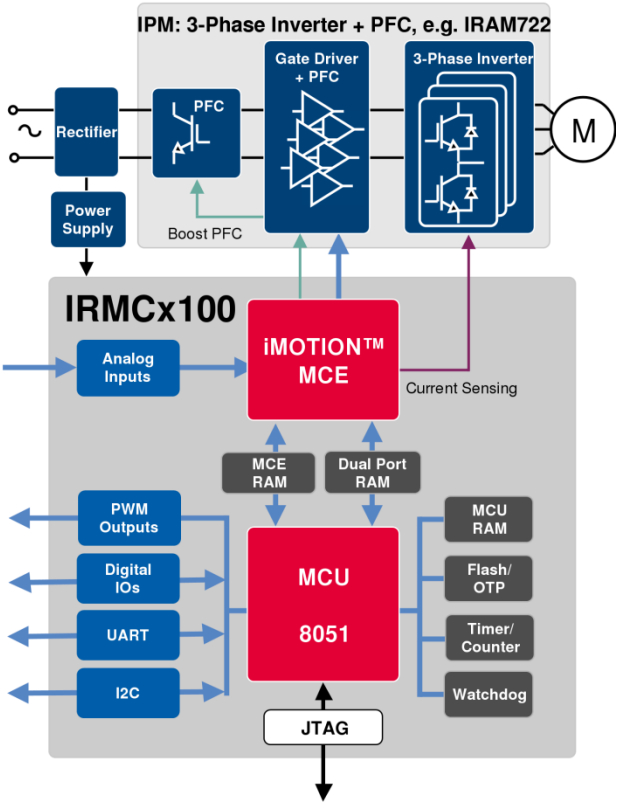
Reference Manual

iMOTION™ Motor Control IC integrating the Motion Control Engine (MCE) with a Microcontroller

Quality Requirement Category: Industry

Features

- Motion Control Engine (MCE) with 100MHz
  Field proven computation engine for high efficiency sinusoidal sensor less motor control
  - Ready-to-use solution for high efficiency variable speed drives
  - Sensor less field oriented control (FOC) of permanent magnet synchronous motors (PMSM)
  - Built-in support for single or two shunt current feedback
  - Integrated protection features
- MCU: 8051 with 30MHz
  - Built-in digital & analog peripherals: oscillator, A/D converter, OP amps & comparators
  - UART & I2C serial interfaces
  - Multiple additional analog & digital application IOs
  - JTAG interface
- Up to 64 KB flash or 32 KB OTP memory
- Packages from tiny 5x5mm² QFN-32 to QFP64



Applications

- Home appliances
- Pumps & fans
- Servo drives (IRMCx143)
- Any other PMSM drive

Description

The IRMCx100 series combines the iMOTION™ motion control engine (MCE) with an additional microcontroller (MCU) to improve application flexibility. The MCE does not require algorithm programming and can be combined with a µIPM™ or a discrete power stage for FOC control of a PMSM.

The industry standard 8051 MCU can be programmed by the customer to control the MCE and to implement additional functionality running almost independently from the motion control algorithm on the MCE.

IRMCx100
Reference Manual
RD-1002

March 25, 2015
Version 2.4

International
IOR Rectifier

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

TABLE OF CONTENTS

1. Overview 10
1.1 System Components 12
1.2 Memory Resources 13
1.2.1 OTP Memory (IRMCK1xx) 13
1.2.2 Flash Memory (IRMCF1xx) 13
1.2.3 Data RAM 13
1.3 Byte Ordering 14
1.4 Reset Mechanism and Boot Process 14
2 8051 Microcontroller 18
2.1 Instruction Set 18
2.2 Special Function Registers 22
2.3 Peripheral Registers 24
2.4 General Functions 26
2.4.1 Processor Registers 26
2.4.2 General Purpose I/O 28
2.4.3 Clock Selection and PLL Frequency Configuration 34
2.4.3.1 PLL Setup Register for IRMCK1xx 34
2.4.3.2 PLL Setup Register for IRMCF1xx 36
2.4.3.3 Boot Clock Operation 39
2.4.4 Sleep Mode Function 39
2.4.4.1 Sleep Mode for IRMCK1xx 39
2.4.4.2 Sleep Mode for IRMCF1xx 40
2.4.5 Miscellaneous Functions 40
2.5 Interrupts 43
2.5.1 Standard Interrupts 43
2.5.2 Extended Interrupts 44
2.5.3 Enabling Interrupts 44
2.5.4 Interrupt Priority 45
2.5.5 Service Order 46
2.5.6 Interrupt Latency 47
2.5.7 Interrupt Vectors 47
2.6 Timers 47
2.6.1 Timer Prescaler 47
2.6.2 General-Purpose Timer/Counters 48
2.6.2.1 Modes of Operation 50
2.6.2.2 Configuring the Timers 50
2.6.2.3 Using the Timers to Measure a Time Interval 50
2.6.2.4 Using the Timers to Signal when a Defined Period has Elapsed 51
2.6.2.5 Using the Timers as Event Counters 51
2.6.2.6 Reading the Timers 51
2.6.3 Watchdog Timer 52
2.6.4 Capture Timer 53
2.6.5 MCE Pin Timers 54
2.7 UART 58
2.8 D/A PWM 60
2.9 I^2C / SPI Serial Interface 62
2.9.1 Command Descriptions for the I^2C Interface 65
2.9.1.1 Device Address 65
2.9.1.2 Read and Write Commands 65
2.9.2 Command Descriptions for the SPI Interface 66
2.9.2.1 Read Instructions 66
2.9.2.2 Write Instructions 67
3 Motion Control Engine 68

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.1 Rotating Frame Notation and Conventions 70
3.2 Control Blocks 71
3.2.1 Frequency Domain Blocks 71
3.2.1.1 PI – Proportional Plus Integral 71
3.2.1.2 LOWPASS_FILT – First Order Low Pass Filter 74
3.2.1.3 HIGHPASS_FILT – First Order High Pass Filter 75
3.2.2 Coordinate Transformation Blocks 78
3.2.2.1 VECROT – Vector Rotation 78
3.2.2.2 CLARK – Inverse Clark Transformation 79
3.2.3 Utility Blocks 81
3.2.3.1 LIMIT 81
3.2.3.2 RAMP – Linear Ramp 82
3.2.3.3 ATAN – Arc Tangent block 84
3.2.3.4 FUNCTION_BLOCK 85
3.2.3.5 COMPARATOR 87
3.2.3.6 SWITCH 87
3.2.3.7 BIT_LATCH 88
3.2.3.8 PEAK_DETECT 89
3.2.3.9 TRANSITION – One Shot Pulse Generator 90
3.2.3.10 INTEGRAL2 – Integral with Limit 91
3.2.4 Math 93
3.2.4.1 DIFF – Subtraction 93
3.2.4.2 SUM – Addition 94
3.2.4.3 ACCUMULATOR 95
3.2.4.4 COUNTER 96
3.2.4.5 SHIFT – Multiply by a Power of Two 97
3.2.4.6 MUL_DIV – Signed / Unsigned Multiplier with Extraction 97
3.2.4.7 DIVIDE 99
3.2.4.8 NOT – Bitwise Inversion 100
3.2.4.9 NEGATE – Two's Complement 101
3.2.4.10 AND – Bitwise Logical AND 101
3.2.4.11 OR – Bitwise Logical OR 102
3.2.4.12 XOR – Bitwise Logical Exclusive OR 102
3.3 Motion Peripherals 103
3.3.1 SENSORLESS_FOC 103
3.3.1.1 Current Control 106
3.3.1.2 Rotor Position Estimation 106
3.3.1.3 Startup Control 108
3.3.1.4 Sequencing 108
3.3.1.5 Catch Spin 108
3.3.1.6 Diagnostic Modes 112
3.3.2 CURRENT_MEAS 113
3.3.2.1 Motor current sampling 113
3.3.3 DC_BUS_VOLTAGE 115
3.3.4 A_D – A/D Converter and A/D Compensation 116
3.3.4.1 AIN5 119
3.3.4.2 AIN6 (Avref/2) 119
3.3.4.3 Scheduling of General Purpose A/D Channels 119
3.3.4.4 A/D Compensation Function 120
3.3.5 Low Loss Space Vector PWM 122
3.3.5.1 SVPWM Transfer Characteristics 125
3.3.5.2 Deadtime Insertion Logic 126
3.3.5.3 Three-Phase and Two-Phase Modulation 126
3.3.5.4 PWM Pre-Charge Control 127
3.3.5.5 Gatekill 128

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.6 FAULTS Block 130
3.3.6.1 Fault State Control 132
3.3.7 MCE_FAULT Generator 133
3.3.8 PFC_PWM 133
3.3.8.1 PFC PWM Generation 135
3.3.8.2 PFC PWM Blanking 136
3.3.9 PFC_SENSE 137
3.3.10 PFC_AC_STATUS 138
3.3.11 PFC_MOTOR_MISC 139
3.4 Motion Peripheral Registers 140
3.4.1 System Write Register Group 141
3.4.2 PWM Configuration Write Register Group 145
3.4.3 Torque Loop Configuration Write Register Group 149
3.4.4 Velocity Control Write Register Group 151
3.4.5 Closed Loop Angle Estimator Write Register Group 152
3.4.6 Open Loop Angle Estimator Write Register Group 157
3.4.7 Startup Angle Estimator Write Register Group 158
3.4.8 Phase Loss Detect Write Register Group 161
3.4.9 Current feedback Write Register Group 162
3.4.10 Catch Spin Write Register Group 166
3.4.11 User Control Write Register Group 168
3.4.12 Field Weakening Control Write Register Group 170
3.4.13 Protection Write Register Group 171
3.4.14 External Signals Write Register Group 172
3.4.15 AD Correction Write Register Group 174
3.4.16 Miscellaneous Signals Write Register Group 175
3.4.17 PFC Control Write Register Group 176
3.4.18 System Read Register Group 180
3.4.19 System Status Read Register Group 185
3.4.20 DC Bus Voltage Read Register Group 186
3.4.21 FOC Diagnostic Data Read Register Group 187
3.4.22 Velocity Status Read Register Group 191
3.4.23 Current Feedback Offset Read Register Group 191
3.4.24 Protection Read Register Group 192
3.4.25 AD Correction Read Register Group 193
3.4.26 PFC Status Read Register Group 194
4 8051 / MCE Interface 197
4.1 Memory Map 197
4.2 IRMCK1xx OTP Memory 200
4.3 IRMCF1xx Flash Memory 202
4.4 The Shared Data RAM 203
4.4.1 Reading and Writing Shared RAM 203
4.4.2 Arbitration 204
4.5 MCE Processor Registers 205
4.6 Interrupts from the MCE to the 8051 206
4.6.1 MCE Interrupt 206
4.6.2 SYNC Interrupt 206
4.6.3 Fault Interrupt 207
5 The MCE Development Process 208
5.1 MCE Design Generation 208
5.2 Creating an MCE Design Using Simulink 209
5.3 Simulink MCE Design Components 210
5.3.1 The MCE Library 210
5.3.1.1 Configuration 210
5.3.1.2 Registers 210

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

5.3.1.3 Control 210
5.3.1.4 Math 210
5.3.1.5 Tools 210
5.3.1.6 Motion Peripherals 211
5.3.1.7 Designs 211
5.3.2 Standard Simulink Library Components 211
5.3.2.1 Enabled Subsystem 211
5.3.2.2 Constant 211
5.3.2.3 Scope 211
5.3.2.4 Goto and From 211
5.3.2.5 Unit Delay 211
5.4 The MCE Compiler 212
5.5 The Host Register Summary Utility 214
5.6 Customizing Motion Peripheral Library Blocks 215
5.7 Providing Information about Compatible Firmware Options 217
5.8 MCE Design Hierarchical Format 218
6 The 8051 Development Process 221
6.1 The Keil and FS2 Tools 221
6.1.1 Software Installation 221
6.1.2 The Keil Compiler 221
6.1.3 Debugging 222

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

LIST OF FIGURES

Figure 1. Typical Two Shunt Leg Current Sensing Application Diagram Using IRMCx100 10
Figure 2. Typical Application of IRMCx188 with Single Shunt Current Sensing and PFC 11
Figure 3. IRMCK171 Internal Block Diagram 11
Figure 4. IRMCK143 Internal Block Diagram 12
Figure 5. Reset Module 15
Figure 6. Reset and Boot Process 16
Figure 7. Port Structure of IRMCx100 Series 29
Figure 8. Clock Selection Diagram 34
Figure 9. GATEKILL Internal Logic 42
Figure 10. Watchdog Timer 52
Figure 11. Capture Timer Operation 53
Figure 12. D/A PWM Output 61
Figure 13. I^2C Pin Structure 66
Figure 14. PI Block 71
Figure 15. LOWPASS_FILT Block 74
Figure 16. LOWPASS_FILT Frequency Response 75
Figure 17. HIGHPASS_FILT Block 76
Figure 18. HIGHPASS_FILT frequency response 77
Figure 19. VECROT Block 78
Figure 20. VECROT Vector Interpretation 78
Figure 21. CLARK Block 79
Figure 22. CLARK Vector Interpretation 80
Figure 23. LIMIT Block 81
Figure 24. RAMP Block 82
Figure 25. ATAN Block 84
Figure 26. FUNCTION_BLOCK Block 85
Figure 27. FUNCTION_BLOCK Example 86
Figure 28. COMPARATOR Block 87
Figure 29. SWITCH Block 87
Figure 30. BIT_LATCH Block 88
Figure 31. PEAK_DETECT Block 89
Figure 32. TRANSITION Block 90
Figure 33. INTEGRAL2 Block 91
Figure 34. DIFF Block 93
Figure 35. SUM Block 94
Figure 36. ACCUMULATOR Block 95
Figure 37. COUNTER Block 96
Figure 38. SHIFT Block 97
Figure 39. MUL_DIV Block 97
Figure 40. DIVIDE Block 99
Figure 41. NOT Block 100
Figure 42. NEGATE Block 101
Figure 43. AND Block 101
Figure 44. OR Block 102
Figure 45. XOR Block 102
Figure 46. SENSORLESS_FOC Block 103
Figure 47. SENSORLESS_FOC Block Diagram 107
Figure 48. Drive Control Modes 108
Figure 49. Sequencer of IRMCx100. 110
Figure 50. State Handling of Catch Spin Algorithm in IRMCx100 111
Figure 51. CURRENT_MEAS Block 113
Figure 52. Single Current Shunt Registers (TCntMin2Phs, TCntMin3Phs) 114
Figure 53. Leg Shunt Register (PwmGuardBand). 114

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Figure 54. DC_BUS_VOLTAGE Block 115
Figure 55. A_D_171 Block 116
Figure 56. A_D_143 Block 116
Figure 57. IRMCx100 Series A/D Converter Structure 118
Figure 58. Single Shunt Current Sense Timing 119
Figure 59. Scheduling of General Purpose A/D Channel 120
Figure 60. Calculation of A/D correction parameters 121
Figure 61. Low Loss SVPWM Block 122
Figure 62. SVPWM Internal Block Diagram 124
Figure 63. Space Vector Diagram 125
Figure 64. Voltage Vector Rescaling 126
Figure 65. Deadtime Insertion 126
Figure 66. Three-Phase and Two-Phase Modulation 127
Figure 67. Types of Space Vector PWM 127
Figure 68. Bootstrap Pre-Charge Sequence 128
Figure 69. Timing of Bootstrap Capacitor Charging 128
Figure 70. Internal Gatekill Generation 129
Figure 71. FAULTS Block 130
Figure 72. State Control of Fault Handling 132
Figure 73. MCE_FAULT Block 133
Figure 74. PFC_PWM Block 133
Figure 75. PFC_PWM Internal Block Diagram 135
Figure 76. Generation of PFC PWM Output and ADC Timing 136
Figure 77. PFC_SENSE Block 137
Figure 78. PFC_ AC_STATUS Block 138
Figure 79. AC Over/Under Voltage Detection 138
Figure 80. PFC_MOTOR_MISC 139
Figure 81. Calculation of VdcRcp 150
Figure 82. Detail Scaling of PLL 156
Figure 83. Measurement of AIN6 (AVREF/2) 181
Figure 84. Implementation of AD Compensation 193
Figure 85. Memory Map of IRMCK100 Series 198
Figure 86. Memory Map of IRMCF100 Series 199
Figure 87. OTP Memory Usage for IRMCK100 Series 200
Figure 88. Flash Memory Usage for IRMCF100 Series 202
Figure 89. Timing of Sync and MCE Computation 206
Figure 90. MCE Simulink Library 210
Figure 91. MCE Compiler Input Screen 212
Figure 92. Example Firmware Selection Window 213
Figure 93. MCE Compiler Results Example 214
Figure 94. The Host Register Summary Utility 215
Figure 95. The CustomMotPer Utility 216
Figure 96. Example Model Info Block Showing Two Firmware Options 217
Figure 97. MCE Design Hierarchy 219

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

LIST OF TABLES

Table 1. IRMCx100 Products 12
Table 1. Access to Memory Resources 13
Table 2. 8051 Instructions 21
Table 3. Special Function Register Memory Map 22
Table 4. Special Function Registers 24
Table 5. Peripheral Registers 26
Table 6. Discrete I/Os from Ports 1, 2, 3 and 5 28
Table 7. Interrupt Source Summary 43
Table 8. Interrupt Service Order 47
Table 9. Registers for I^2C 62
Table 10. Registers for SPI 62
Table 11. MCE Library Elements 69
Table 12. PI User Inputs and Outputs 73
Table 13. PI System Inputs and Outputs 73
Table 14. PI Execution Time 73
Table 15. LOWPASS_FILT User Inputs and Outputs 75
Table 16. LOWPASS_FILT Execution Time 75
Table 17. HIGHPASS_FILT User Inputs and Outputs 77
Table 18. HIGHPASS_FILT Execution Time 77
Table 19. VECROT Inputs and Outputs 78
Table 20. VECROT Execution Time 79
Table 21. CLARK Inputs and Outputs 80
Table 22. CLARK Execution Time 80
Table 23. LIMIT Inputs and Outputs 81
Table 24. LIMIT Execution Time 81
Table 25. RAMP User Inputs and Outputs 83
Table 26. RAMP Execution Time 83
Table 27. ATAN Inputs and Outputs 84
Table 28. ATAN Execution Time 84
Table 29. FUNCTION_BLOCK Inputs and Outputs 86
Table 30. FUNCTION_BLOCK Execution Time 86
Table 31. COMPARATOR Inputs and Outputs 87
Table 32. COMPARATOR Execution Time 87
Table 33. SWITCH Inputs and Outputs 88
Table 34. SWITCH Execution Time 88
Table 35. BIT_LATCH User Inputs and Outputs 89
Table 36. BIT_LATCH Execution Time 89
Table 37. PEAK_DETECT User Inputs and Outputs 90
Table 38. PEAK_DETECT Execution Time 90
Table 39. TRANSITION User Inputs and Outputs 91
Table 40. TRANSITION Execution Time 91
Table 41. INTEGRAL2 User Inputs and Outputs 92
Table 42. INTEGRAL2 Execution Time 92
Table 43. DIFF Inputs and Outputs 93
Table 44. DIFF Execution Time 93
Table 45. SUM Inputs and Outputs 94
Table 46. SUM Execution Time 94
Table 47. ACCUMULATOR User Inputs and Outputs 95
Table 48. ACCUMULATOR Execution Time 96
Table 49. COUNTER User Inputs and Outputs 96
Table 50. COUNTER Execution Time 96
Table 51. SHIFT Inputs and Outputs 97
Table 52. SHIFT Execution Time 97

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Table 53. MUL_DIV Inputs and Outputs 98
Table 54. MUL_DIV Execution Time 99
Table 55. DIVIDE Inputs and Outputs 100
Table 56. DIVIDE Execution Time 100
Table 57. NOT Inputs and Outputs 100
Table 58. NOT Execution Time 100
Table 59. NEGATE Inputs and Outputs 101
Table 60. NEGATE Execution Time 101
Table 61. AND Inputs and Outputs 101
Table 62. AND Execution Time 101
Table 63. OR Inputs and Outputs 102
Table 64. OR Execution Time 102
Table 65. XOR Inputs and Outputs 102
Table 66. XOR Execution Time 102
Table 67. SENSORLESS_FOC Available Inputs and Outputs 105
Table 68. CURRENT_MEAS available Inputs and Outputs 113
Table 69. DC_BUS_VOLTAGE Outputs 115
Table 70. A_D_100 Outputs 117
Table 71. LOWLOSS_SVPWM Available Inputs and Outputs 123
Table 72. FAULTS Block Outputs for IRMCx100 Series 130
Table 73. Summary of Fault Handling 131
Table 74. Relationship between System State and Active Fault Functions 132
Table 75. MCE_FAULT Block Inputs 133
Table 76. PFC_PWM Inputs and Outputs 134
Table 77. PFC_SENSE Outputs 137
Table 78 . PFC_AC_STATUS Outputs 138
Table 79. PFC_MOTOR_MISC Outputs 139
Table 80. MCE Processor Registers 205

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

1. Overview

IRMCK143,IRMCF143,IRMCF188,IRMCF183,IRMCK182,IRMCK172, IRMCK171 and IRMCF171 are components of a new family of International Rectifier integrated circuit devices, collectively called the IRMCx100 series. These devices are primarily designed as one-chip solutions for complete inverterized appliance motor control applications. Unlike a traditional microcontroller or DSP, the IRMCx100-series devices provide a built-in closed loop sensorless control algorithm using the unique Motion Control Engine (MCE™) for permanent magnet motors. The MCE™ consists of a collection of control elements, motion peripherals, a dedicated motion control sequencer and dual port RAM to map internal signal nodes. These devices also employ a unique single shunt current reconstruction circuit, in addition to two leg shunt resistor current sensing, to eliminate additional analog/digital circuitry and enable a direct shunt resistor interface to the IC. The motor can be either a permanent magnet motor or an induction motor depending on application needs, with a common control structure: Sensorless Field Oriented Control. Figure 1 shows a typical application schematic of sensorless control with two shunt current sense resistor while Figure 2 shows sensorless control with a single shunt resistor current sense configuration with power factor correction (PFC). Motion control programming is achieved using a dedicated graphical compiler integrated into the MATLAB/Simulink™ development environment. Sequencing, user interface, host communication, and upper layer control tasks can be implemented in the 8051 high-speed 8-bit microcontroller. The 8051 microcontroller is equipped with a JTAG port to facilitate emulation and debugging tools.

The IRMCK100-series devices contain 32K bytes of One Time Programmable (OTP) memory and 16K bytes RAM for instruction storage, dual port RAM for MCE-8051 communication, and local 8051 RAM. The IRMCF100-series devices replace the 32K-byte OTP memory with a 64K-byte flash memory.

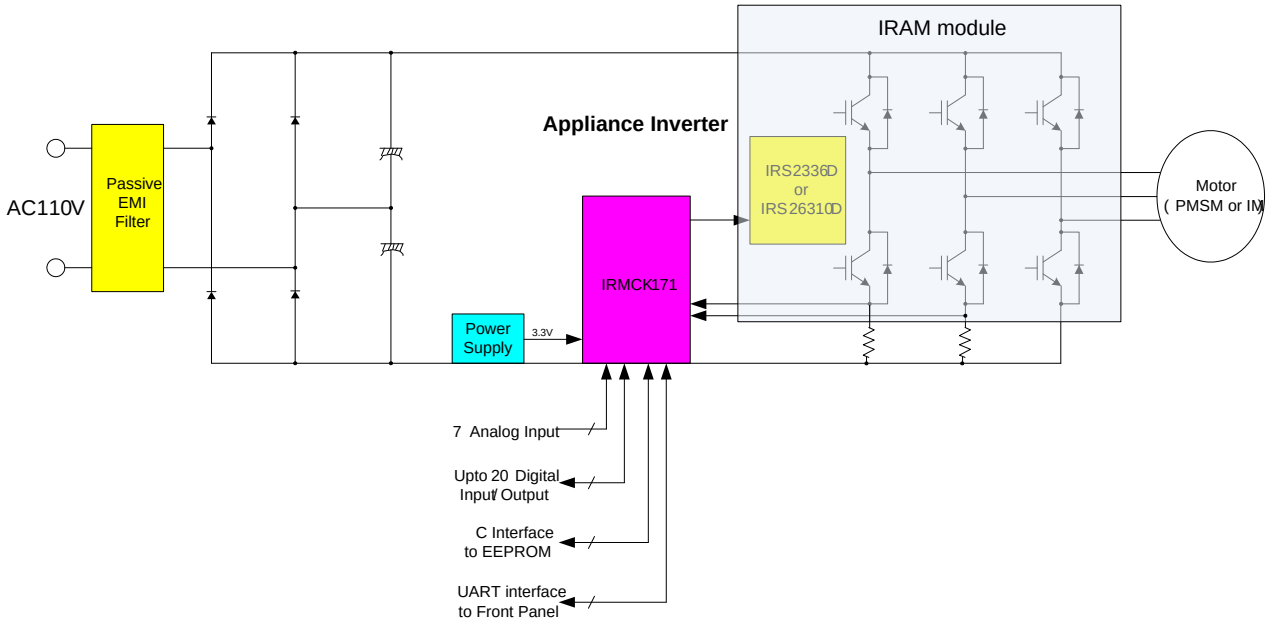


Figure 1. Typical Two Shunt Leg Current Sensing Application Diagram Using IRMCx100

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

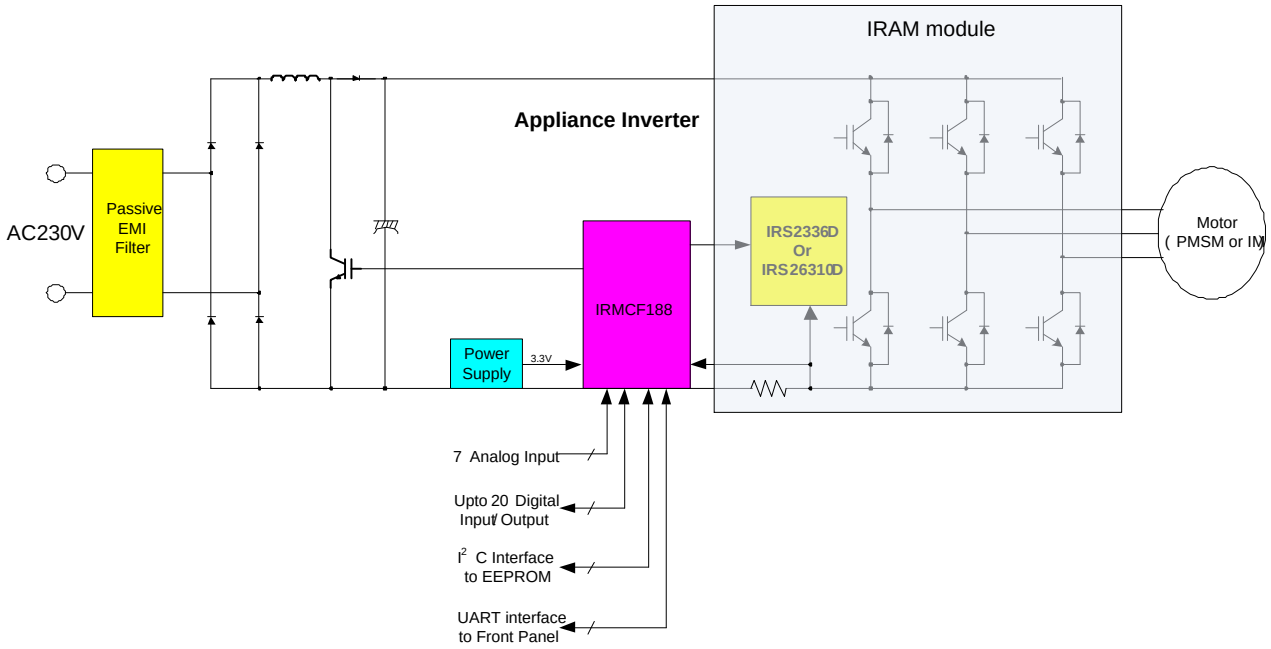


Figure 2. Typical Application of IRMCx188 with Single Shunt Current Sensing and PFC

A block diagram of the IRMCK171 is shown in Figure 3. The IRMCF171 is functionally equivalent, with the 32K-byte OTP memory replaced by a 64K-byte flash memory.

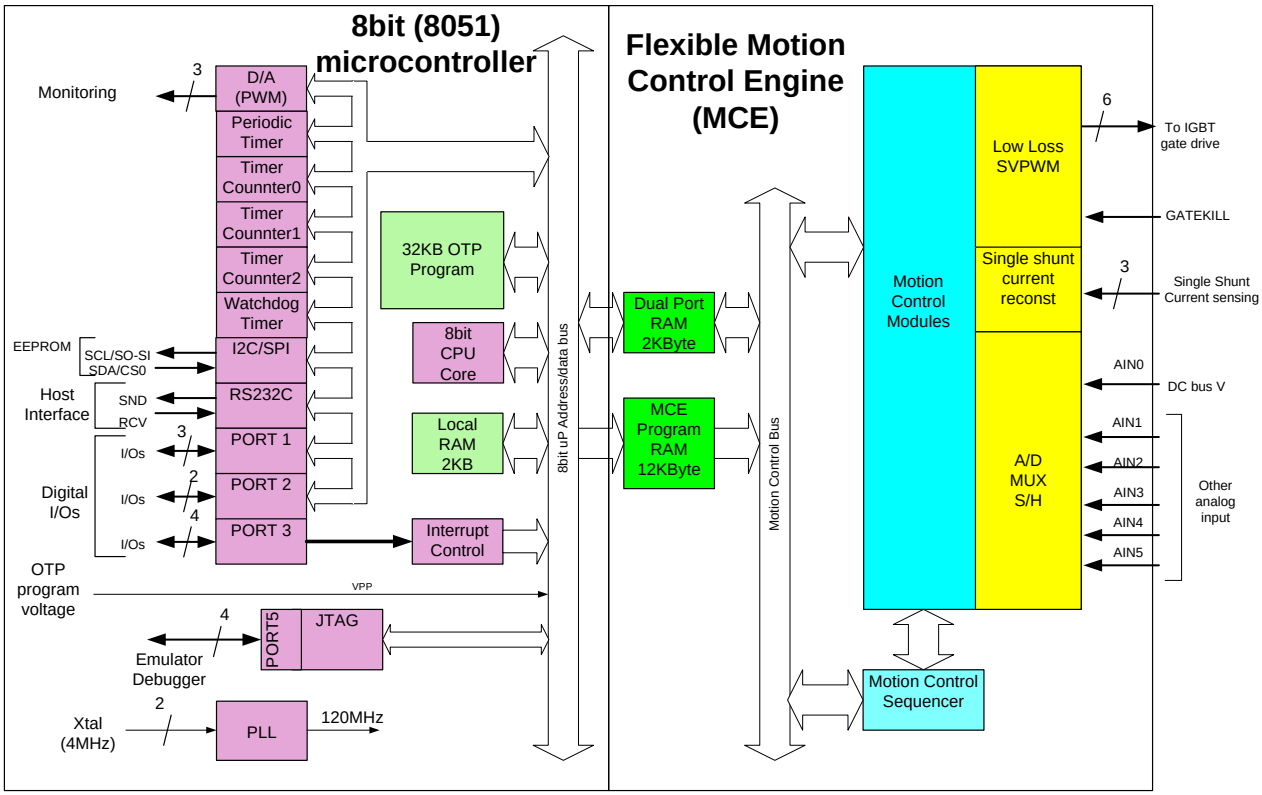


Figure 3. IRMCK171 Internal Block Diagram

A block diagram of the IRMCK143 is shown in Figure 4. The IRMCF143 is functionally equivalent, with the 32K-byte OTP memory replaced by a 64K-byte flash memory.

