AN1277
APPLICATION NOTE

ST92141 AC MOTOR CONTROL SOFTWARE LIBRARY VERSION 2.0 UPDATE

by Vincent ONDE / Microcontroller Division Applications and Dennis NOLAN / Power Supply Applications Lab

INTRODUCTION

In the continuous search for improvement, the Microcontroller Division of STMicroelectronics has investigated different ways of generating PWM sine waves using ST92141 MCUs. This document presents a new implementation of the AC motor control software from the library integration point of view.

Features of this second release of the ST92141 AC Motor Control Software Library will be discussed, as well as upgraded characteristics for applications based on version 1.0.

This new version dramatically reduces the amount of RAM memory needed to run the library, while keeping the CPU load at a reasonable level (approximately 15%). It also enables designers to drive single-phase or 2-phase AC motors using the same module interfaces as for 3-phase AC motors.

Important Note: As the V4 Toolchain is no longer upgraded or supported (does not work with Windows versions above Win98), it is now mandatory to start new developments with release 2.1.1 (or higher) of the software library working with the V6 Toolchain. Most MCU issues, software modules and function prototype descriptions described in this document remain valid.
However section 1.4.3 is obsolete as it refers to the V4 Toolchain.
Enhancements and upgrade issues related to Release 2.1.1 are described in application note AN1367.
The Application Note AN1084 “Getting Started with the ST92141 Software Library Version 1.0" remains required reading even for new developments directly based on Version 2.1.1.
The Version 2.1.1 Software Package is available on internet at http://www.st.com/stonline/products/support/mcu8/mc/st92141.htm#soft

Table of Contents

1 UPGRADED CHARACTERISTICS

1.1 NEW PWM GENERATION METHOD

For reference, RAM look-up tables were used to generate sine wave PWMs in version 1.0. The main task of PWM interrupts was to obtain PWM duty cycles from these tables and load them into the dedicated induction motor control peripheral. Sine wave tables were processed as a background task when necessary and a set of flags was required in PWM interrupts to indicate that the look-up table(s) had been changed. Although this method provides very low CPU loads at low speed, its main drawback is RAM consumption (around 300 bytes for 48-step sine tables).

In Version 2.0, sine waves are generated in real time during PWM interrupt service routines using a 256-step reference sine wave that is stored in ROM. Due to the powerful 8/16-bit architecture of the ST92141, a high frequency internal clock (25 MHz) and an optimized assembly PWM interrupt routine, the CPU load can be maintained at a reasonable level.

The following table summarizes performance changes between the two versions, assuming a PWM frequency (f_{PWM}) of 12.218 kHz and centred switching patterns.

	Version 1.0	**Version 2.0**	**Gain**
Register Use	99	50	49
RAM Use	365	77	288
ROM Use	5652	5087	565
CPU Load	3% @ 60 Hz 6% @ 127 Hz 13% @ 254 Hz < 15% above 254 Hz	15% constant	N/A
Frequency Quantification	0.1 to 0.5 Hz	0.2 Hz	N/A
Frequency Range	1.0 to 680 Hz	0.2 Hz to 680 Hz (theoretical max: $f_{PWM}/2$)[1]	N/A
Number of steps per sine period	Fixed, 48 or 12 at high speed	Variable, up to 256	N/A
Maximum Modulation	< 95% (at high speed)	100% regardless of the speed (including narrow pulse cancellation capability)	N/A

Note 1: Higher output frequencies can be produced using higher PWM switching frequencies to avoid sub-harmonic AC currents generation.

The spreadsheets that are required to compute the 256-step reference sine wave are now included with the library package, in the *utility* directory:

– sine.xls, using Excel (2000 Edition) format,

– sine.13 using Lotus 123 (Version 9) format.

