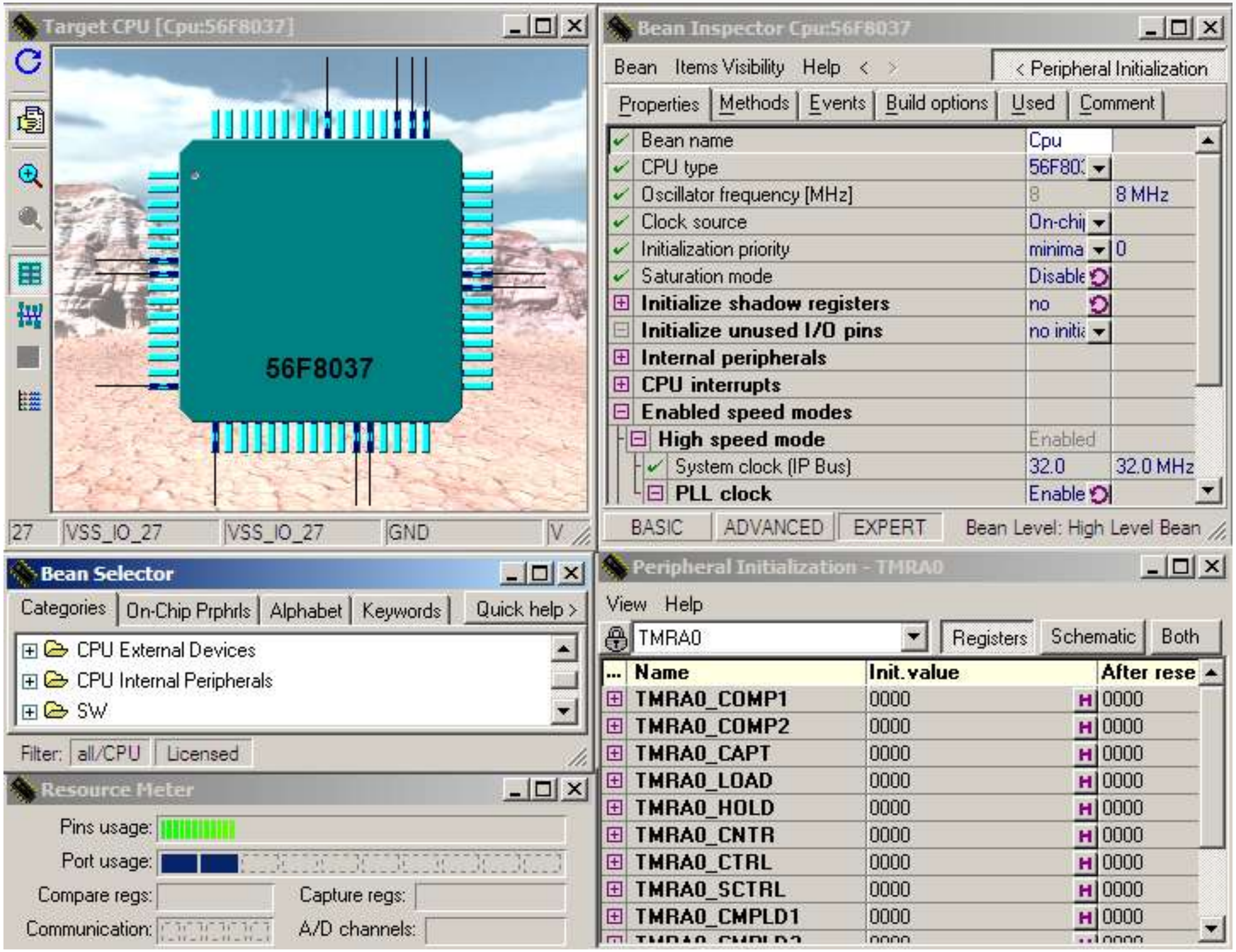
5. Application Tuning

freemaster

Implementation & Setting
Monitoring

ProcessorExpert

- GUI



freescale™

Application Specific Algorithm Libraries

Memory Manager
- Dynamic allocation

Feature Phone Library
- CallerID type 1&2, CallerID Parser, Generic Echo Cancellor

DSP Library
- FIR, IIR, FFT, Auto Correlation, Bit Reversal

Telephony Libraries
- AEC, AGC, Caller ID,
- CAS, CPT, CTG, DTMF
- G165, G168, G711
- G723, G726, G729

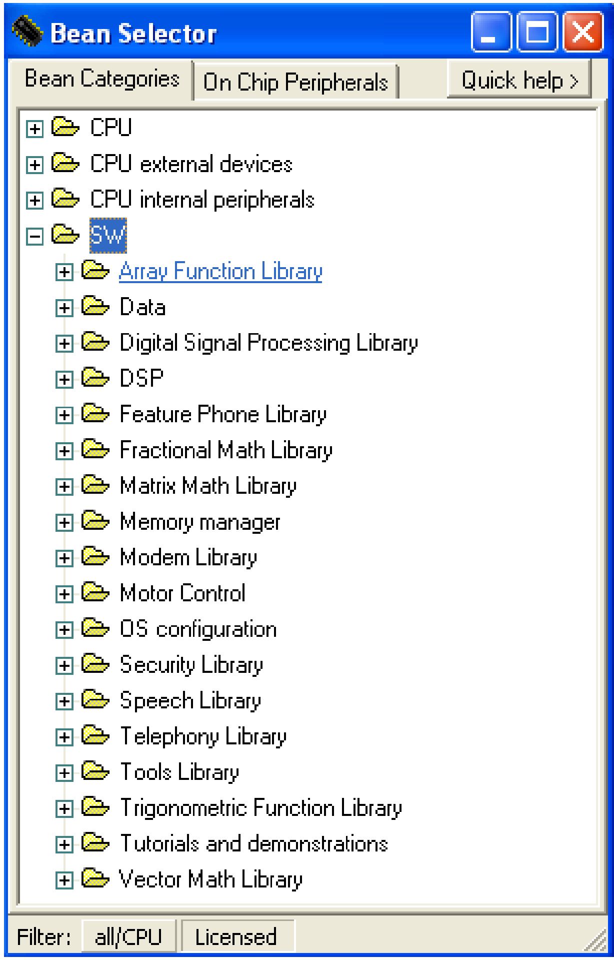
Modem Libraries
- V.8bis, V.21, V.22bis, V.42bis

Security Libraries
- RSA, DES, 3DES,

Motor Control
- BLDC, ACIM, SR motor specific algorithms
- General purpose algorithms

Math Libraries
- Matrix, Fractional, Vector
- Trigonometric

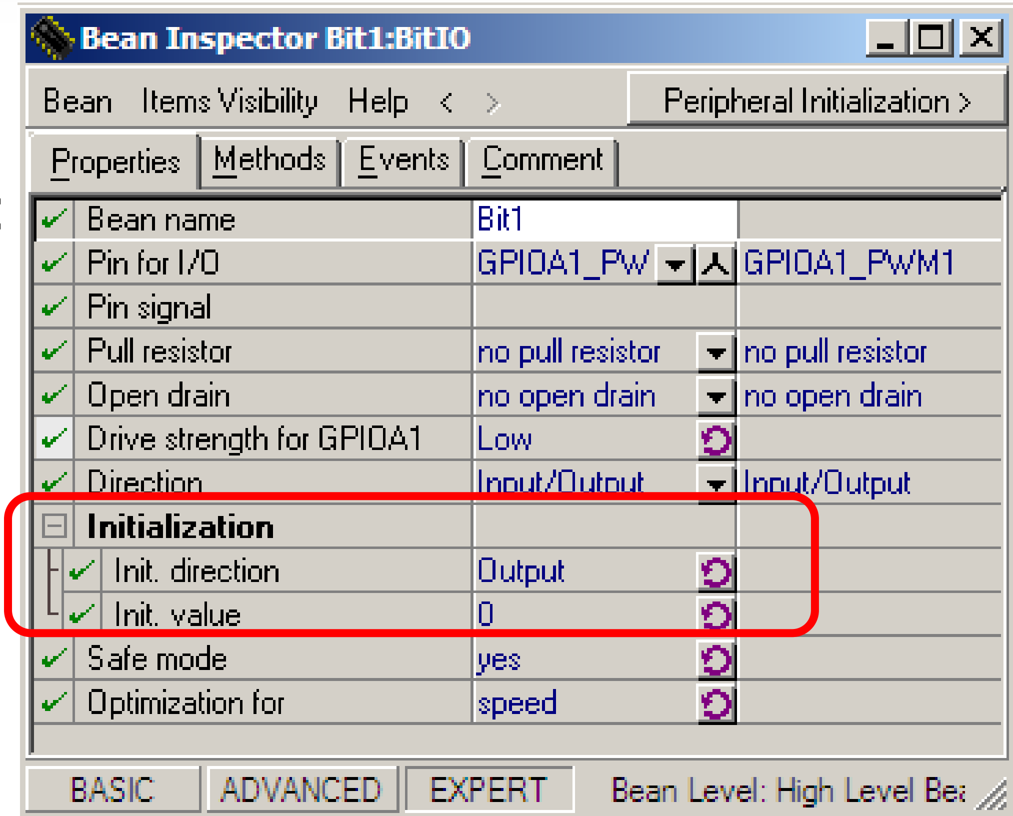
Tools Library
- Cycle Count, FIFO, FileIO, Test



BitIO Bean

- Can change Bean name
- Select a pin
- Configure pin properties:
  - Enable/disable pullup
  - Open drain/pushpull

Must configure the Init. Direction to **Output** and Select a value at initialization



QuickStart

- What is QuickStart?
- QuickStart Low-level Drivers
- Project Stationary
- Graphical Configuration Too
- QuickStart Highlights

What is Quick Start?

Quick Start = Easy-to-use SW Development Environment

- Set of **Low-level Drivers** for **all Peripheral** Modules
- C-language structures of peripheral memory space
- Unified way of accessing peripheral registers
- Highly optimized to achieve an **optimal assembly** generated

- Ready-to-use Project Templates (“Project Stationery”)
- Compiler configurations (RAM-debug, Flash-standalone targets)
- Processor start-up code
- Interrupt tables or Interrupt Dispatcher
- Debugger initialization files

- **Graphical** Configuration Tool
- User-friendly insight to processor configuration (cont.)

freescale™

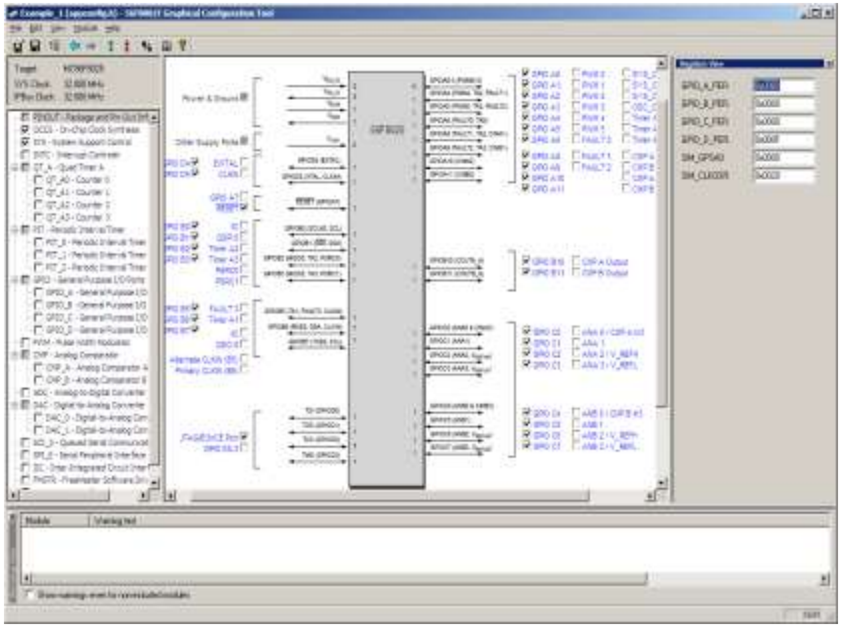
What is Quick Start?

Graphical Configuration Tool

- Edits post-reset processor configuration graphically
- Configuration saved/read from a single ANSI C header file
- GUI to configuration bits of all peripheral module registers
- Possible conflict warnings
- Pin-out view of processor I/O pins

Sample Applications

- Demonstrating usage of GCT, processor peripheral modules and low-level drivers



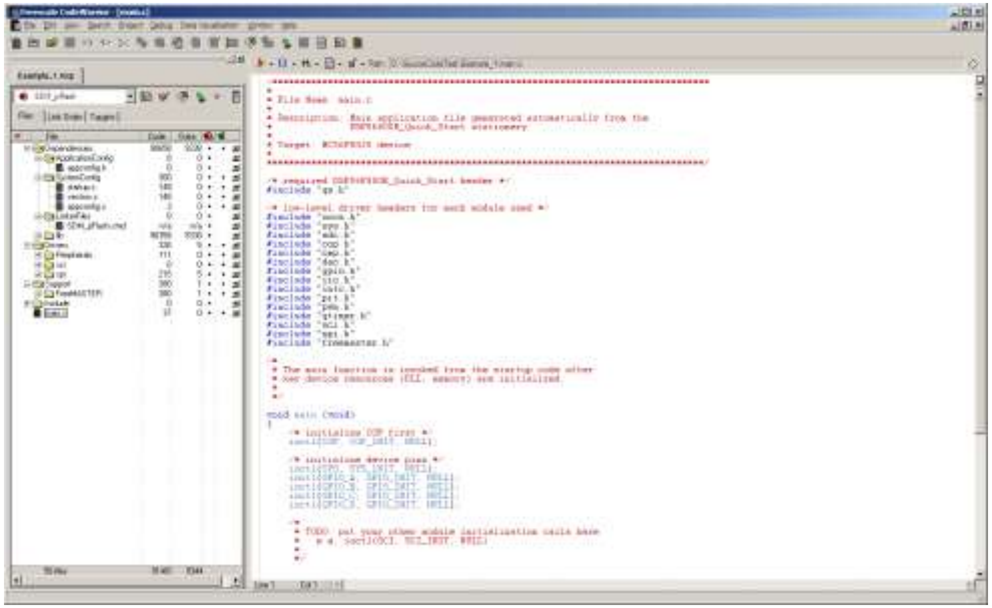
User Manual

- Low-level drivers & tools guide
- Latest device User Manual

Start Environment

CodeWarrior Integration

- Quick Start project stationery is installed directly into the CW
- Support for CW debugger and Flash Programmer
- GCT invoked from CW IDE



Other Tools

- MPC500/MPC5500 supports makefile-based tools (Diab, Green Hills)
- Lauterbach Debugger

ArchIO Structure

- *ArchIO* – global symbol
  - Provides a C interface (structure type) to all peripheral and core registers mapped in data memory
  - All registers are accessed via this structure - no need to know and specify the concrete addresses of the registers to write or read
  - *ArchIO* - declared in the *arch.h* file
  - *ArchIO* structure definition
    - *ArchIO* defined as the *extern* variable
    - Its address defined by a directive in linker command file