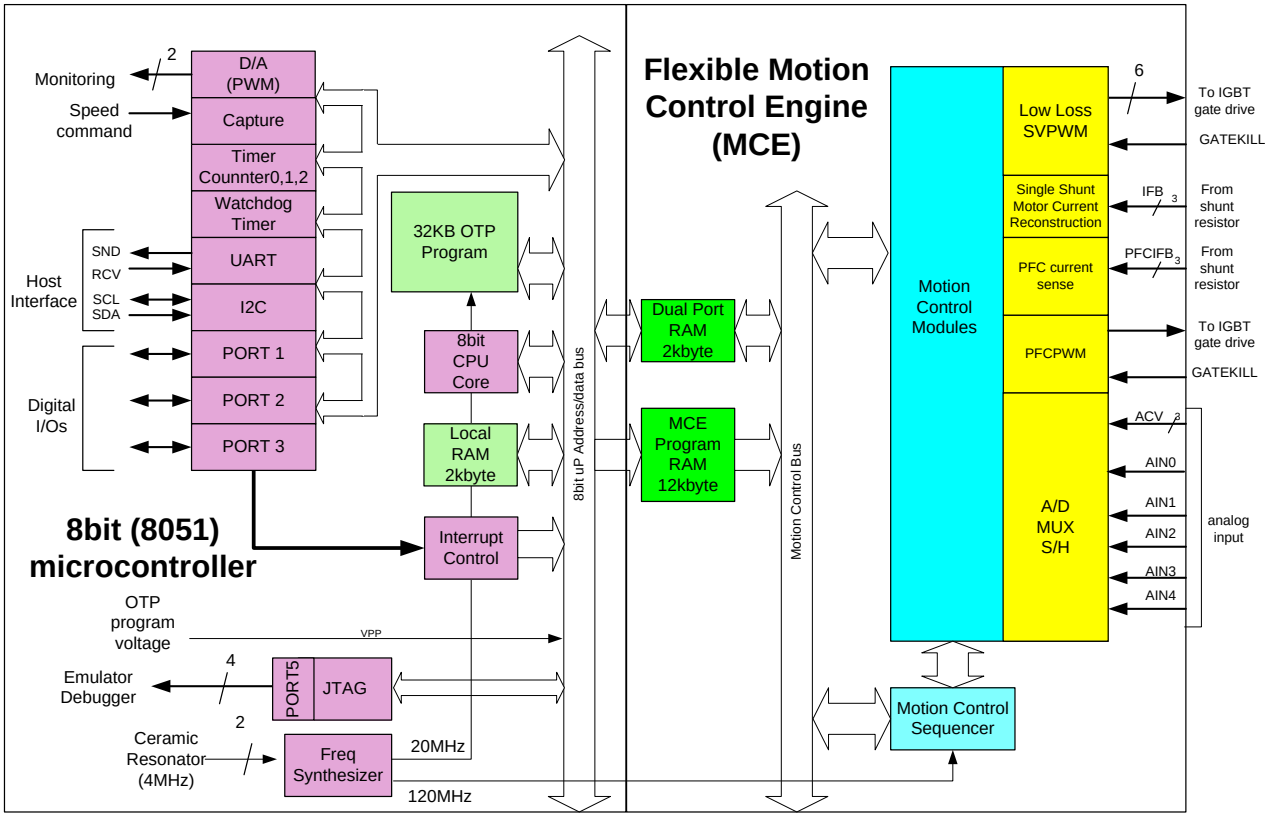


Figure 4. IRMCK143 Internal Block Diagram

Various products offering in 100 series is listed in the table below:

Part Number	Feature	Package
IRMCK171	OTP , Motor Sensorless Control	QFP48
IRMCK172	OTP , Motor Sensorless Control	QFN48
IRMCK182	OTP , Motor Sensorless Control	QFN32
IRMCK143	OTP , Servo Control	QFP64
IRMCF183	Flash, Motor Sensorless Control	QFN32
IRMCF171	Flash, Motor Sensorless Control	QFP48
IRMCF143	Flash, Servo Control	QFP64
IRMCF188	Flash , Motor Sensorless + PFC Control	QFP64

Table 1. IRMCx100 Products

1.1 System Components

The IRMCx100-series devices can be divided into four main components, shown color-coded in Figure 3 and Figure 4. The components are:

- The 8051 microcontroller, shown at the far left in purple
- RAM and non-volatile memory for program and data storage, shown at the center of the diagram in green
- The Motion Control Engine (MCE), shown at right in blue
- Hardware interface or “motion peripheral” modules, shown at the far right in yellow

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

1.2 Memory Resources

The IRMCx100 series provides four memory-mapped resources:

- Non-volatile memory for program storage
- RAM for MCE program execution
- RAM for 8051 and MCE data storage
- Memory-mapped motion hardware registers (MHRs)

Access to some of these resources by the MCE and 8051 is limited, as described in Table 2.

Memory Resource	Size (bytes)	Access by 8051	Access by MCE
OTP Memory (IRMCK1xx)	32K	Read only	None
Flash Memory (IRMCF1xx)	64K	Read only	None
MCE Program RAM	12K	Write only	Read only
Data RAM	4K	Read and write	Read and write
Motion Hardware Registers	1K	Read and write	Read and write

Table 2. Access to Memory Resources

1.2.1 OTP Memory (IRMCK1xx)

For the IRMCK1xx, the 32K-byte OTP memory holds both 8051 and MCE program. The 8051 program executes directly from this area of OTP memory. The MCE cannot address OTP memory. The IRMCK100-series hardware copies MCE program from OTP to MCE program RAM during the boot process and MCE program executes from RAM. Refer to Section 1.4 for more information about the boot process.

OTP memory is divided into two areas with the 8051 program contained in the first section and the MCE program contained in the second. The boundary between the two areas is dynamic and can vary from 20K bytes to 24K bytes. This allows a maximum size of 24K bytes for the 8051 program (limiting MCE program size to 8K bytes) or a maximum size of 12K bytes for the MCE program (limiting the 8051 program to 20K bytes). The total program size (8051 + MCE) cannot exceed 32K bytes. Refer to Section 4.2 for more information about the OTP memory.

1.2.2 Flash Memory (IRMCF1xx)

For the IRMCF1xx, the 64K-byte flash memory holds both 8051 and MCE program. The 8051 program executes directly from this area of flash memory. The MCE cannot address flash memory. The IRMCF100-series hardware copies MCE program from flash to MCE program RAM during the boot process and MCE program executes from RAM. Refer to Section 1.4 for more information about the boot process.

Flash memory is divided into two areas with the 8051 program contained in the first section and the MCE program contained in the second. The boundary between the two areas is fixed at offset 0xCBFF and the upper 1K bytes (beginning at offset 0xFC00) are reserved. This allows a maximum size of 51K bytes for the 8051 program and a maximum size of 12K bytes for the MCE program. Refer to Section 4.3 for more information about the flash memory.

1.2.3 Data RAM

Shared data RAM for the 8051 and MCE has a total size of 4K bytes. This area is used for MCE private data storage, 8051 private data storage and shared data, such as the motion firmware registers (MFRs) described in Section 3.4. For more information about data RAM usage, refer to Section 4.4.

See Section 4.1 for information on how the memory resources described here are mapped to 8051 and MCE memory.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

1.3 Byte Ordering

Byte ordering refers to the convention used to store 16-bit and 32-bit values in memory using a processor, such as the 8051, that has a native addressing mode of 8 bits. The two standard byte ordering conventions are "big endian" or "Motorola" byte ordering and "little endian" or "Intel" byte ordering.

In big endian byte ordering, the "big end" of a value is stored first. That is, the high order byte is stored at the lowest memory address and the low-order byte is stored at the highest memory address. In little endian byte ordering, the "little end" is stored first, with the low-order byte at the lowest memory address.

For example, suppose the 16-bit value 0x2345 is to be stored in memory at address 0x1000. Using big endian byte ordering, 0x23 is stored at address 0x1000 and 0x45 is stored at address 0x1001. Using little endian byte ordering, 0x45 is stored at address 0x1000 and 0x23 is stored at address 0x1001.

The table below shows how the value 0x456789AB would be stored at address 0x1000 using each of the byte ordering conventions.

Address	Big Endian	Little Endian
1000	45	AB
1001	67	89
1002	89	67
1003	AB	45

The Keil compiler used for 8051 software development generates code that uses big endian byte ordering to store 16-bit and 32-bit values in memory.

The MCE is a 16-bit processor and uses little endian byte ordering for data storage. The smallest unit of data storage on the MCE processor is 16 bits (it cannot access a single byte in memory). The shared RAM used to exchange information between the 8051 and MCE processors is 8-bit addressable to the 8051, but 16-bit addressable to the MCE.

All data shared between the 8051 and MCE processors is expected to be in little endian byte ordering. This means that the 8051 must swap bytes before writing to shared RAM and swap bytes after reading from shared RAM. The table below shows how the value 0x456789AB would be stored at address 0x8200 in 8051 RAM, which corresponds to address 0x0100 in MCE RAM. Note that the 8051 reads and writes a byte at a time, but the MCE always accesses the memory a word (16 bits) at a time.

8051 Address	8051 Bytes	MCE Address	MCE Words
8200	AB	0100	89AB
8201	89		
8202	67	0101	4567
8203	45		

1.4 Reset Mechanism and Boot Process

A power on reset does not require any assertion of reset signal from an external circuit. The IRMCx100 series contains an analog power on reset (POR) circuit which issues reset to all internal circuits. Therefore no filter is required at the reset pin. The reset pin is bidirectional and becomes output when one of the following conditions occurs.

1) The watchdog timer times out.
2) Under voltage lockout (UVCC) detects low voltage on 3.3V.
3) Power up

Among these, cases (2) and (3) generate RESET low assertion for a period of 2048 crystal clock time. When the watchdog timeout occurs, RESET low assertion time becomes 31 usec.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

When the RESET input is asserted from an external source, minimum 10 usec low level assertion pulse is required to ensure the internal reset.

Figure 5 shows the Reset Module, which contains two analog circuits, namely Under Voltage Lockout (UVCC) and Power On Reset (POR), and a 12-bit ripple counter to stretch the Reset pulse width. This module is also shown in Figure 6.

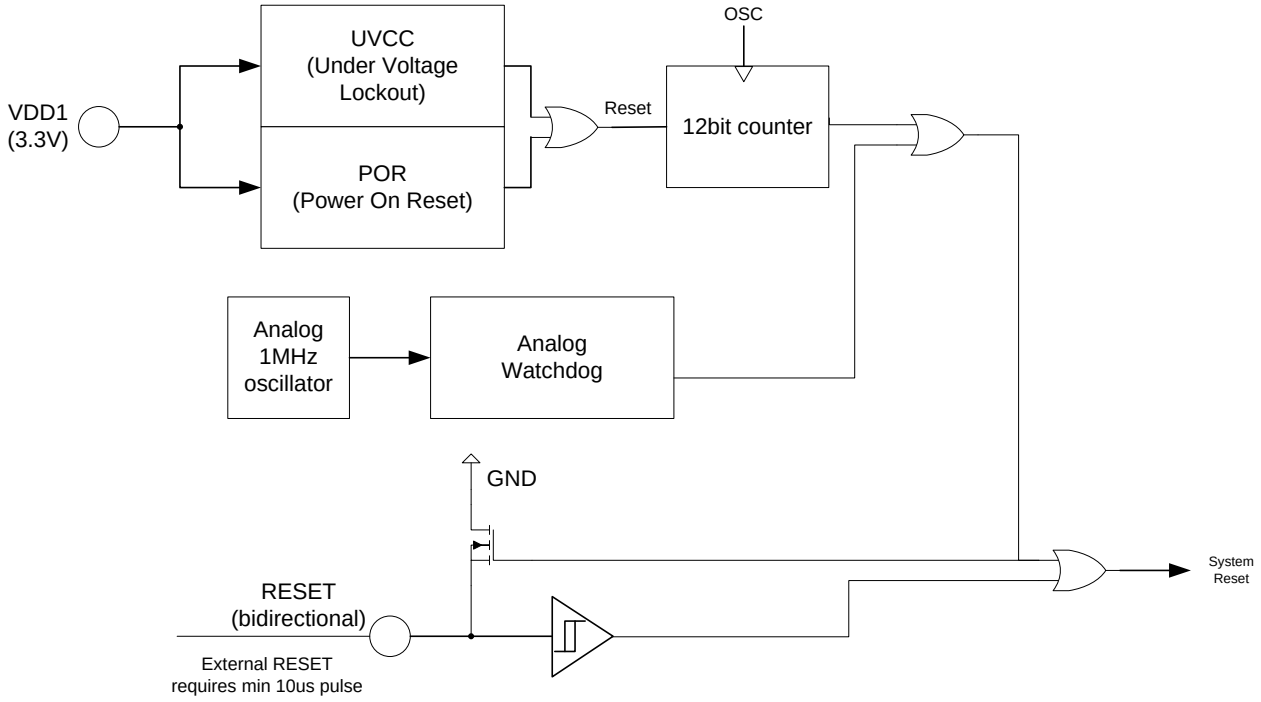


Figure 5. Reset Module

The reset and boot processes are closely tied together. The boot process is automatically accomplished following a proper reset sequence, which is triggered by power on or external RESET. Therefore, a user application cannot intervene during the reset and boot process. The main task of the boot process is to copy the MCE program stored in internal OTP or flash memory to MCE program RAM, initialize the program counter and transfer control to the 8051 CPU. The block diagram of the reset and boot process is shown in Figure 6.

A reset of the device is accomplished by asserting low on the RESET input pin for a minimum of 10 µsec. The RESET is a Schmitt trigger type input with hysteresis to avoid any multiple triggers to the system. Once the reset is recognized in the system, the reset module counts up 2048 clocks at the crystal frequency to ensure that the internal PLL becomes stable for generation of the internal system clock. When this waiting period is complete, the boot sequence is initiated.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

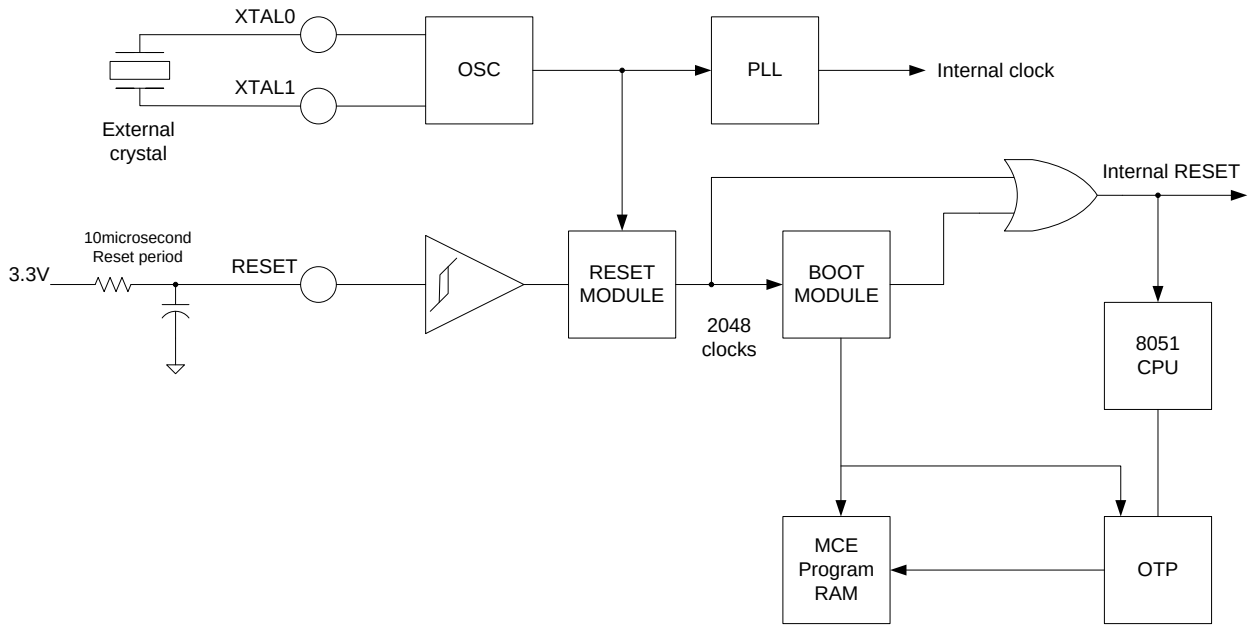


Figure 6. Reset and Boot Process

For the IRMCK1xx, the boot sequence is done in the following way:

- The boot clock source is determined according to the value of VPP/P1.5 when the reset module signal is de-asserted:
  o VPP/P1.5 = 0 : boot_clock = 1MHz oscillator (watchdog) clock
  o VPP/P1.5 = 1 : boot_clock = crystal clock
- The boot sequence starts 16 boot_clocks after reset module signal is de-asserted.
- OTP bytes from offsets 0x7FFE and 0x7FFD are decrypted and written to the 16-bit word at address 0x000 of the MCE program RAM.
- OTP bytes from offsets 0x7FFC and 0x7FFB are decrypted and written to the 16-bit word at address 0x001 of the MCE program RAM.
- This copy procedure continues until the last MCE program bytes are copied from OTP offsets 0x5000 and 0x4FFF to MCE program RAM address 0x17FF.
- A total of 12K bytes are copied.
- Only after the boot sequence is done does the 8051 code start running.

For the IRMCF1xx, the boot sequence is similar, but the address range for the MCE program is different:

- The boot clock source is determined according to the value of P1.5 when the reset module signal is de-asserted:
  o P1.5 = 0 : boot_clock = 1MHz oscillator (watchdog) clock
  o P1.5 = 1 : boot_clock = crystal clock
- The boot sequence starts 16 boot_clocks after reset module signal is de-asserted.
- Flash bytes from offsets 0xFBFE and 0xFBFD are decrypted and written to the 16-bit word at address 0x000 of the MCE program RAM.
- Flash bytes from offsets 0xFBFC and 0xFBFB are decrypted and written to the 16-bit word at address 0x001 of the MCE program RAM.
- This copy procedure continues until the last MCE program bytes are copied from flash offsets 0xCC00 and 0xCBFF to MCE program RAM address 0x17FF.
- A total of 12K bytes are copied.
- Only after the boot sequence is done does the 8051 code start running.

Refer to Section 4.2 for a more detailed description of the copy procedure.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Boot time:
Total MCE program to be copied: 12 * 1024 bytes
Number of clk periods to transfer one byte (including synchronizers and decryption logic): 20
Boot clock freq: 1MHz for internal watchdog clock / 4MHz as example of an external crystal oscillator.

Total transfer time (1MHZ) = 12 * 1024 * 20 * 1 µsec = 244 msec
Total transfer time (4MHZ) = 12 * 1024 * 20 * 0.25 µsec = 61 msec

The boot module holds the internal RESET active while data transfer takes place, then it releases the internal RESET upon completion of the copy process. Immediately after the copy process completes, the 8051 application program begins execution.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2 8051 Microcontroller

This section describes IRMCx100-series features and functions that are specific to the 8051 microcontroller. The interface between the 8051 and the MCE is covered in Section 4.

The instruction set and basic operation of the IRMCx100 series 8051 microcontroller is consistent with the standard Intel 8051 processor. A number of peripheral devices and special functions have been added to customize the operation for the intended application.

2.1 Instruction Set

The instructions of the 8051 microcontroller are 1, 2 or 3 bytes long as listed in the 'Bytes' column below. Each instruction takes either one, two or four PCLK cycles to execute as listed in the following table (one PCLK cycle is equal to two 8051 clock cycles).

Table 3 lists the 8051 instructions. In the table, an entry such as E8-EF indicates a continuous block of hex opcodes used for 8 different registers, the register numbers of which are defined by the lowest three bits of the corresponding code. Non-continuous blocks of codes, shown as 11→F1 (for example), are used for absolute jumps and calls, with the top 3 bits of the code being used to store the top three bits of the destination address.

The CJNE instructions use the abbreviation #d for immediate data; other instructions use #data.

ARITHMETIC				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
ADD A,Rn	Add register to A	1	1	28-2F
ADD A,dir	Add direct byte to A	2	1	25
ADD A,@Ri	Add indirect memory to A	1	1	26-27
ADD A,#data	Add immediate to A	2	1	24
ADDC A,Rn	Add register to A with carry	1	1	38-3F
ADDC A,dir	Add direct byte to A with carry	2	1	35
ADDC A,@Ri	Add indirect memory to A with carry	1	1	36-37
ADDC A,#data	Add immediate to A with carry	2	1	34
SUBB A,Rn	Subtract register from A with borrow	1	1	98-9F
SUBB A,dir	Subtract direct byte from A with borrow	2	1	95
SUBB A,@Ri	Subtract indirect memory from A with borrow	1	1	96-97
SUBB A,#data	Subtract immediate from A with borrow	2	1	94
INC A	Increment A	1	1	04
INC Rn	Increment register	1	1	08-0F
INC dir	Increment direct byte	2	1	05
INC @Ri	Increment indirect memory	1	1	06-07
DEC A	Decrement A	1	1	14
DEC Rn	Decrement register	1	1	18-1F
DEC dir	Decrement direct byte	2	1	15
DEC @Ri	Decrement indirect memory	1	1	16-17
INC DPTR	Increment data pointer	1	2	A3
MUL AB	Multiply A by B	1	4	A4
DIV AB	Divide A by B	1	4	84
DA A	Decimal Adjust A	1	1	D4

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

LOGICAL				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
ANL A,Rn	AND register to A	1	1	58-5F
ANL A,dir	AND direct byte to A	2	1	55
ANL A,@Ri	AND indirect memory to A	1	1	56-57
ANL A,#data	AND immediate to A	2	1	54
ANL dir,A	AND A to direct byte	2	1	52
ANL dir,#data	AND immediate to direct byte	3	2	53
ORL A,Rn	OR register to A	1	1	48-4F
ORL A,dir	OR direct byte to A	2	1	45
ORL A,@Ri	OR indirect memory to A	1	1	46-47
ORL A,#data	OR immediate to A	2	1	44
ORL dir,A	OR A to direct byte	2	1	42
ORL dir,#data	OR immediate to direct byte	3	2	43
XRL A,Rn	Exclusive-OR register to A	1	1	68-6F
XRL A,dir	Exclusive-OR direct byte to A	2	1	65
XRL A,@Ri	Exclusive-OR indirect memory to A	1	1	66-67
XRL A,#data	Exclusive-OR immediate to A	2	1	64
XRL dir,A	Exclusive-OR A to direct byte	2	1	62
XRL dir,#data	Exclusive-OR immediate to direct byte	3	2	63
CLR A	Clear A	1	1	E4
CPL A	Complement A	1	1	F4
SWAP A	Swap Nibbles of A	1	1	C4
RL A	Rotate A left	1	1	23
RLC A	Rotate A left through carry	1	1	33
RR A	Rotate A right	1	1	03
RRC A	Rotate A right through carry	1	1	13

DATA TRANSFER				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
MOV A,Rn	Move register to A	1	1	E8-EF
MOV A,dir	Move direct byte to A	2	1	E5
MOV A,@Ri	Move indirect memory to A	1	1	E6-E7
MOV A,#data	Move immediate to A	2	1	74
MOV Rn,A	Move A to register	1	1	F8-FF
MOV Rn,dir	Move direct byte to register	2	2	A8-AF
MOV Rn,#data	Move immediate to register	2	1	78-7F
MOV dir,A	Move A to direct byte	2	1	F5
MOV dir,Rn	Move register to direct byte	2	2	88-8F
MOV dir,dir	Move direct byte to direct byte	3	2	85
MOV dir,@Ri	Move indirect memory to direct byte	2	2	86-87
MOV dir,#data	Move immediate to direct byte	3	2	75
MOV @Ri,A	Move A to indirect memory	1	1	F6-F7
MOV @Ri,dir	Move direct byte to indirect memory	2	2	A6-A7
MOV @Ri,#data	Move immediate to indirect memory	2	1	76-77
MOV DPTR,#data	Move immediate to data pointer	3	2	90
MOVC A,@A+DPTR	Move code byte relative DPTR to A	1	2	93
MOVC A,@A+PC	Move code byte relative PC to A	1	2	83
MOVX A,@Ri	Move external data(A8) to A	1	2	E2-E3

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

DATA TRANSFER				
MOVX A,@DPTR	Move external data(A16) to A	1	2	E0
MOVX @Ri,A	Move A to external data(A8)	1	2	F2-F3
MOVX @DPTR,A	Move A to external data(A16)	1	2	F0
PUSH dir	Push direct byte onto stack	2	2	C0
POP dir	Pop direct byte from stack	2	2	D0
XCH A,Rn	Exchange A and register	1	1	C8-CF
XCH A,dir	Exchange A and direct byte	2	1	C5
XCH A,@Ri	Exchange A and indirect memory	1	1	C6-C7
XCHD A,@Ri	Exchange A and indirect memory nibble	1	1	D6-D7

BOOLEAN				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
CLR C	Clear carry	1	1	C3
CLR bit	Clear direct bit	2	1	C2
SETB C	Set carry	1	1	D3
SETB bit	Set direct bit	2	1	D2
CPL C	Complement carry	1	1	B3
CPL bit	Complement direct bit	2	1	B2
ANL C,bit	AND direct bit to carry	2	2	82
ANL C,/bit	AND direct bit inverse to carry	2	2	B0
ORL C,bit	OR direct bit to carry	2	2	72
ORL C,/bit	OR direct bit inverse to carry	2	2	A0
MOV C,bit	Move direct bit to carry	2	1	A2
MOV bit,C	Move carry to direct bit	2	2	92

BRANCHING				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
ACALL addr 11	Absolute jump to subroutine	2	2	11→F1
LCALL addr 16	Long jump to subroutine	3	2	12
RET	Return from subroutine	1	2	22
RETI	Return from interrupt	1	2	32
AJMP addr 11	Absolute jump unconditional	2	2	01→E1
LJMP addr 16	Long jump unconditional	3	2	02
SJMP rel	Short jump (relative address)	2	2	80
JC rel	Jump on carry = 1	2	2	40
JNC rel	Jump on carry = 0	2	2	50
JB bit,rel	Jump on direct bit = 1	3	2	20
JNB bit,rel	Jump on direct bit = 0	3	2	30
JBC bit,rel	Jump on direct bit = 1 and clear	3	2	10
JMP @A+DPTR	Jump indirect relative DPTR	1	2	73
JZ rel	Jump on accumulator = 0	2	2	60
JNZ rel	Jump on accumulator ≠ 0	2	2	70
CJNE A,dir,rel	Compare A,direct jne relative	3	2	B5
CJNE A,#d,rel	Compare A,immediate jne relative	3	2	B4
CJNE Rn,#d,rel	Compare register, immediate jne relative	3	2	B8-BF
CJNE @Ri,#d,rel	Compare indirect, immediate jne relative	3	2	B6-B7
DJNZ Rn,rel	Decrement register, jnz relative	2	2	D8-DF
DJNZ dir,rel	Decrement direct byte, jnz relative	3	2	D5

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MISCELLANEOUS				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
NOP	No operation	1	1	00

ADDITIONAL INSTRUCTIONS (selected through EO bit 4)				
Mnemonic	**Description**	**Bytes**	**PCLK Cycles**	**Hex code**
MOVC @(DPTR++),A	Supports software download into program memory (see Section 2.4.5)	1	2	A5
TRAP	Software break command (see Section 2.4.5)	1	1	A5

Table 3. 8051 Instructions

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.2 Special Function Registers

I/O functions, interrupt control and some peripherals are accessed via Special Function Registers (SFRs). These registers occupy direct Internal Data Memory space locations in the range 0x80 to 0xFF. Internal processor registers (accumulator, stack pointer, etc.) can also be accessed through SFRs.

Note 1. WKTMR in the IRMCK1xx is replaced by FLSEL in the IRMCF1xx.
Table 4 shows a summary of the SFR memory map and identifies the bit-addressable registers. Table 5 lists each register individually, shows its value on reset and references the document section that describes the register in detail. Some registers are present only for certain product types, as indicated in the Notes column of Table 5 and in the detailed register descriptions.

In the detailed register descriptions, special function registers are shown shaded with a green color.

	0(8)	**1(9)**	**2(A)**	**3(B)**	**4(C)**	**5(D)**	**6(E)**	**7(F)**
80	P0	SP	DPL	DPH				
88	TCON	TMOD	TL0	TL1	TH0	TH1		
90	P1	P1DIR						
98								
A0	P2	P2DIR	EO	FLDWR				
A8	IE	PSCL	IOCON0	IOCON1				
B0	P3	P3DIR	HWCFG	SHTEST				
B8	IP							
C0	IS				SLPTMR	WKTMR/ FLSEL[1]		
C8	T2CON		RCP2L	RCP2H	TL2	TH2		
D0	PSW					PLLF0	PLLF1	PLLF2
D8		MCEBL	MCEBH	PLLF3	HWREV	PLLF4	INDCOH	INDCOL
E0	ACC	FLDRD		MCESL	MCESH	STOPS		P5
E8	IE1	MCECD0	MCECD1	MCECD2	MCECD3			
F0	B							
F8	IP1							
	Bit Addressable							

Note 1. WKTMR in the IRMCK1xx is replaced by FLSEL in the IRMCF1xx.