1.2 90° AND 180° PHASE SHIFT OPERATION

Operations with sine wave phase shifts other than 120° are now possible. This is to allow the supply of single- and two-phase motors and Uninterruptible Power Supplies (UPS). In practice, this is done by conditional compilation with #define statements in the IMCParam.h file:

```
#define DEGREES_180 /* UPS and single-phase motors with starting capacitors */
#define DEGREES_120 /* Standard 3-phase AC motors */
#define DEGREES_90 /* Two-phase motors */
```

For 90° phase shifts, both the main and auxiliary (starting) motor windings are connected to the two half-bridges generating sine wave PWMs, their common point being supplied by:

– the third half-bridge, if mains rectification is achieved via a bridge (typically for nominal mains voltages of approximately 240 VAC),

– central point of rectification capacitors in voltage doubler configuration (typically for nominal mains voltages of approximately 120 VAC).

Note:
When compiling with #define DEGREES_180, the following warning message is displayed during the make process:

```
imcmodul.c(366): warning: comparison is always 0 due to limited range of data type
```

This message is generated by the *Direction IMC_GetRotationDirection(void)* function which does not make sense when generating single-phase voltages (the direction of rotation cannot be reversed). The above function will therefore always return a “Direction” type variable equal to the COUNTERCLOCKWISE_DIRECTION constant.

The entire contents of the “gmake” command output can be stored by re-directing this output to a file rather than on the screen. Instead of “gmake”, enter the following instruction:

```
gmake > out.txt
```

The “out.txt” file will contain all make process information.

1.3 ADT AND ZPC INTERRUPTS TASKS MODIFICATION

The ADT_IT and ZPC_IT interrupts in the IMC macrocell are now used to perform different tasks. Nevertheless, most of the information concerning IMC software time bases described in application note AN1084 (“Getting started with ST92141 Software Library Version 1.0”) still remain valid.

1.3.1 ZPC_IT

The Zero of PWM Counter (ZPC) interrupt occurs with every PWM cycle. In this interrupt routine, the PWM duty cycles are calculated for sine wave generation.

Active braking is also achieved in this routine, using the same method as described in version 1.0 (DC current injection).

This routine has been written in assembly for speed optimization reasons.

A flowchart of this interrupt is described in the appendix (Figure 1. and Figure 2.).

Duration: 10.5 µs with 25 MHz CPU clock.

1.3.2 ADT_IT

The Automatic Data Transfer (ADT) interrupt now occurs with every PWM cycle and replaces the ZPC interrupt described in version 1.0. This function provides a software time base by increasing several software timers every ms, with each timer being dedicated to a particular task.

In order to minimize the CPU load, this time base function can be replaced by other timer resources (the Extended Function Timer or the Standard Timer, if not used for the UART) having just one interrupt each 1 ms period. This can be done easily by clearing bit 4 (ADTE) of the IMC Interrupt Mask Register (IMR) in the IMC_Init function.

Duration: Average: 4.5 µs, Maximum: 10 µs with 25 MHz CPU clock.

1.4 SOURCE FILE MODIFICATIONS

Among the set of files delivered with version 1.0 of the library, the following 2 modules dedicated to sine wave generation have been modified for the new ACMOTOR library release.

The possible impacts of these changes on previously developed applications are listed in Section 2.

1.4.1 ACMOTOR Module

This module includes the following source files (the revision number of each individual file is also listed):

– acmotor.c (v2.0),

– acmotor.h (v1.0),

– acmparam.h (v1.0).

1.4.1.1 Obsolete Functions

The following 4 functions are no longer supported in this module:

```
u16 ACM_FreqToPeriod(u16 StatorFrequency);
void ACM_SetNewPWMFreq(u16 NewStatorFrequency, u16 NewPeriod);
void Calc_rpt_tab(void); (assembly routine)
void Calc_sin(void); (assembly routine)
```

These functions were closely linked to previous principle of sine wave generation and are now obsolete.

1.4.1.2 Modified Functions

A single function prototype has been modified:

```
BOOL ACM_Update_Sine_Tables (u8 NewVoltage, u16 NewStatorFrequency);
```

and is replaced by:

```
void ACM_Update_Sine_Tables (u8 NewVoltage, u16 NewFrequency)
```

It is no longer necessary to return a Boolean variable indicating if this function was achieved successfully, since this is always true.