ArchIO Structure

```
typedef volatile struct
{
    arch_sTimer     TimerA;                 /* TMRA_BASE    0xF000 */
    arch_sTimer     TimerB_unused;
    arch_sADC       Adc;                    /* ADC_BASE     0xF080 */
    arch_sPWM       Pwm;                    /* PWM_BASE     0xF0C0 */
    arch_sIntc      Intc;                   /* INTC_BASE    0xF0E0 */
    arch_sSIM       Sim;                    /* SIM_BASE     0xF100 */
    arch_sCOP       Cop;                    /* COP_BASE     0xF120 */
    arch_sPLL       Pll;                    /* PLL_BASE     0xF130 */
    arch_sLVI       Lvi;                    /* LVI_BASE     0xF140 */
    .
    .
    UWord16        reserved4[0xFF0600];
    arch_sEOnCE     EOnCE;          /* EOnCE_BASE   0xFFFF00 */
} arch_sIO;
```

freescale™

ArchIO Structure

- COP structure – defined in arch.h file

```
typedef volatile struct
{
    ARCH_REG2(UWord16, copctl,   ControlReg);
    ARCH_REG2(UWord16, copto,    TimeoutReg);
    ARCH_REG2(UWord16, copctr,   ServiceReg);
    ARCH_REG1(UWord16, reserved[13]);
} arch_sCOP;
```

ArchIO Structure

- arch.h file – extern declaration of ArchIO variable

```
/* The location of the following structure is defined in linker.cmd */
   extern arch_sIO    ArchIO;
```

- Linker command file – address assignment to the structure

```
   FArchIO = ADDR(.x_onchip_peripherals);
```

Using the ArchIO Structure

- Example of read/write operation using ArchIO structure

UWord16 RegValue;   // variable definition

RegValue = ArchIO.TimerA.Channel0.HoldReg; // read register

ArchIO.TimerA.Channel0.CompareReg1 = 0x8000; // write number to reg

- Example of the same operation as previous case using *periphMemRead* and *periphMemRead* macros

UWord16 RegValue;   // variable definition

RegValue = periphMemRead(&ArchIO.TimerA.Channel0.HoldReg);

periphMemWrite(0x8000 , &ArchIO.TimerA.Channel0.CompareReg1);

freescale™

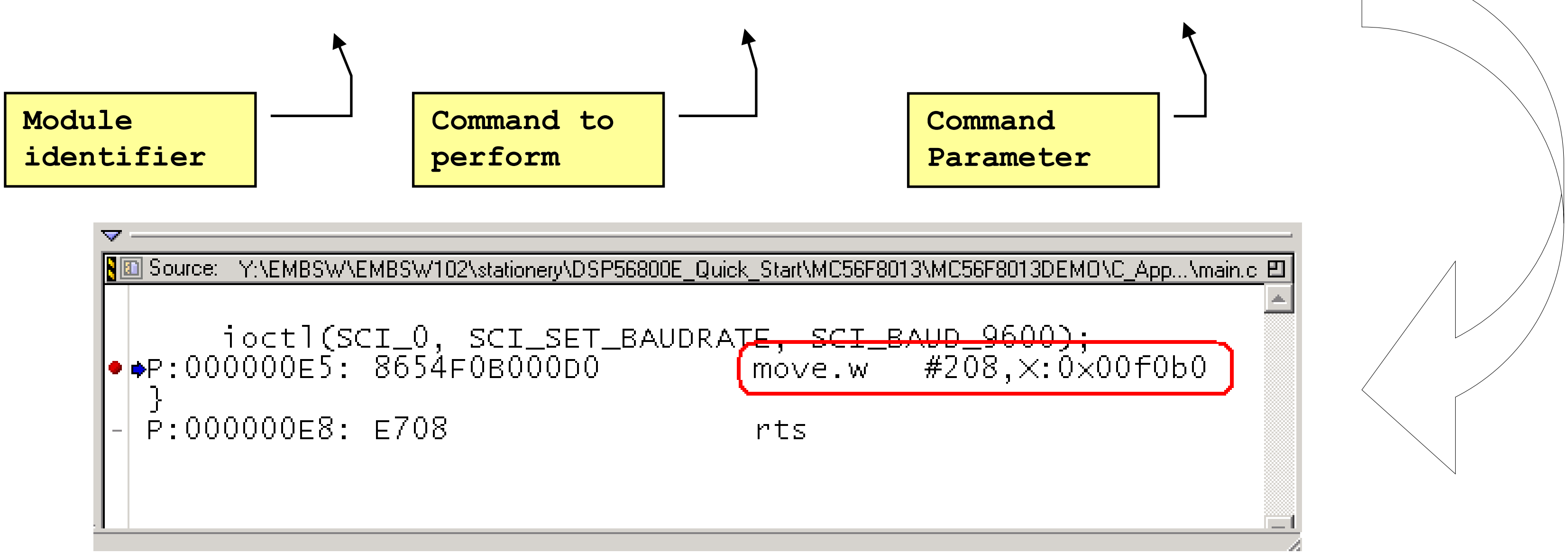
Tools - QuickStart

- What is QuickStart?
- QuickStart Low-level Drivers
- Project Stationary
- Graphical Configuration Too
- QuickStart Highlights

Low-level Drivers

- **Quick Start Low-level Drivers**
  - Full control over and full access to all processor resources
  - Unifies access to peripheral memory space (**`ioctl`** call)
  - Registers are not accessed directly, although this is still possible
  - **`ioctl`** calls are optimally compiled macros or functions

`ioctl(SCI_0, SCI_SET_BAUDRATE, SCI_BAUD_9600)`



ioctl Commands

- ioctl – Input Output Control
- ioctl – general syntax

  ioctl(module_ID, cmd_name, cmd_spec_param);
- **module_ID** – module identifier
  - Predefined symbolic constant corresponding to names of peripheral modules
    - Example: GPIO_A, GPIO_B, ADC, ADC_A, ADC_B, PWM, PWM_A, PWM_B, COP, etc.
  - The base address of the peripheral module
  - List of module identifiers – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.

freescale™

ioctl Commands

- **cmd_name** – specifies action performed on a peripheral module
  - Command is depended to performed operation
  - List of commands – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.
  - Set of commands for each peripheral
    - Example for pwm.h:
      - PWM_SET_PRESCALER
      - PWM_SET_RELOAD_FREQUENCY
      - PWM_FAULT_INT_ENABLE
      - Etc.
    - Self-explaining name of commands
    - No need to dive into deep documentation studying
  - INIT command – essential command for each peripheral
    - Example: COP_INIT, ADC_INIT, PWM_INIT, GPIO_INIT, etc.

freescale™

ioctl Commands

- **cmd_spec_param** – command specific parameter
  - Specifies other data required to execute the command
  - In general, it can be
    - Pointer to the structure
    - NULL value
    - Variable-value in dependency with the specific command
  - List of recommended parameters – “*.h” corresponding to managed peripheral
    - Example: gpio.h, adc.h, pwm.h, sci.h, spi.h, qtimer.h, etc.
    - Example for pwm.h:
      - #define PWM_PRESCALER_DIV_1 0
      - #define PWM_PRESCALER_DIV_2 1
      - #define PWM_PRESCALER_DIV_4 2
      - #define PWM_PRESCALER_DIV_8 3
      - Etc.

ioctl Commands Implementation

- ioctl command - macro

  #define ioctl(fd,cmd,prm) ioctl##cmd((fd),(prm))

- Macro definition – periph.h
- fd
  - Peripheral module base address
  - Address assigned from ArchIO structure

ioctl Commands Implementation

- Example for GPIO – general command
  - gpio.h
    - **#define GPIO_A (&ArchIO.PortA) // GPIO_A base address**
  - User source code - *.c
    - **ioctl(GPIO_A, GPIO_SET_PIN, BIT_0);**
  - periph.h
    - **#define periphBitSet(mask, addr) (*(addr) |= (mask))**
  - gpio.h
    - **#define ioctlGPIO_SET_PIN(pGpioBase, param)**

      periphBitSet(param, &((pGpioBase)->dr))
  - Compiler result – assembly code

    ioctl(GPIO_A, GPIO_SET_PIN, BIT_0);

    P:0000414A: 8254F1510001 bfset #1,X:0x00f151

ioctl Commands Implementation

- Example for GPIO – INIT command
  - gpio.h
    - **#define GPIO_A (&ArchIO.PortA) // GPIO_A base address**
  - User source code - *.c
    - **ioctl(GPIO_A, GPIO_INIT, NULL);**
  - gpio.h
    - **void gpioInit(arch_sPort *pGpioBase); // declaration**
    - **#define ioctlGPIO_INIT(pGpioBase, param) gpioInit(pGpioBase)**
  - gpioInit() function execution
    - Function definition - gpio.c
    - Usually executed just ones during chip initialization
    - Performs setting stored in appconfig.h file
    - appconfig.h file modified by GCT (Graphical Configuration Tool)

Low-level Drivers

- **Why not to use direct access to peripheral registers?**
  - Most of **`ioctl`** calls are “macroized” to direct register access anyway (either read/write or bit-set/bit-clear instructions used)
  - Some registers do need special attention, **`ioctl`** usage brings kind-of **abstraction** and **transparency** to an application code while still being optimally compiled

Decoder Control Register (DECCR)

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write	HIRQ	HIE	HIP	HNE	SWIP	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

`Clear-by-write-one interrupt request flags`

Exercise:

Suppose you want to clear DIRQ bit only, while not modifying the rest of the register. Also you must not clear the HIRQ and XIRQ bits.
What C or assembly statement will you use on 56F800E?

solution on the next slide...

Low-level Drivers: Exercise

Decoder Control Register (DECCR)

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write	HIRQ	HIE	HIP	HNE	SWIP	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010   /* DIRQ bit constant */
ArchIO.Decoder0.deccr       /* register in the peripheral structure */
```

```
C-language:
    ArchIO.Decoder0.deccr = DECCR_DIRQ;
```

?

```
56F800E Assembler:

    asm ( move.w #>16,X:0x00f180 );
```

- DIRQ gets cleared ... OK
- XIRQ and HIRQ remain unchanged ... OK
- All other bits get reset! ... Wrong!

freescale™

Low-level Drivers: Exercise

Decoder Control Register (DECCR)

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clear-by-write-one interrupt request flags

```
#define DECCR_DIRQ 0x0010    /* DIRQ bit constant */
ArchIO.Decoder0.deccr        /* register in the peripheral structure */
```

C-language:

```
ArchIO.Decoder0.deccr |= DECCR_DIRQ;
```

?

56F800E Assembler:

```
asm ( bfset #0x10,X:0x00f180 );
```

- DIRQ gets cleared ... OK
- Other register bits unchanged ... OK
- XIRQ or HIRQ gets reset if they read as “1” (i.e. when interrupt request is pending!)

Low-level Drivers: Exercise

Decoder Control Register (DECCR)

Base + $0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	MODE	
Write					SWIP											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Clear-by-write-one interrupt request flags

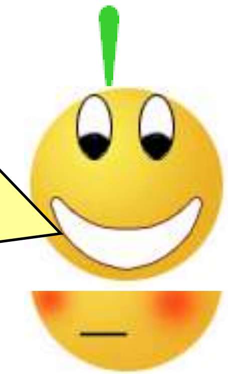
```
#define DECCR_DIRQ 0x0010        /* DIRQ bit constant */
#define DECCR_HIRQ 0x8000        /* HIRQ bit constant */
#define DECCR_XIRQ 0x0100        /* XIRQ bit constant */
ArchIO.Decoder0.deccr            /* register in the peripheral structure */
```

```
C-language:
    ArchIO.Decoder0.deccr &= ~(~(DECCR_DIRQ) &

56F8
```

Better work with QuickStart and use the "Clear Interrupt Request" command:

```
ioctl(DEC_0, DEC_INT_REQUEST_CLEAR, DEC_DECCR_DIRQ);
```



Low-level Drivers: Highlights

- Full control over all processor resources

- Real-world application development **know-how** inside
  - transparent solution to tricky register access
  - higher abstraction and code readability without loosing performance

- Delivered as source code

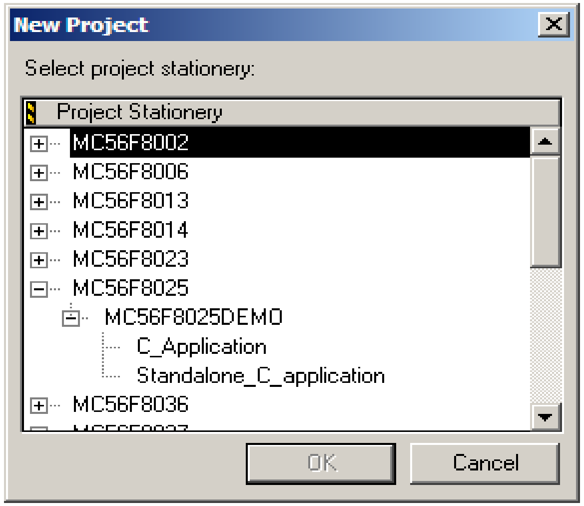
- Fully tested and documented

Tools - QuickStart

- What is QuickStart?
- QuickStart Low-level Drivers
- Project Stationary
- Graphical Configuration Too
- QuickStart Highlights

Project Stationery

- **CodeWarrior concept of creating a new project**
  - CodeWarrior “clones” the project template and creates a ready-to-use skeleton of a new application
  - In Quick Start, a dedicated project stationery exists for each processor and evaluation board (EVB)
    - Processors differ in memory layout, peripheral modules etc.
    - For a given processor, more than one EVB may exist, differing in how the processor is connected with external components



Project Stationery

- Multiple Compiler configurations per project
  - RAM-based debugging targets
  - Standalone Flash-based (release) targets
  - CPU Simulator target

- Start-up code, Board Initialization, Interrupt tables

- Linker Command Files
  - provide the linker with information about how to arrange a C-code in memory

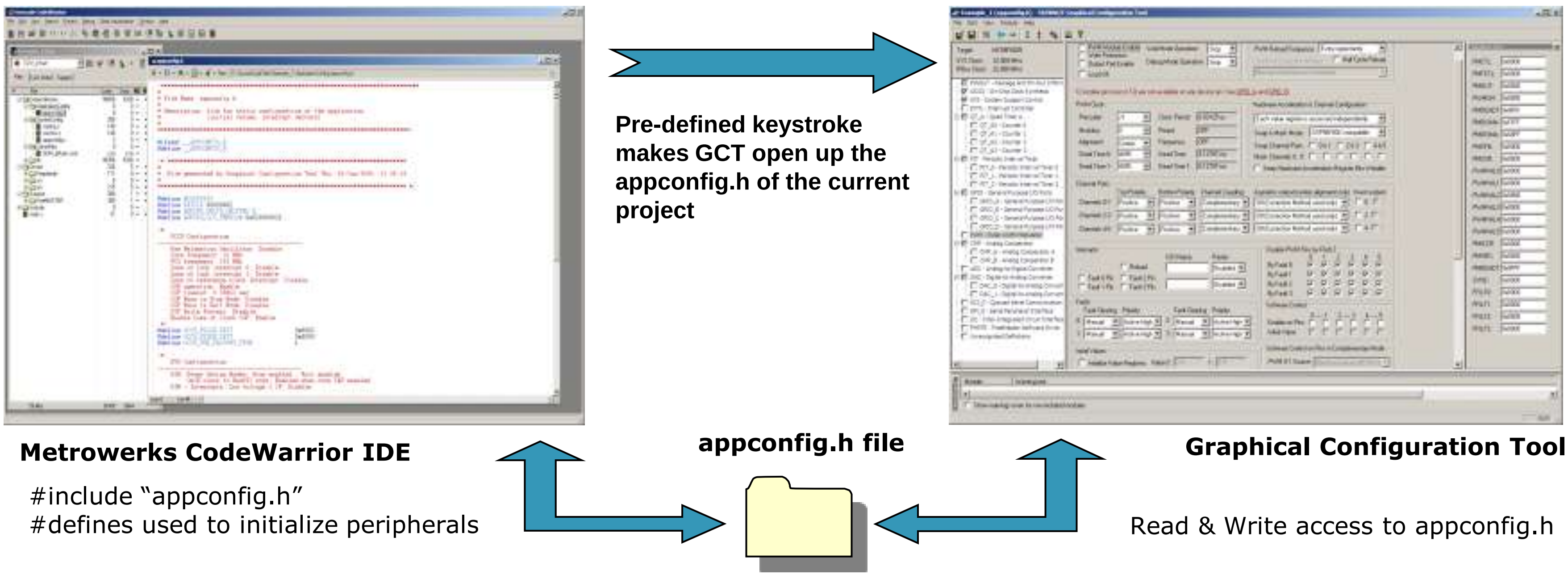
- Debugger Configuration Files
  - Making the EVB ready for RAM-based debugging
  - Making the EVB ready for Flash Programmer
  - Memory description files

Tools - QuickStart

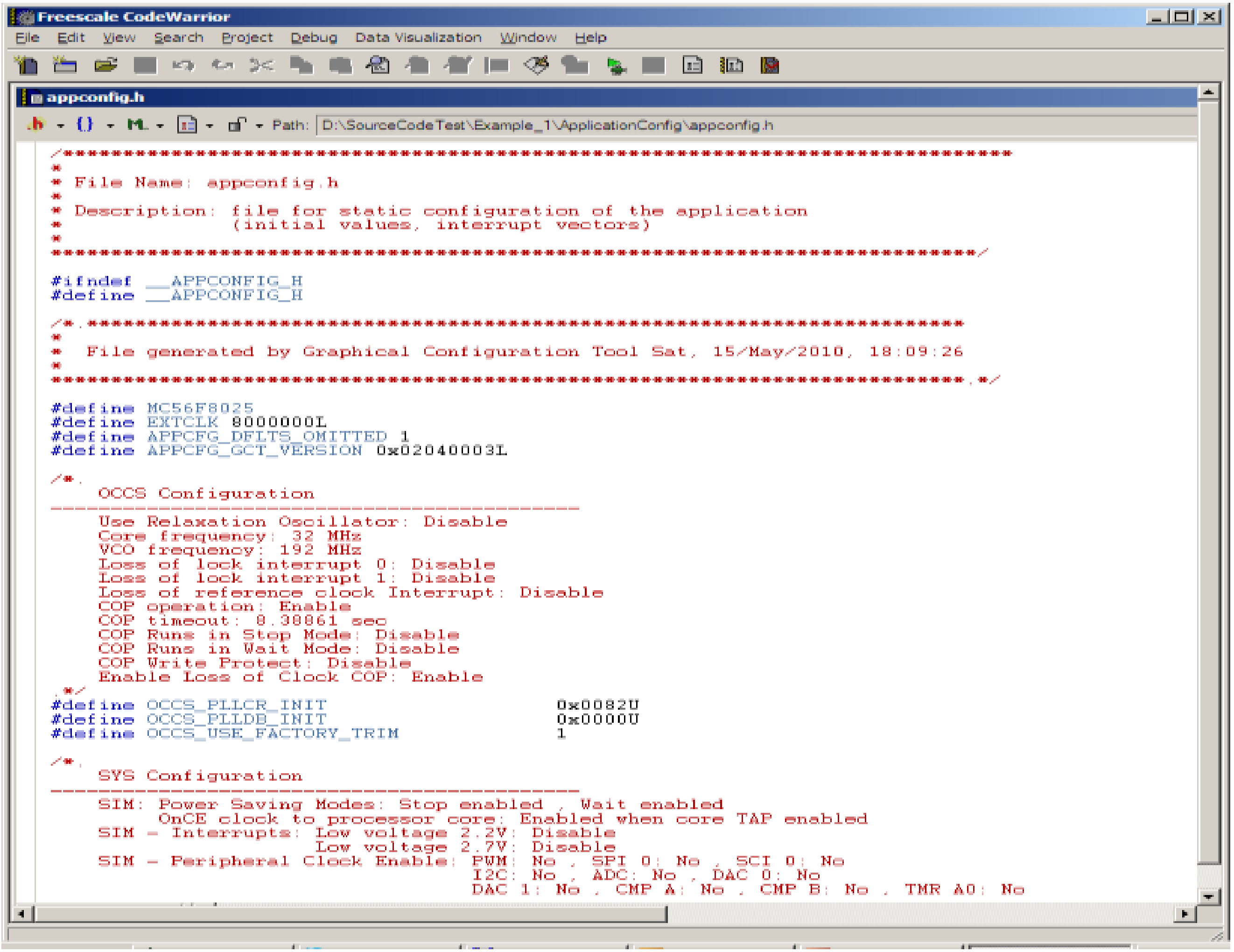
- What is QuickStart?
- QuickStart Low-level Drivers
- Project Stationary
- Graphical Configuration Tool
- QuickStart Highlights

Graphical Configuration Tool

- **A desktop application for MS Windows**
  - Used to edit the ANSI C-compatible application configuration header file (typically appconfig.h for QuickStart applications)



Graphical Configuration Tool: appconfig.h

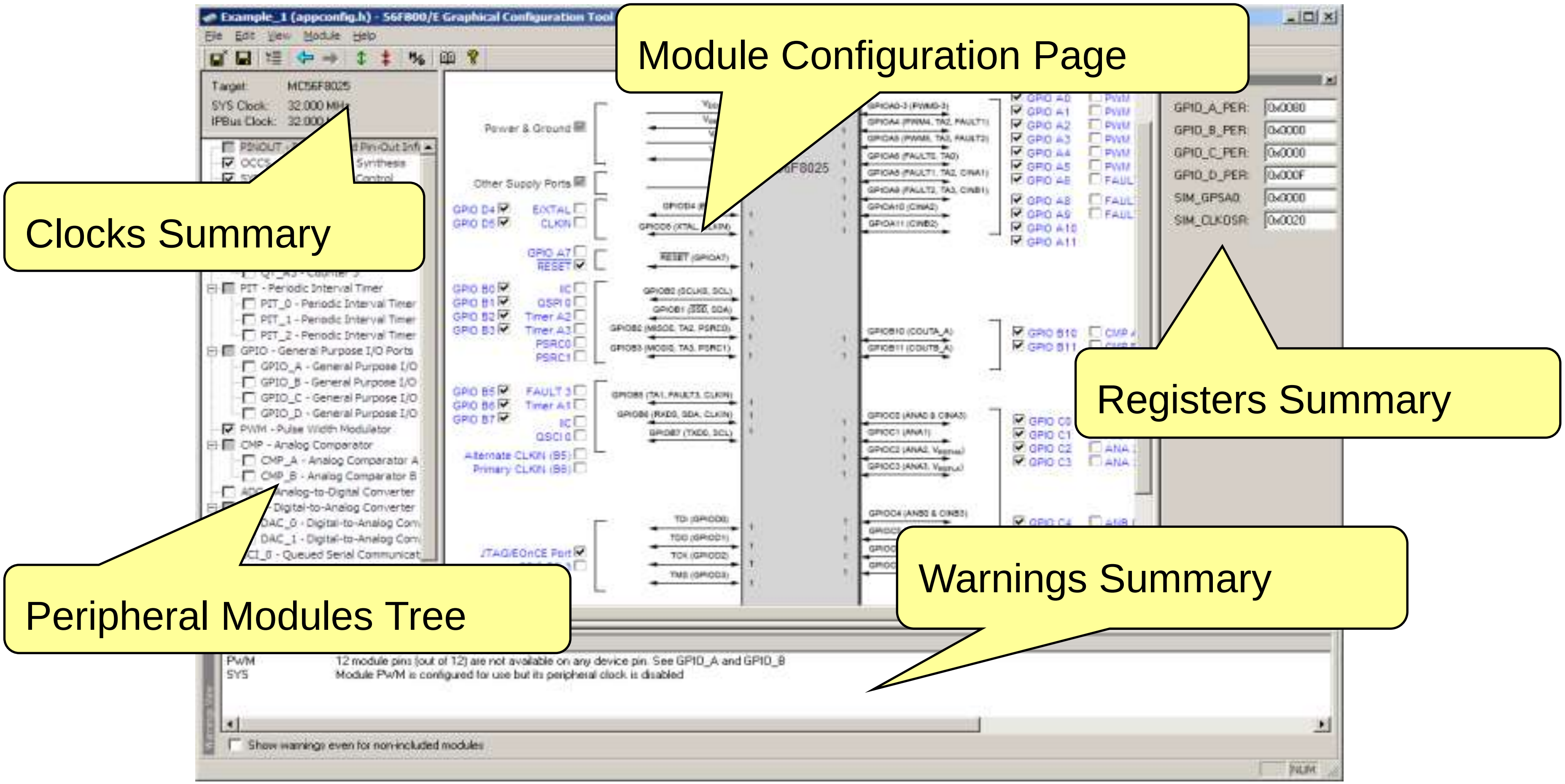


Graphical Configuration Tool: appconfig.h

- **A single macro constant per peripheral register**
- **Configuration summary comments**
- **Read / Write in GCT**
  - Enables manual editing of the **`appconfig.h`** file
  - Copy & paste migrating to other CPUs
  - GCT supports importing of module configuration within a single project or between projects
- **Private section in `appconfig.h` file**
  - Users put other global symbols & definitions here
  - The file can be a real application configuration file (not only the processor configuration)

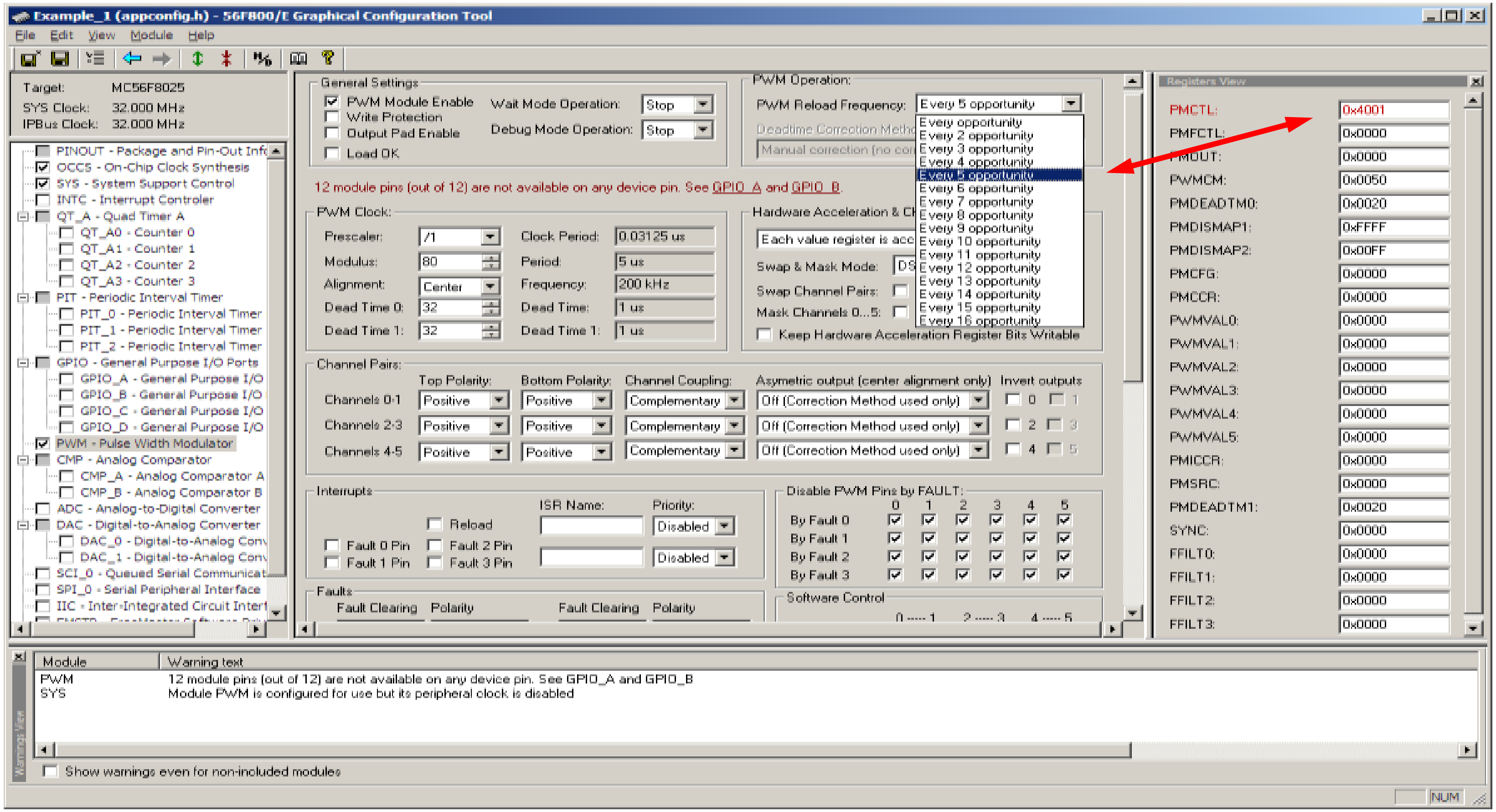
Graphical Configuration Tool

- Different Control Page for each Peripheral Module



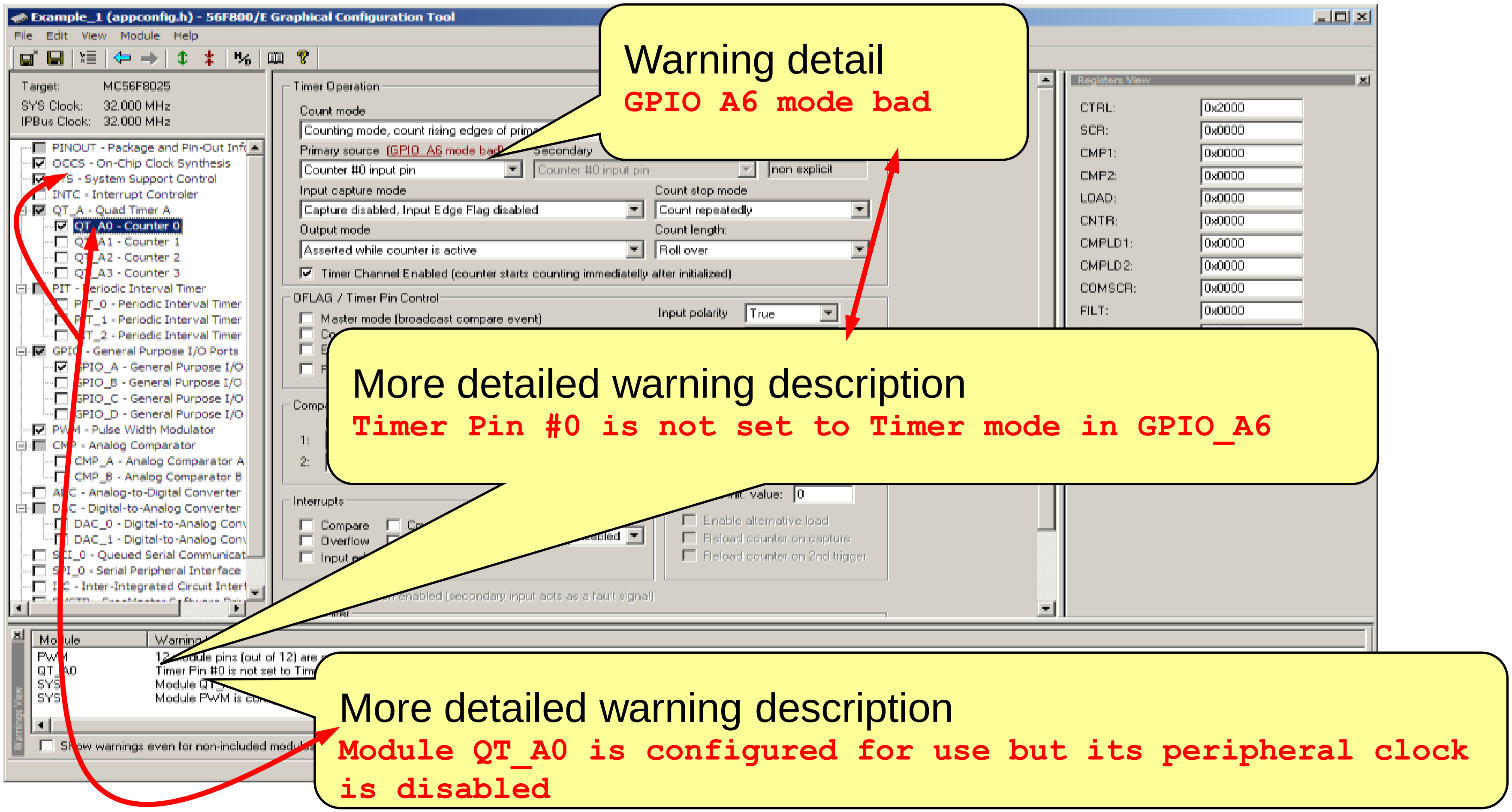
Graphical Configuration Tool

- **Direct Register Value View**



Graphical Configuration Tool

- **Conflict Warnings**



Tools - QuickStart

- What is QuickStart?
- QuickStart Low-level Drivers
- Project Stationary
- Graphical Configuration Tool
- QuickStart Highlights

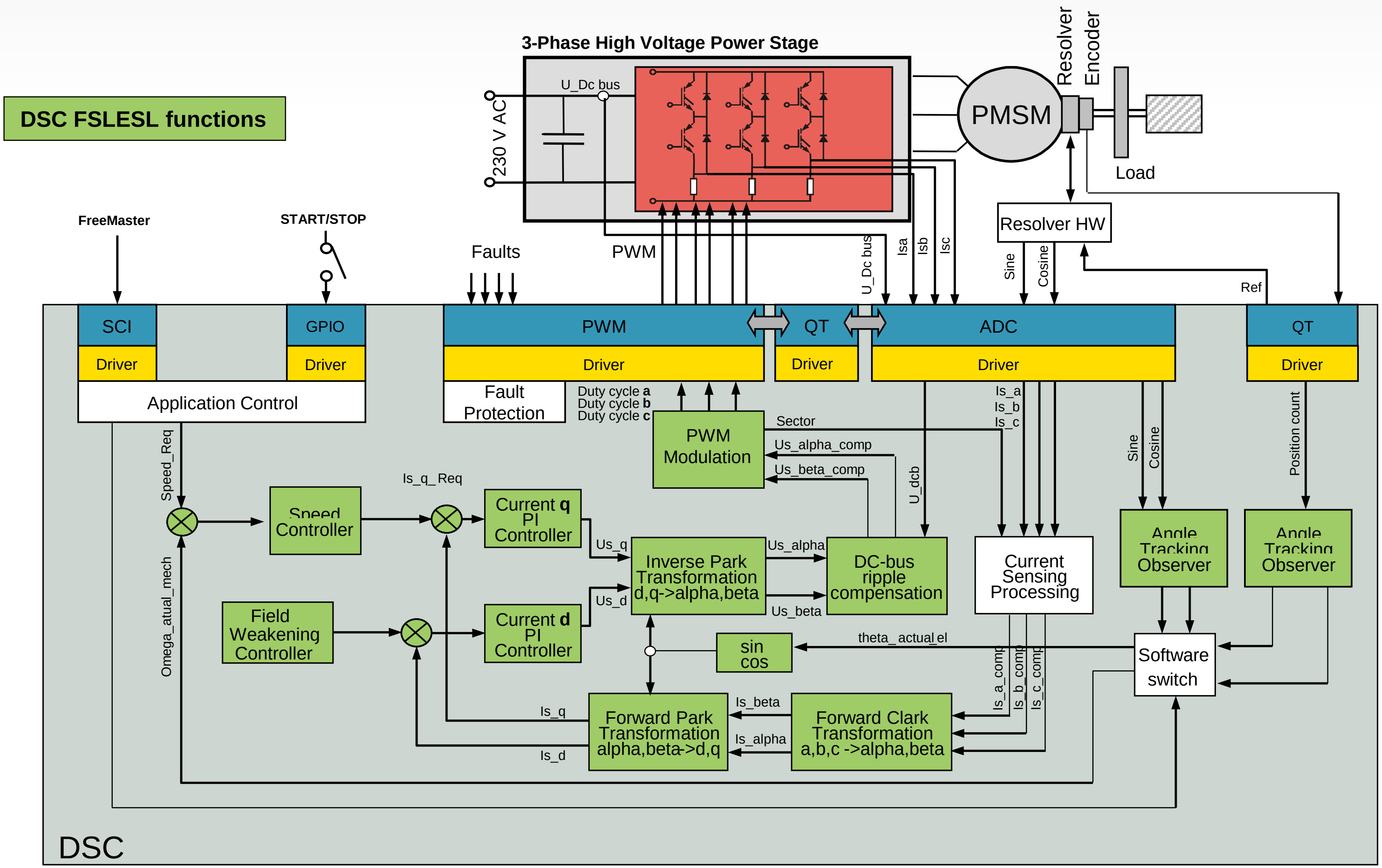
QuickStart Highlights

- **Highlights**