Table 4. Special Function Register Memory Map

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Address (hex)	Label	Description	Reset Value (hex)	Bit Address-able	Notes	Document Section
80	P0	Port 0	FF	*	Not used	
81	SP	Stack Pointer	07			2.4.1
82	DPL	Data Pointer Low Byte	00			2.4.1
83	DPH	Data Pointer High Byte	00			2.4.1
88	TCON	Timer/Counter Control	00	*		2.6.2
89	TMOD	Timer/Counter Mode Control	00			2.6.2
8A	TL0	Timer/Counter 0 Low Byte	00			2.6.2
8B	TL1	Timer/Counter 1 Low Byte	00			2.6.2
8C	TH0	Timer/Counter 0 High Byte	00			2.6.2
8D	TH1	Timer/Counter 1 High Byte	00			2.6.2
90	P1	Port 1	FF	*		2.4.2
91	P1DIR	Port 1 Direction Register	00			2.4.2
A0	P2	Port 2	FF	*		2.4.2
A1	P2DIR	Port 2 Direction Register	00			2.4.2
A2	EO	Extended Operation Register	00			2.4.5
A3	FLDWR	Flash Write Data	00		Internal use IRMCF1xx only	
A8	IE	Interrupt Enable Register 0	00	*		2.5.3
A9	PSCL	Prescaler Control	00			2.6.1
AA	IOCON0	I/O Control Register 0	B0			2.4.2
AB	IOCON1	I/O Control Register 1	00			2.4.2
B0	P3	Port 3	FF	*		2.4.2
B1	P3DIR	Port 3 Direction Register	00			2.4.2
B2	HWCFG	Hardware Configuration	01			2.4.5
B3	SHTEST	Sample & Hold Test Enable	00			
B8	IP	Interrupt Priority Register 0	00	*		2.5.4
C0	IS	Interrupt Status & Acknowledge	00	*		2.5.3
C4	SLPTMR	Power saving mode sleep timer	00			2.4.4
C5	WKTMR	Power saving mode wake up timer	00		IRMCK1xx only	2.4.4
C5	FLSEL	Flash Select Register	00		Internal use IRMCF1xx only	
C8	T2CON	Timer /Counter 2 Control	00	*		2.6.2
CA	RCP2L	Timer 2 Capture Register Low	00			2.6.2
CB	RCP2H	Timer 2 Capture Register High	00			2.6.2
CC	TL2	Timer /Counter 2 Low Byte	00			2.6.2
CD	TH2	Timer /Counter 2 High Byte	00			2.6.2
D0	PSW	Program Status Word	00	*		2.4.1
D5	PLLF0	PLL Setup bits 0 – 7	Device dependent			2.4.3
D6	PLLF1	PLL Setup bits 8 – 15	Device dependent			2.4.3
D7	PLLF2	PLL Setup	Device			2.4.3

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Address (hex)	Label	Description	Reset Value (hex)	Bit Address-able	Notes	Document Section
		bits 16 – 23	dependent			
D9	MCEBL	MCE Sequencer Breakpoint Address Low	00		Internal use	
DA	MCEBH	MCE Sequencer Breakpoint Address High	00		Internal use	
DB	PLLF3	PLL Setup bits 24 – 31	Device dependent			2.4.3
DC	HWREV	Hardware Model and Revision	Device dependent		Read only	2.4.5
DD	PLLF4	PLL Setup bits 32 – 40	08		IRMCF1xx only	2.4.3
DE	INDCOH	PLL DCO Control Word (High)	00		Internal use IRMCF1xx only	
DF	INDCOL	PLL DCO Control Word (Low)	00		Internal use IRMCF1xx only	
E0	ACC	Accumulator	00	*		2.4.1
E1	FLDRD	Flash Read Data	00		Internal use Read only IRMCF1xx only	
E3	MCESL	MCE Sequencer Stack Low	00		Read only Internal use	
E4	MCESH	MCE Sequencer Stack High	00		Read only Internal use	
E5	STOPS	GATEKILL Configuration	00			2.4.5
E7	P5	Port 5	00	*		2.4.2
E8	IE1	Interrupt Enable Register 1	00	*		2.5.3
E9	MCECD0	MCE Coherent Data 0	00			4.4.1
EA	MCECD1	MCE Coherent Data 1	00			4.4.1
EB	MCECD2	MCE Coherent Data 2	00			4.4.1
EC	MCECD3	MCE Coherent Data 3	00			4.4.1
F0	B	B Register	00	*		2.4.1
F8	IP1	Interrupt Priority Register 1	00	*		2.5.4

Table 5. Special Function Registers

2.3 Peripheral Registers

This register group provides control of 8051 peripherals unique to the IRMCx100 series, including the UART, I^2C/SPI interface and special-purpose timers. The peripheral registers are directly addressable through an area of shared data RAM in the address range 0xF100 – 0xF7FF. They can be accessed from the MCE as well as from the 8051, although the MCE does not generally use these registers since their functions are associated with 8051 peripheral devices.

Peripheral registers range in size from 8 bits to 32 bits. On the 8051 memory bus, the registers appear in little endian byte ordering (see Section 1.3), low-order byte first. On the 16-bit MCE bus, registers larger than 16 bits appear in little endian word order, low-order word first.

For peripheral registers larger than 8 bits (UART0 baud rate, for example), two or more 8-bit registers are defined for the 8051, while the MCE accesses the register as a single 16-bit or 32-bit value. For any register, the MCE address can be converted to an 8051 address using the following formula:

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

8051 address = (MCE address – 0xE80) * 2 + 0xF100

Note 1. The Watchdog Timer Count register is present in IRMCF1xx devices only.
Table 6 lists each register, shows its value on reset and references the document section that describes the register in detail. In the detailed register descriptions, peripheral registers are shown shaded with a pink color.

8051 Address (hex)	MCE Address (hex)	Label	Description	Reset Value (hex)	Document Section
F114	E8A	UTXD	UART0 Transmit Data Buffer	00	2.7
F116	E8B	UCTRL	UART0 Control	00	2.7
	E8C	UBD	UART0 Baud Rate (16 bits)	0000	2.7
F118		UBDL	UART0 Baud Rate (low)	00	2.7
F119		UBDH	UART0 Baud Rate (high)	00	2.7
F11A	E8D	ISFILT	I^2C Noise Filter	00	2.9
	E8E	ISBD	I^2C /SPI Baud Rate Control (16 bits)	0000	2.9
F11C		ISBDL	I^2C /SPI Baud Rate Control (low)	00	2.9
F11D		ISBDH	I^2C /SPI Baud Rate Control (high)	00	2.9
F11E	E8F	ISCMD	I^2C /SPI Command	00	2.9
F120	E90	ISTXD	I^2C /SPI Write Data	00	2.9
F122	E91	ISACKIN	I^2C /SPI ACK Data	00	2.9
	E93	WDLIM	Watchdog Timer Limit (16 bits)	FFFE	2.6.3
F126		WDLIML	Watchdog Timer Limit (low)	FE	2.6.3
F127		WDLIMH	Watchdog Timer Limit (high)	FF	2.6.3
F1CC	EE6	DAPRE	D/A PWM Prescaler	00	2.8
F1CE	EE7	DAD0	D/A PWM Output 0 Data	00	2.8
F1D0	EE8	DAD1	D/A PWM Output 1 Data	00	2.8
F1D2	EE9	DAD2	D/A PWM Output 2 Data	00	2.8
	EF0	CAPPS	Capture Prescaler (16 bits)	0000	2.6.4
F1E0		CAPPSL	Capture Prescaler (low)	00	2.6.4
F1E1		CAPPSH	Capture Prescaler (high)	00	2.6.4
F1E2	EF1	CAPEG	Capture Edge Selection	00	2.6.4
F1E6	EF3	MTENB	MCE Pin Timer Enable	00	2.6.5
	EF4	INSEL	Input Signal Selection (32 bits)	FFFFFFFF	2.4.2
F1E8		INSEL0	Input Signal Selection (low)	FF	2.4.2
F1E9		INSEL1	Input Signal Selection (mid-low)	FF	2.4.2
F1EA		INSEL2	Input Signal Selection (mid-high)	FF	2.4.2
F1EB		INSEL3	Input Signal Selection (high)	FF	2.4.2
F1F0	EF8	MTPER1	MCE Pin Timer 1 Period	00	2.6.5
	EFA	MTC1	MCE Pin Timer 1 Count (24 bits)	000000	2.6.5
F1F4		MTC1L	MCE Pin Timer 1 Count (low)	00	2.6.5
F1F5		MTC1M	MCE Pin Timer 1 Count (mid)	00	2.6.5
F1F6		MTC1H	MCE Pin Timer 1 Count (high)	00	2.6.5
F1F8	EFC	MTMD1	MCE Pin Timer 1 Mode	00	2.6.5
F1FA	EFD	MTPER2	MCE Pin Timer 2 Period	00	2.6.5
	EFE	MTC2	MCE Pin Timer 2 Count (24 bits)	000000	2.6.5
F1FC		MTC2L	MCE Pin Timer 2 Count (low)	00	2.6.5
F1FD		MTC2M	MCE Pin Timer 2 Count (mid)	00	2.6.5
F1FE		MTC2H	MCE Pin Timer 2 Count (high)	00	2.6.5
F200	F00	MTMD2	MCE Pin Timer 2 Mode	00	2.6.5
F202	F01	MTPER3	MCE Pin Timer 3 Period	00	2.6.5
	F02	MTC3	MCE Pin Timer 3 Count (24 bits)	000000	2.6.5

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

8051 Address (hex)	MCE Address (hex)	Label	Description	Reset Value (hex)	Document Section
F204		MTC3L	MCE Pin Timer 3 Count (low)	00	2.6.5
F205		MTC3M	MCE Pin Timer 3 Count (mid)	00	2.6.5
F206		MTC3H	MCE Pin Timer 3 Count (high)	00	2.6.5
F208	F04	MTMD3	MCE Pin Timer 3 Mode	00	2.6.5
F21C	F0E	URXD	UART0 Receive Data Buffer	00	2.7
F21E	F0F	USTAT	UART0 Status	00	2.7
F224	F12	ISRXD	I^2C /SPI Read Data	00	2.9
F226	F13	ISSTAT	I^2C /SPI Status	00	2.9
	F1E	CAPLST	Capture Last Time (16 bits)	0000	2.6.4
F23C		CAPLSTL	Capture Last Time (low)	00	2.6.4
F23D		CAPLSTH	Capture Last Time (high)	00	2.6.4
	F1F	CAPMID	Capture Previous Time (16 bits)	0000	2.6.4
F23E		CAPMIDL	Capture Previous Time (low)	00	2.6.4
F23F		CAPMIDH	Capture Previous Time (high)	00	2.6.4
	F2C	WDCNT	Watchdog Timer Count (16 bits)[1]	0000	2.6.3
F258		WDCNTL	Watchdog Timer Count (low)[1]	00	2.6.3
F259		WDCNTH	Watchdog Timer Count (high)[1]	00	2.6.3

Note 1. The Watchdog Timer Count register is present in IRMCF1xx devices only.

Table 6. Peripheral Registers

2.4 General Functions

This section describes general 8051 functions controlled through Special Function Registers (SFRs). The 8051 peripheral devices, such as UART, timers and the I^2C/SPI interface are described in later sections.

2.4.1 Processor Registers

Some of the 8051 processor registers can be accessed as SFRs. These include the stack pointer (SP), data pointer (DPTR), program status word (PSW), accumulator (A or ACC) and the B register.

STACK POINTER (SP)

Address: 0x81 Not Bit Addressable Reset value: 0x07

The SP register contains the Stack Pointer. The Stack Pointer is used to load the program counter into Internal Data Memory during LCALL and ACALL instructions and to retrieve the program counter from memory during RET and RETI instructions.

Data may also be saved on or retrieved from the stack using PUSH and POP instructions. Instructions that use the stack automatically pre-increment or post-decrement the Stack Pointer so that the Stack Pointer always points to the last byte written to the stack, i.e. the top of the stack. On reset the Stack Pointer is set to 0x07.

It falls to the programmer to ensure that the location of the stack in Internal Data Memory does not interfere with other data stored therein.

DATA POINTER (DPTR)

Address: 0x82 (DPL), 0x83 (DPH) Not Bit Addressable Reset value: 0x0000

The Data Pointer (DPTR) is a 16-bit register that is used to form 16-bit addresses for External Data Memory accesses (MOVX A,@DPTR and MOVX @DPTR,A), for program byte moves (MOVC A,@A+DPTR) and for indirect program jumps (JMP @A+DPTR).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Two true 16-bit operations are allowed on the Data Pointer – load immediate (MOV DPTR,#data) and increment (INC DPTR).

On reset all data pointers are set to 0x00.

PROGRAM STATUS WORD (PSW)

Address: 0xD0 Bit Addressable Reset value: 00000000b

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**F1**	**P**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R

This register contains status information resulting from CPU and ALU operation. The bit definitions are given below:

PSW.7	**CY**	ALU carry flag.
PSW.6	**AC**	ALU auxiliary carry flag.
PSW.5	**F0**	General-purpose user-definable flag.
PSW.4	**RS1**	Register Bank Select bit 1.
PSW.3	**RS0**	Register Bank Select bit 0.
PSW.2	**OV**	ALU overflow flag.
PSW.1	**F1**	User-definable flag.
PSW.0	**P**	Parity flag. Set each instruction cycle to indicate odd/even parity in the accumulator.

The Register Bank Select bits PSW[4:3] operate as follows.

RS1	RS0	Register Bank Select
0	0	RB0. Registers from 00 – 07 hex.
0	1	RB1. Registers from 08 – 0F hex.
1	0	RB2. Registers from 10 – 17 hex.
1	1	RB3. Registers from 18 – 1F hex.

On reset this register returns 0x00.

ACCUMULATOR (ACC)

Address: 0xE0 Bit Addressable Reset value: 0x00

This register provides one of the operands for most ALU operations. It is denoted as ‘A’ in the instruction set summary (Table 3).

On reset this register returns 0x00.

B REGISTER (B)

Address: 0xF0 Bit Addressable Reset value: 0x00

This register provides the second operand for multiply or divide instructions. Otherwise, it may be used as a scratch pad register.

On reset this register returns 0x00.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.4.2 General Purpose I/O

I/O PORTS (P1, P2, P3, P5)

Address:	*0x90 (P1), 0xA0 (P2), 0xB0 (P3), 0xE7 (P5)*	*Bit Addressable*	*Reset value:*	*0xFF, undefined(P5)*

P1, P2 and P3 are latches used to drive the quasi-bidirectional I/O lines. On reset P1 – P3 are set to the value FF hex. Some of the ports have dual functions, as shown in Table 4. Port P0 (at address 0x80) is not used. The external pin number associated with each port is dependent on the pinout for the specific IC, and some ports are not available on all ICs.
If the discrete I/O function is selected, any P1 – P3 port has a bi-directional I/O shown in Figure 7. The input buffer has Schmitt trigger type input with hysteresis and a 70k ohm pull-up resistor.

P5 is shared with the JTAG function and these ports are always inputs. (No selection of port function is required.)

Port Name	Pin Number					Function
	K182	F188	x143	F183	x171 K172	
P1.0/T2/MT1	-	3	3	-	3	Discrete I/O, or Timer/Counter 2 input or MCE pin timer 1 output
P1.1/RXD	2	60	4	1	4	Discrete I/O, or UART0 RXD input
P1.2/TXD	3	61	5	2	5	Discrete I/O, or UART0 TXD output
P1.3/ SYNC/SCK	-	6	6	-	6	Discrete I/O, or SYNC output or SPI clock output
P1.4	-	7	7	-	7	Discrete I/O
P1.5 / BCLK source	-	50	50	-	39	Discrete I/O, Boot Clock Source input (during reset)
P1.6	-	8	8	-	-	Discrete I/O
P1.7	-	9	9	-	-	Discrete I/O
P2.0/NMI	-	13	13	-	11	Discrete I/O, or Non-maskable interrupt input
P2.1	-	47	53	-	-	Discrete I/O
P2.2	-	15	15	-	-	Discrete I/O
P2.3 / QCI	-	16	16	-	-	Discrete I/O, or interrupt output reserved for future use
P2.4	-	-	-	-	-	not used
P2.5	-	-	17	-	-	Discrete I/O
P2.6/AOPWM0	9	17	18	9	-	Discrete I/O, or AOPWM0 output
P2.7/AOPWM1/MT2	-	18	19	-	13	Discrete I/O, or AOPWM1 output or MCE pin timer 2 output
P3.0/CS1	-	54	51	-	40	Discrete I/O, or SPI CS1
P3.1/AOPWM2/MT3	-	41	52	-	-	Discrete I/O, or AOPWM2 output or MCE pin timer 3 output
P3.2/NINT0	-	14	14	8	12	Discrete I/O, or NINT0 input
P3.3/NINT1	-	64	64	-	48	Discrete I/O, or NINT1 input
P3.4	-	62	-	-	-	not used
P3.5/T1	-	63	63	-	-	Discrete I/O, or Timer/Counter 1 input
P3.6	-	-	54	-	-	Discrete I/O
P3.7	-	46	-	-	-	not used
P5.1/TDI	30	57	57	30	43	Discrete input, or TDI input
P5.2/TMS	28	55	55	28	41	Discrete input, or TMS input
P5.3/TDO	29	56	56	29		Discrete input, or TDO output

Table 7. Discrete I/Os from Ports 1, 2, 3 and 5

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

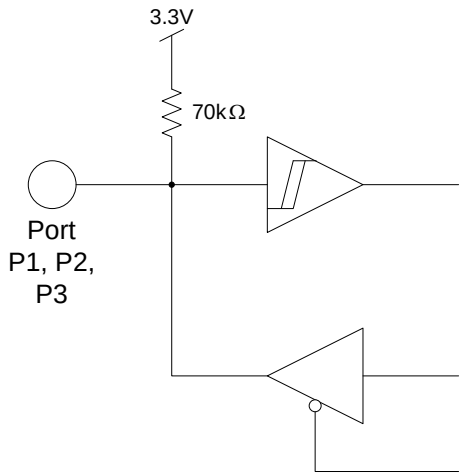


Figure 7. Port Structure of IRMCx100 Series

Port Direction registers (P1DIR, P2DIR, P3DIR) configure the direction (input or output) for each port. P5 pins are always inputs.

PORT 1 DIRECTION (P1DIR)

Address: 0x91 *Not Bit Addressable* *Reset value: 00000000b*

P1DIR.7	P1DIR.6	P1DIR.5	P1DIR.4	P1DIR.3	P1DIR.2	P1DIR.1	P1DIR.0
P1.7D	**P1.6D**	**P1.5D**	**P1.4D**	**P1.3D**	**P1.2D**	**P1.1D**	**P1.0D**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P1DIR.7 – P1DIR.0	**P1.xD**	0: input, 1: output

PORT 2 DIRECTION (P2DIR)

Address: 0xA1 *Not Bit Addressable* *Reset value: 00000000b*

P2DIR.7	P2DIR.6	P2DIR.5	P2DIR.4	P2DIR.3	P2DIR.2	P2DIR.1	P2DIR.0
P2.7D	**P2.6D**	**P2.5D**	**P2.4D**	**P2.3D**	**P2.2D**	**P2.1D**	**P2.0D**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P2DIR.7 - P2DIR.0	**P2.xD**	0: input, 1: output

PORT 3 DIRECTION (P3DIR)

Address: 0xB1 *Not Bit Addressable* *Reset value: 00000000b*

P3DIR.7	P3DIR.6	P3DIR.5	P3DIR.4	P3DIR.3	P3DIR.2	P3DIR.1	P3DIR.0
P3.7D	**P3.6D**	**P3.5D**	**P3.4D**	**P3.3D**	**P3.2D**	**P3.1D**	**P3.0D**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P3DIR.7 - P3DIR.0	**P3.xD**	0: input, 1: output

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The I/O Control registers are used to configure alternate functions for the I/O ports and related options.

I/O CONTROL 0 (IOCON0)

Address: 0xAA *Not Bit Addressable* *Reset value: 00000000b*

IOCON0.7	IOCON0.6	IOCON0.5	IOCON0.4	IOCON0.3	IOCON0.2	IOCON0.1	IOCON0.0
-	**UART0E**	**AOPWM2**	**AOPWM1**	**AOPWM0**	**AOPWMF**	**SYNC**	**NMI**
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

IOCON0.7	**-**	*Reserved*
IOCON0.6	**UART0E**	UART0 Enable. 1: P1.1 is RXD input and P1.2 is TXD output. 0: P1.1 and P1.2 are general purpose I/Os.
IOCON0.5	**AOPWM2**	AOPWM2 Enable. 1: P3.1 is AOPWM2 output, 0: P3.1 is a general purpose I/O
IOCON0.4	**AOPWM1**	AOPWM1 Enable. 1: P2.7 is AOPWM1 output, 0: P2.7 is a general purpose I/O
IOCON0.3	**AOPWM0**	AOPWM0 Enable. 1: P2.6 is AOPWM0 output, 0: P2.6 is a general purpose I/O
IOCON0.2	**AOPWMF**	Frequency selection for AOPWM output. 1: SYSCLK/1024, 0: SYSCLK/256
IOCON0.1	**SYNC**	SYNC output select. 1: P1.3 is SYNC output, 0: P1.3 is a general purpose I/O
IOCON0.0	**NMI**	NMI input select. 1: P2.0 is NMI input, 0: P2.0 is a general purpose I/O

I/O CONTROL 1 (IOCON1)

Address: 0xAB *Not Bit Addressable* *Reset value: 00000000b*

IOCON1.7	IOCON1.6	IOCON1.5	IOCON1.4	IOCON1.3	IOCON1.2	IOCON1.1	IOCON1.0
-	**-**	**CAPE**	**-**	**INT2M1**	**INT2M0**	**-**	**IOMODE**
R/W	R/W	R/W	R	R/W	R/W	R	R/W

The bit definitions for this register are as follows:

IOCON1.7	**-**	*Reserved*
IOCON1.6	**-**	*Reserved*
IOCON1.5	**CAPE**	Capture Timer enable. 1: enable, 0: disable
IOCON1.4	**-**	*Reserved*
IOCON1.3	**INT2M1**	Interrupt 2 Mode selection bit 1.
IOCON1.2	**INT2M0**	Interrupt 2 Mode selection bit 0.
IOCON1.1	**-**	*Reserved*
IOCON1.0	**IOMODE**	Controls the direction mode of all general purpose I/Os. 1: PxDIR determines the direction. 0: 8051 compatibility mode. In 8051 compatibility mode (default at power-up), all I/Os are open drain. Writing 0 to a bit drives the output to 0. Writing 1 causes the output to float.

Interrupt 2 mode selection is encoded as follows:

INT2M1	INT2M0	Operating Mode
0	0	Interrupt 2 is level sensitive (low signal generates the interrupt)
0	1	Positive edge sensitive
1	0	Negative edge sensitive
1	1	Both Positive and Negative edge sensitive

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The input signal selection register, defined below, is used to select an input source for the capture timer (Section 2.6.4) and the external interrupt (INT2). This is a peripheral register, accessible from both the MCE and the 8051. On the MCE, the input signal selection register is accessed as a single 32-bit register. On the 8051, the register is divided into four 8-bit registers (high, mid-high, mid-low and low).

INPUT SIGNAL SELECTION (INSEL)

Address: 0xEF4 (MCE) Range: Bit field definitions Reset value: 0xFFFFFFFF

INPUT SIGNAL SELECTION LOW (INSEL0)

Address: 0xF1E8 (8051) Range: Bit field definitions Reset value: 0xFF

INSEL0.7	INSEL0.6	INSEL0	INSEL0	INSEL0.3	INSEL0	INSEL0	INSEL0.0
QAS1	**QAS0**	**CPS5**	**CPS4**	**CPS3**	**CPS2**	**CPS1**	**CPS0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

INPUT SIGNAL SELECTION MID-LOW (INSEL1)

Address: 0xF1E9 (8051) Range: Bit field definitions Reset value: 0xFF

INSEL1.7	INSEL1.6	INSEL1.5	INSEL1.4	INSEL1.3	INSEL1.2	INSEL1.1	INSEL1.0
QBS3	**QBS2**	**QBS1**	**QBS0**	**QAS5**	**QAS4**	**QAS3**	**QAS2**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

INPUT SIGNAL SELECTION MID-HIGH (INSEL2)

Address: 0xF1EA (8051) Range: Bit field definitions Reset value: 0xFF

INSEL2.7	INSEL2.6	INSEL2.5	INSEL2.4	INSEL2.3	INSEL2.2	INSEL2.1	INSEL2.0
QZS5	**QZS4**	**QZS3**	**QZS2**	**QZS1**	**QZS0**	**QBS5**	**QBS4**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

INPUT SIGNAL SELECTION HIGH (INSEL3)

Address: 0xF1EB (8051) Range: Bit field definitions Reset value: 0xFF

INSEL3.7	INSEL3.6	INSEL3.5	INSEL3.4	INSEL3.3	INSEL3.2	INSEL3.1	INSEL3.0
-	**-**	**I2S5**	**I2S4**	**I2S3**	**I2S2**	**I2S1**	**I2S0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Each six-bit field within the input signal selection register selects an input signal for a particular function, as described below:

Bit Field	Description
CPS0 – CPS5	Selects capture timer input signal
QAS0 – QAS5	Reserved for future use
QBS0 – QBS5	Reserved for future use
QZS0 – QZS5	Reserved for future use
I2S0 – I2S5	Selects external interrupt (INT2) input signal

The input signal selection bits form a 6-bit unsigned value in the range 0 – 34 and are used to select the input signal as follows:

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

xxS5 – xxS0	Selected Input Signal
0	P1.0
1	P1.1
2	P1.2
3	P1.3
4	P1.4
5	P1.5
6	P1.6
7	P1.7
8	P2.0
9	P2.1
10	P2.2
11	P2.3
12	P2.4
13	P2.5
14	P2.6
15	P2.7
16	P3.0
17	P3.1
18	P3.2
19	P3.3
20	P3.4
21	P3.5
22	P3.6
23	P3.7
24	PWMUL
25	PWMVL
26	PWMWL
27	PWMUH
28	PWMVH
29	PWMWH
30	PFC PWM (Not used in IRMCx171)
31	SDA/CS0 (I^2C data or SPI chip select 0)
32	SCL/SDI-SDO (I^2C clock or SPI data)
33	GATEKILL (motor)
34	PFC GATEKILL (Not used in IRMCx171)

Any input selection value not defined above (35 – 63) sets the input for the associated function to a constant zero. The value of each field on power-up/reset is 0xFF, so all inputs are disabled (tied to zero) by default.

General Port Value Register 0 (GPVR0)

Address: 0xF260 (8051) 0xF30 (MCE) Range: Bit field definitions Reset value: 0x00

GPVR0.7	GPVR0.6	GPVR0.5	GPVR0.4	GPVR0.3	GPVR0.2	GPVR0.1	GPVR0.0
P1.4	**P1.3**	**P1.2**	**P1.1**	**-**	**P3.3**	**P3.5**	**P3.4**
RO	RO	RO	RO	RO	RO	RO	RO

This Read only register reflects several port values. This register is accessible both by 8051 and the MCE

General Port Value Register 1 (GPVR1)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Address:	*0xF261 (8051)* *0xF30 (MCE)*	*Range:*	*Bit field definitions*	*Reset value:*	*0x00*

GPVR1.7	GPVR1.6	GPVR1.5	GPVR1.4	GPVR1.3	GPVR1.2	GPVR1.1	GPVR1.0
P2.3	**P2.2**	**P2.1**	**P2.0**	**-**	**P1.7**	**P1.6**	**P1.5**
RO	RO	RO	RO	RO	RO	RO	RO

This Read only register reflects several port values. This register is accessible both by 8051 and the MCE

General Port Value Register 2 (GPVR2)

Address:	*0xF262 (8051)* *0xF31 (MCE)*	*Range:*	*Bit field definitions*	*Reset value:*	*0x00*

GPVR2.7	GPVR2.6	GPVR2.5	GPVR2.4	GPVR2.3	GPVR2.2	GPVR2.1	GPVR2.0
-	**-**	**-**	**-**	**P3.7**	**P3.1**	**P2.7**	**P2.6**
RO	RO	RO	RO	RO	RO	RO	RO

This Read only register reflects several port values. This register is accessible both by 8051 and the MCE

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.4.3 Clock Selection and PLL Frequency Configuration

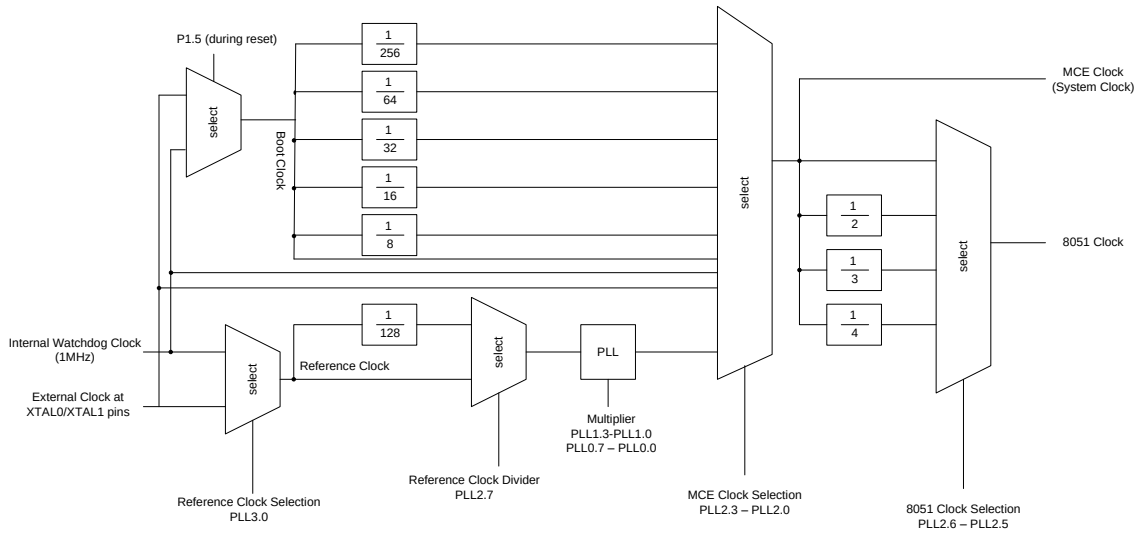


Figure 8. Clock Selection Diagram

Figure 9 shows the clock selection diagram. The user can basically choose either the internal watchdog clock (1MHz) or an external crystal oscillator. When the internal watchdog clock is used, substantial inaccuracy may result since it has approximately maximum +/- 30% error at 1MHz. In other words, it may vary between 700KHz and 1.3MHz. Therefore, it is recommended to use an external crystal clock source. Since the 1MHz internal watchdog clock is always present, the user can switch between two clock choices, which is sometimes useful when implementing a deep energy saving mode while the 8051 still functions. A 256 divider off the internal watchdog clock was specifically designed for this purpose.

All configuration of clock selection is achieved through the PLL SETUP REGISTER. The definition of this register depends on the product type:

- For the IRMCK1xx, the PLL Setup Register is 32 bits, divided into four 8-bit registers located in SFRs as described in Section 2.4.3.1. Except for the PLLF3.7 (BS) bit, which is read only, all bits are read/write. OTP300NS (PLLF3.6 – PLLF3.3) is used for sleep mode operation and is not related to clock setup.
- For the IRMCF1xx, the PLL Setup Register is 40 bits, divided into five 8-bit registers located in SFRs as described in Section 2.4.3.2. Except for the PLLF4.7 (BS) bit, which is read only, all bits are read/write.

All the peripherals do not operate off the same clock, as described in later sections. The following clocks are described in this document:

- MCE Clock is called the “System Clock,” “SYSCLK,” or “SysClk.”
- 8051 Clock as shown in Figure 8, above.
- 8051 Peripheral Clock is half the 8051 Clock frequency and is referred to as the “PCLK.”

2.4.3.1 PLL Setup Register for IRMCK1xx

PLL SETUP REGISTER (PLLF3, PLLF2, PLLF1, PLLF0)

Address:	*0xD5 – PLLF0*	*Bit Addressable*	*Reset value:*	*0x2C (PLLF0)*
	0xD6 – PLLF1			*0xA1 (PLLF1)*
	0xD7 – PLLF2			*0x00 (PLLF2)*
	0xDB – PLLF3			*0x00 (PLLF3)*

PLLF3.7	PLLF3.6	PLLF3.5	PLLF3.4	PLLF3.3	PLLF3.2	PLLF3.1	PLLF3.0	PLLF2.7	PLLF2.6	PLLF2.5	PLLF2.4	PLLF2.3	PLLF2.2	PLLF2.1	PLLF2.0

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

BS	OTP300NS	LLIMIT	RC	B1	8DIV	NA	MCEC
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The upper 16 bits of the PLL Setup register contain control bits related to the PLL clock oscillator, as described in the following table.