Nevertheless, the following two functions have been modified, but keep the same interface (i.e. functionality is maintained but care must be taken if the previously delivered source files were re-used in a project, see Section 2.2):

```
void ACM_RampUp(u16 EndFrequency, u16 RampTime)
void ACM_SustainSpeed(u16 Time)
```

1.4.2 IMCMODUL Module

This module includes the following source files (the revision number of each individual file is also listed):

– IMCModul.c (v1.0)

– IMCModul.h (v1.0)

– IMCParam.h (v1.0)

1.4.2.1 New Functions

The following 4 functions have been created to meet the requirements of the new PWM generation method and to maintain a certain modularity between the IMCMODUL and ACMOTOR modules:

```
void IMC_Reset_Brake_Flag(void);
void IMC_Set_NeutralOffset(void);
void IMC_SetWaveform(WaveForm_t);
Direction IMC_GetRotationDirection(void);
```

These functions are described in Section 3.1.

1.4.3 MAKEFILE File

The makefile revision number is now 1.0.

This file has been modified since an assembly file is no longer needed other than the crt9.asm start-up file (the sine.asm file has been removed from the makefile's "ASM_List" sources list, see Section 1.4.5). All other files, settings (compiler, assembler and linker options) and utilities are maintained.

1.4.4 REGISTER File

The Reg_file.h revision number is now 1.0.

The mapping of registers has been modified (approximately half of library variables in version 1.0 have been deleted). Registers are now mapped as follows:

– Register group 0 (R0 to R15) is reserved for the C compiler,

– Register group 1 (R16 to R31) is reserved for PWM processing in the ZPC interrupt,

– Register group 2 and part of group 3 (R32 to R53) are reserved for the software library (miscellaneous use, including UART),

– The upper part of the register file (register group D and below) is still reserved for the system stack. System stack use must be carefully monitored during development to avoid overlapping with user-defined registers.

Assuming a system stack of 30 bytes (i.e. R223 down to R193) is used, this provides 138 user-free registers (R54 to R192).

1.4.5 SINE.ASM File

This file previously contained assembly routines used to fill RAM tables required for sine wave generation. As these routines are now obsolete, this file was removed from the set of library files.

2 UPGRADING VERSION 1.0 APPLICATIONS

This chapter is divided into 3 sections which should be studied depending on work performed using version 1.0 of this library:

– Source code was not modified and no new functions based on the example functions delivered with the library (such as ACM_RampUp, ACM_SustainSpeed) were written. Refer to Section 2.1.

– Some functions were modified or new functions were written based on example functions delivered with the library, except for sine wave generation routines. Refer to Section 2.2.

  – The sine wave generation source code was modified. Refer to Section 2.3.

2.1 DIRECT UPGRADE

In order to directly upgrade the 1.0 version of your application, the files listed above (except the *reg_file.h, acmparam.h* and *IMCParam.h* files) must be replaced with their corresponding files in version 2.0. The *sine.asm* file can be removed.

In addition:

– Registers must be re-mapped using the new *reg_file.h* baseline (registers R54 to R99 are now user-free),

– User parameters stored in prior versions of *acmparam.h* and *IMCParam.h* files must be replaced in the new version.

2.2 UPGRADE WITH POSSIBLE MINOR CHANGES ON APPLICATION

In order to directly upgrade the 1.0 version of your application with possible minor changes, the files listed above (except the *reg_file.h, acmparam.h* and *IMCParam.h* files) must be replaced with their corresponding files in version 2.0. The *sine.asm* file can be removed.

In addition:

– Registers must be re-mapped using the new *reg_file.h* baseline (registers R54 to R99 are now user-free),

– User parameters stored in prior versions of *acmparam.h* and *IMCParam.h* files must be replaced in the new version.

Tests using Boolean variables returned by ACM_Update_Sine_Tables must be removed in the applicable code in order to comply with the new version of this function. This is now pos-

sible since updates are now done on the fly (worst case latency being one PWM period). If a new update is required before the PWM interrupt, the previous one will simply be overwritten.

New functions developed using the ACM_RampUp and ACM_SustainSpeed functions as examples must be re-validated. The ACM_GetCurrentStatorFreq() function is now stored at the beginning of the module to determine the starting point of ramp or the stator frequency value to be sustained, rather than systematically called each time the Voltage variable is updated by the Regulation loop.