  - QuickStart helps users to get familiar with the processor quickly
    - GCT helps to understand individual bits of peripheral registers
    - Sample applications demonstrate how to access the peripheral modules
  - QuickStart helps users to jump in the SW development quickly
    - A ready-to-use project stationery to start a new project
    - GCT immediately available
  - No performance penalty when using QuickStart
    - Optimal code, each instruction matters
    - Suitable for hard real-time applications (motor control)
    - Source files available, everything under control, no hidden code
- **Quality**
  - Developed under CMM-Level 3 certified process

freescale™

Freescale Library

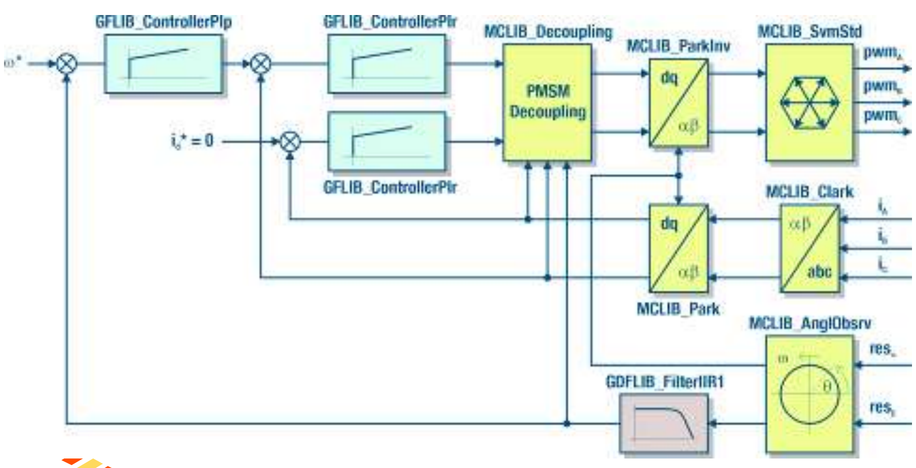
FOC Application Block Diagram



freescale™

Freescale Embeded Software Libraries

- **Library Provides:**
  - Optimized and tested algorithms
  - Full algorithms documentation
  - S/W library in ".lib" form that can be included into any project
- **Algorithms:**
  - ASM coded
  - optimized
  - fully tested using Matlab models
- **Algorithm Sets:**
  - General Functions / Math
  - Motor Control
  - Digital Filters
  - Advanced Library (sensorless)
- **Supported devices**
  - Anquilla/Hawk V2 DSC
  - ColdFire V1 (selected algorithms)
  - CortexM4

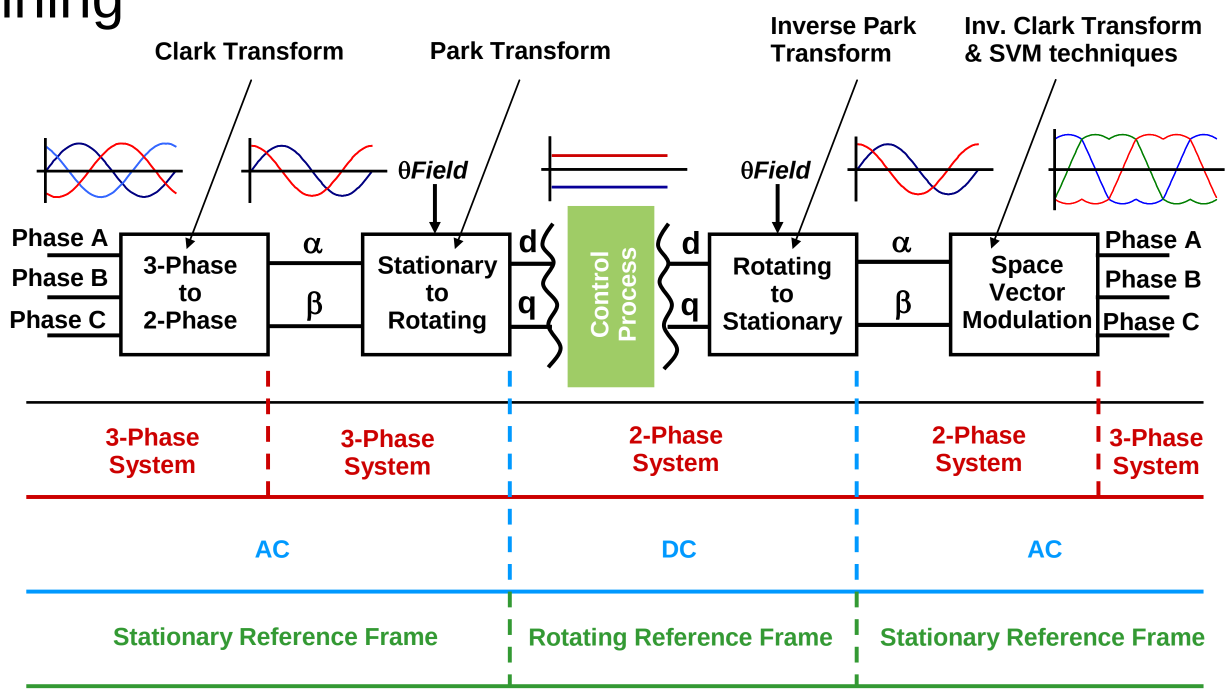


Implemented Algorithms				
Library	**Core**	**56800E**	**MCF51**	**Cortex M4**
GFLIB	Sine	3	1	1
	Cosine	3	1	1
	Tangent	1	1	1
	Arcus Sine	1	0	1
	Arcus Cosine	1	0	1
	Arcus Tangent	1	0	1
	Arcus Tangent YX	1	0	1
	Sifted Arcus Tangnet YX	1	0	1
	Square Root	2	1	1
	Ramp	2	2	1
	Dynamic Ramp	2	0	0
	Limiter	6	2	3
	Hysteresis	1	0	1
	Signum	2	0	1
	Look-up Table	1	0	1
	PI Controller	3	1	2
	PID Controller	2	0	0
MCLIB	Clarke Transformation	1	1	1
	Inverse Clarke Transformation	1	1	1
	Park Transformation	1	1	1
	Inverse Park Transformation	1	1	1
	Space Vector Modulation	6	1	1
	Vector Limiter	2	1	1
	PMSM Decoupling	1	1	1
	DC Bus Ripple Elimination	2	1	1
GDFLIB	IIR Filter	2	1	1
	Moving Avg. Filter	1	1	1
ACLIB	Angle Tracking Observer	2	1	0
	Tracking Observer	1	0	0
	PMSM BEMF Observer in Alpha/Beta	2	1	0
	PMSM BEMF Observer in D/Q	1	0	0
	Integrator	1	0	0

Library – Park & Clarke Transformations

- Written in assembler
- Documentation describes transformation theory and implemented equations
- Properly tested and used on many millions of running applications

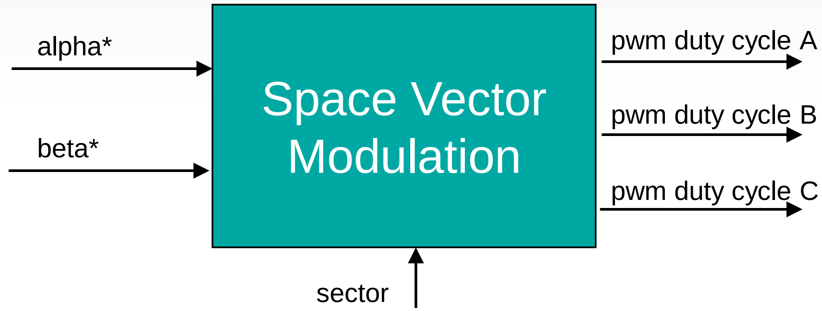
Function	Code Size (words)	Execution Clocks
MCLIB_ClarkTrfm	9	21/22
MCLIB_ClarkTrfmInv	12	24/25
MCLIB_ParkTrfm	9	24/25
MCLIB_ParkTrfmInv	9	24

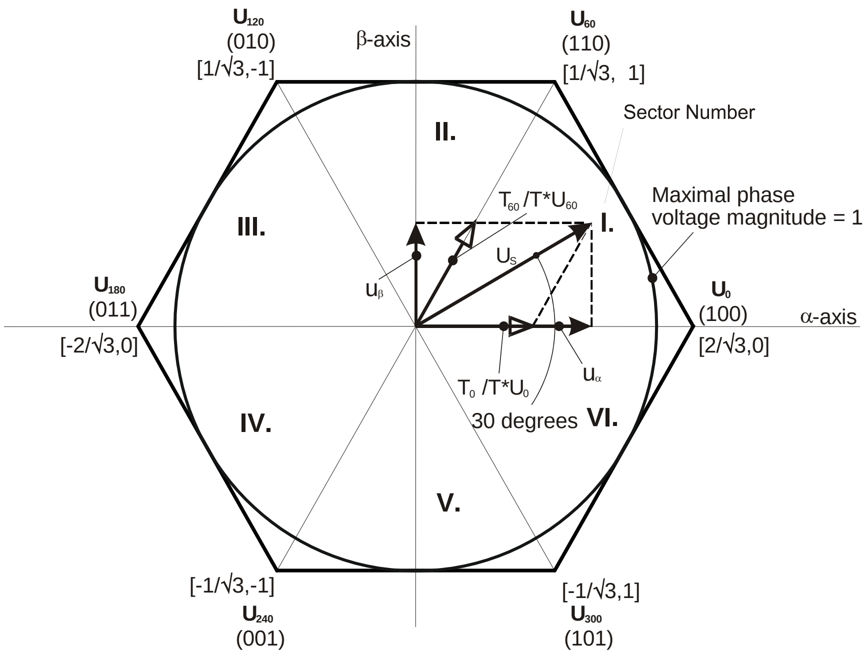


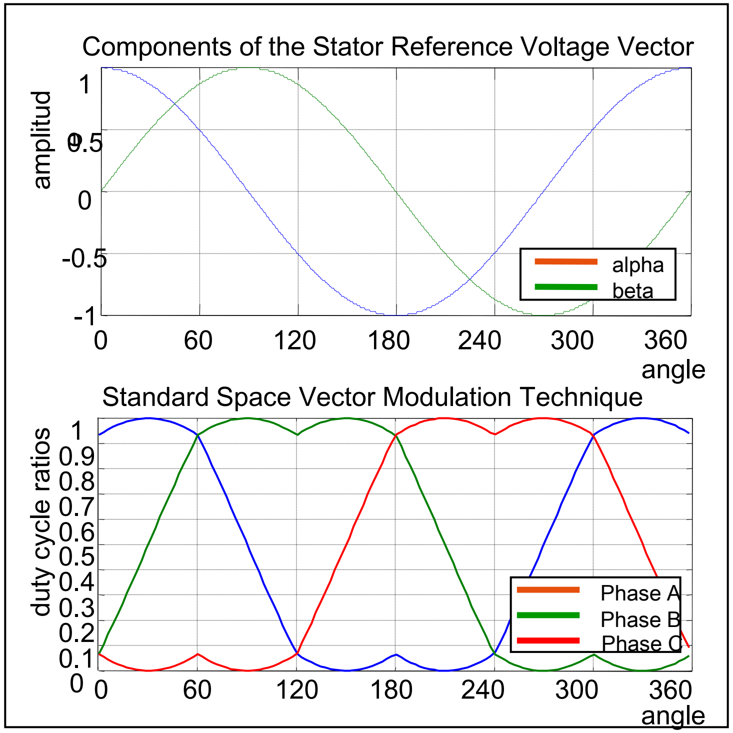
freescale™

Space Vector Modulation Basics

- Transforms directly the stator voltage vectors from the two-phase coordinate system fixed with stator to PWM signals
- Output voltage vector is created by continuous switching of two adjacent vectors and the “NULL” vectors







freescale™

3.17 GFLIB_Ramp16

The function calculates a 16-bit version of the up/down ramp with the step increment/decrement defined in the pParam structure.

3.17.1 Synopsis