PLLF3.7	**BS**	Busy status bit for PLL SETUP REGISTER. Any write to the PLL SETUP REGISTER causes this bit to be set to "1". Once the hardware update is complete, the bit is automatically set back to "0". The PLL SETUP REGISTER should not be written while the BS bit is set to "1".
PLLF3.6 PLLF3.5 PLLF3.4 PLLF3.3	**OTP300NS**	OTP memory setup time after wakeup from sleep mode. See Section 2.4.4.1 for a detailed description of this field.
PLLF3.2 PLLF3.1	**LLIMIT**	Lock Limit. This value establishes how many reference cycles within Lock Range 0 is enough to enter lock condition. The actual Lock Limit value is computed by LLIMIT * 4 + 8. Set these bits to 01.
PLLF3.0	**RC**	Reference Clock selection: "0" – use external clock off XTAL0/XTAL1 pins "1" – use internal watchdog clock (1Mhz)
PLLF2.7	**B1**	Reference Clock divider: "0" – divided by 128 "1" – no divider
PLLF2.6 PLLF2.5	**8DIV**	8051 Clock divider selection: 00 – 8051 clock is same as system clock 01 – 8051 clock is system clock divided by 2 10 – 8051 clock is system clock divided by 3 11 – 8051 clock is system clock divided by 4
PLLF2.4	**NA**	Must be set to "0"
PLLF2.3 PLLF2.2 PLLF2.1 PLLF2.0	**MCEC**	MCE Clock (System Clock) selection: 0000 – Internal watchdog clock (1MHz) is selected 0001 – External clock is selected (1:1 mode) 0010 – PLL clock is selected 0011 – Internal watchdog clock (1Mhz) divided by 8 0100 – Internal watchdog clock (1MHz) divided by 16 0101 – Internal watchdog clock (1Mhz) divided by 32 0110 – Internal watchdog clock (1MHz) divided by 64 0111 – Internal watchdog clock (1Mhz) divided by 256 1xxx – Reserved – must not be used.

PLLF1.7	PLLF1.6	PLLF1.5	PLLF1.4	PLLF1.3	PLLF1.2	PLLF1.1	PLLF1.0	PLLF0.7	PLLF0.6	PLLF0.5	PLLF0.4	PLLF0.3	PLLF0.2	PLLF0.1	PLLF0.0
LR1		**LR0**		**PLLMPY**											
R/W		R/W		R/W											

The lower 16 bits of the PLL setup register contain control bits related to the PLL clock oscillator as described in the following table.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

PLLF1.7	**LR1**	Lock Range 1 selector. This bit influences the step size in searching for lock condition. Set this bit to 1.
PLLF1.6 PLLF1.5 PLLF1.4	**LR0**	Lock Range 0 selector. These bits influence the step size in searching for lock condition. Set these bits to 010.
PLLF1.3 PLLF1.2 PLLF1.1 PLLF1.0 PLLF0.7 PLLF0.6 PLLF0.5 PLLF0.4 PLLF0.3 PLLF0.2 PLLF0.1 PLLF0.0	**PLLMPY**	PLL clock multiplier. When PLLF2.7 = 0: MCE clock (System Clock) = Reference Clock * (1/128) * PLLMPY When PLLF2.7 = 1: MCE clock(System Clock) = Reference Clock * PLLMPY The recommended range of values is 768 <= PLLMPY <= 4086. If PLLMPY is set out of this range, the System Clock becomes unstable.

Here is an example of clock set up based on an external 4MHz crystal oscillator.

PLLF0 = 0x5A
PLLF1 = 0xAF
PLLF2 = 0x62
PLLF3 = 0x52

This combination will give MCE clock = 122.81MHz and 8051 clock = 30.703MHz. This is the maximum recommended clock speed for the 8051, limited by the 8051 program memory access speed.

Important: All 32 bits of the PLL setup register are processed at the same time, triggered by a write to PLLF3. Therefore, when initializing or modifying the PLL setup, the last operation must always be a write to PLLF3, even if its value is not being changed.

2.4.3.2 PLL Setup Register for IRMCF1xx

PLL SETUP REGISTER (PLLF4, PLLF3, PLLF2, PLLF1, PLLF0)

Address:	*0xD5 – PLLF0*	*Bit Addressable*	*Reset value:*	*0x2C (PLLF0)*
	0xD6 – PLLF1			*0x01 (PLLF1)*
	0xD7 – PLLF2			*0x00 (PLLF2)*
	0xDB – PLLF3			*0x02 (PLLF3)*
	0xDD – PLLF4			*0x08 (PLLF4)*

PLLF4.7	PLLF4.6	PLLF4.5	PLLF4.4	PLLF4.3	PLLF4.2	PLLF4.1	PLLF4.0
BS	**LR0**				**-**	**RCDIV**	
R	R/W				R/W	R/W	

Bits 32 – 39 of the PLL Setup register contain control bits related to the PLL clock oscillator, as described in the following table.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

PLLF4.7	**BS**	Busy status bit for PLL SETUP REGISTER. Any write to the PLL SETUP REGISTER causes this bit to be set to “1”. Once the hardware update is complete, the bit is automatically set back to “0”. The PLL SETUP REGISTER should not be written while the BS bit is set to “1”.
PLLF4.6 PLLF4.5 PLLF4.4 PLLF4.3	**LR0**	Lock Range 0 selector. These bits influence the step size in searching for lock condition. Set these bits to 0010.
PLLF4.2	**-**	Reserved. Must be set to “0”.
PLLF4.1 PLLF4.0	**RCDIV**	Reference Clock divider: 00 – divide by 512 01 – divide by 256 10 – divide by 128 11 – divide by 64

PLLF3.7 PLLF3.6 PLLF3.5 PLLF3.4 PLLF3.3	PLLF3.2 PLLF3.1	PLLF3.0	PLLF2.7	PLLF2.6 PLLF2.5 PLLF2.4	PLLF2.3	PLLF2.2 PLLF2.1 PLLF2.0
-	**LLIMIT**	**RC**	**B1**	**8DIV**	**NA**	**MCEC**
R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bits 16 – 31 of the PLL Setup register contain control bits related to the PLL clock oscillator, as described in the following table.

PLLF3.7 PLLF3.6 PLLF3.5 PLLF3.4 PLLF3.3	**-**	Reserved. Must be set to “0”.
PLLF3.2 PLLF3.1	**LLIMIT**	Lock Limit. This value establishes how many reference cycles within Lock Range 0 is enough to enter lock condition. The actual Lock Limit value is computed by LLIMIT * 4 + 8. Set these bits to 01.
PLLF3.0	**RC**	Reference Clock selection: “0” – use external clock off XTAL0/XTAL1 pins “1” – use internal watchdog clock (1Mhz)
PLLF2.7	**B1**	Reference Clock divider bypass (see PLLMPY): “0” – divide according to RCDIV “1” – no divider
PLLF2.6 PLLF2.5 PLLF2.4	**8DIV**	8051 Clock divider selection: 000 – 8051 clock is same as system clock 001 – 8051 clock is system clock divided by 2 010 – 8051 clock is system clock divided by 3 011 – 8051 clock is system clock divided by 4 100 – 8051 clock is system clock divided by 5 101 – 8051 clock is system clock divided by 6 110 – 8051 clock is system clock divided by 7 111 – *Reserved*
PLLF2.3	**NA**	Must be set to “0”

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

PLLF2.2 PLLF2.1 PLLF2.0	**MCEC**	MCE Clock (System Clock) selection: 000 – Internal watchdog clock (1MHz) is selected 001 – External clock is selected (1:1 mode) 010 – PLL clock is selected 011 – Internal watchdog clock (1Mhz) divided by 8 100 – Internal watchdog clock (1MHz) divided by 16 101 – Internal watchdog clock (1Mhz) divided by 32 110 – Internal watchdog clock (1MHz) divided by 64 111 – Internal watchdog clock (1Mhz) divided by 256

PLLF1.7	PLLF1.6	PLLF1.5	PLLF1.4	PLLF1.3	PLLF1.2	PLLF1.1	PLLF1.0	PLLF0.7	PLLF0.6	PLLF0.5	PLLF0.4	PLLF0.3	PLLF0.2	PLLF0.1	PLLF0.0
LR1				**PLLMPY**											
R/W				R/W											

The lower 16 bits of the PLL setup register contain control bits related to the PLL clock oscillator as described in the following table.

PLLF1.7 PLLF1.6 PLLF1.5 PLLF1.4	**LR1**	Lock Range 1 selector. These bits influence the step size in searching for lock condition. Set these bits to 0001.
PLLF1.3 PLLF1.2 PLLF1.1 PLLF1.0 PLLF0.7 PLLF0.6 PLLF0.5 PLLF0.4 PLLF0.3 PLLF0.2 PLLF0.1 PLLF0.0	**PLLMPY**	PLL clock multiplier. When PLLF2.7 = 0: MCE clock (System Clock) = Reference Clock * (1/RCDIV) * PLLMPY When PLLF2.7 = 1: MCE clock(System Clock) = Reference Clock * PLLMPY The recommended range of values is 768 <= PLLMPY <= 4086. If PLLMPY is set out of this range, the System Clock becomes unstable.

Here is an example of clock set up based on an external 4MHz crystal oscillator.

PLLF0 = 0x40
PLLF1 = 0x16
PLLF2 = 0x32
PLLF3 = 0x02
PLLF4 = 0x13

This combination will give MCE clock = 100MHz and 8051 clock = 25MHz,which is the maximum recommended clock speed for the 8051, limited by the 8051 program memory access speed.

Important: All 40 bits of the PLL setup register are processed at the same time, triggered by a write to PLLF4. Therefore, when initializing or modifying the PLL setup, the last operation must always be a write to PLLF4, even if its value is not being changed.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.4.3.3 Boot Clock Operation

During the boot process, the clock selection follows the status of the VPP/P1.5 pin. At power up, this I/O status is sampled and uses its clock accordingly for the entire boot process. This clock selection will not be changed until the first write to the PLL SETUP REGISTER to specify a runtime clock configuration.

During the boot process, VPP/P1.5 determines the clock selection as follows:

VPP/P1.5 = 1 (pull up high): external crystal oscillator is selected
VPP/P1.5 = 0 (pull down low): internal watchdog clock (1MHz) is selected

Both the MCE clock and 8051 clock follow this clock selection with the same clocks. The boot process uses this clock to copy the MCE program located in the non-volatile memory area to MCE program RAM area, followed by 8051 user program execution. During this boot process, the RESET pin is activated low.

Note that there is an internal pull up resistor to 3.3V on VPP/P1.5, so if left unconnected, the external oscillator will be used as the clock for the boot process.

2.4.4 Sleep Mode Function

The sleep mode function is provided to support a power saving mode of operation. Because of differences between OTP and flash memory, sleep mode is implemented differently for IRMCK1xx and IRMCF1xx. IRMCK1xx sleep mode is described in Section 2.4.4.1and IRMCF1xx sleep mode is described in Section 2.4.4.2.

2.4.4.1 Sleep Mode for IRMCK1xx

SLEEP MODE TIMER (SLPTMR)

Address:	*0xC4*	*Not Bit Addressable*	*Reset value:*	*0x00*

WAKE UP TIMER (WKTMR)

Address:	*0xC5*	*Not Bit Addressable*	*Reset value:*	*0x00*

PLL SETUP REGISTER (PLLF3)/OTP300NS

Address:	*0xDB*	*Not Bit Addressable*	*Reset value:*	*0x00*

PLLF3.7	PLLF3.6	PLLF3.5	PLLF3.4	PLLF3.3	PLLF3.2	PLLF3.1	PLLF3.0
-	**OTP300NS**				**-**		**-**
-	R/W				-		-

(See Section 2.4.3 for a definition of the other bit fields in this register.)

The sleep mode timer and wake up timer together with the OTP300NS bit field in the PLL SETUP REGISTER are provided to support a power saving mode of operation. In order to save power, the 8051 and OTP memory can be put into sleep mode. Sleep mode disables the OTP and makes sure it consumes the minimum power needed. In the sleep mode, power consumption drops to the microamps range at 3.3V if the MCE is also halted.

The following three registers control the sleep mode operation:

- SLPTMR – This register is located at SFR address 0xC4. When a value *n* is written to this register, the OTP immediately enters sleep mode for a number of 8051 clock cycles specified by *n* * 16 and then wakes up. For example, if a value of 100 is written to SLPTMR, the OTP enters sleep mode for 1600 clock cycles.
- WKTMR – This register is located in SFR address 0xC5 and specifies the number of 8051 clock cycles the OTP should run after exiting sleep mode before re-entering sleep mode. If WKTMR is set to zero, OTP does not re-enter sleep mode.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

- OTP300NS (PLLF3.6, PLLF3.5, PLLF3.4, PLLF3.3) – This 4-bit field specifies for how many 8051 clock cycles the OTP enable signal should be set before any read attempt can be initiated after wakeup mode. In other words, this field indicates the setup time between enable and read on the OTP. The OTP memory requires a minimum of 300 nanoseconds of wakeup time from sleep mode. For example, with a 32MHz configuration of the 8051 clock, this field should be set to 10.

There are two different modes of operation that the software can choose for sleep mode:

- Manual mode – For this mode the WKTMR register must be set to 0x00. At any given time the software can write a value to the SLPTMR register. When the register is written, the OTP enters sleep mode immediately for the specified number of 8051 clock cycles (SLPTMR * 16) after which it exits sleep mode.
- Automatic mode – In this mode the OTP enters and exits sleep mode automatically according to the values configured in WKTMR and SLPTMR. The software configures the WKTMR with the "runtime value". Writing the "sleeptime value" to SLPTMR triggers the automatic sleep mode mechanism. The OTP then alternately enters sleep mode for the specified number of 8051 clock cycles (SLPTMR * 16) and then runs for the number of 8051 clock cycles specified by WKTMR. This will go on until the software configures WKTMR with 0x00.

Remarks:

- During the chip reset the OTP is automatically put into sleep mode.
- While the OTP is in sleep mode, the 8051 is stalled and no program fetch occurs.
- Sleep mode does not automatically stall or halt the MCE processor. The MCE processor can be halted from the 8051 application by writing to the MCE CTRL register as described in Section 4.5.
- In both modes, the OTP is enabled before the read access is done according to the OTP300NS field.

2.4.4.2 Sleep Mode for IRMCF1xx

SLEEP MODE TIMER (SLPTMR)

Address:	*0xC4*	*Not Bit Addressable*	*Reset value:*	*0x00*

The sleep mode timer is provided to support a power saving mode of operation. In order to save power, the 8051 processor can be put into sleep mode to make sure it consumes the minimum power needed. In the sleep mode, power consumption drops to the microamps range at 3.3V if the MCE is also halted.

The SLPTMR register controls the sleep mode operation. When a value *n* is written to the SLPTMR register, the 8051 immediately enters sleep mode for a number of 8051 clock cycles specified by *n* * 16 and then wakes up. For example, if a value of 100 is written to SLPTMR, the 8051 enters sleep mode for 1600 clock cycles.

The software can write a value to the SLPTMR register at any time to enter sleep mode. When the register is written, the 8051 enters sleep mode immediately for the specified number of 8051 clock cycles (SLPTMR * 16) after which it exits sleep mode.

Remarks:

- During the chip reset the 8051 is automatically put into sleep mode.
- While in sleep mode, the 8051processor is stalled and no program fetch occurs.
- Sleep mode does not automatically stall or halt the MCE processor. The MCE processor can be halted from the 8051 application by writing to the MCE CTRL register as described in Section 4.5.

2.4.5 Miscellaneous Functions

EXTENDED OPERATION (EO)

Address:	*0xA2*	*Not Bit Addressable*	*Reset value:*	*00000000b*

EO.7	EO.6	EO.5	EO.4	EO.3	EO.2	EO.1	EO.0
-	**-**	**-**	**TRAP_EN**	**0**	**-**	**-**	**-**
R	R	R	R/W	R	R	R	R

EO.4 is used to select the instruction executed with the opcode 0xA5 (which is unused in the standard 8051), with bits 5 – 7 reserved for future expansion of this feature.

Bit definitions for this register are as follows:

EO.7 – EO.5	-	*Reserved for future use.*
EO.4	**TRAP_EN**	Selects the instruction to be executed by opcode 0xA5 as follows: 1 Selects software TRAP instruction. 0 Selects MOVC @(DPTR++),A – a specific instruction that supports software download into program memory implemented as RAM (see Section 2.1).
EO.3	-	*Always returns 0.*
EO.2 – EO.0	-	*Reserved for future use.*

HARDWARE CONFIGURATION (HWCFG)

Address: 0xB2 Not Bit Addressable Reset value: 00000001b

HWCFG.7	HWCFG.6	HWCFG.5	HWCFG.4	HWCFG.3	HWCFG.2	HWCFG.1	HWCFG.0
-	**-**	**-**	**OP_VAC**	**-**	**OP5**	**-**	**OP1**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

HWCFG.7	**-**	*Reserved for future use.*
HWCFG.6	**-**	*Reserved for future use.*
HWCFG.5	**-**	*Reserved for future use.*
HWCFG.4	**OP_VAC**	Op amp for AC voltage sensing control. If 1, Op amp is enabled. This Op amp is used for AC voltage sensing for PFC. Only available for PFC IRMCx100 IC.
HWCFG.3	**-**	*Not implemented. Should be set to 0.*
HWCFG.2	**OP5**	OP amp 5 control. If 1, OP amp 5 is enabled. OP amp 5 is used for leg shunt current measurement.or PFC current measurement.
HWCFG.1	**-**	*Not implemented. Should be set to 0.*
HWCFG.0	**OP1**	OP amp 1 control. If 1, OP amp 1 is enabled. OP amp 1 is used for single shunt current measurement or, in the case of leg shunts, for one of the shunts.

HARDWARE REVISION (HWREV)

Address: 0xDC Not Bit Addressable Reset value: Revision code

The hardware revision register identifies the hardware revision level. The value is fixed and the register is read-only.

IRMCK1xx (OTP memory version) –HWREV = 0x72
IRMCF1xx (Flash memory version) –HWREV = 0x77

GATEKILL CONFIGURATION (STOPS)

Address: 0xE5 Not Bit Addressable Reset value: 00000000b

STOPS.7	STOPS.6	STOPS.5	STOPS.4	STOPS.3	STOPS.2	STOPS.1	STOPS.0
-	**-**	**-**	**-**	**RSV**	**USE_ITRIP**	**-**	**GKILL_EN**
R	R	R	R	0	R/W	R	R/W

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

STOPS.7 – STOPS.4	-	*Reserved.*
STOPS.3	**RSV**	Reserved – Must be set to ‘0’
STOPS.2	**USE_ITRIP**	“0” – disables use of internal Itrip comparator for generating GATEKILL “1” – enables use of internal Itrip comparator for generating GATEKILL
STOPS.1	-	*Reserved*
STOPS.0	**GKILL_ASSERT**	“0” – sets GATEKILL signal, cannot be cleared, shuts down motor PWM “1” – GATEKILL is input from external circuit

The GATEKILL pin in the IRMCx100 series, shown in Figure 10, is a bidirectional and low true logic pin. The user can drive this pin to activate low or receive an external gatekill signal to shut off the motor PWM.

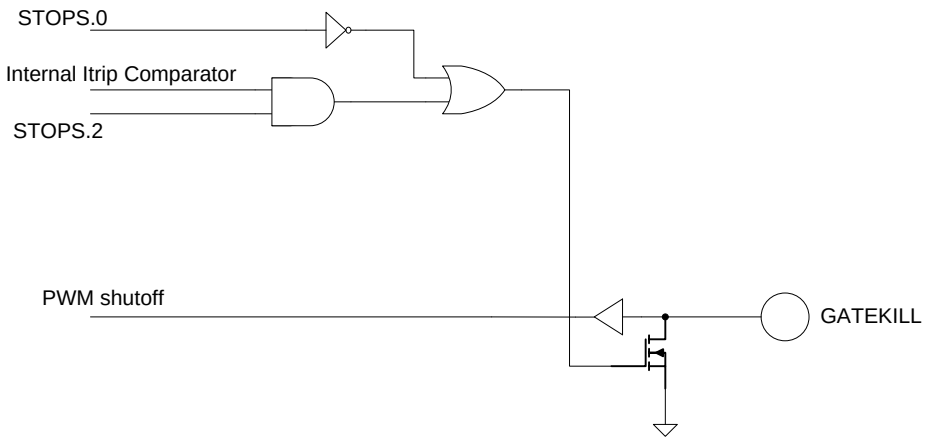


Figure 9. GATEKILL Internal Logic

As can be seen from the schematic, when the GATEKILL pin is defined as output, it will always drive the external pin to zero due to the open drain output. The GATEKILL pin should be externally pulled up to 3.3V.

The GATEKILL pin is defined as an output in either of the following cases:

- When the chip is in reset, the STOPS.0 bit is set to '0' and as a result the gatekill signal is asserted. The software normally sets this bit after reset in order to stop the gatekill activation and start normal operation. The software can, at any time, write a zero to this bit to assert GATEKILL (active low) on this pin, at which time the motor PWM is shut off.
- Use of internal over-current trip comparator: In this case, when STOPS.2 = 1 (also STOPS.0 = 1) and the internal comparator (itrip) trigger activates GATEKILL active low, the motor PWM output is shut off.

Upon power up, the user must do the following:

- Set the STOP.0 bit to '1' in order to stop the gatekill activation.
- Map the external GATEKILL signal to the relevant PWMs.
- Clear the previously pending GATEKILL indications by writing “1” to the FaultClear register.
- Set the STOPS.2 bit to '1' if the internal comparator should be used to trigger GATEKILL.

Use the FaultClear register defined in Section 3.4.1 to clear a GATEKILL fault.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.5 Interrupts

The IRMCx100 series supports various interrupt sources. These comprise the standard 8051/8052 internal interrupts and additional interrupts. The standard and extended interrupts each have separate SFR enable bits associated with them, allowing full software control. There are two levels of interrupt priority. The non-maskable interrupt source is always enabled as long as the NMI bit is set (IOCON0.0). NMI has a higher priority than any other interrupt source, and is not controllable by software.

Table 8 provides a summary of all the supported interrupts. The first column shows the interrupt number and the second column shows the 8051 interrupt vector address at the base of 8051 internal data RAM. The third column identifies the interrupt source. The forth and fifth columns indicate which interrupt enable bit, in either register IE or IE1, is associated with the interrupt. (See Section 2.5.3 for more information about the interrupt enable registers.)

Interrupt Number	Vector Address (hex)	Interrupt Source	IE Bit Number	IE1 Bit Number
0	0003	NINT0	0	
1	000B	Timer 0	1	
2	0013	NINT1	2	
3	001B	Timer 1	3	
4	0023	UART	4	
5	002B	Timer 2	5	
6	0033	INT2		0
7	003B	Sync		1
8	0043	Fault		2
9	004B	*Reserved*		3
10	0053	MCE		4
11	005B	Capture		5
12	0063	*Unused*	-	-
13	006B	*Unused*	-	-
14	0073	NMI	-	-

Table 8. Interrupt Source Summary

2.5.1 Standard Interrupts

The standard interrupts comprise three timer (Timer/Counter 0, 1 and 2) overflow interrupts, an interrupt associated with the core's built-in serial interface, and two maskable external interrupts (NINT0 and NINT1).

The Timer overflow interrupts, TF0, TF1, and TF2, are set whenever Timers 0, 1, and 2, respectively, rollover to zero. The states of these interrupts are stored in the TCON and T2CON registers. TF0 and TF1 (but not TF2) are automatically cleared by hardware on entry to the corresponding interrupt service routine.

The legacy external interrupts, NINT0 and NINT1, are driven from inputs P3.2 and P3.3 respectively. These interrupts may be either edge or level sensitive, depending on settings within the TCON register. Two further TCON register bits, IE0 and IE1, act as interrupt flags. If the external interrupt is set to edge-triggered, the corresponding register bit IE0/1 is set by a falling edge on NINT0/1 and cleared by hardware on entry to the corresponding interrupt service routine. If the interrupt is set to be level sensitive, IE0/1 reflects the logic level on NINT0/1.

Note: All events on NINT0 and NINT1, whether level-triggered or edge-triggered, are detected by sampling the relevant interrupt line on the rising edge of the 8051 clock at the end of Phase 1 of every PCLK cycle. Where NINT0/NINT1 is level-triggered, a response is made to the signal being sampled low and, to ensure detection, the external source needs to hold the line low until the resulting interrupt is generated. (It also needs to ensure that the request is deactivated before the end of the associated service routine.) Where NINT0/NINT1 is edge-triggered, the response is made to a transition on the signal from high to low between successive samples. This means that, to ensure detection, NINT0/NINT1 needs to have been high for at least two clocks before it goes low and then needs to be held low for at least two clocks after this transition.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.5.2 Extended Interrupts

The extended interrupts include the following:

- external interrupt 2 (INT2)
- SYNC interrupt
- MCE interrupt
- Fault interrupt
- Capture Timer
- two software interrupts

These interrupts are all level-sensitive (low true) except External Interrupt 2 (INT2), which is configured using register IOCON1 (Section 2.4.2).

The internal interrupt line is sampled on the rising edge of PCLK at the beginning of Phase 2 of the last cycle of the current instruction.

See Section 4.6 for a description of the SYNC, MCE and fault interrupts, which are generated by the MCE.

2.5.3 Enabling Interrupts

The Non-Maskable Interrupt is always enabled. The maskable interrupts are enabled through a pair of bit-addressable Interrupt Enable registers (IE and IE1).

The bits of the IE register and the IE1 register each individually enable/disable a particular interrupt source, except IE.7. Overall control is provided by bit 7 of IE (EA). When EA is set to '0', all interrupts (except the NMI) are disabled; when EA is set to '1', interrupts are individually enabled or disabled through the other bits of the Interrupt Enable Registers.

Both IE and IE1 are bit-addressable. The details of the registers are given below.

INTERRUPT ENABLE 0 (IE)

Address: 0xA8 Bit Addressable Reset value: 00000000b

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	**-**	**ET2**	**EU0**	**ET1**	**EX1**	**ET0**	**EX0**
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

For each bit in this register, "1" enables the corresponding interrupt and "0" disables it. The allocation of interrupts to bits is as follows:

IE.7	**EA**	Enable or disable all interrupt bits.
IE.6	-	*Not implemented. Returns zero when read.*
IE.5	**ET2**	Enable Timer 2 overflow interrupt.
IE.4	**EU0**	Enable UART 0 interrupt.
IE.3	**ET1**	Enable Timer 1 overflow interrupt.
IE.2	**EX1**	Enable External Interrupt 1 (NINT1, P3.3)
IE.1	**ET0**	Enable Timer 0 overflow interrupt.
IE.0	**EX0**	Enable External Interrupt 0 (NINT0, P3.2)

INTERRUPT ENABLE 1 (IE1)

Address: 0xE8 Bit Addressable Reset value: 00000000b

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

IE1.7	IE1.6	IE1.5	IE1.4	IE1.3	IE1.2	IE1.1	IE1.0
-	**-**	**ET4**	**EMCE**	**-**	**EFLT**	**ESYNC**	**EINT2**
R	R	R/W	R/W	R	R	R/W	R/W

For each bit in this register, a 1 enables the corresponding interrupt, and a 0 disables it. The allocation of interrupts to bits is as follows:

IE1.7	**-**	*Not implemented. Returns zero when read.*
IE1.6	**-**	*Not implemented. Returns zero when read.*
IE1.5	**ET4**	Enable Capture Timer interrupt
IE1.4	**EMCE**	Enable MCE interrupt
IE1.3	**-**	*Reserved*
IE1.2	**EFLT**	Fault Interrupt
IE1.1	**ESYNC**	Enable SYNC Interrupt
IE1.0	**EINT2**	Enable External Interrupt 2 (INT2)

INTERRUPT STATUS AND ACKNOWLEDGE (IS)

Address: 0xC0 Bit Addressable Reset value: 00000000b

IS.7	IS.6	IS.5	IS.4	IS.3	IS.2	IS.1	IS.0
-	**-**	**T4S**	**MCES**	**-**	**FLTS**	**SYNCS**	**INT2S**
R/W	R/W	R/W	R/W	R	R	R/W	R/W

For each bit in this register, a 1 indicates that the associated interrupt has occurred and is pending. When written 0, the pending interrupt is cleared. (Note that in general it is not necessary to write to this register since most interrupts are cleared automatically when the interrupt is serviced.)

The allocation of status and acknowledge bits are as follows:

IS.7	**-**	*Not implemented. Returns zero when read.*
IS.6	**-**	*Not implemented. Returns zero when read.*
IS.5	**T4S**	Capture Timer interrupt is pending. Write 0 to reset
IS.4	**MCES**	MCE interrupt is pending. Write 0 to reset.
IS.3	**-**	*Reserved*
IS.2	**FLTS**	Fault interrupt is pending. Write 0 to reset.
IS.1	**SYNCS**	SYNC interrupt is pending. Write 0 to reset.
IS.0	**INT2S**	External Interrupt 2 (INT2) is pending. Write 0 to reset.

2.5.4 Interrupt Priority

The standard 8051 architecture supports a two-level interrupt priority scheme. Under the two-level priority scheme, the priority level is decided solely on the IP and IP1 value.

Details of the registers are given below. IP and IP1 are bit-addressable.

Note: No priority level is assigned to the NMI. It simply takes precedence over all other interrupts.

INTERRUPT PRIORITY (IP)

Address: 0xB8 Bit Addressable Reset value: 00000000b

IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
-	**-**	**PT2**	**PU0**	**PT1**	**PX1**	**PT0**	**PX0**

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

R	R	R/W	R/W	R/W	R/W	R/W	R/W

For each bit of the IP register, "1" selects high priority for the interrupt enabled by the corresponding bit of the IE register, while "0" selects low priority for this interrupt.

The allocation of interrupts to bits is as follows:

IP.7	**-**	*Reserved.*
IP.6	**-**	*Reserved.*
IP.5	**PT2**	Select priority for Timer 2 overflow Interrupt.
IP.4	**PU0**	Select priority for UART 0 Interrupt.
IP.3	**PT1**	Select priority for Timer 1 overflow Interrupt.
IP.2	**PX1**	Select priority for External Interrupt 1. (P3.3)
IP.1	**PT0**	Select priority for Timer 0 overflow Interrupt.
IP.0	**PX0**	Select priority for External Interrupt 0. (P3.2)

INTERRUPT PRIORITY 1 (IP1)

Address: 0xF8 Bit Addressable Reset value: 00000000b

IP1.7	IP1.6	IP1.5	IP1.4	IP1.3	IP1.2	IP1.1	IP1.0
-	**-**	**PT4**	**PMCE**	**-**	**PFLT**	**PSYNC**	**PINT2**
R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

For each bit of the IP1 register, "1" selects high priority for the interrupt enabled by the corresponding bit of the IE1 register, while "0" selects low priority for this interrupt.

The allocation of interrupts to bits is as follows:

IP1.7	**-**	*Not implemented. Returns zero when read.*
IP1.6	**-**	*Not implemented. Returns zero when read.*
IP1.5	**PT4**	Select priority for Capture Timer Interrupt
IP1.4	**PMCE**	Select priority for MCE Interrupt
IP1.3	**-**	*Reserved*
IP1.2	**PFLT**	Select priority for Fault Interrupt
IP1.1	**PSYNC**	Select priority for SYNC Interrupt
IP1.0	**PINT2**	Select priority for External Interrupt 2

2.5.5 Service Order

An interrupt service routine may only be interrupted by an interrupt of higher priority and, if two interrupts of different priority occur at the same time, the higher level interrupt will be serviced first. An interrupt cannot be interrupted by another interrupt of the same or a lower priority level.