We recommend examining the source code of these two functions in order to have a clear idea of how the functions are now achieved (although there are a few slight changes) and compare them to the functions based on the examples given in version 1.0.

2.3 UPGRADE WITH POSSIBLE MAJOR CHANGES ON APPLICATION

In order to directly upgrade the 1.0 version of your application with possible major changes, the application upgrade must only be done if:

– The functions listed in Section 1.4.1.1 as being obsolete are not required,

– The application will greatly benefit from the increased performance of version 2.0, compared to that of version 1.0.

If the conditions listed above are met, the recommendations listed in Section 2.2 must be followed.

3 APPENDIX

3.1 NEW FUNCTIONS DOCUMENTATION

The prototypes functions listed below have been added to the IMCModul.h header file.

With the exception of the IMC_GetRotationDirection function, the main purpose of these functions is to setup the PWM sine wave generation in the ACM_Init function. A full description is provided in order to give user an exhaustive library reference and a full visibility over source code.

IMC_Reset_Brake_Flag

Synopsis

#include “IMCModul.h”

void IMC_Reset_Brake_Flag (void);

Description

The aim of this function is to reset the variable needed to enable active DC braking of the motor. This function is required in order to properly initialize the sine wave generation in the ACM_Init function.

See also

ZPC interrupt flowchart in Figure 1.

Note

This function must not be confused with the IMC_Brake_Disable function, which also disables PWM signals on dedicated outputs by imposing a fixed pattern. We recommend using the IMC_Brake_Disable function rather than the IMC_Reset_Brake_Flag to disable active braking.

IMC_Set_NeutralOffset

Synopsis

#include “IMCModul.h”

void IMC_Set_NeutralOffset(void);

Description

The aim of this function is to set the neutral point of sine waves to 50% of the duty cycle. This function is required in order to properly initialize the sine wave generation in the ACM_Init function.

Caution

We recommend not modifying this function unless the PWM frequency must be modified. This function is closely linked to CMP0 register setting (defining the PWM frequency), where setting an inconsistent value will result in a sine voltage imbalance and parasitic DC current injections.

IMC_SetWaveform

Synopsis

#include “IMCModul.h”

void IMC_SetWaveform(WaveForm_t);

Description

During sine wave generation interrupts, values from a reference wave form (sine for instance) stored in a 256 data array in ROM are accessed via a pointer to this array before being computed to get the desired voltage and frequency commands.

The IMC_SetWaveform function sets this pointer to the base address of the desired reference waveform. This function is required in order to properly initialize the sine wave generation in the ACM_Init function.

Inputs

Input value for reference wave shape selection is “typedef”-ed in IMC-Modul.h:

```
typedef enum
{
PURE_SINE
} WaveForm_t;
```

This a convenient way to avoid manual entry of #define instructions. Although at that time there is only a single type of shape available (sine), this could be extended easily in the future to include trapezoid shapes, fundamental plus third harmonic sine, etc.

Note

Default value if input is out of range is `PURE_SINE`.

IMC_GetRotationDirection

Synopsis

#include “IMCModul.h”

Direction IMC_GetRotationDirection(void);

Description

This function returns the current rotation direction (note that this information is relative, physical direction being imposed by motor phases hardware connections).

Returns

Output value for direction as only two values and is “typedef”-ed in IMCModul.h:

```
typedef enum
{
CLOCKWISE_DIRECTION,
COUNTERCLOCKWISE_DIRECTION
} Direction;
```

Note

As explained in Section 1.2, a warning message will appear when compiling with the “`#define DEGREES_180`” option.