```
#include "gflib.h"
Frac16 GFLIB_Ramp16(Frac16 f16Desired, Frac16 f16Actual, const
GFLIB_RAMP16_T *pudtParam)
```

3.17.2 Prototype

```
asm Frac16 GFLIB_Ramp16FAsm(Frac16 f16Desired, Frac16 f16Actual, const
GFLIB_RAMP16_T *pudtParam)
```

3.17.3 Arguments

Table 3-38. Function Arguments

Name	In/Out	Format	Range	Description
f16Desired	In	SF16	0x8000... 0x7FFF	Desired value; the **Frac16** data type is defined in header file GFLIB_types.h
f16Actual	In	SF16	0x8000... 0x7FFF	Actual value; the **Frac16** data type is defined in header file GFLIB_types.h
*pudtParam	In	N/A	N/A	Pointer to structure containing the ramp-up and -down increments

Table 3-39. User Type Definitions

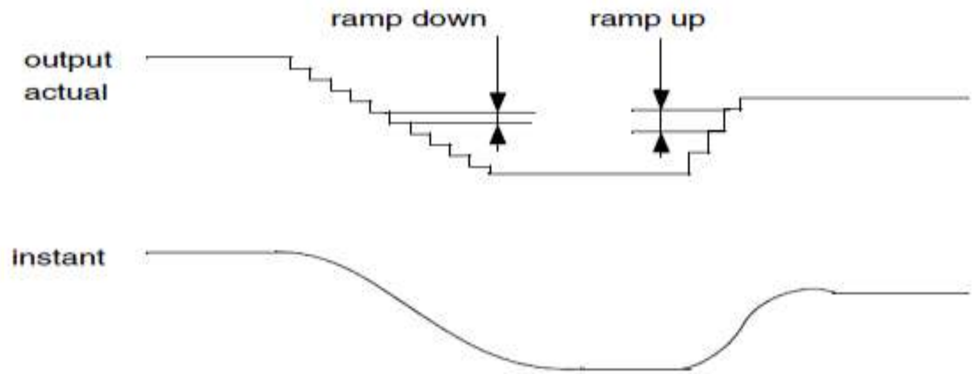
Typedef	Name	In/Out	Format	Range	Description
GFLIB_RAMP16_T	f16RampUp	In	SF16	0x8000... 0x7FFF	Ramp up increment
	f16RampDown	In	SF16	0x8000... 0x7FFF	Ramp down increment

3.17.6 Description

The **GFLIB_Ramp16** calculates the 16-bit ramp of the actual value by the up or down increments contained in the pudtParam structure.

If the desired value is greater than the actual value, the function adds the ramp-up value to the actual value. The output cannot be greater than the desired value.

If the desired value is lower than the actual value, the function subtracts the ramp-down value from the actual value. The output cannot be lower than the desired value.



3.17.7 Returns

If f16Desired is greater than f16Actual, the function returns f16Actual + the ramp-up value until f16Desired is reached.

If f16Desired is less than f16Actual, the function returns f16Actual - the ramp-down value until the f16Desired is reached.

3.17.8 Range Issues

The input data value is in the range of <-1,1) and the output data values are in the range <-1,1).

3.17.9 Special Issues

The function **GFLIB_Ramp16** is the saturation mode independent.

3.17.10 Implementation

The **GFLIB_Ramp16** function is implemented as a function call.

Example 3-17. Implementation Code

```
#include "gflib.h"

static Frac16 mf16DesiredValue;
static Frac16 mf16ActualValue;

/* Ramp parameters */
static GFLIB_RAMP16_T mudtRamp16;

void Isr(void);

void main(void)
{
        /* Ramp parameters initialization */
        mudtRamp16.f16RampUp = FRAC16(0.25);
        mudtRamp16.f16RampDown = FRAC16(0.25);

        /* Desired value initialization */
        mf16DesiredValue = FRAC16(1.0);