If two interrupts of the same priority level occur simultaneously, a polling sequence is observed as follows:

Source	Level	Description
NMI	0 (Highest)	Non-Maskable Interrupt
NINT0	1	External Interrupt 0 from P3.2/NINT0
TF0	2	Timer/Counter 0 Interrupt
NINT1	3	External Interrupt 1 from P3.3/NINT1
TF1	4	Timer/Counter 1 Interrupt
U0	5	UART Interrupt
TF2	6	Timer/Counter 2 Interrupt

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

INT2	7	External Interrupt 2 (INT2)
SYNC	8	SYNC Interrupt
FLT	9	Fault Interrupt
-	10	*Reserved*
MCE	11	MCE Interrupt
T4	12 (Lowest)	Capture Timer Interrupt

Table 9. Interrupt Service Order

2.5.6 Interrupt Latency

The response time in a single interrupt system is between three and nine PCLK cycles.

2.5.7 Interrupt Vectors

When an interrupt is serviced, a long call instruction is executed to an interrupt vector address determined by the source of the interrupt. The vector associated with each interrupt source is shown in Table 8.

2.6 Timers

This section describes the three general-purpose timer/counter devices and the special-purpose timers: a watchdog timer, a capture timer and three MCE pin timers.

2.6.1 Timer Prescaler

A prescaler register PSCL can be configured to divide the clock on input to the three general-purpose timers. The clock rate on input to each of the three timers can be individually configured as described below.

The clock input to the watchdog timer and capture timer cannot be prescaled.

TIMER PRESCALER (PSCL)

Address: 0xA9 Not Bit Addressable Reset value: 00000000b

PSCL.7	PSCL.6	PSCL.5	PSCL.4	PSCL.3	PSCL.2	PSCL.1	PSCL.0
-	**-**	**P1(2)**	**P0(2)**	**P1(1)**	**P0(1)**	**P1(0)**	**P0(0)**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for the PSCL register are as follows:

PSCL.7	**-**	*Not implemented. Returns zero when read.*
PSCL.6	**-**	*Not implemented. Returns zero when read.*
PSCL.5	**P1(2)**	Timer 2 prescale control bit P1.
PSCL.4	**P0(2)**	Timer 2 prescale control bit P0.
PSCL.3	**P1(1)**	Timer 1 prescale control bit P1.
PSCL.2	**P0(1)**	Timer 1 prescale control bit P0.
PSCL.1	**P1(0)**	Timer 0 prescale control bit P1.
PSCL.0	**P0(0)**	Timer 0 prescale control bit P0.

For all three timers, the prescale bits P0 and P1 apply as follows:

P1	P0	Function
0	0	No prescaler. The counting clock is half the 8051 clock.
0	1	PCLK is divided by 16 (or 8051 clock divided by 32).
1	0	PCLK is divided by 256 (or 8051 clock divided by 512).
1	1	PCLK is divided by 4096 (or 8051 clock divided by 8192).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.6.2 General-Purpose Timer/Counters

Two 16-bit timer/counters are provided, Timer 0 and Timer 1. The TCON and TMOD registers are used to set the mode of operation and to control the running and interrupt generation of these two devices, with the timer/counter values stored in two pairs of 8-bit registers (TL0, TH0 and TL1, TH1).

TIMER /COUNTER CONTROL (TCON)

Address: 0x88 Bit Addressable Reset value: 00000000b

TCON.7	TCON.6	TCON.5	TCON.4	TCON.3	TCON.2	TCON.1	TCON.0
TF1	**TR1**	**TF0**	**TR0**	**IEDG1**	**IT1**	**IEDG0**	**IT0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

TCON.7	**TF1**	Timer 1 overflow flag. Set by hardware when Timer/Counter 1 overflows. Cleared by hardware when the processor calls the interrupt service routine.
TCON.6	**TR1**	Timer 1 run control. If 1, timer runs; if 0, timer is halted.
TCON.5	**TF0**	Timer 0 overflow flag. Set by hardware when Timer/Counter 0 overflows. Cleared by hardware when the processor calls the interrupt service routine.
TCON.4	**TR0**	Timer 0 run control. If 1, timer runs; if 0, timer is halted.
TCON.3	**IEDG1**	External Interrupt 1 edge flag. Set by hardware when an External Interrupt 1 edge is detected.
TCON.2	**IT1**	External Interrupt 1 control bit. If 1, External Interrupt 1 is “edge-triggered”; if 0, External Interrupt 1 is “level triggered” (see Section 2.5.1).
TCON.1	**IEDG0**	External Interrupt 0 edge flag. Set by hardware when an External Interrupt 0 edge is detected.
TCON.0	**IT0**	External Interrupt 0 control bit. If 1, External Interrupt 0 is “edge-triggered”; if 0, External Interrupt 0 is “level triggered” (see Section 2.5.1).

TIMER /COUNTER MODE (TMOD)

Address: 0x89 Not Bit Addressable Reset value: 00000000b

TMOD.7	TMOD.6	TMOD.5	TMOD.4	TMOD.3	TMOD.2	TMOD.1	TMOD.0
GATE1	**C/NT1**	**M1(1)**	**M0(1)**	**GATE0**	**C/NT0**	**M1(0)**	**M0(0)**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

TMOD.7	**GATE1**	Timer 1 gate flag. When TCON.6 is set and GATE1 = 1, Timer/Counter 1 will only run if NINT1 pin is 1 (hardware control). When GATE1 = 0, Timer/Counter 1 will only run if TCON.6 = 1(software control).
TMOD.6	**C/NT1**	Timer/Counter 1 selector. If 0, input is from 8051 clock; if 1, input is from T1 pin.
TMOD.5	**M1(1)**	Timer 1 Mode control bit M1.
TMOD.4	**M0(1)**	Timer 1 Mode control bit M0.
TMOD.3	**GATE0**	Timer 0 gate flag. When TCON.4 is set and GATE0 = 1, Timer/Counter 0 will only run if NINT0 pin is 1 (hardware control). When GATE0 = 0, Timer/Counter 0 will only run if TCON.4 = 1(software control).
TMOD.2	**C/NT0**	Timer/Counter 0 selector. If 0, input is from 8051 clock; if 1, input is from T0 pin.
TMOD.1	**M1(0)**	Timer 0 Mode control bit M1.
TMOD.0	**M0(0)**	Timer 0 Mode control bit M0.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

For both timer/counters, the mode bits M0 and M1 apply as follows:

M1	M0	Operating Mode
0	0	13-bit timer/counter (M8048 compatible mode).
0	1	16-bit timer/counter.
1	0	8-bit auto-reload timer/counter.[1]
1	1	Timer 0 is split into two halves. TL0 is an 8-bit timer/counter controlled by the standard Timer 0 control bits. TH0 is an 8-bit timer/counter controlled by the standard Timer 1 control bits. TH1 and TL1 are held (Timer 1 is stopped).

Note 1. When the IRMCK1xx is used in auto-reload mode with a non-zero prescaler value (PSCL register), no interrupt is generated when the counter overflows. For the IRMCF1xx, an interrupt is generated in this condition.

TIMER / COUNTER DATA (TL0, TL1, TH0, TH1)

Address: 0x8A (TL0), 0x8B (TL1), 0x8C (TH0), 0x8D (TH1) *Not Bit Addressable* *Reset value: 0x00*

TL0 and TH0 are the low and high bytes of Timer/Counter 0 respectively. TL1 and TH1 are the low and high bytes of Timer/Counter 1 respectively. In Mode 2 (TMOD[5:4] = 10), the TL register is an 8-bit counter while TH stores the reload value.

On reset, all timer/counter registers are 0x00.

TIMER / COUNTER 2 CONTROL (T2CON)

Address: 0xC8 *Bit Addressable* *Reset value: 00000000b*

T2CON.7	T2CON.6	T2CON.5	T2CON.4	T2CON.3	T2CON.2	T2CON.1	T2CON.0
TF2	**EXF2**	**RCLK**	**TCLK**	**EXEN2**	**TR2**	**C/NT2**	**CP/NRL2**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit definitions for this register are as follows:

T2CON.7	**TF2**	Timer 2 overflow flag. Set by hardware when Timer/Counter 2 overflows unless either RCLK or TCLK is set to 1. This bit is not cleared by hardware when the processor calls the interrupt service routine.
T2CON.6	**EXF2**	Timer 2 external flag. This bit is set when a capture or reload is triggered by a negative transition on T2EX and EXEN2 is set to 1. If the Timer 2 interrupt is enabled, setting this bit will cause an interrupt to the Timer 2 vector.
T2CON.5	**RCLK**	If this bit is set, the Serial Port receive clock is driven from the overflow pulses of Timer 2.
T2CON.4	**TCLK**	If this bit is set, the Serial Port transmit clock is driven from the overflow pulses of Timer 2.
T2CON.3	**EXEN2**	Timer 2 External interrupt enable flag. When set, a negative edge on T2EX will trigger a capture or auto-reload.
T2CON.2	**TR2**	Run control bit for Timer 2. If set to 1, the timer is enabled.
T2CON.1	**C/NT2**	Timer/Counter select. A 0 selects internal timer mode; a 1 selects external counter mode.
T2CON.0	**CP/NRL2**	Capture/Reload control. When set, captures occur on negative transitions of T2EX (if EXEN2 is set). When 0, auto-reloads are performed on timer overflows or on negative transitions of T2EX(if EXEN2 is set). If either RCLK or TCLK is 1, this bit is ignored and auto-reloads are performed on timer overflows.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

TIMER / COUNTER 2 DATA (RCP2L, RCP2H, TL2, TH2)

Address:	*0xCA (RCP2L), 0xCB (RCP2H), 0xCC (TL2), 0xCD (TH2)*	*Not Bit Addressable*	*Reset value:*	*0x00*

TL2 and TH2 are the low and high bytes of Timer/Counter 2 data. RCP2L and RCP2H are the low and high bytes of the Timer 2 capture registers. These registers are also used for auto-reload.

On reset all these registers are 0x00.

2.6.2.1 Modes of Operation

The IRMCx100 series includes three general-purpose counter/timers (Timers 0, 1 and 2). For each timer's counter value, there are two 8-bit special function registers (SFRs), one providing the low byte of the timer/counter value and the other the high byte. Further SFRs are used to configure and control the timers.

In Timer mode, the counter/timers count PCLK cycles; in Counter mode, the counter/timers count high-to-low transitions on the corresponding input pin (T0, T1, T2). Applications for the timers include measuring the time interval between events, counting events and generating a signal at regular intervals.

Timers 0 and 1 can each be configured either as a 16-bit counter/timer, a 13-bit counter/timer, or an 8-bit auto-reload counter/timer. (The auto-reload option allows automatic resetting in a counter counting up to 256.) Alternatively, Timer 0 can be split into two 8-bit timers. (The 13-bit mode provides an 8-bit counter with a divide-by-32 prescaler and is included solely for compatibility with Intel 8048 devices.)

Timer 2 has two modes of operation: a capture mode in which the current value of the timer is captured into the RCP2L and RCP2H registers; and an auto-reload mode in which Timer 2 is automatically reloaded with the contents of RCP2L and RCP2H.

2.6.2.2 Configuring the Timers

Timers 0 and 1 are configured by writing to the TMOD register. The high-order nibble of the register controls Timer 1 while the lower nibble controls Timer 0.

The controls included comprise a GATE flag which sets whether the timer is under hardware or software control (Bits 7 and 3 respectively); a Counter/Timer selector which controls whether the timer is to be triggered by the internal clock or from the corresponding input pin (Bits 6 and 2 respectively) and a pair of bits which select the mode in which the timer is used (Bits 5 & 4 and 1 & 0, respectively).

Once the mode is configured, the operation of Timers 0 and 1 is controlled using the TCON register.

2.6.2.3 Using the Timers to Measure a Time Interval

When triggered by the internal clock, the timers are incremented once every PCLK cycle, which is comprised of two 8051 clock cycles.

To use any of the timers to measure a time interval, you therefore need to:

1. Set the timer you propose to use into the required timer mode by setting the appropriate mode bits.
2. Set the corresponding C/NT bit to 0 so that the timer is triggered by the internal clock.
3. Use the GATE bit to put the running of the timer under hardware or software control as appropriate.
4. Set the required initial value for the timer by writing to the associated SFRs.
5. Start timing by setting the appropriate Run bit with an instruction such as SETB TR1.
6. Stop timing by clearing the Run bit (CLR TR1)
7. Calculate the elapsed time by dividing the difference between the initial and the final setting by half of the 8051 clock frequency.

Note: In 16-bit mode, a timer can count up to 65,536, which is equivalent to 65,536x2/f_{osc} seconds (where f_{osc} is the input clock frequency). In 13-bit mode, it can count up to 8,192 or 8,192x2/f_{osc} seconds, while in an 8-bit mode, it can

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

count up to 256 or 256x2/f_{osc} seconds. To measure longer periods than these, a count must be kept of the number of times the timer overflows.

Note also that the total of a number of separate time intervals, for example the total time that some device is switched on, can be determined simply by stopping and starting the timer at the appropriate points during the period over which the time is measured.

Each time the timer is restarted, it will continue counting from the value at which it was previously halted, giving the required cumulative total.

2.6.2.4 Using the Timers to Signal when a Defined Period has Elapsed

A timer can be used to signal when a defined period has elapsed by letting the timers count PCLK cycles. There are two approaches that can be taken.

The first option is to calculate the value that you will require the timer to count up to (either once or many times) and use a CJNE instruction, for example, to watch for when the timer reaches the required value.

The alternative approach is to set the initial value of the timer equal to its maximum setting minus the value you calculate and use the corresponding overflow flag to signal when the required time has elapsed. In particular, you may be able to use a timer in its Auto-Reload mode to count from this initial value up to overflow as many times as are required. The application just needs to keep track of the number of times the timer has overflowed.

Note: The overflow flags for Timer 0 and Timer 1 (TF0 and TF1) are in the TCON register. The overflow flag for Timer 2 (TF2) is included in the T2CON register.

2.6.2.5 Using the Timers as Event Counters

To use any of the timers as an event counter, you need to put it into counter mode by setting the corresponding C/NT bit to 1. Then, rather than counting PCLK cycles, the timer will count high-to-low transitions on the corresponding timer input line (T0, T1 or T2).

Note: T0, T1 and T2 are alternate uses of PORT2[5], PORT2[4] and PORT1[0] respectively. You should also note that the timer input lines are sampled once every PCLK cycle (at the end of the second phase) and the count is incremented when the samples record a high in one cycle and a low in the following cycle. Recognizing a transition therefore takes two PCLK cycles or four 8051 clock cycles. Events that have a shorter time period will be undersampled and will therefore be not recognized reliably.

2.6.2.6 Reading the Timers

Timers 0 and 1 are straightforward to read when they are operating in one of their 8-bit modes as all that is required is a simple read of the appropriate SFR. Reading a timer that is operating in either a 13-bit or a 16-bit mode, however, takes two cycles – leaving you open to the risk that the timer may change its value between the two reads.

Two strategies may help here. One approach is to read the high byte, then read the low byte, then read the high byte again. If the high byte hasn't changed, the readings made correctly record the value of the timer at the time the low byte was read. If the high byte has changed, however, the readings should be made again as the values read give an uncertain result.

The other option is to stop the timer by clearing the appropriate Run bit while the reading is made. However, this approach should only be taken if the application for which the IRMCx100 series is being used can tolerate the timer being stopped for a short while.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.6.3 Watchdog Timer

The watchdog timer is a fail-safe mechanism that generates an automatic chip reset if the 8051 software fails to execute its intended program sequence for any reason. As with typical watchdog timers, the program must reset a counter on a regular basis to maintain normal operation; if the software fails to reset the counter (the program is not operating properly) the watchdog "times out" and resets the device. Unlike a typical watchdog timer, however, the IRMCx100-series watchdog is driven by an internal analog oscillator so the timer will continue to run even if the external oscillator fails. In this case software execution halts, but the watchdog times out as it should and generates a chip reset so that all signals are reset to their initial power-on values and the motor drive reverts to an idle state.

The watchdog timer is a 16-bit timer with an associated 16-bit watchdog limit register. The clock is taken directly from a special on-chip independent 1 MHz (approximate) analog oscillator, divided by 16 and fed to the watchdog timer. When the 16-bit watchdog counter reaches (counts up to) the value in the 16-bit watchdog limit, an internal reset is generated. Whenever a user program writes or reads the 16-bit watchdog limit register, the 16-bit watchdog counter is reset to zero and starts counting again toward the specified value in the limit register. The block diagram is shown in Figure 10.

After power-on reset, the watchdog limit register is set to 0xFFFE and the watchdog timer is enabled. Every time the high byte of the limit register is read while the timer is running, the counter starts again from zero. The counter is also reset to zero if the limit register is written with a non-zero value while the timer is running. Writing "0" to the limit register disables the watchdog timer.

With the watchdog enabled and the limit register set to the maximum value, the counter reaches the limit value in approximately one second. Therefore, the user program must reset the counter (by reading or writing the limit register) at regular intervals of less than one second to avoid a system reset. If it is desirable to detect a software "hang" in less than one second, the value in the limit register can be adjusted accordingly.

The watchdog timer is always disabled when the system is in the debug mode (the JTAG debug port is active).

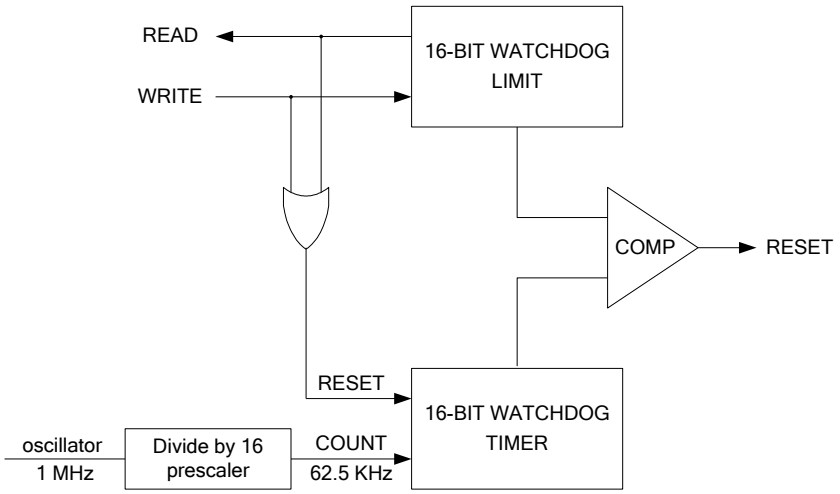


Figure 10. Watchdog Timer

WATCHDOG LIMIT REGISTER (WDLIM)

Address:	*0xF126 (8051, low byte)* *0xF127 (8051, high byte)* *0xE93 (MCE)*	*Range:*	*Unsigned, 0 – 65535*	*Reset value:*	*0xFFFE*

The watchdog limit register is read/write. When the limit register is read, the 16-bit watchdog counter is cleared to zero and begins counting up again. When the limit register is written, the specified 16-bit value is stored as the new counter limit and at the same time the 16-bit watchdog timer is cleared and begins counting up again.

WATCHDOG COUNT REGISTER (WDCNT)

Address:	*0xF258 (8051, low byte)* *0xF259 (8051, high byte)* *0xF2C (MCE)*	*Range:*	*Unsigned, 0 – 65535*	*Reset value:*	*0x0000*

The watchdog count register is read-only and available only for the IRMCF1xx. It can be used to read the current 16-bit watchdog counter value to verify that the watchdog timer is operational.

2.6.4 Capture Timer

The capture timer is used to measure the duty cycle and frequency period of an external signal. A number of different source signals can be assigned to capture timer input through the input signal selection register (INSEL). The counting frequency resolution is selected using the capture prescaler register (CAPPS).

Two 16-bit counter values are latched into the capture previous pulse time (CAPPRV) and capture most recent pulse time (CAPLST) registers as the pulse times are measured. The interrupt polarity setup in register capture edge selection register (CAPEG) determines at which edge an interrupt is generated, and also which waveform segment is considered the “last pulse time” and which is considered the “previous pulse time”.

Figure 11 shows the behavior of the capture timer for both rising-edge and falling-edge configuration.

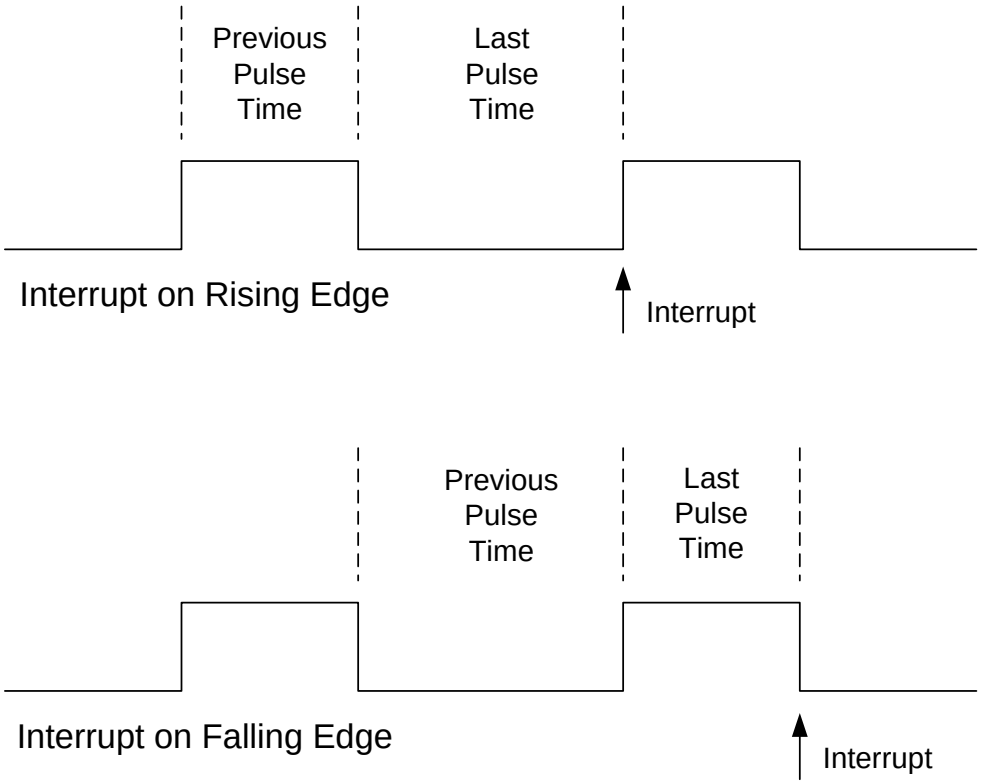


Figure 11. Capture Timer Operation

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The pulse time register CAPLST must be read before the CAPPRV register. When CAPLST is read, the values in both registers are held (no new values are latched into the registers). This ensures that coherent data from a single duty cycle can be retrieved. When CAPPRV is read, the registers are released and the timer begins to sample new values. The CAPPRV register must be read in order to re-enable the timer.

CAPTURE PRESCALER CONFIGURATION (CAPPS)

Address:	*0xF1E0 (8051, low))* *0xF1E1 (8051, high)* *0xEF0 (MCE)*	*Range:*	*Unsigned, 0 – 65535*	*Reset value:*	*0x0000*

This 16-bit register defines a prescale value for the input clock (SYSCLK) to determine the counting frequency resolution. For example, if the CAPPS register is set to 256, the input signal is captured once every 256 clock cycles.

CAPTURE EDGE SELECTION (CAPEG)

Address:	*0xF1E2 (8051)* *0xEF1 (MCE)*	*Range:*	*Unsigned, 0 – 34*	*Reset value:*	*0x00*

CAPEG.7	CAPEG.6	CAPEG.5	CAPEG.4	CAPEG.3	CAPEG.2	CAPEG.1	CAPEG.0
-	-	-	-	-	-	-	**CE 0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

A value of 1 in the CE0 bit configures falling-edge operation. A value of 0 configures rising-edge operation. Refer to Figure 11 for an illustration of capture timer behavior in each mode.

CAPTURE LAST TIME DATA (CAPLST)

Address:	*0xF23C (8051, low)* *0xF23D (8051, high)* *0xF1E (MCE)*	*Range:*	*Unsigned, 0 – 65535*	*Reset value:*	*0x0000*

The read-only CAPLST data register is used to read the 16-bit counter value for the most recent pulse time period measured by the capture timer. CAPLST should be read first to hold the current values in the capture timer data registers. On the 8051, read the register's low byte (CAPLSTL) first.

CAPTURE PREVIOUS TIME DATA (CAPPRV)

Address:	*0xF23E (8051, low)* *0xF23F (8051, high)* *0xF1F (MCE)*	*Range:*	*Unsigned, 0 – 65535*	*Reset value:*	*0x0000*

The read-only CAPPRV data register is used to read the 16-bit counter value for the pulse time period immediately preceding the "last time" period. CAPPRV should be read last to release the capture timer data registers, and must be read to re-enable the capture timer. (On the 8051, read the register's high byte (CAPPRVH) last.

2.6.5 MCE Pin Timers

The three MCE pin timers can be used to produce a level change or pulse on an output pin at a specific point within a global PWM cycle, where the term "global PWM cycle" is defined as follows:

- In IRMCx100 IC with only motor control, where the system consists of a single motor component, a global PWM cycle is the same as a PWM cycle, and the length of the cycle depends on the PWM period configuration for the motor.
- In IRMCx100 IC with PFC and motor control, the system consists of motor and PFC components and the two components can be configured to different PWM frequencies (where one is a multiple of the other). In this

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

case, the global PWM cycle refers to the longer of the two periods. For example, if the PFC is configured for three times the PWM frequency of the motor, the motor PWM frequency determines the length of the global PWM cycle and there are three PFC PWM cycles within each global cycle.

Each of the three MCE pin timers is associated with a specific general-purpose I/O pin, as follows:

MCE pin timer 1	P1.0
MCE pin timer 2	P2.7
MCE pin timer 3	P3.1

Note: The MCE pin timers are primarily intended for use by the MCE program. However, the MCE program implementation varies depending on the MCE product application library (PAL) and a given implementation may not use all three of the MCE pin timers. If availability of a particular MCE pin timer has been confirmed with International Rectifier for the selected MCE PAL, the 8051 application may program that timer as described in this section.

To use an MCE pin timer, its I/O pin must be configured as an output as described in Section 2.4.2 and the appropriate bit must be set in the MCE pin timer enable register described below.

Three configuration registers are associated with each MCE pin timer:

Timer Mode	The mode register configures the timer's mode of operation (described below) and the starting level of the associated output pin (0 or 1). The timer is armed when the mode register is written.
Timer Period	The timer period register must be set according to the PWM cycle configuration.
Timer Count	The timer count determines the point within a PWM cycle that the timer is to expire.

When enabled, each timer can be configured in one of four modes:

Asynchronous Mode	In asynchronous mode, the output pin is set to the specified level at the time the mode register is written. In this mode the level change is immediate and the timer count is not used.
Single Shot Mode	In single shot mode, the output pin is set to the specified level at the time the mode register is written. The timer begins counting at the start of the next PWM cycle after the mode register is written and the output pin changes level when the timer count expires within that cycle. The level of the output pin is not changed again until the mode register is reprogrammed.
Single Pulse Mode	In single pulse mode, the output pin is set to the specified level at the time the mode register is written. The timer begins counting at the start of the next PWM cycle after the mode register is written and the output pin changes level when the timer count expires within that cycle. At the end of the same cycle, the output pin reverts to the starting level and the level is not changed again until the mode register is reprogrammed.
Repetitive Mode	In repetitive mode, the output pin is set to the specified level at the time the mode register is written. The timer begins counting at the start of the next PWM cycle after the mode register is written and the output pin changes level when the timer count expires within that cycle. At the end of the same cycle, the output pin reverts to the starting level and the counter is reset for the next cycle. Until the mode register is reprogrammed or the timer is disabled, the output pin changes level at the same point within each cycle and reverts to the original level at the end of each cycle.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MCE PIN TIMER ENABLE (MTENB)

Address:	*0xF1E6 (8051)* *0xEF3 (MCE)*	*Range:*	*Bit field definitions*	*Reset value:*	*0x00*

MTENB.7	MTENB.6	MTENB.5	MTENB.4	MTENB.3	MTENB.2	MTENB.1	MTENB.0
-	**-**	**-**	**-**	**-**	**ENB3**	**ENB2**	**ENB1**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Each of the MCE pin timer enable bits enables or disables the associated MCE pin timer as shown below:

MTENB.7	**-**	*Not implemented. Returns zero when read.*
MTENB.6	**-**	*Not implemented. Returns zero when read.*
MTENB.5	**-**	*Not implemented. Returns zero when read.*
MTENB.4	**-**	*Not implemented. Returns zero when read.*
MTENB.3	**-**	*Not implemented. Returns zero when read.*
MTENB.2	**ENB3**	0 = disable MCE pin timer 3; 1 = enable.
MTENB.1	**ENB2**	0 = disable MCE pin timer 2; 1 = enable.
MTENB.0	**ENB1**	0 = disable MCE pin timer 1; 1 = enable.

MCE PIN TIMER PERIOD (MTPER1, MTPER2, MTPER3)

MTPER1	*Address:*	*0xF1F0 (8051)* *0xEF8 (MCE)*	*Range:*	*Unsigned, 0 – 15*	*Reset value:*	*0x00*
MTPER2		*0xF1FA (8051)* *0xEFD (MCE)*				
MTPER3		*0xF202 (8051)* *0xF01 (MCE)*				

MTPER.7	MTPER.6	MTPER.5	MTPER.4	MTPER.3	MTPER.2	MTPER.1	MTPER.0
-	**-**	**-**	**-**	**PER3**	**PER2**	**PER1**	**PER0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The four-bit timer period value (PER3 – PER0) for each of the three MCE pin timers must be set to a defined value based on the PWM cycle ratio configured for the system. For the IRMCx171, the ratio is always 1 and the MCE pin timer period registers should always be set to 0. For IRMCF188, the ratio between motor PWM period and PFC PWM period is configurable and the MCE pin timer period registers should be set to the ratio minus 1. For example, if the PFC PWM frequency is three times the motor PWM frequency, the PWM ratio is 3 and the MCE pin timer period registers should be set to 2.

MCE PIN TIMER COUNT (MTC1, MTC2, MTC3)

MTC1	*Address:*	*0xEFA (MCE)*	*Range:*	*0 – 1048575*	*Reset value:*	*0x00000*
MTC2		*0xEFE (MCE)*				
MCT3		*0xF02 (MCE)*				

On the MCE, the timer count for each of the three MCE pin timers is accessed as a single 20-bit register as shown above. On the 8051, the 20-bit timer count for each of the three MCE pin timers is divided into three 8-bit registers (high, middle and low) as shown below.