3.2 ZPC INTERRUPT FLOWCHARTS

Figure 1. Interrupt Overview (Phase A, B and C Processing for 120° Phase Shift)

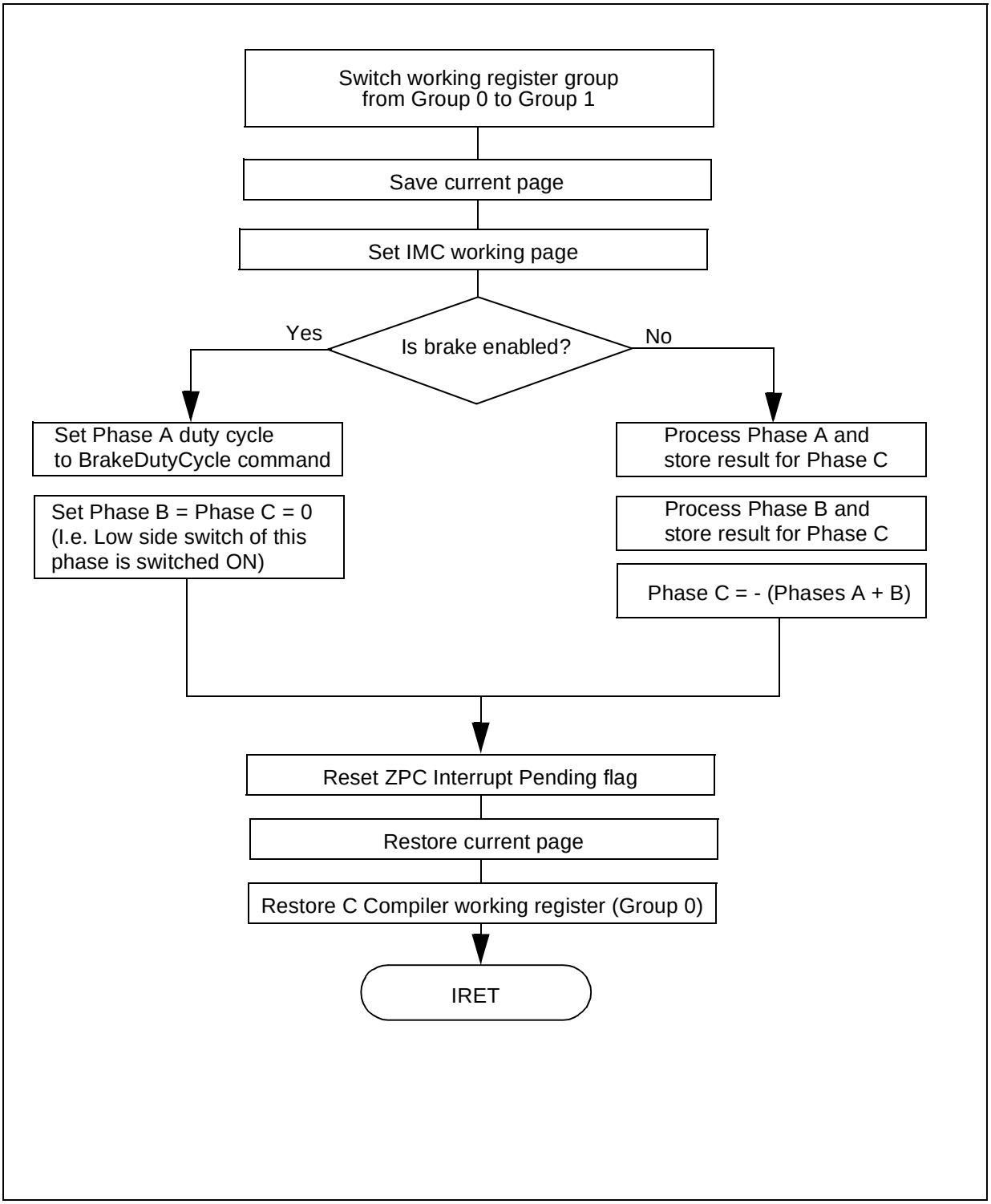
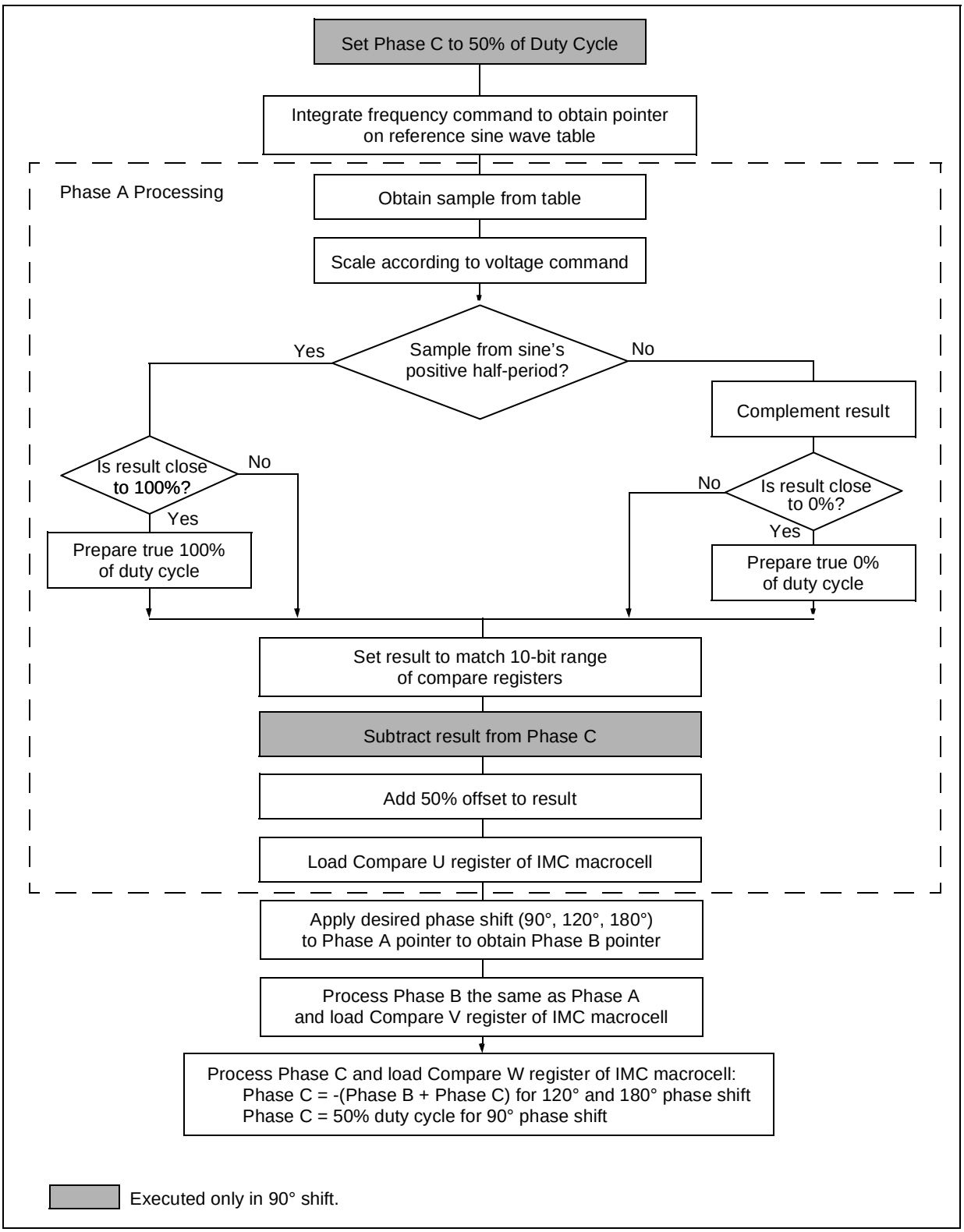


Figure 2. Details of Phase Duty Cycle Processing (including 90° Shift Situation)



THE PRESENT NOTE WHICH IS FOR GUIDANCE ONLY AIMS AT PROVIDING CUSTOMERS WITH INFORMATION REGARDING THEIR PRODUCTS IN ORDER FOR THEM TO SAVE TIME. AS A RESULT, STMICROELECTRONICS SHALL NOT BE HELD LIABLE FOR ANY DIRECT, INDIRECT OR CONSEQUENTIAL DAMAGES WITH RESPECT TO ANY CLAIMS ARISING FROM THE CONTENT OF SUCH A NOTE AND/OR THE USE MADE BY CUSTOMERS OF THE INFORMATION CONTAINED HEREIN IN CONNEXION WITH THEIR PRODUCTS."

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.