        /* Actual value initialization */
        mf16ActualValue = 0;
}

/* Periodical function or interrupt */
void Isr(void)
{
        /* Ramp generation */
        mf16ActualValue = GFLIB_Ramp16(mf16DesiredValue,
mf16ActualValue, &mudtRamp16);
}
```

3.17.11 See Also

See GFLIB_Ramp32, GFLIB_DynRamp16 and GFLIB_DynRamp32 for more information.

3.17.12 Performance

Table 3-40. Performance of GFLIB_Ramp16 function

Code Size (words)	18	
Data Size (words)	0	
Execution Clock	Min	36/37 cycles
	Max	36/37 cycles

3.19 GFLIB_DynRamp16

This calculates a 16-bit version of the ramp with a different set of up/down parameters depending on the state of uw16SatFlag. If uw16SatFlag is set, the ramp counts up/down towards the f16Instant value.

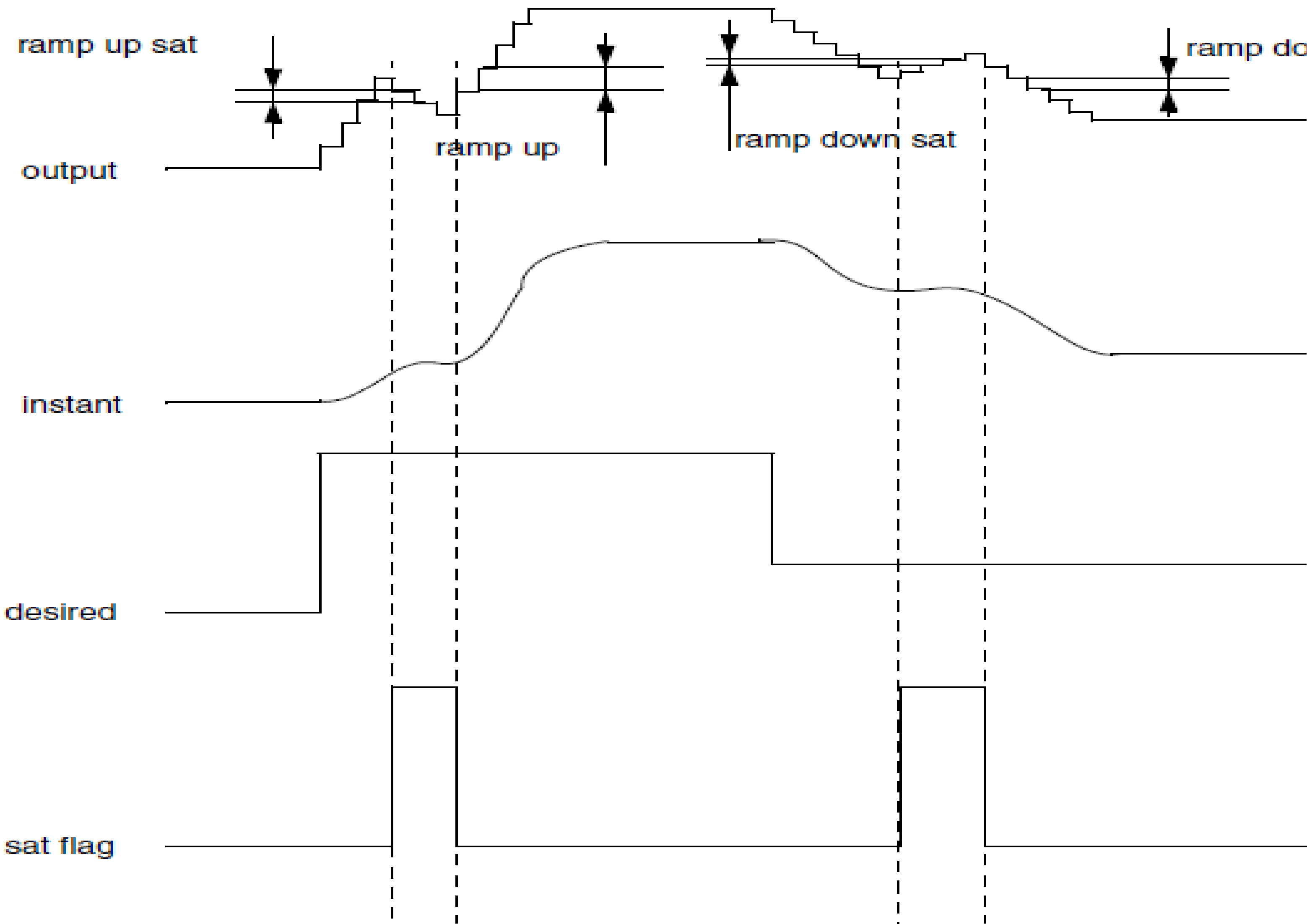


Figure 3-10. Algorithm Diagram

3.2 ACLIB_PMSMBemfObsrvAB

The function calculates the algorithm of back electro-motive force observer in stationary reference frame.

$$\begin{bmatrix} u_\alpha \\ u_\beta \end{bmatrix} = R_S \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + \begin{bmatrix} sL_D & \Delta L\omega_r \\ -\Delta L_D\omega_r & sL_D \end{bmatrix} \cdot \begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} + (\Delta L \cdot (\omega_e i_D - i_Q') + k_e\omega_r) \cdot \begin{bmatrix} -\sin(\theta_r) \\ \cos(\theta_r) \end{bmatrix} \quad \textbf{Eqn. 3-1}$$

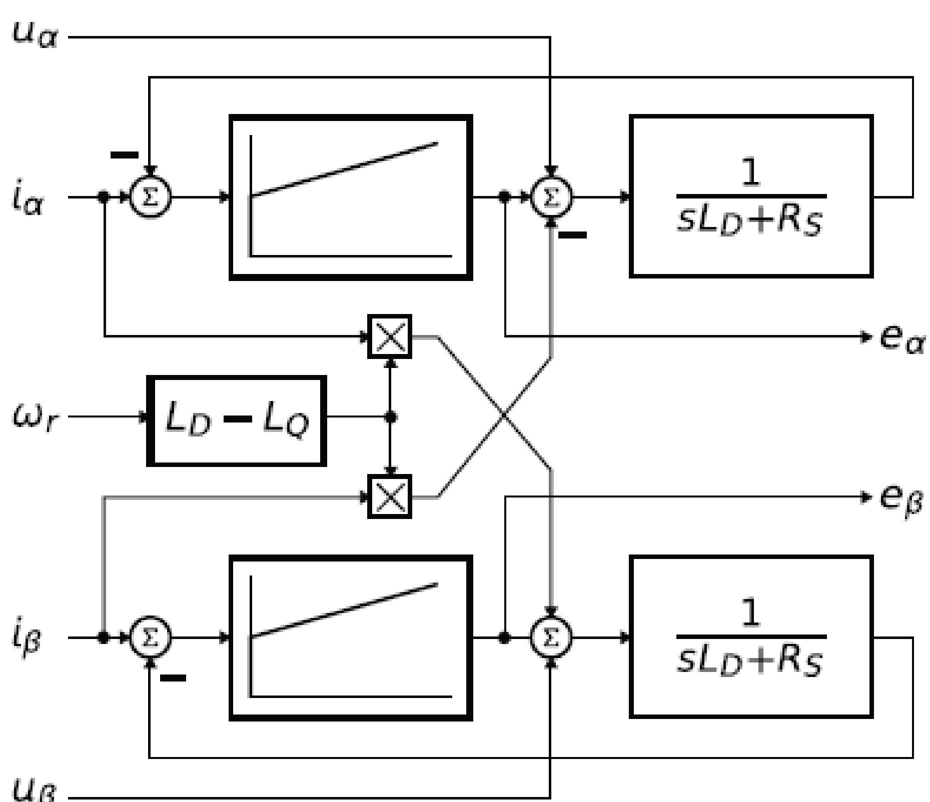


Figure 3-1. Block diagram of back-emf observer

3.4 ACLIB_AngleTrackObsrv

The function calculates angle tracking observer for determination angular speed and position of input functional signal.

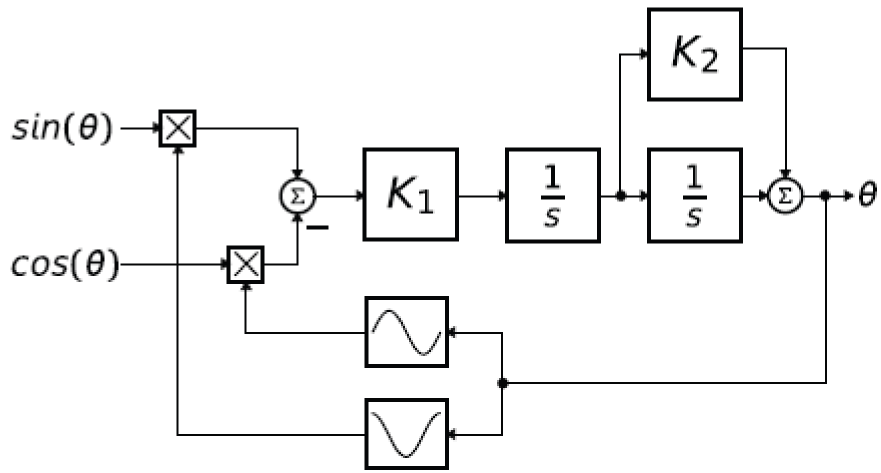


Figure 3-3. Block scheme of the angle tracking observer

Analogue Quantities Scaling

- Analogue quantities (voltage, current, frequency) are scaled to the maximum measurable range – depended on hardware
- Relation between a real and a fractional representation

$$\boxed{\text{Fractional Value} = \frac{\text{Real value}}{\text{Real quantity Range}}}$$

  - Fractional Value – fractional representation of the real value [Frac16]
  - Real Value – real value of the quantity [V, A, RPM, etc.]
  - Real Quantity Range – maximum range of the quantity, defined in the application [V,A,RPM, etc.]

- Angles are represented as a 16-bit fractional values in the range [-1,1] which corresponds to the angle [-PI,PI]

$$-\mathrm{pi} \approx 0\mathrm{x}8000$$

$$\mathrm{pi} \bullet (1.0 - 2^{-15}) \approx 0\mathrm{x}7\mathrm{FFF}$$

Analogue Quantities Scaling

- Example:
  - Vmax = 407 V - maximum measurable voltage range of the power stage
  - Vmeasured = 303.5 – DC-Bus voltage measured with ADC

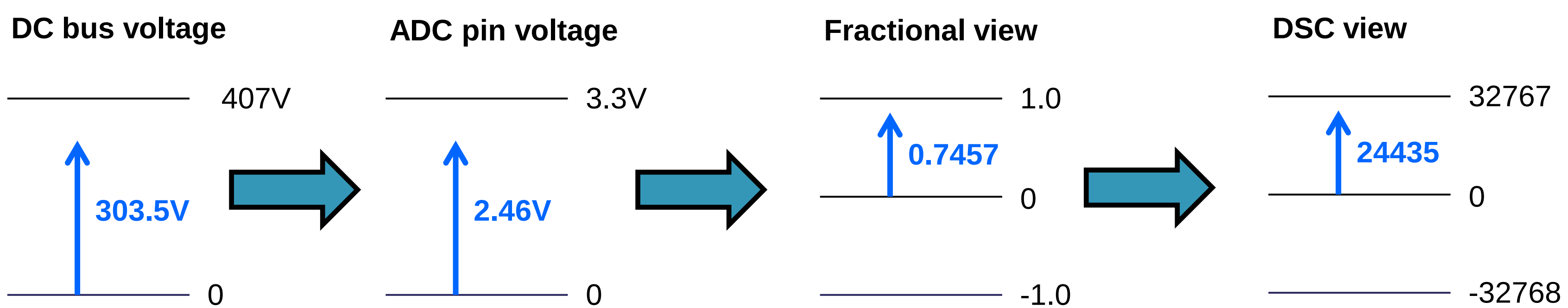
$$(Frac16)voltage\_variable = \frac{V_{MEASURED}}{V_{MAX}} = \frac{303.5}{407} = 0.7457$$

  - Fractional variables are internally stored as signed 16-bit integer values

$$(Int16)voltage\_variable = (Frac16)voltage\_variable \bullet 2^{15} = 0.7457 \bullet 2^{15} = 24435$$

5000(Max)
/32768(16bit)
=0.153008

Resolution :
0.153008[rpm]



freescale™

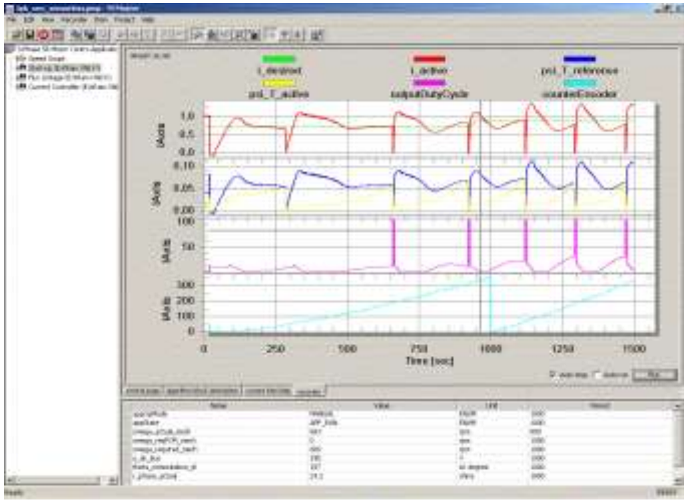
FreeMASTER

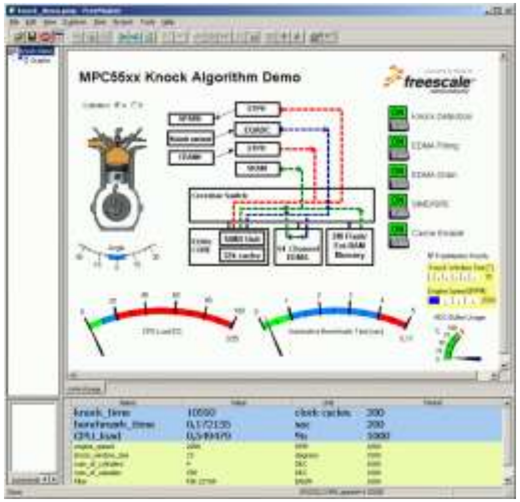
- What is FreeMASTER?
- Real-Time Monitor
- Graphical User Interface to the Embedded Application
- Demonstration Platform & Selling Tool

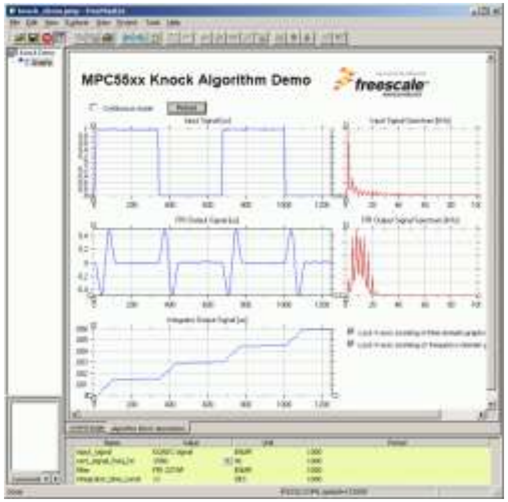
What is FreeMASTER?

- Real-time Monitor
- Graphical Control Panel
- Demonstration Platform & Selling Tool

FOR YOUR EMBEDDED APPLICATION







Tools - FreeMASTER

- What is FreeMASTER?

- Real-Time Monitor

- Graphical User Interface to the Embedded Application

- Demonstration Platform & Selling Tool

freescale™

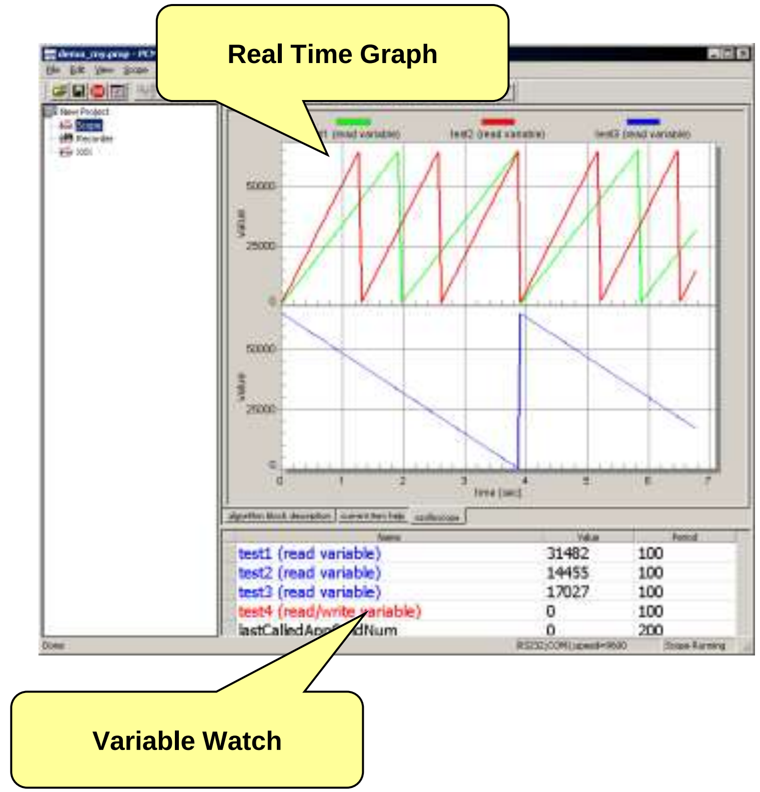
FreeMASTER as a Real-Time Monitor

- Connects to an embedded application
  - SCI, UART
  - JTAG/EOnCE (56F8xxx only)
  - BDM (HCS08, HCS12 only)
  - CAN Calibration Protocol
  - Ethernet, TCP/IP
  - Any of the above remotely over the network

- Enables access to application memory
  - Parses ELF application executable file
  - Parses DWARF debugging information in the ELF file
  - Knows addresses of global and static C-variables
  - Knows variable sizes, structure types, array dimensions etc.

FreeMASTER as a Real-Time Monitor

- Displays the variable values in various formats:
  - **Text**, tabular grid
    - variable name
    - value as hex, dec or bin number
    - min, max values
    - number-to-text labels
  - **Real-time waveforms**
    - up to 8 variables simultaneously in an oscilloscope-like graph
  - **High-speed recorded data**
    - up to 8 variables in on-board memory **transient recorder**



FreeMASTER as a Real-Time Monitor

Additional features:

- Variable Transformations
  - Variable value can be transformed to custom unit
  - Variable transformations may reference other variable values
  - Values are transformed back when writing a new value to variable

- Application Commands
  - Command code and parameters are delivered to an application for arbitrary processing
  - After processed (asynchronously to a command delivery) the command result code is returned to PC

- Ability to protect memory regions
  - Describing variables visible to FreeMASTER
  - Declaring variables as read-write to read-only for FreeMASTER - the access is guarded by the embedded-side driver

freescale™

FreeMASTER as a Real-Time Monitor

Highlights:

- FreeMASTER helps developers to debug or tune their applications
- Replaces debugger in situations when the processor core can not be simply stopped (e.g. motor control)
- Recorder may be used to visualize transitions in near 10-us resolution

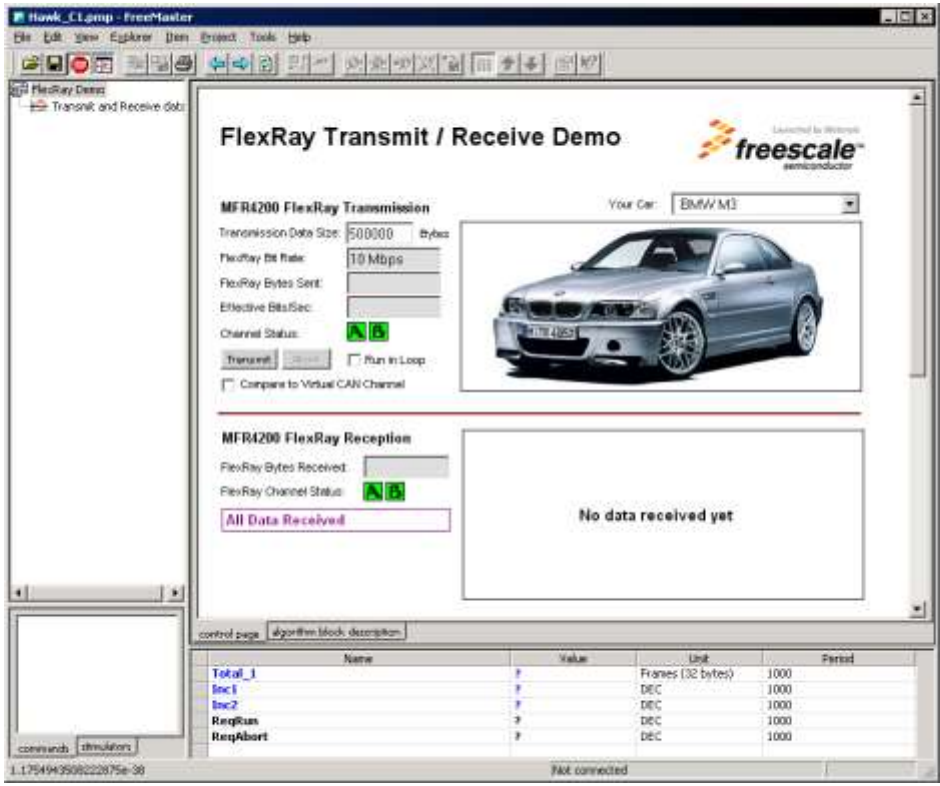
Tools - FreeMASTER

- What is FreeMASTER?
- Real-Time Monitor
- Graphical User Interface to the Embedded Application
- Demonstration Platform & Selling Tool

FreeMASTER as a Graphical User Interface

- Variable Watch pane enables direct setting of the variable value
- Sending Application Commands from the application GUI
- Time-table stimulation of the variable value

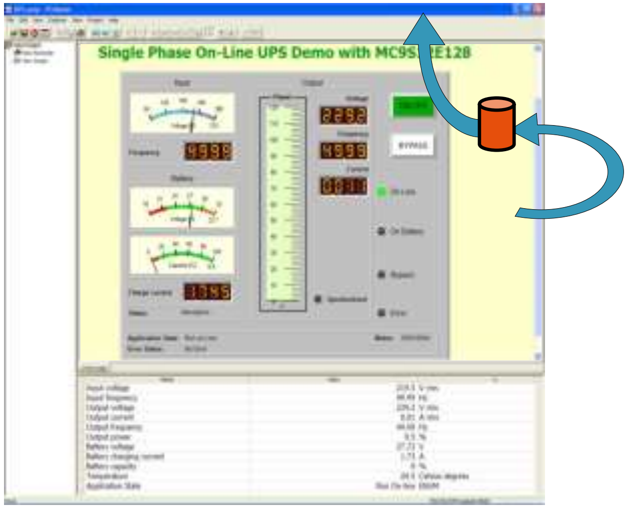
- HTML Pages and Forms
  - JScript or VBScript
  - Push buttons
  - Images, indicators
  - Sounds, videos
  - Sliders, gauges and other 3rd party ActiveX controls



FreeMASTER as a Graphical User Interface`

Scripting in FreeMASTER

- HTML pages are displayed directly in the FreeMASTER window
- HTML may contain scripts and ActiveX objects



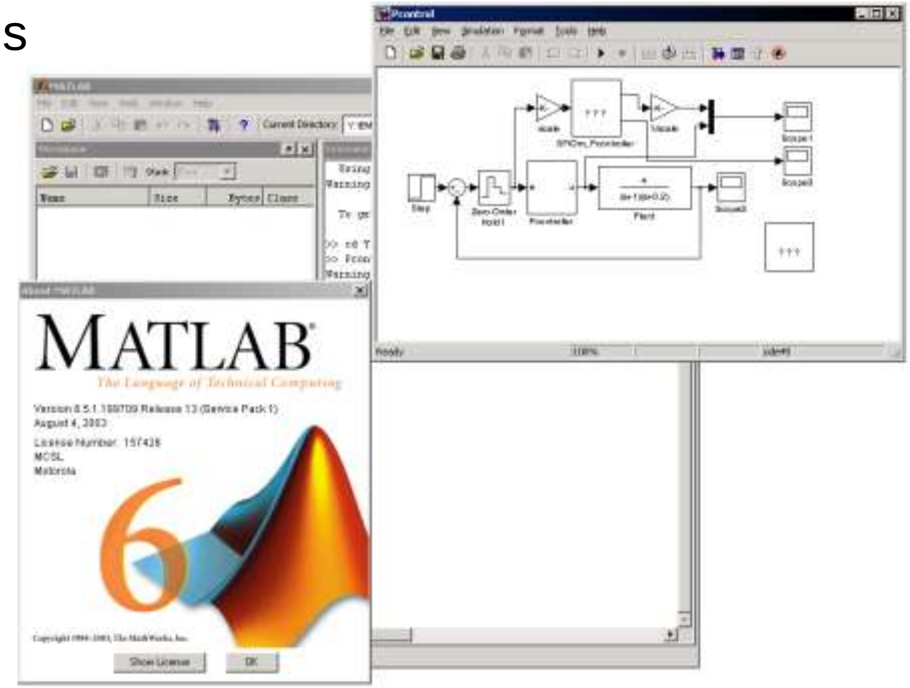
  – FreeMASTER itself implements an invisible ActiveX object
  – Script accesses the FreeMASTER functionality through this object
    - Variable access
    - Stimulator access
    - Application Commands
    - Recorder Data

- HTML may host whole applications, for example Excel
  – Excel Visual Basic macros may access FreeMASTER as well

FreeMASTER as a Graphical User Interface`

Target-in-loop Simulations

- FreeMASTER invisible ActiveX object is accessible also by external standalone applications
  - Standard C++ or VB applications
  - Excel & Visual Basic for Applications
  - Matlab, Simulink
- Target-in-loop Simulation
  - Matlab or Simulink engine lets embedded application to perform calculations



freescale™

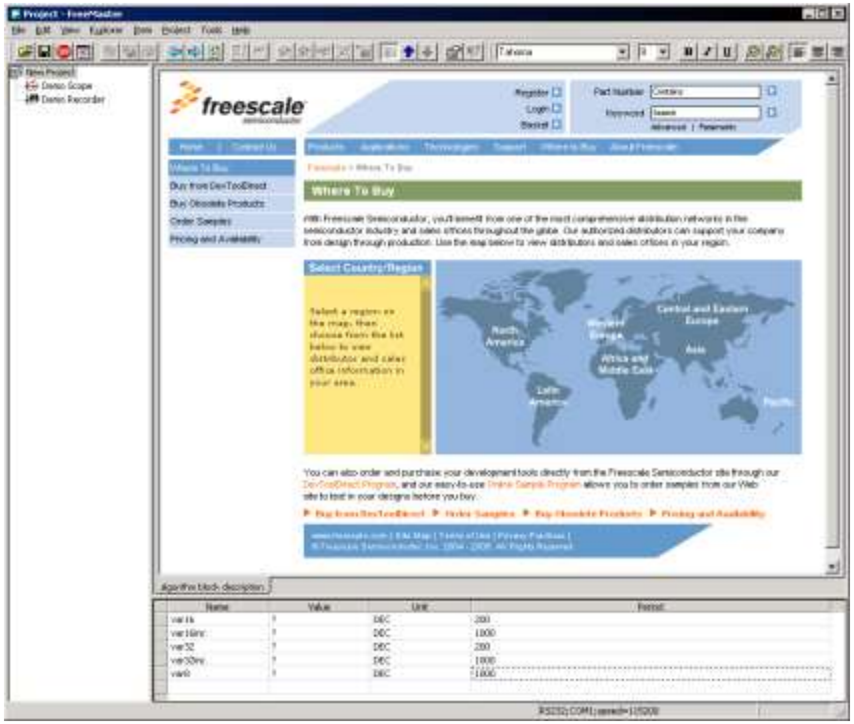
Tools - FreeMASTER

- What is FreeMASTER?
- Real-Time Monitor
- Graphical User Interface to the Embedded Application
- Demonstration Platform & Selling Tool

FreeMASTER as a Selling Tool

FreeMASTER helps Freescale Marketers to sell our work

- FreeMASTER project can visualize any detail of how the embedded application works
- HTML Pages embed text images, videos together with live application data
- FreeMASTER acts as a web-browser so it is possible to navigate to online shop directly without even leaving a FreeMASTER environment
- **FreeMASTER helps Freescale customers to sell their work**



FreeMASTER as a Selling Tool

FreeMASTER is Free!

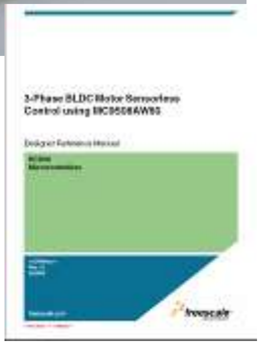
- The FreeMASTER is freely available from the Freescale web
- License agreement prevents using FreeMASTER with processors from competition
- Free redistribution enables Freescale customers to pack FreeMASTER with their products

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FREEMASTER&fsrch=1

Reference Design & etc.

IMM Motor Control / Power Conversion Team Focus

- Experienced team with15 years of motor control history
- Focusing on **Advanced Motor Control** and **Digital Power Conversion** for Industrial and Appliance – Freescale Centre of Excellence
- Covering all application specific products from 8-bit S08 up to16-bit DSC and 32-bit ColdFire & Kinetis)
- Providing global customer projects and support
- Developing
  - Demos
  - Reference designs
  - S/W Libraries
  - Application Notes
- Sharing the expertise's world wide (trainings, FAE support, exhibitions)
- Publishing research results at conferences world wide, covering the technology with patents
- Supporting NPI definition from application point of view



IMM Motor Control / Power Conversion Team Expertise's

- **Motor Control**
  - Running all kinds of 3-phase motors: ACIM, PMSM, BLDC, SR
  - Focus on advanced sensorless techniques (PMSM, SR)