MCE PIN TIMER COUNT HIGH (MTCH1, MTCH2, MTCH3)

MTCH1	*Address:*	*0xF1F6 (8051)*	*Range:*	*See description*	*Reset value:*	*0x00*
MTCH2		*0xF1FE (8051)*				
MTCH3		*0xF208 (8051)*				

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MTCH.7	MTCH.6	MTCH5	MTCH4	MTCH.3	MTCH2	MTCH1	MTCH.0
-	**-**	**-**	**-**	**CNT19**	**CNT18**	**CNT17**	**CNT16**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MCE PIN TIMER COUNT MIDDLE (MTCM1, MTCM2, MTCM3)

MTCM1	*Address:*	*0xF1F5 (8051)*	*Range:*	*See description*	*Reset value:*	*0x00*
MTCM2		*0xF1FD (8051)*				
MTCM3		*0xF207 (8051)*				

MTCM.7	MTCM.6	MTCM.5	MTCM.4	MTCM.3	MTCM.2	MTCM.1	MTCM.0
CNT15	**CNT14**	**CNT13**	**CNT12**	**CNT11**	**CNT10**	**CNT9**	**CNT8**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

MCE PIN TIMER COUNT LOW (MTCL1, MTCL2, MTCL3)

MTCL1	*Address:*	*0xF1F4 (8051)*	*Range:*	*See description*	*Reset value:*	*0x00*
MTCL2		*0xF1FC (8051)*				
MTCL3		*0xF206 (8051)*				

MTCL.7	MTCL.6	MTCL.5	MTCL.4	MTCL.3	MTCL.2	MTCL.1	MTCL.0
CNT7	**CNT6**	**CNT5**	**CNT4**	**CNT3**	**CNT2**	**CNT1**	**CNT0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The MCE pin timer count (CNT0 – CNT19) is a 20-bit value that determines the point within the global PWM cycle at which the pin timer expires and the level of the output pin is changed. The valid range of values depends on the MCE pin timer period configuration. If the period register is set to a value *p*, then the MCE pin timer count value can range from 0x00000 to 0xpFFFF. For example, if period is 2 the count can range from 0x00000 to 2FFFF. For the IRMCx171, period is always 0 and count can range from 0x00000 to 0x0FFFF. For timer expiration at a particular point within the global PWM cycle, set the count value to the corresponding percentage of the maximum value.

Example for the IRMCx171: Period is 0 and the maximum count value is 0x0FFFF. Set the count to 0x04000 for timer expiration after the first quarter of the PWM cycle.

Example for the IRMCF188 with PFC PWM frequency twice the motor PWM frequency: Period is 1 and the maximum count value is 0x1FFFF. Set the count to 0x10000 for timer expiration at the start of the second PFC PWM period (the center of the global PWM period).

MCE PIN TIMER MODE (MTMD1, MTMD2, MTMD3)

MTMD1	*Address:*	*0xF1F8 (8051)*	*Range:*	*Bit field definitions*	*Reset value:*	*0x00*
		0xEFC (MCE)				
MTMD2		*0xF200 (8051)*				
		0xF00 (MCE)				
MTMD3		*0xF208 (8051)*				
		0xF04 (MCE)				

MTMD.7	MTMD.6	MTMD.5	MTMD.4	MTMD.3	MTMD.2	MTMD.1	MTMD.0
-	**-**	**-**	**-**	**-**	**MD1**	**MD0**	**LEVEL**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The bit definitions for the three MTMD registers are as follows:

MTMD.7	-	*Not implemented. Returns zero when read.*
MTMD.6	-	*Not implemented. Returns zero when read.*
MTMD.5	-	*Not implemented. Returns zero when read.*
MTMD.4	-	*Not implemented. Returns zero when read.*
MTMD.3	-	*Not implemented. Returns zero when read.*
MTMD.2	**MD1**	MCE Timer mode select bit MD1.
MTMD.1	**MD0**	MCE Timer mode select bit MD0.
MTMD.0	**LEVEL**	Starting level for output pin (0 = low; 1 = high).

For all three timers, the mode select bits MD1 and MD0 select the timer mode (described at the beginning of this section) as follows:

MD1	MD0	Function
0	0	Select asynchronous mode.
0	1	Select single shot mode.
1	0	Select single pulse mode.
1	1	Select repetitive mode.

In asynchronous mode, the specified level of the output pin is set when the mode register is written. For all other modes, the timer is armed when the mode register is written. Therefore, the MCE pin timer enable, count and period registers should be written first, followed by a write to the mode register.

2.7 UART

There are five registers used to operate the UART. These are:

- Transmit data buffer (UTXD)
- Receive data buffer (URXD)
- Baud rate code (UBD)
- Control (UCTRL)
- Status (USTAT)

The features of the UART can be summarized as follows:

- Programmable to 8-bit or 9-bit parity (odd/even) or 9-bit data operation.
- Programmable 8-bit baud rate. SYSCLK divided by the baudrate code (UBD) gives the sampling clock frequency, which is 16 times the UART bit clock.
- The 16 times clock samples three times in the middle of each bit. If the data is different, a noisy indication is set. This noisy bit is sticky across the whole character.
- The receive and transmit data buffers each hold two characters.
- An overrun error is detected if new data cannot be stored in the receive buffer because the buffer is full.
- A framing error is detected if no stop bit present.
- In discard mode bytes with errors are simply discarded so the CPU doesn't have to process them.
- An interrupt is issued when the receive buffer contains data or when the transmit buffer has available space.
- When the hunt mode bit is set, the UART generates an interrupt only when the ninth bit is equal to the UCTRL.2 bit. This feature may be used to hunt for the start of a message.
- In case the UART receives a constant input of "0", it will generate an interrupt at a byte interval. (The number of bits per byte depends on the parity setting.) This interval includes a start bit, data (8/9) bits and a stop bit. Then, because the stop bit is '0', a framing error will occur.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

UART0 CONTROL (UCTRL)

Address: 0xF116 (8051) 0xE8B (MCE) Range: Bit field definitions Reset value: 0x00

UCTL.7	UCTL.6	UCTL.5	UCTL.4	UCTL.3	UCTL.2	UCTL.1	UCTL.0
-	**UXINTE**	**UHUNT**	**UEN**	**UDISC**	**UDATA**	**UM1**	**UM0**
	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

UCTL.7	**UTXDATA**	In UART Mode 3 (UM1=1, UM0=1), this bit is the data for bit 9.
UCTL.6	**UXINTE**	UART transmit interrupt enable
UCTL.5	**UHUNT**	When this bit is set, the UART hunts for a byte with bit 9 set as specified by UCTL.2 (UDATA) and only then generates a receive interrupt. All bytes before that are ignored. The "hunt" feature is only available in UART Mode 3 (UM1=1, UM0=1).
UCTL.4	**UEN**	Enables the activity of the UART
UCTL.3	**UDISC**	Setting the "discard" bit causes the UART to discard data received with errors such as framing error, parity error or noise error.
UCTL.2	**URXDATA**	This is the 9th value received
UCTL.1	**UM1**	UART Mode 1
UCTL.0	**UM0**	UART Mode 0

The mode control bits operate as follows:

U0M1	U0M0	Operating Mode	Parity
0	0	Mode 0: 8-bit UART	No Parity
0	1	Mode 1: 9-bit UART	Odd Parity
1	0	Mode 2: 9-bit UART	Even Parity
1	1	Mode 3: 9-bit UART	Data Parity

The UART receive interrupt is asserted whenever a character has been received and is ready to be read out of the UART Receive Buffer (uart_rxdata). The interrupt is no longer asserted after the 8051 reads the character out of the UART Receive Buffer.

The UART transmit interrupt is asserted whenever there is room in the UART Transmit Buffer (uart_txdata) for another character to be transmitted. The 8051 software must disable this interrupt by writing "0" to the U0XINTE bit in the control register (uart_control.6) when there are no characters to be sent.

UART0 STATUS (USTAT)

Address: 0xF21E (8051) 0xF0F (MCE) Range: Bit field definitions Reset value: 0x00

USTAT.7	USTAT.6	USTAT.5	USTAT.4	USTAT.3	USTAT.2	USTAT.1	USTAT.0
-	**UBIT9**	**UOE**	**UPE**	**UFE**	**UNE**	**UTXEM**	**URXV**
	R	R	R	R	R	R	R

The bit definitions for this read-only register are as follows:

USTAT.7	**-**	*Reserved*
USTAT.6	**UBIT9**	Ninth bit of received data
USTAT.5	**UOE**	Current reception buffer was over-run
USTAT.4	**UPE**	Current reception buffer had parity error

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

USTAT.3	**UFE**	Current reception buffer had framing error
USTAT.2	**UNE**	Current reception buffer had noise
USTAT.1	**UTXEM**	Transmission buffer is empty
USTAT.0	**URXV**	Reception buffer has data

UART0 BAUD RATE (UBD)

Address:	*0xF118 (8051, low)* *0xF119 (8051, high)* *0xE8C (MCE)*	*Range:*	*Unsigned, 0 - 65535*	*Reset value:*	*0x0000*

The value in the 16-bit baud rate register divides the system clock to create the sampling clock. The sampling clock is 16 times faster than the bit rate. The formula for calculating the baud rate code is:

*code = (SYSCLK / (baud rate * 16)) - 1*

UART0 TRANSMIT DATA BUFFER (UTXD)

Address:	*0xF114 (8051)* *0xE8A (MCE)*	*Range:*	*Unsigned, 0 - 255*	*Reset value:*	*0x00*

To transmit an 8-bit character, write it to the UTXD register. A character should be written only when the UTXEM (USTAT.1) is "1" (transmit buffer is empty).

UART0 RECEIVE DATA BUFFER (URXD)

Address:	*0xF21C (8051)* *0xF0E (MCE)*	*Range:*	*Unsigned, 0 - 255*	*Reset value:*	*0x00*

When the **URXV** (USTAT.0) bit is "1" (receive buffer has data), read the URXD register to read an 8-bit character from the UART receive buffer. Prior to reading the received data, the error status bits in the USTAT register should be examined.

2.8 D/A PWM

The IRMCx100 series has three digital-to-analog PWM output ports. All three are based on 8-bit data with a resolution of 256 counts. Each one has an associated 8-bit data register and an internal 8-bit counter. The value written into the 8-bit data register sets the modulation index of the PWM output. On reset, the data registers are set to zero, which effectively disables the PWM. Writing any value other than zero enables the PWM and produces the desired waveform.

By setting the DAPRE.0 bit (PRESC) in the D/A PWM output prescaler register to "1", the counter for the D/A PWM outputs increments at the rate of SYSCLK/16. When the DAPRE.0 bit is set to "0", the counter increments at the system clock rate. The period of the PWM is fixed to 256 cycles. The data register determines the duty cycle.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

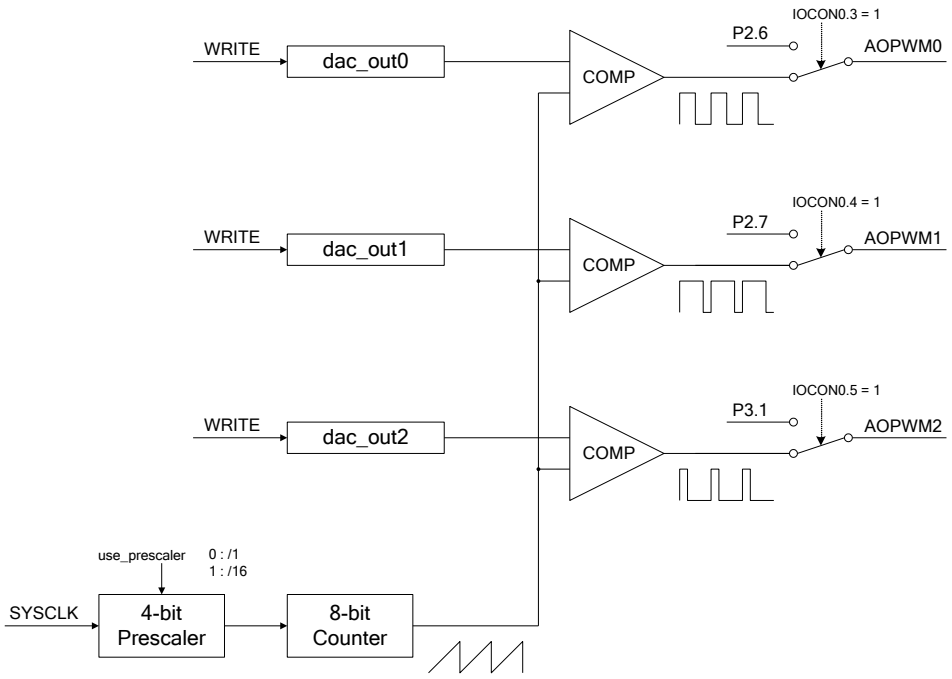


Figure 12. D/A PWM Output

D/A PWM OUTPUT PRESCALER (DAPRE)

Address: 0xF1CC (8051) 0xEE6 (MCE) Range: Bit field definitions Reset value: 0x00

DAPRE.7	DAPRE.6	DAPRE.5	DAPRE.4	DAPRE.3	DAPRE.2	DAPRE.1	DAPRE.0
-	-	-	-	-	-	-	**PRESC**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The bit definitions for the DAPRE register are as follows:

DAPRE.7	-	*Not implemented. Returns zero when read.*
DAPRE.6	-	*Not implemented. Returns zero when read.*
DAPRE.5	-	*Not implemented. Returns zero when read.*
DAPRE.4	-	*Not implemented. Returns zero when read.*
DAPRE.3	-	*Not implemented. Returns zero when read.*
DAPRE.2	-	*Not implemented. Returns zero when read.*
DAPRE.1	-	*Not implemented. Returns zero when read.*
DAPRE.0	**PRESC**	0 = increment counter at system clock rate; 1 = increment at SYSCLK / 16

D/A PWM OUTPUT DATA (DAD0, DAD1, DAD2)

DAD0 Address: 0xF1CE (8051) 0xEE7 (MCE) Range: Unsigned, 0 – 255 Reset value: 0x00
DAD1 0xF1D0 (8051) 0xEE8 (MCE)
DAD2 0xF1D2 (8051) 0xEE9 (MCE)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

2.9 I²C / SPI Serial Interface

The I²C/SPI interface is used for communication with simple external devices such as EEPROM. The registers used for the I²C interface are summarized in Table 10. The registers used for the SPI interface are summarized in Table 11.

Note that the user must choose between the two interfaces. The registers and I/O pins for the I²C and SPI interfaces are shared and only one type of device can be supported for a given application.

Register	8051 Address	MCE Address	Description	Reset Value
ISTXD	0xF120	0xE90	I²C write data	0x00
ISRXD	0xF224	0xF12	I²C read data	0x00
ISCMD	0xF11E	0xE8F	I²C command	0x00
ISSTAT	0xF226	0xF13	I²C status	0x00
ISACKIN	0xF122	0xE91	I²C ACK data	0x00
ISBD		0xE8E	I²C baud rate control (16 bits)	0x0000
ISBDL	0xF11C		I²C baud rate control (low)	0x00
ISBDH	0xF11D		I²C baud rate control (high)	0x00
ISBD	0xF11C	0xE8E	I²C baud rate control	0x00
ISFILT	0xF11A	0xE8D	I²C noise filter	0x00

Table 10. Registers for I²C

Register	8051 Address	MCE Address	Description	Reset Value
ISTXD	0xF120	0xE90	SPI write data	0x00
ISRXD	0xF224	0xF12	SPI read data	0x00
ISCMD	0xF11E	0xE8F	SPI command	0x00
ISSTAT	0xF226	0xF13	SPI status	0x00
ISBD		0xE8E	SPI baud rate control (16 bits)	0x0000
ISBDL	0xF11C		SPI baud rate control (low)	0x00
ISBDH	0xF11D		SPI baud rate control (high)	0x00

Table 11. Registers for SPI

I²C / SPI WRITE DATA REGISTER (ISTXD)

Address: 0xF120 (8051) 0xE90 (MCE) Range: Unsigned, 0 - 255 Reset value: 0x00

The write data register is read/write. This register is used for both I²C and SPI to transmit a byte of data to the target device. The data transmitted may be an address or other parameter associated with a command or data to be transferred to the target device.

I²C / SPI READ DATA REGISTER (ISRXD)

Address: 0xF224 (8051) 0xF12 (MCE) Range: Unsigned, 0 - 255 Reset value: 0x00

The read data register is read only. This register is used for both I²C and SPI to read a byte of data from the target device.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

I²C / SPI COMMAND REGISTER (ISCMD)

Address: 0xF11E (8051) 0xE8F (MCE) Range: Bit field definitions Reset value: 0x00

ISCMD.7	ISCMD.6	ISCMD.5	ISCMD.4	ISCMD.3	ISCMD.2	ISCMD.1	ISCMD.0
-	-	-	**SPISEL**	**SPIEN**	**CMD2**	**CMD1**	**CMD0**
			R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

ISCMD.7	-	*Reserved.*
ISCMD.6	-	*Reserved.*
ISCMD.5	-	*Reserved.*
ISCMD.4	**SPICS1**	SPI CS1 enable. 0: CS1 is not used, 1: CS1 is used. If an SPI device is configured at CS1, this bit should always be set to 1, even when issuing commands for the device at CS0. Always set this bit to 0 for I²C.
ISCMD.3	**SPIEN**	SPI interface enable. 0: enable I²C, 1: enable SPI.
ISCMD.2	**CMD2**	SPI / I²C command bit 2.
ISCMD.1	**CMD1**	SPI / I²C command bit 1.
ISCMD.0	**CMD0**	SPI / I²C command bit 0.

SPI / I²C commands are initiated by writing to the command register. The command type is encoded in bits CMD2 – CMD0. Command codes for the I²C interface are defined as follows:

CMD2	**CMD1**	**CMD0**	**I²C Command**	**Description**
0	0	0	Start	Send a start condition (precedes all commands)
0	0	1	Stop	Send a stop condition (terminates a command sequence)
0	1	0	Release	Send a release condition (terminates a command when a subsequent command is to follow immediately)
0	1	1	*Reserved*	
1	0	0	Control	Write a byte and report arbitration status (ABORT bit in i2c_spi_status register)
1	0	1	Write	Write a byte with no arbitration
1	1	0	Read	Read a byte
1	1	1	*Reserved*	

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The command codes for the SPI interface are:

CMD2	CMD1	CMD0	SPI Command	Description
0	0	0	Byte write CS0, CS0 low on completion	Write a byte to device 0, more bytes to follow
0	0	1	Byte read CS0, CS0 low on completion	Read a byte from device 0, more bytes to follow
0	1	0	Byte write CS1, CS1 low on completion	Write a byte to device 1, more bytes to follow
0	1	1	Byte read CS1, CS1 low on completion	Read a byte from device 1, more bytes to follow
1	0	0	Byte write CS0, CS0 high on completion	Write a byte to device 0, last byte
1	0	1	Byte read CS0, CS0 high on completion	Read a byte from device 0, last byte
1	1	0	Byte write CS1, CS1 high on completion	Write a byte to device 1, last byte
1	1	1	Byte read CS1, CS1 high on completion	Read a byte from device 1, last byte

I^2C / SPI STATUS REGISTER (ISSTAT)

Address: 0xF226 (8051) 0xF13 (MCE) Range: Bit field definitions Reset value: 0x00

ISSTAT.7	ISSTAT.6	ISSTAT.5	ISSTAT.4	ISSTAT.3	ISSTAT.2	ISSTAT.1	ISSTAT.0
-	**WRACC**	**BUSST2**	**BUSST1**	**BUSST0**	**ACKO**	**ABORT**	**BUSY**
			R/W	R/W	R/W	R/W	R/W

The bit definitions for this register are as follows:

ISSTAT.7	-	*Reserved.*
ISSTAT.6	**WRACC**	I^2C current or most recent access type. 0: read; 1: write.
ISSTAT.5	**BUSST2**	I^2C bus status bit 2.
ISSTAT.4	**BUSST1**	I^2C bus status bit 1.
ISSTAT.3	**BUSST0**	I^2C bus status bit 0.
ISSTAT.2	**ACKO**	I^2C command acknowledgement. Reports the value returned from the device.
ISSTAT.1	**ABORT**	I^2C command completion status. 0: normal; 1: command aborted.
ISSTAT.0	**BUSY**	SPI / I^2C operational status. 0: idle; 1: busy.

Only bit 0 (BUSY) is used for SPI.

The I^2C bus status is encoded in bits BUSST2 – BUSST0, as follows:

BUSST2	BUSST1	BUSST0	I^2C Bus Status Description
0	0	0	Idle
0	0	1	Writing slave address
0	1	0	Writing start address
0	1	1	Writing data
1	0	0	Reading data
1	0	1	ACK following read
1	1	0	*Reserved*
1	1	1	*Reserved*

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

I²C ACK DATA REGISTER (ISACKIN)

Address:	*0xF122 (8051)* *0xE91 (MCE)*	*Range:*	*Bit field definitions*	*Reset value:*	*0x00*

The ACK data register is read/write and only the low-order bit (bit 0) is used. This register is used for I²C only, and determines the ACK value transmitted at the end of a read operation. The ACK value should be set to "0" if additional data bytes are to be read (sequential read operation) or "1" if there are no additional bytes to read in the current operation (current address read, random read, or the last byte of a sequential read).

I²C / SPI BAUD RATE CONTROL (ISBD)

Address:	*0xF11C (8051, low)* *0xF11D (8051, high)* *0xE8E (MCE)*	*Range:*	*Unsigned, 0 - 4095*	*Reset value:*	*0x0000*

The baud rate control register configures the bit rate for I²C or SPI serial communication. The clock frequency of the SCL pin is determined by this register value. The value is a frequency divider and can range from 0 to 4095 (12 bits, unsigned). The SCL clock frequency is given by the equation:

SCL clock frequency (kHz) = SYSCLK /3. Value

where SYSCLK is the system clock rate.

I²C NOISE FILTER (ISFILT)

Address:	*0xF11A (8051)* *0xE8D (MCE)*	*Range:*	*Unsigned, 0 - 15*	*Reset value:*	*0x00*

The noise filter register configures the I²C noise filter time constant and is not used for the SPI interface. The valid range of values is: 0 – 15 (4 bits). The filter is a counter clocked by the system clock (SYSCLK).

For example, if ISFILT is set to 12, any signal level change on the SDA or SCL pin that persists for fewer than 12 clock cycles is ignored.

2.9.1 Command Descriptions for the I²C Interface

This section describes how to use the I²C interface to read and write typical I²C EEPROM devices.

2.9.1.1 Device Address

Each I²C device has an 8-bit address. Bits 3 – 6 must always be set to "1010" in order to address any device on the I²C bus. Bits 0 – 2 select an individual device. EEPROM devices typically have the upper five bits of address hard-wired to "10100" and have two configurable address pins A1 and A0.

2.9.1.2 Read and Write Commands

The commands defined for the ISCMD register provides the building blocks for the read and write operations defined by an I²C device.

Note: After every command written to the ISCMD register, software must wait for the BUSY bit in the ISSTAT register to be cleared ("0") before writing another command or reading the ISRXD register. The sequences described below assume that the required wait is performed at each step.

The typical sequence used from the 8051 to **write** to an EEPROM device is as follows:

1. Issue a Start command (ISCMD register).
2. Write the slave address to the ISTXD register and issue a Write command.
3. If the ACKO bit in the ISSTAT register is set ("1"), issue a Stop command and then repeat steps 1 and 2. (An ACK response of "1" after writing the device address indicates that the specified device is busy.)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4. Write the high byte of the desired EEPROM offset to the ISTXD register and issue a Write command. Repeat for the low byte of the offset. (For smaller EEPROM devices, the offset is one byte and only one Write operation is required.)
5. Write the data byte to the ISTXD register and issue a Write command.
6. Repeat step 5 to write additional data bytes.
7. Issue a Stop command.

The typical sequence used from the 8051 to **read** from an EEPROM device is as follows:

1. Issue a Start command (ISCMD register).
2. Write the slave address to the ISTXD register and issue a Write command.
3. If the ACKO bit in the ISSTAT register is set ("1"), issue a Stop command and then repeat steps 1 and 2. (An ACK response of "1" after writing the device address indicates that the specified device is busy.)
4. Write the high byte of the desired EEPROM offset to the ISTXD register and issue a Write command. Repeat for the low byte of the offset. (For smaller EEPROM devices, the offset is one byte and only one Write operation is required.)
5. Issue a Release command.
6. Issue a Start command.
7. Write the slave address OR'd with 0x01 (to enable a read operation) to the ISTXD register and issue a Write command.
8. If this is the only (or last) data byte to be read, write a "1" to the ISACKIN register. Otherwise (additional data bytes to follow), write a "0".
9. Issue a Read command.
10. Read the data byte from the ISRXD register.
11. Repeat steps 8 – 10 to read additional data bytes.

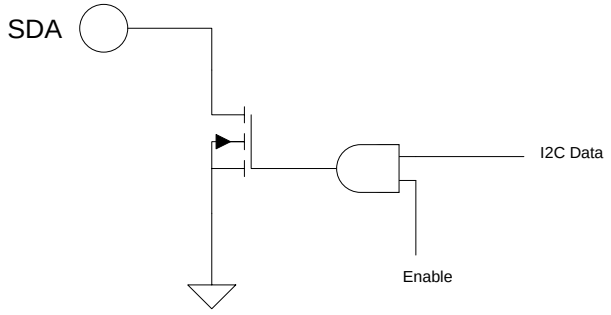


Figure 13. I^{2}C Pin Structure

2.9.2 Command Descriptions for the SPI Interface

This section describes how to use the SPI interface to read and write typical SPI EEPROM devices.

Each SPI command reads or writes a single byte of data. Typical SPI devices define instruction sequences composed of multiple bytes. A series of SPI commands is required to issue a single multi-byte instruction sequence to an SPI device. When issuing multi-byte instructions, use the set of command codes that specifies "CSn low on completion" for all but the last command of the sequence. For the last command, use a command code from the set that specifies "CSn high on completion". The SPI device recognizes the end of the instruction sequence when the CS signal returns to the high state. For single-byte instructions (such as write enable), always use a command code that sets CS high on completion.

2.9.2.1 Read Instructions

Reading a byte of data from an SPI EEPROM device typically requires a sequence of four commands to be issued to the SPI interface: three write commands are issued to send the read instruction and the 16-byte address, followed by a read command to read the data. Reading multiple bytes of data from sequential locations can usually be accomplished

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

by writing the read instruction and 16-byte starting offset, followed by a series of read commands, one for each data byte. Only the last read command sets the CS signal high on completion.

Note: After issuing any command, software must wait for the BUSY bit in the ISSTAT register to be cleared ("0") before writing another command or reading the ISRXD register. The sequence described below assumes that the required wait is performed at each step.

The typical sequence used from the 8051 to **read** from an SPI EEPROM is as follows:

1. Write a Read instruction code (specific to the SPI device) to the ISTXD register.
2. Write a byte write command code (CS low on completion) to the ISCMD register.
3. Write the high byte of the 16-bit EEPROM offset to the ISTXD register.
4. Write a byte write command code (CS low on completion) to the ISCMD register.
5. Write the low byte of the 16-bit EEPROM offset to the ISTXD register.
6. Write a byte write command code (CS low on completion) to the ISCMD register.
7. Write a byte read command code (CS high on completion) to the ISCMD register.
8. Read the data byte from the ISRXD register.

To read more than one byte, follow the steps above, but use a read command code with CS low on completion at step 7. Repeat steps 7 and 8 for each additional data byte to be read. For the last read only, use the read command code with CS high on completion.

2.9.2.2 Write Instructions

Writing to an SPI EEPROM device typically requires the device to be write enabled first as a separate instruction (setting the CS signal high on completion). The write instruction follows, which typically requires a sequence of four write command to send the write instruction, the 16-bit address and the data byte. Writing multiple bytes of data to sequential locations can usually be accomplished by sending the write instruction and 16-byte starting offset, followed by a series of write commands, one for each data byte. Only the last write command sets the CS signal high on completion.

Note: After issuing any command, software must wait for the BUSY bit in the ISSTAT register to be cleared ("0") before writing another command or reading the ISTXD register. The sequence described below assumes that the required wait is performed at each step.

The typical sequence used from the 8051 to **write** to an SPI EEPROM is as follows:

1. Write a Write Enable instruction code (specific to the SPI device) to the ISTXD register.
2. Write a byte write command code (CS high on completion) to the ISCMD register.
3. Write a Write instruction code (specific to the SPI device) to the ISTXD register.
4. Write a byte write command code (CS low on completion) to the ISCMD register.
5. Write the high byte of the 16-bit EEPROM address to the ISTXD register.
6. Write a byte write command code (CS low on completion) to the ISCMD register.
7. Write the low byte of the 16-bit EEPROM address to the ISTXD register.
8. Write a byte write command code (CS low on completion) to the ISCMD register.
9. Write the data byte to the ISTXD register.
10. Write a byte write command code (CS high on completion) to the ISCMD register.

To write more than one byte, follow the steps above, but use a write command code with CS low on completion at step 10. Repeat steps 9 and 10 for each additional data byte to be written. For the last write only, use the write command code with CS high on completion.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3 Motion Control Engine

The Motion Control Engine (MCE) is a collection of hardware and firmware modules—the building blocks necessary to implement high efficiency sinusoidal sensorless control for Permanent Magnet motors. The MCE Library provides a graphical representation of these modules.

The MCE library contains various control block modules specific to motor control applications as well as a number of general-purpose modules for miscellaneous operations and support functions. The user can select and connect the library blocks to form a control algorithm. Using the MCE library in the MATLAB/Simulink™ environment, the user can design custom control loops (closed loop sensorless current control, closed loop speed control, etc.) based on application requirements. A graphic compiler analyses the completed design and automatically translates it into a sequence of MCE-specific machine code for integration with the IRMCx100 series. Operating on the IRMCx100-series device, the MCE machine code essentially customizes the device for the user's specific application requirements.

The two basic types of firmware resources available on the IRMCx100 series are Motion Peripherals and Control blocks.

Motion peripherals process analog and digital signals and interface to "the outside world"—hardware external to the IRMCx100 series IC. Examples are the Low Loss Space Vector PWM module, A/D converter module, and single shunt current reconstruction module. These modules are colored yellow throughout this document and the design entry tool (Matlab/Simulink™) to distinguish them from other firmware elements. Each motion peripheral module is used only once in an application design since it corresponds to a single firmware module. The motion peripheral blocks are described in Section 3.3. Registers used to configure and monitor the operation of the motion peripheral blocks from an 8051 application or a host system are described in Section 3.4.

Control Blocks are the math, control, and logic elements implemented in hardware or firmware. These modules can be used in an application design as many times as needed. Control block signals can be connected to another Control Block or to a Motion Peripheral module. Control Blocks are colored green (for math) or blue (all others) throughout this document and the design entry tool (MATLAB/Simulink™) to distinguish them from the Motion Peripherals. The control blocks are described in Section 3.2. There are no pre-defined registers for control block configuration and monitoring as there are for the motion peripherals.

Additional blocks are provided for support functions such as data initialization and monitoring, signal delays and page-to-page connections. Some support functions are implemented using standard Simulink library components. The support blocks are described in Section 5.3.1.

Table 12 summarizes all the blocks in the MCE library. For each block, the table entry references the document section that describes the block in detail. For more information about using the MCE library in MATLAB/Simulink, refer to Section 5.

All blocks are based on 16-bit signed or unsigned integer input and output.