  - Applications include washers, vacuum cleaners, dryers, dishwashers, fans, HVAC, compressors, etc.

- **Digital Power Conversion**
  - Switched Mode Power Supplies
  - Solar Panel Inverters
  - Uninterruptable Power Supplies
  - Light Ballast, PFC

- **NPI Support**
  - Supporting definition of new Freescale products inline with market requirements in motor control and power conversion area.
  - Integral part is the validation and application testing of new products

Developed by the Rožnov Motor Control team

DC motor
DC Motor with Speed and Current Closed Loops, driven by eTPU on MPC5554
Power Drill Control Software for MC68HC908QY4.

Universal motor
Open Loop PWM Control of Univ. Motor for Vacuum Cleaner using MC68HC908QT4

BrushLess DC Motor
BLDC Control using Kinetis
BLDC Control using Anguilla Black
BLDC Sensorless Control using MC56F8006
BLDC Sensorless Control using MCF51AG128
BLDC Sensorless Control using S08MP16 – ADC utilization
BLDC Sensorless Control using S08MP16 – Comaprators utilization
BLDC Sensorless Control using MC56F8013
BLDC Sensorless Control - very high speed – using MC56F8013
BLDC Control using MC9S08GT60 and MC33927
BLDC Sensorless Control using MC9S08AW60
BLDC Drive using DC/DC Inverter on MC56F8013
BLDC Control with Quadrature Encoder using DSP56F8346 - the PE solution
Low Power BLDC Drive for Fan using the MC68HC908QY4 MCU
High Voltage BLDC Drive for Domestic Appliances using MC68HC908MR8 MCU
BLDC Sensorless Control with BEMF Zero Crossing using MC68HC908MR32
BLDC Sensorless Control with BEMF Zero Crossing Using ADC for DSP56F805
Number of BLDC applications using TPU and eTPU

AC Induction Motor
Washing Machine 3-Phase ACIM Vector Control Based on MC56F8013
Washer 3-Phase ACIM Indirect Vector Control Based on MC56F8013
PWM Control of the Single-Phase ACIM Using the MC68HC908QT4 MCU
3-Ph. ACIM V/Hz Control using Hybrid Controller 56F8346 - the PE solution
3-Ph. ACIM Vector Control Using DSP56F80x
3-Ph. ACIM Vector Control with Single Shunt Current Sensing using 56F8013/23
3-Ph. ACIM Vector Control Using MPC555
3-Ph. ACIM Control V/Hz Application using MC68HC908MRxx
3-Ph. ACIM Control with Dead Time Distortion Correction using MC68HC908MR32
3-Ph. ACIM Volt Per Hertz Control System Based on DSP56F80x
Power Factor Correction for Motor Control Applications using 56F8013
DSP56F8xx Resolver Driver and Hardware Interface

Permanent magnet Synchronous Motor
Sensorless PMSM VC for appliance using DSC
Sensorless PMSM VC for appliance using Celis
PMSM VC with Encoder using Celis
PMSM VC with Encoder using Pictus
PMSM VC with Encoder using leopard
Sensorless PMSM VC with Sliding Mode Observer for Compressors using 56F8013
Permanent Magnet Synchronous Motor Vector Control, driven by eTPU on MCF523x
3-Phase PMSM Vector Control using MC56F8346
3-ph. PMSM Torque VC with Encoder and Resolver with MC56F80x/83xx (EPS Demo)
Electro-Mechanical Brake Demonstration Kit using PMSM motors
Synchronous PM Motor Control with Quadrature Encoder using DSP56F805
3-ph PM Synchronous Motor Torque Vector Control on DSP56F80x
3-Phase PM Synchronous Motor Vector Control using DSP56F80x
3-Phase PM Synchronous Motor Vector Control using DSP56F8013/23
3-Phase PM Synchronous Motor Vector Control using MCF51AC256
Sine Voltage Powered 3ph PM Synchronous Motor using MC68HC908MRxx
DSP56F8xx Resolver Driver and Hardware Interface

Stepper Motor
LIN-bus HID Lamp Levelling Stepper Motor Control Using MC908E625

Switched Reluctance Motor
3-Phase SR Motor Control with Hall Sensors Using DSP56F80x
3-Phase SR Sensorless Motor Control using DSP56F80x
Advanced 3-Phase SR Motor Control with Encoder Using DSP56F80x
Sensorless 2-phase SRM for Vacuum Cleaner using 56F8013

TPU and eTPU controlling motors
Four BLDC Motors Driven by One eTPU
3-Phase BLDC Motor Sensorless Control using MPC565
BLDC Motor with Speed Closed Loop driven by eTPU on MPC5554
DC Motor with Speed and Current Closed Loops, driven by eTPU on MPC5554
AC Induction Motor V/Hz Control, driven by eTPU on MCF523x
BLDC Motor with Quad. Enc. and Speed Closed Loop, driven by eTPU on MPC5554
3-Phase BLDC Motor with HS and Speed Closed Loop, driven by eTPU on MPC554
3 BLDC Motor Control with Hall Sensors driven by eTPU on MCF5235
Permanent Magnet Synchronous Motor Vector Control, driven by eTPU on MCF523x
TPU and eTPU Library Routines

Analogue support
Small Electric Vehicle with Analog DC Motor Driver (DMD)
3-phase Power Stage with DC/DC Inverter Lite using MC33883
3-Phase 12-Volt BLDC Power Stage with 33395 Driver

Specific Motor Control Hardware
Pictus Controller Board
Leopard Controller Board
Komodo Controller Boa
K40, Ang. Black, Ang. Blue / White, 51AG128, Leopard, S08MP16 – HV Power Stage card
MC56F8013/23/25 Controller Board
MC9S08AW60 Controller Board
MC56F8013/23 Controller Board
DSP56F802 Controller Board
MC56F8346 Controller Board
MC9S12E128 Controller Board
DSP56F805 Controller Board
Power Factor Correction Board
3-phase AC/BLDC High Voltage Power Stage Board
MC33927 Evaluation Board
3-phase Power Stage with DC/DC Inverter using MC33883
3-phase Micro Power Stage
3-Phase 12-Volt BLDC Power Stage with 33395 Driver
EVM Motor Board (3ph Low Voltage BLDC Power Stage)
3-Phase Low Voltage SR Power Stage
3-Phase High Voltage SR Power Stage
3-Phase Low Voltage AC/BLDC Power Stage
3-Phase High Voltage AC/BLDC Power Stage
Tower Power Stage

Specific Motor Control Software Libraries
TPU Library Routines for MPC555
eTPU Motor Control Libraries
Motor Control Libraries for 56F80xx
Motor Control Libraries for ColdFireV1
Motor Control Libraries for CortexM4
Motor Control Libraries for Pictus
Motor Control Libraries for Leopard / Komodo

3-ph. ACIM Vector Control Drive for Washer MC56F8013

Key Features

- Speed-close loop with PID controller
- Speed sensor on motor shaft (tachogenerator)
- Motor 3-phase currents reconstruction from DC-Bus current using single shunt sensor
- Rotor flux position evaluation from sensed currents and speed using rotor flux estimator
- Adaptive control circuit minimizes error of rotor flux estimator caused by motor parameter drift
- Motor current is decomposed into torque (Isq) and flux producing (Isd) components
- Field weakening algorithm controls excitation above nominal speed
- Space Vector Modulation is applied to generate output voltage
- Wide range of motor speed (0 – 18000 RPM)
- Washer algorithms implementation (tumble-wash, unbalance detection, spindry)
- FreeMASTER control interface

Description

This application demonstrates a direct vector control algorithm of a three-phase AC induction motor based on Freescale's MC56F8013 / MC56F8023 dedicated motor control devices.

The presented design is targeted mainly for consumer applications.

The cost-effective solution and high reliability are two key requirements considered. Minimizing system cost the algorithm implements a single-shunt current sensing eliminating three current sensors to one. High range of motor operating speed up to 18000RPM is another advantage of the presented design. Adaptive closed loop rotor flux estimator enhances control performance and increases overall robustness of the system.

The demo consists of the washing machine, controller board based on MC56F8013/23 and high voltage power stage.

Featured Products

- MC56F80xx

Key Markets

- Appliance (washers)

freescale™

BLDC Sensorless Drive – MC9S08MP16

Key Features

- Sensorless control of BLDC motor based on Back-EMF zero crossing sensing
- Targeted for the MC9S08MP16 Microcontroller
- Running on the 3-phase motor control drive universal low power board (24V) with MC9S08MP16 daughter board
- Using on-chip comparators for zero crossing sensing
- Closed-loop speed control with automatic current regulation and limitation
- Start from any motor position with rotor alignment
- Manual interface (Run / Stop switch, Up / Down push button control)
- FreeMASTER software control interface (motor run / stop, speed/torque set-up)
- FreeMaster software remote monitor

Description

This application is a 3-phase Brushless DC (BLDC) motor sensorless drive for fans, pumps and compressors. It is based on the low-cost Freescale MC9S08MP16 hybrid controller. The concept of the application is a closed-loop speed-controlled BLDC drive, with no need for position or speed sensors. It serves as a reference design for a BLDC motor sensorless control system, especially for fan, pump and compressor applications. Demo is based on 3-phase motor control drive universal low power board (24V) with MC56F8006 daughter board. Application uses an on-chip comparators for back-EMF zaro-crossing evaluation. A designer reference manual provides a detailed description of the application, including the design of the hardware and the software.

Featured Products

- MC9S08MP16
- MC33395 3-Ph. Pre-Driver

Key Markets

- Appliance (compressors, fans, HVAC, pumps)
- Industrial Drives

freescale™

Pancake PMSM Sensorless VC Demo – MC56F8013

Key Features

- Sensorless Vector Control of Panckake Permanent Magnet Synchronous Motor in whole speed range
- Application based on MC56F80XX digital signal controller