Module Category	Module Name	Description	Document Section
Control Blocks – Frequency Domain	PI	Proportional plus integral	3.2.1.1
	LOWPASS_FILT	First order low pass filter	3.2.1.2
	HIGHPASS_FILT	First order high pass filter	3.2.1.3
Control Blocks – Coordinate Transformation	VECROT	Vector rotator	3.2.2.1
	CLARK	Inverse Clark transformation	3.2.2.2
Control Blocks – Utility	RAMP	Linear ramp function	3.2.3.2
	ATAN	Arc tangent lookup table	3.2.3.3
	LIMIT	Limit function	3.2.3.1
	FUNCTION_BLOCK	Five-input two-dimensional function	3.2.3.4
	COMPARATOR	Compare two inputs	3.2.3.5

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Module Category	Module Name	Description	Document Section
Control Blocks – Utility	SWITCH	Switch between two inputs based on a third	3.2.3.6
	BIT_LATCH	Bit latch	3.2.3.7
	PEAK_DETECT	Peak detect	3.2.3.8
	TRANSITION	One shot pulse generator	3.2.3.9
	INTEGRAL2	Integral with limit	3.2.3.10
Control Blocks – Math	MUL_DIV	Multiply with extraction (signed/unsigned)	3.2.4.6
	DIVIDE	Divide (signed/unsigned)	3.2.4.7
	SUM	Adder	3.2.4.2
	ACCUMULATOR	Accumulator	3.2.4.3
	COUNTER	Counter with limit	3.2.4.4
	DIFF	Difference	3.2.4.1
	SHIFT	Multiply by a power of two	3.2.4.5
	AND	Logical AND (16 bit)	3.2.4.10
	OR	Logical OR (16 bit)	3.2.4.11
	XOR	Logical XOR (16 bit)	3.2.4.12
	NOT	Logical NOT (16 bit)	3.2.4.8
	NEGATE	Logical NEGATE (16 bit)	3.2.4.9
Motion Peripheral Blocks	SENSORLESS_FOC	Sensorless field orientation control	3.3.1
	LOWLOSS_SVPWM	Low loss space vector modulator	3.3.5
	CURRENT_MEAS	Motor current measurement	3.3.2
	DC_BUS_VOLTAGE	DC bus voltage monitor	3.3.3
	A_D	A/D converters	3.3.4
	FAULTS	Drive fault status	3.3.6
	MCE_FAULT	MCE fault generator	3.3.7
	PFC_PWM	PFC PWM pulse generator	1.1.1
	PFC_SENSE	PFC feedback signals	3.3.9
	PFC_AC_STATUS	AC Input voltage status	3.3.10
	PFC_MOTOR_MISC	Miscelaneous system data	3.3.11
Custom Support Blocks	Configure PWM	Hierarchical connection to PWM	5.3.1.1
	Configure Control Loop	Hierarchical connection to control loops	5.3.1.1
	Read Register	Output to host/8051 interface	5.3.1.2
	Write Register	Inputs from host/8051 interface	5.3.1.2
	MCE Compiler	Shortcut to the MCE Compiler	5.3.1.5
	Host Register Summary	Read and Write register summary display	5.3.1.5
Standard Simulink Support Blocks	Enabled Subsystem	Hierarchical organization block	5.3.2.1
	Constant	Defines a constant input value	5.3.2.2
	Scope	Defines an output for MCEDesigner trace function	5.3.2.3
	Goto	Off-page connector (output)	5.3.2.4
	From	Off-page connector (input)	5.3.2.4
	Unit Delay	PWM cycle signal delay	5.3.2.5

Table 12. MCE Library Elements

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.1 Rotating Frame Notation and Conventions

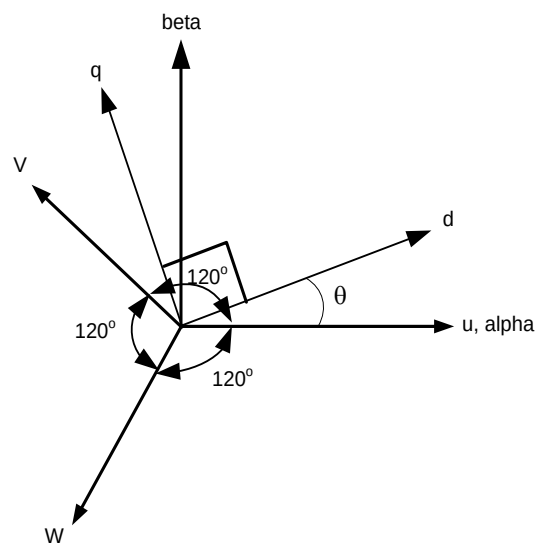
Throughout this document, the following convention is used to describe input and output notation with regard to the rotating frame for field orientation control.

The stationary frame variables "alpha" and "beta" are orthogonal. The three-phase stationary frame variables "u", "v", and "w" are 120 degree apart. "alpha" is aligned to "u". The synchronously rotating frame variables "d" and "q" are relative to the moving angle of θ.

Forward Vector Rotation:

$$\begin{bmatrix} alpha \\ beta \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} d \\ q \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \cdot \begin{bmatrix} alpha \\ beta \end{bmatrix}$$



Inverse Vector Rotation:

$$\begin{bmatrix} alpha \\ beta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} alpha \\ beta \end{bmatrix}$$

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2 Control Blocks

This section describes the MCE control blocks, which include frequency domain and coordinate transformation modules, general utility functions and math operations. In the Simulink MCE libraries, the math blocks are found in the Math library and the remaining control blocks are located in the Control library.

The connections between the various control blocks in the MCE design determine their order of execution, and they execute sequentially (not in parallel). A worst case timing estimate is shown for each control block. The designer should take care to ensure that the total execution time for all control blocks in the design does not exceed the configured PWM preiod. The MCE Compiler provides a worst case execution time estimate for an MCE design, as described in Section 5.4

3.2.1 Frequency Domain Blocks

3.2.1.1 PI – Proportional Plus Integral

The PI block performs the following function in the s-domain:

$$\frac{Output}{Input} = Kp + \frac{Ki}{s}$$

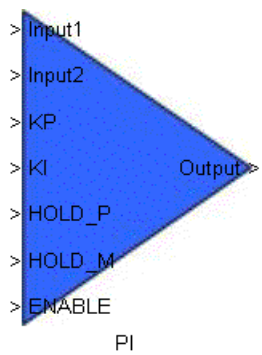
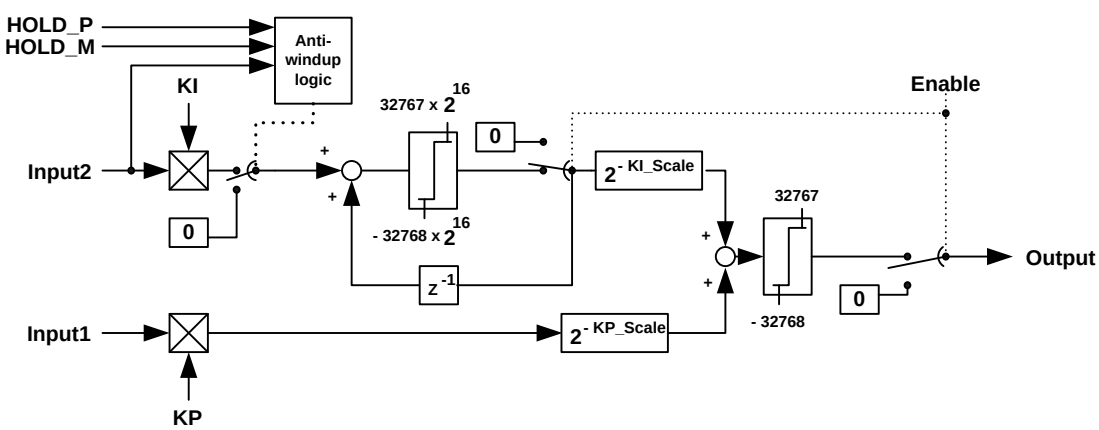


Figure 14. PI Block

And in the discrete domain:



In terms of pseudo code, the PI regulator can be represented by:

```
KP_Prod(n) = INPUT1(n)   x   KP
[s31:0]        [s15:0]          [s15:0]
KP_Prod(n) =  KP_Prod(n) right shift  KP_SCALE
[s31:0]         [s31:0]

If((INPUT2(n) > 0 and HOLD_P) or (INPUT2(n) < 0 and HOLD_M))  // antiwindup
     [s15:0]                [1]                [s15:0]                    [1]
        KI_Prod(n) = 0
        [s31:0]
Else
        KI_Prod(n)   = INPUT2(n)   x   KI
        [s31:0]        [s15:0]            [s15:0]

If(ENABLE)
          Int(n)    =     KI_Prod(n)  +    Int(n-1)
          [s31:0]          [s31:0]           [s31:0]
          Protect sign 32-bit overflow on Int(n)
Else
          Int(n) = 0
          [s31:0]

KI_Int(n)      =  Int(n) right shift KI_SCALE
[s31:0]            [s31:0]

Temp1(n)  =    KP_Prod(n)  +    KI_Int(n)
[s31:0]          [s31:0]             [s31:0]
 Protect 32 bit overflow on Temp1(n)
Temp2(n) = Limit Temp1(n) to [s15:0]
 [s15:0]

If(ENABLE)
      OUTPUT  =  Temp2(n)
       [s15:0]        [s15:0]
Else
       OUTPUT = 0
```

The scalers (KP_Scaler, KI_Scaler) are accessible by double clicking the PI block in the MODEL file. The scalers must be from 0 – 31. Note: The scalers can only be updated during compile time (MCE compiler).

Also note that when the output limits at the maximum or minimum 16-bit number, the internal 32-bit recursive register (I_output) continues to accumulate. To prevent this "wind-up" action, the PI block should be used in conjunction with the LIMIT block. The integration can be halted by feeding the SATP or SATM outputs of LIMIT back to HOLD_P and HOLD_M inputs of PI, respectively.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Type
Input1	Input for proportional path	Input	16 bit, signed integer
Input2	Input for integral path	Input	16 bit, signed integer
KP	Proportional gain	Input	16 bit, signed integer [1]
KI	Integral gain	Input	16 bit, signed integer [1]
HOLD_P	Hold Integrator Positive going	Input	Boolean, 0 = no hold 1 = hold integrator positive going
HOLD_M	Hold Integrator Negative going	Input	Boolean, 0 = no hold 1 = hold integrator negative going
ENABLE	Enable/Reset	Input	Boolean, 0 = reset all recursive data and output 1 = normal operation
Output	Output	Output	16 bit, signed integer

Table 13. PI User Inputs and Outputs

Signal name	Description	I/O	Type
KP_Scale	Scale factor for Kp gain	Input	8 bit scaler [1]
KI_Scale	Scale factor for Ki gain	Input	8 bit scaler [1]

Table 14. PI System Inputs and Outputs

Note 1. The range of KP and KI is from 0 to 32767 and can be interpreted as simple floating format. For KP and KI, the system uses a simple floating point format as shown below. The mantissa is a 16-bit unsigned integer and scaler is an 8-bit value, however only the **5** LSB are used (0 – 31 range). The scaler represents a negative power of two applied to the mantissa.

Mantissa:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Scaler positions decimal point

Scaler:

7	6	5	4	3	2	1	0
0	0	0					

Data range:

$1 * 2^{-31} – 65,535$

Status	Clock cycles
Min. case	34
Max. case	74

Table 15. PI Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.1.2 LOWPASS_FILT – First Order Low Pass Filter

The low pass filter block performs the following function:

$$\frac{Output}{Input} = \frac{1}{1 + S \bullet Tau}$$

where Tau is the filter time constant (1/Wc).

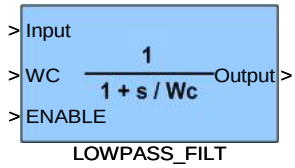
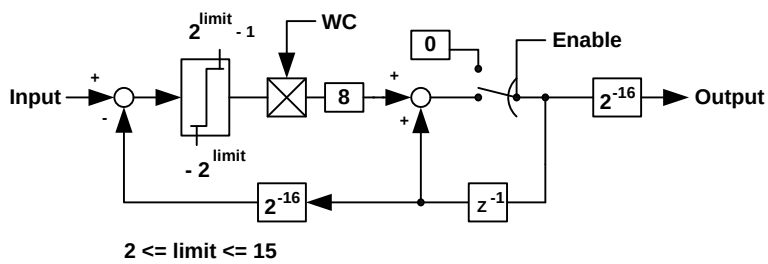


Figure 15. LOWPASS_FILT Block

Realization in the discrete domain:



In terms of Psuedo code, the lowpass filter can be realized as:

```
Temp1(n)   =   Input(n)    -   Output(n-1)
 [s15:0]        [s15:0]          [s15:0]
 Overflow protect Temp1(n) to 16-bit signed
 Temp1(n)  =  Temp1(n)  x   WC
  [s31:0]         [s15:0]          [s15:0]
 Temp1(n) = Shift Temp1(n) left 3 bit
  [s31:0]              [s31:0]

  If (ENABLE)
          Int(n)   =  Int(n)  +  Temp1(n)
          [s31:0]     [s31:0]     [s31:0]
           Overflow protected to 32 bit sign
  Else
          Int(n) = 0

  Output(n) =  Int(n) [s31:16]
  [s15:0]
```

Double clicking on the block in the Simulink model file will allow the designer to access a limit value parameter. The parameter restricts the difference between the filtered output and the new input to the range [-2^{LIMIT}, $2^{LIMIT} - 1$]. The parameter can be any value between 2 and 15, with default value of 15. Note: The LIMIT value parameter can only be updated during compile time (MCE compiler).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Relationship between the actual filter time constant (sec.) and the configurable parameter WC (input) is given by:

$$Tau = \frac{DeltaT}{WC} \times 2^{13} \quad \text{[sec]}$$

where: Tau is defined as the filter time constant in seconds, which corresponds to 1/Wc as shown in Figure 15.
WC (in digital counts) is the configurable input of the filter block (Figure 15).
DeltaT is the sampling time of the filter in seconds. This is generally 1/PWM Frequency.

Note: Theoretically, the minimum filter time constant should be larger than 2 times the filter sampling time to prevent digital filter instability. The filter gain and phase characteristics is shown in Figure 16 where the normalized frequency 1 corresponds to 1/Tau (rad./sec).

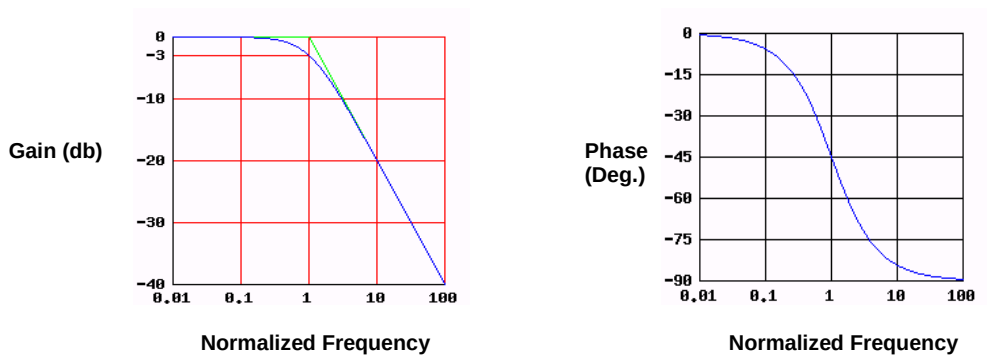


Figure 16. LOWPASS_FILT Frequency Response

Signal name	Description	I/O	Type
ENABLE	1 – enable filter 0 – disable filter, output = 0	Input	Boolean,
WC	Filter bandwidth in digital counts.	Input	16 bit, signed integer [1]
Input	Filter input	Input	16 bit, signed integer
Output	Filter Output (rounding)	Output	16 bit, signed integer

Table 16. LOWPASS_FILT User Inputs and Outputs

Note 1: The allowable data range of WC is 0 to 2^{13}. Numerical overflow will occur if WC is outside the specified range. To pass Input unchanged to Output (turn off filter action), set WC to 2^{13}.

Status	Clock cycles
Max. case	37

Table 17. LOWPASS_FILT Execution Time

3.2.1.3 HIGHPASS_FILT – First Order High Pass Filter

The HIGHPASS_FILT block performs the following S-domian function:

$$\frac{Output}{Input} = \frac{s \bullet Tau}{1 + s \bullet Tau} \qquad \text{where Tau = 1/Wc}$$

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

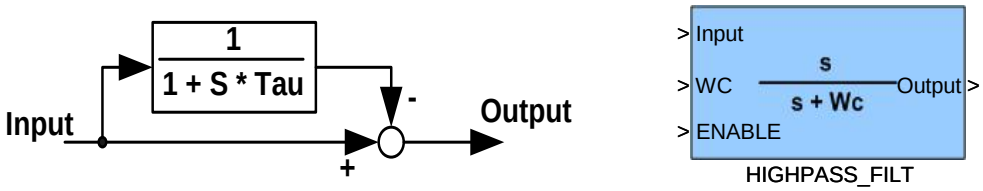


Figure 17. HIGHPASS_FILT Block

The highpass filter can be realized by the following psuedo code:

```
Temp1(n)   =   Input(n)    -   Output(n-1)
 [s15:0]         [s15:0]          [s15:0]
 Overflow protect Temp1(n) to 16-bit signed
 Temp1(n)  =  Temp1(n)  x   WC
  [s31:0]          [s15:0]          [s15:0]
 Temp1(n) = Shift Temp1(n) left 3 bit
  [s31:0]                [s31:0]

  If (ENABLE)
          Int(n)   =  Int(n)  +  Temp1(n)
          [s31:0]    [s31:0]    [s31:0]
           Overflow protected to 32 bit sign
  Else
          Int(n) = 0

   Temp2(n) = Input(n) – Int(n)[s31:16]
    [s15:0]       [s15:0]      [s15:0]
    Temp3(n) = Overflow protect Temp2 to 16-bit signed
    If(ENABLE)
          Output(n) = Temp3(n)
          [s15:0]         [s15:0]
    Else
           Output(n) = 0
```

When the ENABLE signal is low, the output of the block is zero. In order for the Output to pass the Input signal unchanged, set WC = 0 and pulse the ENABLE input to clear any recursive data. WC should be in the range [0, 2^{13}], otherwise Output could have over/under shoot spikes or even large oscillations.
The high pass filter is constructed by a low pass filter and a summing as shown in Figure 17. The low pass filter time contant (Tau) determines the high pass filter cutoff frequency (Wc).
Relationship between the actual filter time constant (sec.) and the configurable parameter WC (input) is given by:

$$Tau = \frac{DeltaT}{WC} \times 2^{13} \quad \text{[sec]}$$

where: Tau is defined as the low pass filter time constant in seconds.
WC (in digital counts) is the configurable input of the filter block (Figure 17).
DeltaT is the sampling time of the filter in seconds. This is generally 1/PWM Frequency.

Note: Theorectically, the minimum filter time constant should be larger than 2 times the filter sampling time to prevent digital filter instability.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

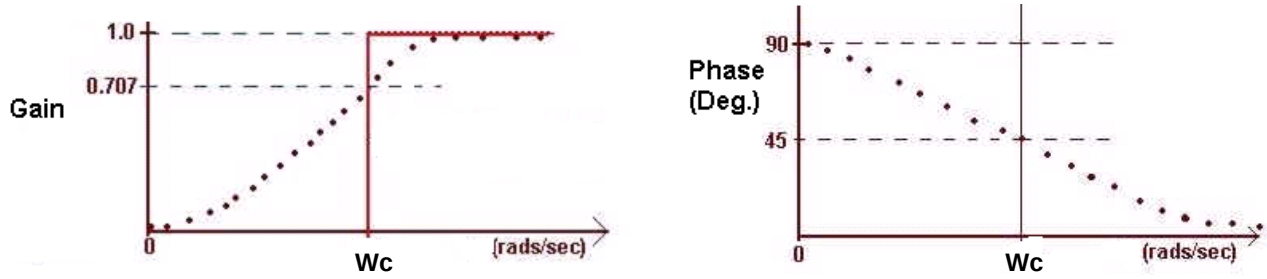


Figure 18. HIGHPASS_FILT frequency response

Figure 18 shows the frequency response of the high pass filter, the filter cutoff frequency (Wc in rad./sec) corresponds to 1/Tau.

Signal name	Description	I/O	Type
Input	Filter input	Input	16 bit, signed integer
WC	Filter bandwidth	Input	16 bit, unsigned integer [1]
ENABLE	1 – enable filter 0 – disable filter, output = 0	Input	Boolean,
Output	Filter output	Output	16 bit, signed integer

Table 18. HIGHPASS_FILT User Inputs and Outputs

Note 1: The allowable data range of WC is 0 to 2^{13}. Numerical overflow will occur if WC is outside the specified range. To pass Input unchanged to Output (turn off filter action), set WC to 0.

Status	Clock cycles
Max. case	87

Table 19. HIGHPASS_FILT Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.2 Coordinate Transformation Blocks

3.2.2.1 VECROT – Vector Rotation

The VECROT block performs the following function in the discrete domain:

$$\begin{bmatrix} Output1(n) \\ Output2(n) \end{bmatrix} = 1.64676 \cdot \begin{bmatrix} \cos\theta(n) & -\sin\theta(n) \\ \sin\theta(n) & \cos\theta(n) \end{bmatrix} \cdot \begin{bmatrix} Input1(n) \\ Input2(n) \end{bmatrix}$$

Where $0 \le \theta(n) \le 2\pi$, $\theta(n) = \text{THETA(n)} \times 2\pi/4096$.

The VECROT block follows the vector rotation convention described in section 3.1.

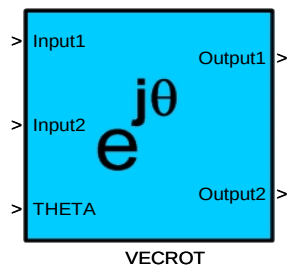


Figure 19. VECROT Block

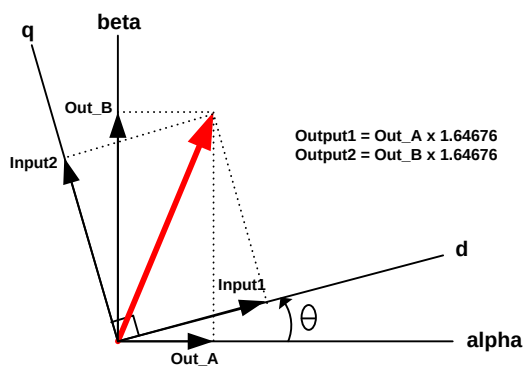


Figure 20. VECROT Vector Interpretation

Signal name	Description	I/O	Type
Input1	Input1 aligns with d axis of Figure 20	Input	16 bit, signed integer [1]
Input2	Input 2 aligns with q axis of Figure 20.	Input	16 bit, signed integer [1]
THETA	Rotator angle (4096 digital counts = 2π)	Input	16 bit, signed integer [2]
Output1	Output1 aligns with alpha axis of Figure 20.	Output	16 bit, signed integer
Output2	Output2 aligns with beta axis of Figure 20.	Output	16 bit, signed integer

Table 20. VECROT Inputs and Outputs

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Note 1: Valid data range of input 1 and 2 is $-2^{12} \le Input1 \le 2^{12}$, $-2^{12} \le Input2 \le 2^{12}$.
Note 2: THETA mapping: $0 \le THETA \le 4096$ digital counts maps to 0 – 2π radians.

Status	Clock cycles
Nominal	45

Table 21. VECROT Execution Time

3.2.2.2 CLARK – Inverse Clark Transformation

The inverse Clark block is a 3-phase to 2-phase transformation.

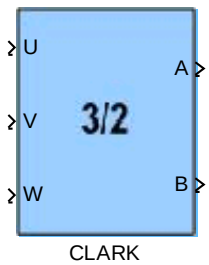


Figure 21. CLARK Block

The module implements the following equation in the discrete domain:

$$\begin{bmatrix} A(n) \\ B(n) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} \end{bmatrix} \cdot \begin{bmatrix} U(n) \\ V(n) \\ W(n) \end{bmatrix}$$

Pseudo code for Inverse Clark is realized by:

```
 A(n)      =      U(n)
[s11:0]          [s11:0]
Temp1(n) =  V(n)      –     W(n)
[s12:0]       [s11:0]        [s11:0]
Temp2(n) =  shift right 11 (Temp1(n)   *    2365)
 [s14:0]                          [s12:0]          [s12:0]
 B(n)  =  Temp2(n) [12:1]
 [s11:0]
```

Figure 22 shows the CLARK vector operation, the relationship (2/3 scaler) of B and B”is used to preserve power invariant transformation.

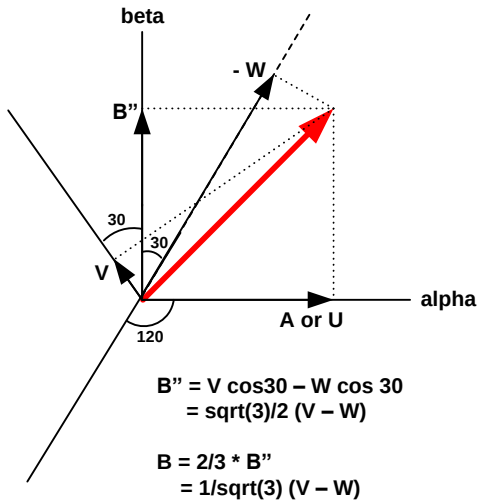


Figure 22. CLARK Vector Interpretation

Signal Name	Description	I/O	Type
U	Phase U (one of 3-phase)	Input	16 bit, signed integer [1]
V	Phase V (one of 3-phase)	Input	16 bit, signed integer [1]
W	Phase W (one of 3-phase)	Input	16 bit, signed integer [1]
A	A (orthogonal set with B)	Output	16 bit, signed integer
B	B (orthogonal set with A)	Output	16 bit, signed integer

Table 22. CLARK Inputs and Outputs

Note 1: range of U, V and W should be restricted between $\pm 2^{12}$.

Status	Clock cycles
Nominal	21

Table 23. CLARK Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.3 Utility Blocks

3.2.3.1 LIMIT

Figure 23 shows the inputs and outputs of the LIMIT block, which are also described in Table 24. Note that if LIMIT_P < LIMIT_M then OUT = LIMIT_M, SATP = 0 and SATM = 1 for all values of Input. Also, if LIMIT_P = LIMIT M, then OUT = LIMIT_M, SATP = 1 and SATM = 1 for all values of Input.

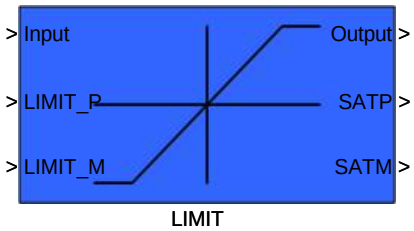


Figure 23. LIMIT Block

The LIMIT block performs the following function:

```
Output = Input
SATM = 0
SATP = 0
If (Input >= LIMIT_P) then
        Output = LIMIT_P
If (Input <= LIMIT_M ) then
        Output = LIMIT_M
If (Output == LIMIT_P) then
        SATP = 1
If (Output == LIMIT_M) then
        SATM = 1
```

Note: Input, Output, LIMIT_M and LIMIT_P are [s15:0].

Signal name	Description	I/O	Type
Input	Input	Input	16 bit, signed integer
LIMIT_P	Positive Limit Threshold	Input	16 bit, signed integer
LIMIT_M	Negative Limit Threshold	Input	16 bit, signed integer
SATP	Status flag for indicating upper limit (Limit_P) saturation	Output	Boolean, 0 = not saturated 1 = saturated
SATM	Status flag for indicating lower limit (Limit_M) saturation	Output	Boolean, 0 = not saturated 1 = saturated
Output	Output	Output	16bit, signed integer

Table 24. LIMIT Inputs and Outputs

Status	Clock cycles
Max. case	38

Table 25. LIMIT Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.3.2 RAMP – Linear Ramp

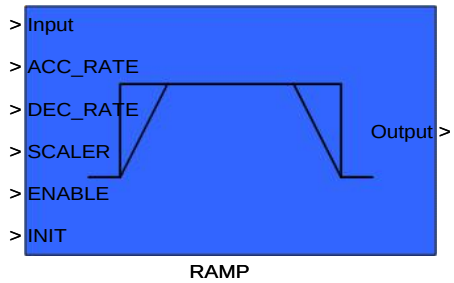
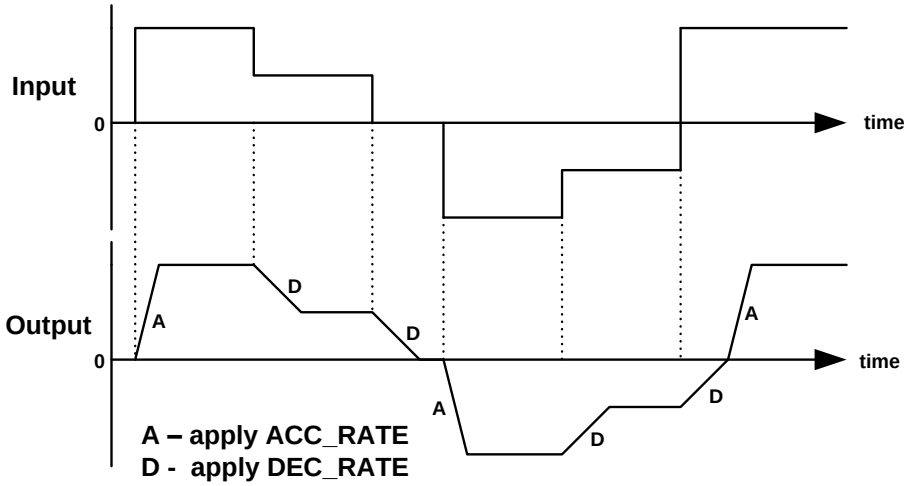


Figure 24. RAMP Block

The RAMP block provides rate limiting function. The rate of change of input is limited by configurable parameters ACC_RATE and DEC_RATE of the RAMP block inputs. Figure below illustrates (ACC_RATE = 4x DCC_RATE) the application of rate limiting under step changes of the input.



Example: ACC_RATE = 4 x DEC_RATE

The scaling of ACC_RATE and DEC_RATE is given by:

$$\text{Rate limit} = \frac{ACC_RATE}{2^{Scaler} \times \Delta T} \quad \text{[digital counts per sec.]}$$

$$\text{or} \quad = \frac{DEC_RATE}{2^{Scaler} \times \Delta T} \quad \text{[digital counts per sec.]}$$

Where ΔT = sampling time of the RAMP block

Note: It is not recommend to change SCALER when the RAMP block is enabled. Doing so could result in a discontinuous jump of the Output value.

The ramp function can be realized by the following Pseudo code:

```
if(ENABLE)
    Delta  = Input x 2^SCALER – Int32(n)
    [s31:0]  [s15:0]            [s31:0]
    If((Int32(n)>0 and Delta>0) or (Int32(n)<0 and Delta<0))
          RATE = ACC_RATE
    Elseif((Int32(n)<0 and Delta>0) or (Int32(n)>0 and Delta<0))
          RATE = DEC_RATE

    If(Delta > RATE) Delta = RATE
    Elseif (Delta < -RATE) Delta = -RATE

    Int32(n)= Int32(n-1) +  RATE
    [s31:0]    [s31:0]     [s31:0]
    Temp(n) =   Int32(n) x 2^(-SCALER)
    [s31:0]    [s31:0]
    Output  = temp[s15:0]
    [s15:0]
else
    Int32(n) = 2^SCALER x INIT
    [s31:0]           [s15:0]
    Output  = INIT
    [s15:0]   [s15:0]
endif
```

Signal name	Description	I/O	Type
Input	Input	Input	16-bit, signed integer
ACC_RATE	Acceleration rate limit	Input	16-bit, signed integer. (range: 0 – 32767)
DEC_RATE	Decceleration rate limit	Input	16-bit, signed integer. (range: 0 – 32767).
SCALER	scaler for ramp rate range accomodation	Input	8-bit unsigned integer [1]
ENABLE	Block enable control bit	Input	Boolean, 0 = ramp function disabled (output = INIT) 1 = ramp function enabled
INIT	Reset value of block output	Input	16-bit, signed integer If(ENABLE = 0) Output = INIT
Output	Output	Output	16-bit, signed integer

Table 26. RAMP User Inputs and Outputs

Note 1: Although SCALER is an 8-bit unsigned integer, in order to avoid integer arithmetic overflow, the maximum value of SCALER should be restricted to 16.