- 3-phase AC/BLDC High Voltage Power Stage with 1-ph. line input 110/230VAC @ 50/60Hz
- Pancake Permanent Magnet Synchronous Motor with AC Induction motor as a brake
- Initial position detection using high frequency injection
- standstill torque generation
- low speed operation using high frequency injection
- nominal speed operation using back-EMF observer
- Application based on C-callable library functions (GFLIB, GDFLIB, MCLIB, ACLIB)
- FreeMASTER based control pages
- Fault Protection

Description

Presented demo of sensorless control maintains the electric drive performance and requires no mechanical position or speed sensor. Application of this sensorless control allows generation throughout motor whole speed range starting from zero up to the nominal speed and even motor reversal is achievable. The control of PM motor is based on field oriented control with implemented speed control loop. This includes inner current control loop with implemented decoupling of cross-coupled variables achieving good torque control performance. Application is a single chip solution based on MC56F80xx digital signal controller series without any additional supportive circuitry. The demo consists of the pancake PMSM and motor load, control board based on MC56F8013/23 and high voltage power stage.

Featured Products

- MC56F80xx

Key Markets

- Appliance
    - V-axis washing machine
- Industrial Drives

freescale™

PMSM Sensorless Vector Control - 56F8023

Key Features

- Sensorless Control of Permanent Magnet Synchronous Motor based on Back-EMF Observer
- Application based on MC56F80XX digital signal controller
- 3-phase AC/BLDC High Voltage Power Stage with 1-ph. line input 110/230VAC @ 50/60Hz
- Industrial Permanent Magnet Synchronous Motor with braking mechanism
- Initial rotor position detection using high frequency injection
- Full torque at motor start-up
- Field weakening at high speeds
- Application based on C-callable library functions (GFLIB, GDFLIB, MCLIB, ACLIB)
- Current control loop execution time: 38us
- Speed control loop with Field weakening execution time : 11us
- Flash: ~ 6KB, RAM ~ 1.5KB
- FreeMASTER based control pages
- Fault Protection

Description

This application presents a motor control technique of permanent magnet motor (PM motor) without a need to use a rotor position transducer. This technique particularly targets horizontal axis (H-axis) washing machine with belt drive in fractional horsepower range

The PM motor control solution is based on field oriented control (FOC) with implemented speed control loop. This includes inner current control loop achieving good torque control performance. To maximize converter efficiency and minimize its rating, current loop. Even such sensorless control technique can be realized on low-cost 32-MIPS digital signal controller. Application is a single chip solution based on MC56F80XX digital signal controller series

The demo consists of the 3-phace PM motor, control board based on MC56F8025 and high voltage power stage.

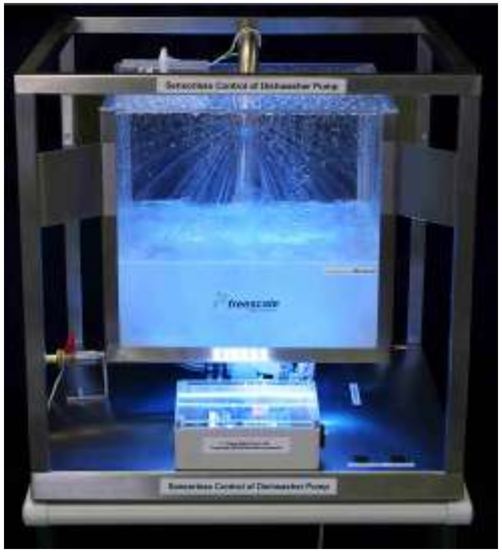
Featured Products

- MC56F80xx

Key Markets

- Industrial Drives
- Appliance

PMSM Sensorless VC for Dishwashers – MC56F8006



Key Features

- Sensorless Control of Permanent Magnet Synchronous Motor
- Control algorithm based on Back-EMF Observer tailored to dishwasher pump requirements
- Application based on MC56F8006 digital signal controller
- Low-cost 3-phase High Voltage Power Stage
- Dishwasher Permanent Magnet Synchronous Motor with water pump
- Typical pressure from 103 kPa (15 psi) to 827 kPa (120 psi) - speed range 1500-3500 rpm
- Fault Protection

Description

This application demonstrates a low cost dishwasher pump control solution. This new dishwasher pump employs a 3-phase Permanent Magnet Synchronous Motor (PMSM), which provides quieter, more efficient, and more reliable operation than previous solutions. The PMSM requires a more complex hardware and software solution than conventional universal AC motor based pumps. To minimize system cost, it is essential to design the most inexpensive drive possible. The extremely low cost Freescale MC56F8006 device is an ideal solution, allowing designers to build an effective drive for dishwasher pumps based on a sensorless algorithm that eliminates a relatively expensive position sensor. A back EMF observer tailored to the dishwasher pump motor is implemented here. It allows to control the dishwasher pump over required speed and torque range as required by the dishwasher application.

Featured Products
- MC56F8006

Key Markets
- Appliance (dishwashers, dryers)
- Industrial drives (pumps, etc.)
- Handheld power tools
- Medical devices & equipments

freescale™

PMSM Sensor / Sensorless Vector Ctrl - MCF51AC256

Key Features

- Vector control of PMSM using the Quadrature Encoder as a position sensor
- Vector control with speed closed-loop
- Two algorithms implemented:
    - Encoder based position and speed measurement
    - Sensorless position and speed estimation using Back-EMF Observer
- Start from any motor position (with rotor alignment)
- 4-quadrant operation
- 3-shunt current sensing
- Wide speed range
- FreeMASTER Control Interface
- Fault protection – over-current, over-voltage, under-voltage

Description

This application demonstrates an advanced design of a 3-phase Permanent Magnet (PM) synchronous motor drive that is controlled sensorless or using an encoder. It is based on Freescale Semiconductor's MCF51AC256 controller. The concept of the application is a speed closed loop PM synchronous drive using a Vector Control technique. It serves as an example of a PMSM control. The application uses the Freescale libraries (GFLIB, MCLIB, GDFLIB, ACLIB) that contained algorithms already compiled and optimized in assembler. This application utilizes a 3-phase power stage equipped with Freescale gate driver and a Freescale chip that creates a virtual COM port via USB for the Freemaster communication. The application contains very attractive graphical gauges web page control for the Freemaster software plus a lot of real time charts to explain the behavior of the system. .

Featured Products

- S08MRxxx

Key Markets

- Appliance
    - Diswasher pump drives
    - Washing machine
    - High-end pumps & Fans
- Industrial Drives

Tools and Software

Modular, expandable and cost-effective development platform TWR-56F84789-KIT

FreeMASTER

Allows control of an application remotely from a graphical environment running on a PC

CodeWarrior

Comprehensive IDE that provides a highly visual, automated framework to accelerate development of some of the most complex embedded applications

Motor Control Libraries

Market-focused software components increasing ease of use and helping decrease time to market

QEDesign

Complimentary filtering tool ideal for designing FIR and IIR filters

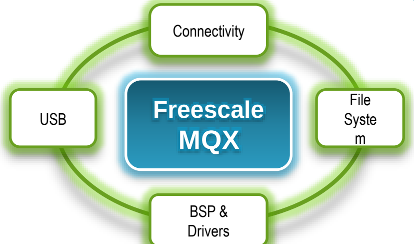
Processor Expert

Rapid application design tool that combines easy-to-use component-based application creation with an expert knowledge system

Reference Designs

Complimentary gerbers, code and schematics for:

- PMSM/BLDC motor control
- LLC resonant converter
- Solar power conversion



Accelerate design success with complimentary RTOS that is simple to fine-tune for custom applications and scalable to fit requirements

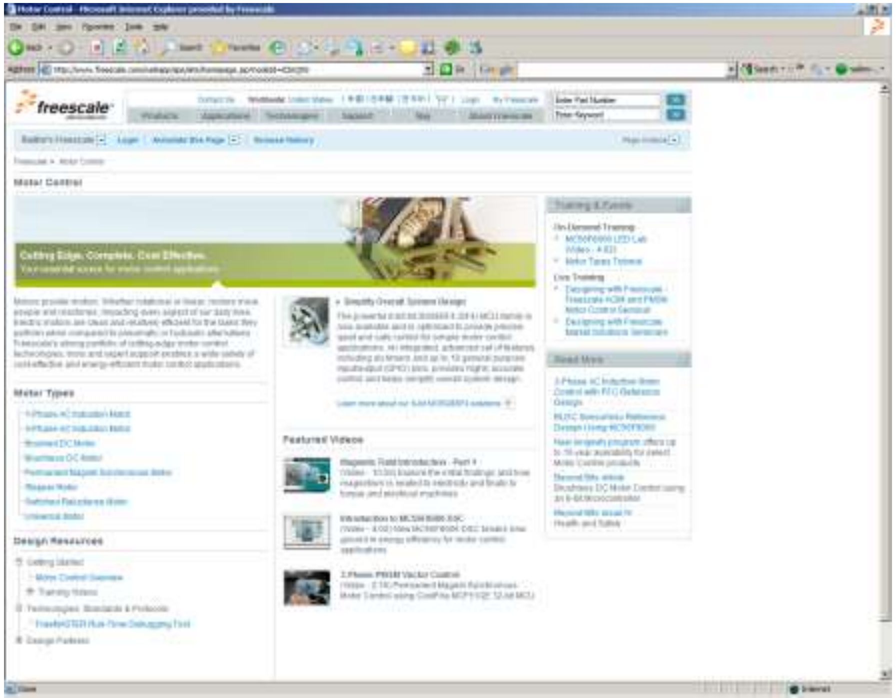
freescale™

Links of Motor Control Reference Designs

Motor Control materials available at:

For each motor types available:

- System description
- Typical applications
- Highlighted products
- Documentation (AN's, brochures)
- Reference designs
- HW tools
- SW tools



External Freescale Web (official doc)
www.freescale.com/motorcontrol

Resources

BLDC
http://www.freescale.com/webapp/sps/site/application.jsp?nodeId=02nQXG7C9C&code=APLBDCM&tab=Training_Support_Tab&aspll=1#ref_designs

3 PHASE AC Induction
http://www.freescale.com/webapp/sps/site/application.jsp?code=APLINDMOT&fasp=1&tab=Training_Support_Tab

1 PHASE AC Induction
http://www.freescale.com/webapp/sps/site/application.jsp?code=APLPHACIND&fasp=1&tab=Training_Support_Tab

PMSM
http://www.freescale.com/webapp/sps/site/application.jsp?code=APLPMSYNCMO&fasp=1&tab=Training_Support_Tab

STEP
http://www.freescale.com/webapp/sps/site/application.jsp?code=APLSTEMOT&fasp=1&tab=Training_Support_Tab

SRM
http://www.freescale.com/webapp/sps/site/application.jsp?code=APLSWRMOT&fasp=1&tab=Training_Support_Tab

Freescale Product Longevity Program

- Freescale has a longstanding track record of **providing long-term production support** for our products
- Freescale is pleased to offer a formal **product longevity program** for the market segments we serve
  - For the automotive and medical segments, Freescale will make a broad range of program devices available for a minimum of **15 years**
  - For all other market segments in which Freescale participates, Freescale will make a broad range of devices available for a minimum of **10 years**
  - Life cycles begin at the time of launch
- For terms and conditions and a list of participating **Freescale products** visit: www.freescale.com/productlongevity

freescale™

freescale™