Status	Clock cycles
Min. case	65
Max. case	73

Table 27. RAMP Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.3.3 ATAN – Arc Tangent block

The ATAN block performs an arc tangent function. In order to optimize memory utilization, instead of using look-up table, the arc tangent function is implemented by Cordic algorithm inside the IRMCx100 series. The block symbol and its mathematical approximation are shown in Figure 25.

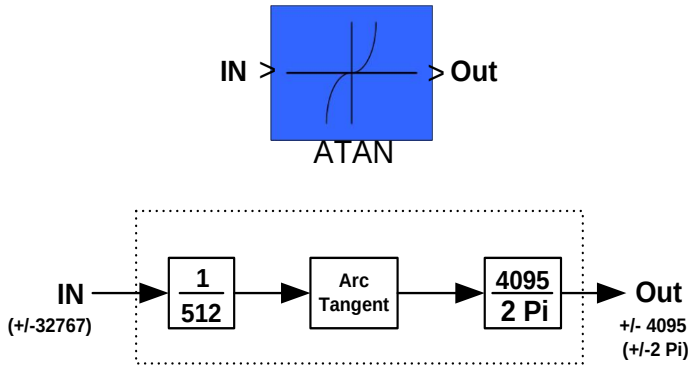


Figure 25. ATAN Block

Signal name	Description	I/O	Type
INPUT	Input	Input	16bit, signed integer
OUTPUT	Angle 4095 = 2π	Output	16bit, signed integer

Table 28. ATAN Inputs and Outputs

Status	Clock cycles
Nominal	24

Table 29. ATAN Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.3.4 FUNCTION_BLOCK

The FUNCTION_BLOCK block maps a 16-bit signed input into a 16-bit signed integer output. The shape of the function is defined by the user providing six points (p0 – p5 of Figure 27) in two dimensional space. The six points are literal (constant) values provided at compile time. Figure 26 shows the FUNCTION_BLOCK symbol and Figure 27 shows a sample function in which the values between points are linearly interpolated. Table 30 describes all the inputs and the output.

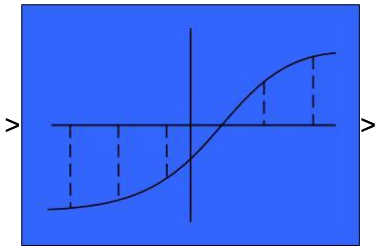


Figure 26. FUNCTION_BLOCK Block

The user provides six pairs of literal integer values, entered using a Simulink custom mask dialog, which is accessed by double-clicking the Function block: (X0,Y0) (X1,Y1) ….. (X5,Y5). It is the user's responsibility to ensure that no overflow can occur over the usable function range, or the results may be unpredictable. Outside the six points, the function is extrapolated. For input values less than point1, the slope p0-p1 is used. For inputs greater than point4, the slope p4-p5 is used.

Internally the function is represented by four coordinates (p1 – p4) and five slopes. (Points p0 and p5 are used only to determine the slope of the function outside the defined range.) These coordinates and slopes are computed by the MCE compiler and the thirteen values (four coordinate pairs and five slopes) are packed into ten input registers (each 16 bits wide). All values are signed.

The coordinates are represented internally by the high-order 12 bits of the input coordinate value appended with 4 binary zeroes, so the minimal X resolution is 16 while the minimal Y resolution is 16*slope. Slopes are represented by a 13-bit signed integer (except the p0 – p1 slope, which has only 12 bits) with a valid range of –4096 to 4095 (–2048 to 2047 for the p0 – p1slope). Slopes are normalized per 256 X distance, so the slope value is how much the Y changes when X increments by 256. In practice, this limits the slopes to –16 to 15.996 (–8 to 7.996 for the p0 – p1 slope). The accuracy of slopes is enough for practical cases.

In addition, the user provides a single (variable) input value, which the compiler passes directly to the MCE FUNCTION_BLOCK module. The module returns a signed 16-bit integer, which is the Y value corresponding to the input X value for the user defined function.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

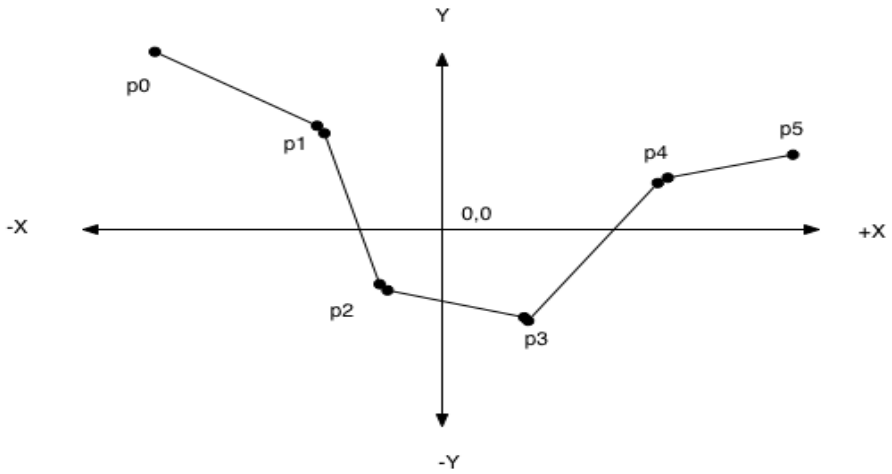


Figure 27. FUNCTION_BLOCK Example

Signal name	Description	I/O	Type
Input	x-coordinate input	Input	16 bit, signed integer
X0	1st point x-coordinate	Input	16 bit, signed integer
X1	2nd point x-coordinate	Input	16 bit, signed integer
X2	3rd point x-coordinate	Input	16 bit, signed integer
X3	4th point x-coordinate	Input	16 bit, signed integer
X4	5th point x-coordinate	Input	16 bit, signed integer
X5	6th point x-coordinate	Input	16 bit, signed integer
Y0	1st point y-coordinate	Input	16 bit, signed integer
Y1	2nd point y-coordinate	Input	16 bit, signed integer
Y2	3rd point y-coordinate	Input	16 bit, signed integer
Y3	4th point y-coordinate	Input	16 bit, signed integer
Y4	5th point y-coordinate	Input	16 bit, signed integer
Y5	6th point y-coordinate	Input	16 bit, signed integer
Output	y-coordinate interpolated output	Output	16 bit, signed integer

Table 30. FUNCTION_BLOCK Inputs and Outputs

Status	Clock cycles
Max. case	82

Table 31. FUNCTION_BLOCK Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.3.5 COMPARATOR

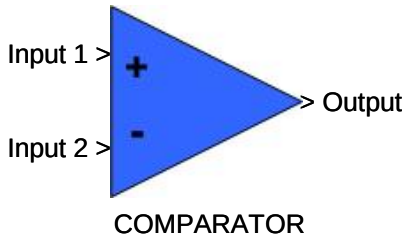


Figure 28. COMPARATOR Block

The following pseudo-code describes the function of the COMPARATOR block:

```
If (Input1 > = Input2) then
   [s15:0]       [s15:0]
          Output = 1
Else
          Output = 0
```

Signal name	Description	I/O	Type
Input1	Input 1	Input	16 bit, signed integer
Input2	Input 2	Input	16 bit, signed integer
Output	Output	Output	Boolean, 0 if Input1 < Input2 1 if Input1 >= Input2

Table 32. COMPARATOR Inputs and Outputs

Status	Clock cycles
Max. case	17

Table 33. COMPARATOR Execution Time

3.2.3.6 SWITCH

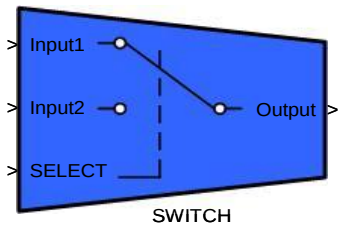


Figure 29. SWITCH Block

The operation of the SWITCH block can be described with the following pseudo-code:

If (SELECT = 0) then
Output = Input1
Else
Output = Input2

Note: if the SELECT input is non-Boolean, the LSB of the SELECT input will be used for logic decision.

Signal name	Description	I/O	Type
Input1	Input 1	Input	16 bit, signed integer
Input2	Input 2	Input	16 bit, signed integer
SELECT	Select between Input 1 or 2	Input	Boolean 0 = select input1 1 = select input2
Output	Output	Output	16 bit, signed integer

Table 34. SWITCH Inputs and Outputs

Status	Clock cycles
Max. case	7

Table 35. SWITCH Execution Time

3.2.3.7 BIT_LATCH

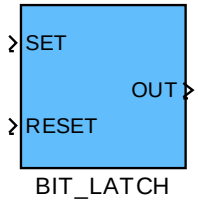
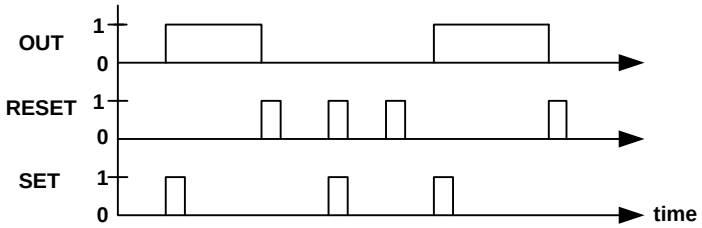


Figure 30. BIT_LATCH Block

The following pseudo-code describes the operation of the BIT_LATCH block:

If (positive edge (0 to 1) transition on RESET) then
OUT = 0
Elseif (positive edge (0 to 1) transition on SET) then
OUT = 1
Else no change on OUT

A Bit latch example is shown below to demonstrate the operation of the block:



The minimum duration of SET and RESET pulses should be larger than or equal to the execution rate of the BIT_LATCH block. In case inputs are connected by mistake to non-Boolean signals, the LSB of the non-Boolean signals will be used to determine logic operation.

Signal name	Description	I/O	Type
SET	Positive edge triggered input	Input	Boolean
RESET	Positive edge triggered input	Input	Boolean
OUT	Latched output	Output	Boolean

Table 36. BIT_LATCH User Inputs and Outputs

Status	Clock cycles
Max. case	26

Table 37. BIT_LATCH Execution Time

3.2.3.8 PEAK_DETECT

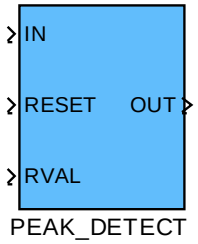


Figure 31. PEAK_DETECT Block

When RESET changes from a low to high value, the peak value of the input (IN) will appear at the output (OUT) for one block execution cycle. Thereafter, if RESET persists, the output will change to RVAL (OUT = RVAL). When RESET goes low, the input will be scanned for peak value again. It is recommended that the designer use the TRANSITION block to generate the RESET signal for continuous peak detection.

The following pseudo-code describes the operation of the PEAK_DETECT block:

```
If (RESET == 1)
        OUT     =   Store_Max
        [s15:0]     [s15:0]
        Store_Max = RVAL
        [s15:0]         [s15:0]
else
        if (IN        >=   Store_Max)
           [s15:0]
                Store_Max = IN
endif
```

Signal name	Description	I/O	Type
IN	Input	Input	16 bit, signed integer
RESET	Reset	Input	Boolean 0 = scan max. value of input 1 = output max and reset
RVAL	Reset Value	Input	16 bit, signed integer
OUT	Output	Output	16 bit, signed integer

Table 38. PEAK_DETECT User Inputs and Outputs

Status	Clock cycles
RESET = 0	18
RESET = 1	16

Table 39. PEAK_DETECT Execution Time

3.2.3.9 TRANSITION – One Shot Pulse Generator

The TRANSITION block provides transition detection of a Boolean input signal (IN). Transition detection format (positive and/or negative edge) can be configured by input POL (see Table 40).

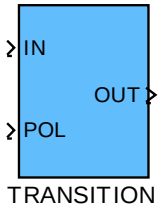


Figure 32. TRANSITION Block

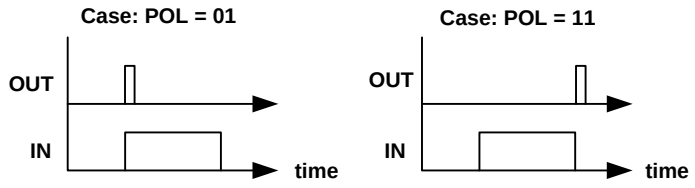
The following pseudo-code describes the function of the TRANSITION block:

```
If  (POL > 0)  // case positive transition detection
{
        If ( IN = 1 and  IN_old = 0) OUT = 1
        Else OUT = 0
 }
Elseif  (POL = 0) // case positive or negative transition detection
{
        If ( IN not equal to IN_OLD)  OUT = 1
        Else OUT = 0
 }
Elseif  (POL < 0)  // case negative transition detection
{
        If ( IN= 0 and  IN_old = 1) OUT = 1
        Else OUT = 0
 }

IN_old = IN
```

Note: IN_old is the previous sample of IN (input).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.



Examples of transition detection

Signal name	Description	I/O	Type
IN	Input signal	Input	Boolean
POL	Configure detection format 00 detects ↑↓ (+ & -) 01 detects ↑ (+) 10 detects ↓ (-) 11 detects ↓ (-)	Input	2-bit, signed integer
OUT	One pulse output for an edge being detected on the input.	Output	Boolean

Table 40. TRANSITION User Inputs and Outputs

Status	Clock cycles
Min. case	20
Max. case	36

Table 41. TRANSITION Execution Time

3.2.3.10 INTEGRAL2 – Integral with Limit

This blocks performs integration with configurable limits.

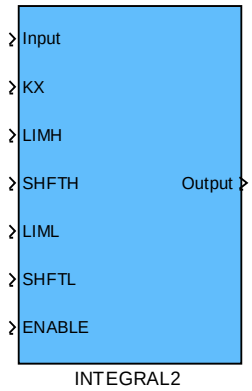


Figure 33. INTEGRAL2 Block

The following pseudo-code describes the function of the INTEGRAL2 block:

```
If ENABLE == 0
        Int(n) = 0
        [s31:0]
else
        Temp1(n)  =  KX       x     Input
        [s31:0]      [s15:0]        [s15:0]
        Int(n)    =  Temp1(n)  +   Int(n-1)
        [s31:0]      [s31:0]        [s31:0]
        Limit Int(n) to 32-bit signed value

        Temp1(n)  =  Shift left LIMH  by SHIFTH bits
        [s31:0]                [s31:0]
        Temp2(n)  =  Shift left LIML by SHIFTL bits
        [s31:0]                [s31:0]

        If Int(n) > Temp1(n)
        [s31:0]    [s31:0]
                Int(n) = Temp1(n)
        If Int(n) < Temp2(n)
        [s31:0]    [s31:0]
                Int(n) = Temp2(n)

 Output(n) = Int(n) [s31:16]
 [s15:0]
```

SHIFTH and SHIFTL are 5-bit unsigned integers with valid range of 0 – 31. Choosing SHIFTH and SHIFTL outside this range could overflow internal registers. Please ensure that LIMH * 2^{SHIFTH} > LIML * 2^{SHIFTL} for proper block utilization.

Signal name	Description	I/O	Type
Input	Input	Input	16 bit, signed integer
KX	Input Gain	Input	16 bit, signed integer
LIMH	Upper limit	Input	16 bit, signed integer
SHIFTH	Scaler for LIMH	Input	5 bit, unsigned integer
LIML	Lower limit	Input	16 bit, signed integer
SHIFTL	Scaler for LIML	Input	5 bit, unsigned integer
ENABLE	Enable/reset	Input	Boolean 0 = Reset recursive data 1 = Normal operation
Output	Output	Output	16 bit, signed integer

Table 42. INTEGRAL2 User Inputs and Outputs

Status	Clock cycles
Min. case	49
Max. case	78

Table 43. INTEGRAL2 Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4 Math

3.2.4.1 DIFF – Subtraction

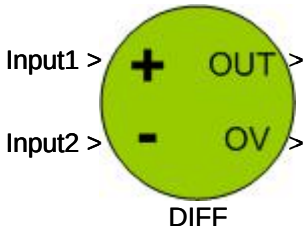


Figure 34. DIFF Block

The following pseudo-code describes the function of the DIFF block:

```
Temp  =   Input1 – Input2
[s15:0]   [s15:0]   [s15:0]
If Temp overflow then
        Temp = 32,767
        OV = 1
Elseif Temp underflow then
        Temp = -32,768
        OV = 1
Else
        OV =  0
 OUT = Temp
 [s15:0]
```

Signal name	Description	I/O	Type
Input1	DIFF block 1st input	Input	16 bit, signed integer
Input2	DIFF block 2nd input	Input	16 bit, signed integer
OUT	DIFF block output	Output	16 bit, signed integer
OV	overflow/underflow status bit	Output	Boolean, 1 16-bit signed integer overflow 0 no overflow

Table 44. DIFF Inputs and Outputs

Status	Clock cycles
Max. case	22

Table 45. DIFF Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4.2 SUM – Addition

The function of the SUM block can be described using the following pseudo-code:

```
Temp     =    IN1     +     IN2
[s15:0]       [s15:0]      [s15:0]
If Temp positive overflow (greater than 32,767) then
            Temp = 32,767
            OV = 1
Elseif OUT negative underflow (less than -32,768) then
            Temp = -32,768
            OV = 1
Else
            OV = 0
 OUT  =  Temp
 [s15:0]
```

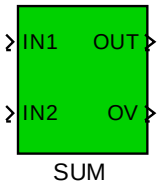


Figure 35. SUM Block

Signal name	Description	I/O	Type
IN1	SUM block 1st input	Input	16 bit, signed integer
IN2	SUM block 2nd input	Input	16 bit, signed integer
OUT	SUM block output	Output	16 bit, signed integer
OV	overflow/underflow status flag	Output	Boolean 1 16-bit signed integer overflow 0 no overflow

Table 46. SUM Inputs and Outputs

Status	Clock cycles
Max. case	21

Table 47. SUM Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4.3 ACCUMULATOR

This accumulator block is an integrator without overflow protect. This block can be used to build counters.

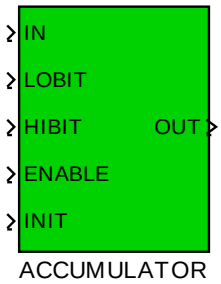


Figure 36. ACCUMULATOR Block

The following pseudo-code describes the operation of the ACCUMULATOR block:

```
If (ENABLE == 0)
        Sign extend INIT to 32 bits
        Output32 = INIT      *    2 LOBIT
         [s31:0]     [s15:0]

Output32 = Output32   +    IN
[31:0]      [31:0]         [s15:0]

If (HIBIT – LOBIT) > 15
        OUT = Output32 [LOBIT +15 : LOBIT]
        [15:0]
Else
        OUT = Output32 [HIBIT : LOBIT]
         [15:0]
```

If HIBIT or LOBIT are outside of the allowed range (0 <= LO/HIBIT <= 31) or LOBIT > HIBIT, then the OUT signal will be invalid. When HIBIT – LOBIT is less than 15, unfilled MSBs of the output will be zero.

Note: When enable = 0, the output of accumulator block is equal to (INIT + IN).

Signal Name	Description	I/O	Type
IN	Accumulator input	Input	16 bit, signed integer
LOBIT	Output extraction low bit.	Input	5 bit, unsigned integer
HIBIT	Output extraction high bit.	Input	5 bit, unsigned integer
ENABLE	Enable control bit	Input	Boolean, 0 freeze output Out = INIT + IN 1 normal accumulating
INIT	Reset value	Input	16 bit, unsigned integer
OUT	Output	Output	16 bit, unsigned integer

Table 48. ACCUMULATOR User Inputs and Outputs

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Status	Clock cycles
Normal Operation	42

Table 49. ACCUMULATOR Execution Time

3.2.4.4 COUNTER

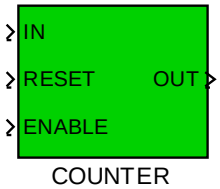


Figure 37. COUNTER Block

The pseudo-code shown below describes the operation of the COUNTER block.

```
If (RESET == 1 )
        OUT(n) = 0
         [s15:0]
Else
        If (ENABLE == 1 )
                OUT(n) = OUT(n-1) + IN(n)
                [s15:0]     [s15:0]        [s15:0]
                 Overflow protect OUT(n) to 16-bit signed integer
        Endif
 Endif
```

For example, if ENABLE = 1 and IN = 1, OUT will count up by 1 for each block execution until it reaches 32,767. The output (OUT) will remain at 32,767 until a negative integer is supplied at IN, or RESET = 1.

Signal name	Description	I/O	Type
IN	Counter Input	Input	16 bit, signed integer
RESET	Reset counter control bit	Input	Boolean 0 = normal operation, 1 = initialize output to 0
ENABLE	Enable Count control bit	Input	Boolean 0 = freeze output value 1 = unfreeze output
OUT	Output	Output	16-bit signed integer

Table 50. COUNTER User Inputs and Outputs

Status	Clock cycles
Nominal	19

Table 51. COUNTER Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4.5 SHIFT – Multiply by a Power of Two

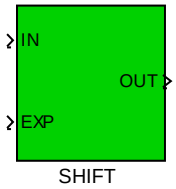


Figure 38. SHIFT Block

The SHIFT block multiplies an input by a power of two, as shown below.

$$\underset{[s15:0]}{OUT} = \underset{[s15:0]}{IN} \times 2^{EXP}$$

This operation can also be viewed as a binary shift left or right, where EXP specifies the number of bits to shift the IN value. A positive value for EXP shifts left and a negative value shifts right. The SHIFT block retains the sign bit of the input when shifting right (negative EXP).

Note: Please ensure that $-32768 <= IN * 2^{EXP} <= 32767$ *to avoid output overflow*

Signal name	Description	I/O	Type
IN	Input	Input	16-bit signed integer
EXP	Exponent	Input	-15 < EXP < 15
OUT	Output	Output	16-bit signed integer

Table 52. SHIFT Inputs and Outputs

Status	Clock cycles
Nominal	5

Table 53. SHIFT Execution Time

3.2.4.6 MUL_DIV – Signed / Unsigned Multiplier with Extraction

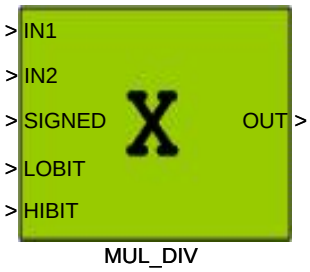


Figure 39. MUL_DIV Block

Pseudo code for MUL_DIV is given by:

```
If (SIGNED == 1)
        Temp =     IN1     x     IN2
        [s31:0]    [s15:0]       [s15:0]
        If Temp overflows HIBIT + 1 bits
                Temp = max signed value for HIBIT + 1 bits (see Notes)
        OUT = Temp[HIBIT : LOBIT]
        [s15:0]
else
        Temp =     IN1     x     IN2
        [31:0]     [15:0]        [15:0]
        If Temp overflows HIBIT + 1 bits
                Temp = max unsigned value for HIBIT + 1 bits (see Notes)
        OUT = Temp[HIBIT : LOBIT]
        [15:0]
End
```

Notes:
HIBIT – LOBIT + 1 bits are extracted from Temp, sign extended and stored in OUT. For example, if LOBIT = 2, HIBIT = 13, 12 bits are extracted from Temp (OUT [11 : 0] = Temp [13: 2]). The upper significant bits of OUT (OUT [15:12]) are filled with zeros if the result is positive or ones if the result is negative.

In the case that the 32-bit result (Temp) overflows N bits, where N = HIBIT + 1, Temp is set to the largest value for an N-bit signed or unsigned integer, depending on the mode. For signed underflow, Temp is set to the most negative N-bit signed integer. OUT is extracted from Temp, as described above, *after* this overflow adjustment is performed.

SIGNED, LOBIT and HIBIT are compile time parameters and should be connected to Simulink Constant blocks rather than variables. If (HIBIT – LOBIT) > 15, the MCE compiler will issue an error.

Examples:
LOBIT = 5, HIBIT = 14, SIGNED = 1, IN1 = -32, IN2 = 3
Temp = -32 * 3 = -96 = 0xFFFFFFA0. There is no overflow on a signed 15-bit integer. Extraction of bits [14 : 5] with sign extension gives an output value of 0xFFFD or -3.

LOBIT = 5, HIBIT = 14, SIGNED = 1, IN1 = 273, IN2 = 844
Temp = 273 * 844 = 230,412 = 0x3840C. This value overflows a signed 15-bit integer, so Temp = 0x3FFF (max positive 15-bit integer). Extraction of bits [14 : 5] gives an output value of 0x1FF or 511.

LOBIT = 5, HIBIT = 14, SIGNED = 0, IN1 = 3200, IN2 = 3
Temp = 3200 * 3 = 9600 = 0x2580. There is no overflow on an unsigned 15-bit integer. Extraction of bits [14 : 5] gives an output value of 0x12C or 300.

Signal name	Description	I/O	Type
IN1	Input1	Input	16 bit, signed integer
IN2	Input2	Input	16 bit, signed integer
SIGNED	Signed / Unsigned	Input	Boolean constant, 0 = unsigned, 1 = signed
LOBIT	Starting bit	Input	5 bit, unsigned (0 – 31)
HIBIT	Ending bit	Input	5 bit, unsigned (0 – 31)
OUT	Output	Output	16bit, signed integer

Table 54. MUL_DIV Inputs and Outputs

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Status	Clock cycles
Signed	15
Unsigned	23

Table 55. MUL_DIV Execution Time

3.2.4.7 DIVIDE

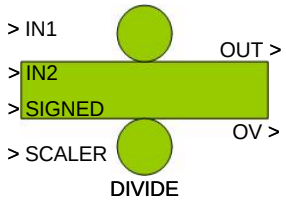


Figure 40. DIVIDE Block

The following pseudo-code describes the operation of the DIVIDE block:

```
If (IN2 == 0)
        Output = 0
        OV = 1
Else
        Temp =    IN1      x      2^SCALER
        [s31:0]   [s15:0]         [4:0]
        Output =  Temp    /    IN2
        [s31:0]   [s31:0]    [s15:0]
        If SIGNED = 0
            If (Output > 65535)
                Output = 65535
                OV = 1
        Else
            If (Output > 32767)
                Output = 32767
                OV = 1
            Elseif (Output < -32768)
                Output = -32768
                OV = 1
            Else OV = 0
        Endif
Endif
OUT = Output[15:0]
```

In the case of signed division, if (IN1*2^{SCALER})/IN2 overflows the 32-bit signed internal register (Output), the output may have incorrect sign and the overflow (OV) bit will not be set.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Type
IN1	Dividend	Input	16 bit, signed integer
IN2	Divisor	Input	16 bit, signed integer
SCALER	Dividend scale factor	Input	5 bit, unsigned integer (0 – 16)
SIGNED	Signed / Unsigned selector	Input	Boolean 0 = unsigned 1 = signed
OUT	Output	Output	16 bit, signed integer
OV	Overflow flag	Output	Boolean 0 = no overflow 1 = overflow

Table 56. DIVIDE Inputs and Outputs

Status	Clock cycles
Signed	39
Unsigned	38

Table 57. DIVIDE Execution Time

3.2.4.8 NOT – Bitwise Inversion

The NOT block performs a bitwise (logical) inversion of the unsigned 16-bit input value. That is, all 0 bits are set to 1 and all 1 bits are set to 0. For example, for Input = 30764, Output = 2003

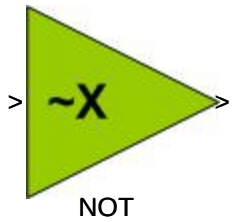


Figure 41. NOT Block

Signal name	Description	I/O	Type
Input	Input	Input	16 bit, unsigned integer
Output	Output	Output	16 bit, unsigned integer

Table 58. NOT Inputs and Outputs

Status	Clock cycles
Nominal	5

Table 59. NOT Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4.9 NEGATE – Two's Complement

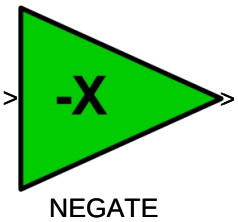


Figure 42. NEGATE Block

The NEGATE block performs a two's complement of the signed 16-bit input value.

Output = - Input
[s15:0] [s15:0]

Note: 16-bit signed input range excludes -32768 (-32767 <= Input <= 32767). If input = –32768, output = –32768.

Signal name	Description	I/O	Type
Input	Input	Input	16 bit signed integer
Output	Output	Output	16 bit,signed integer

Table 60. NEGATE Inputs and Outputs

Status	Clock cycles
Nominal	5

Table 61. NEGATE Execution Time

3.2.4.10 AND – Bitwise Logical AND

The AND block performs a bitwise logical AND operation on two 16-bit unsigned input values. For example, INPUT1 = 5319 and INPUT2 = 30764 yields OUTPUT = 4100.

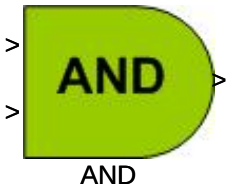


Figure 43. AND Block

Signal name	Description	I/O	Type
INPUT1	Input1	Input	16 bit, unsigned integer
INPUT2	Input2	Input	16 bit, unsigned integer
OUTPUT	Output	Output	16 bit, unsigned integer

Table 62. AND Inputs and Outputs

Status	Clock cycles
Nominal	10

Table 63. AND Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.2.4.11 OR – Bitwise Logical OR

The OR block performs a bitwise logical OR operation on two 16-bit unsigned input values. For example, INPUT1 = 5319 and INPUT2 = 30764 yields OUTPUT = 31983.

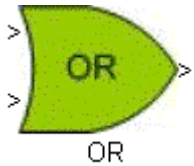


Figure 44. OR Block

Signal name	Description	I/O	Type
INPUT1	Input 1	Input	16 bit, unsigned integer
INPUT2	Input 2	Input	16 bit, unsigned integer
OUTPUT	Output	Output	16 bit, unsigned integer

Table 64. OR Inputs and Outputs

Status	Clock cycles
Nominal	10

Table 65. OR Execution Time

3.2.4.12 XOR – Bitwise Logical Exclusive OR

The XOR block performs a bitwise logical exclusive OR operation on two 16-bit unsigned input values. For example, INPUT1 = 5319 and INPUT2 = 30764 yeilds OUTPUT = 27883.

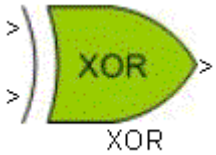


Figure 45. XOR Block

Signal name	Description	I/O	Type
INPUT1	Input 1	Input	16 bit, unsigned integer
INPUT2	Input 2	Input	16 bit, unsigned integer
OUTPUT	Output	Output	16 bit, unsigned integer

Table 66. XOR Inputs and Outputs

Status	Clock cycles
Nominal	10

Table 67. XOR Execution Time

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3 Motion Peripherals

This section describes the motion peripheral blocks of the MCE library. Each motion peripheral block provides an interface to fixed elements of the IRMCx100-series firmware or hardware. Therefore, unlike other blocks in the library, the motion peripherals may be used only once in a design.

Motion peripherals differ from control blocks in their manner of execution. The connections to motion peripherals within the design do not determine the order of module execution. Motion peripheral inputs and outputs simply provide an interface to the registers that configure, control and monitor the associated firmware and hardware elements. These elements run before the MCE design during the PWM cycle and execute whether or not the corresponding motion peripheral blocks are included in the MCE design.

3.3.1 SENSORLESS_FOC

The inputs and outputs of the SENSORLESS_FOC module can be customized using the CustomMotPer utility described in Section 5.6. Figure 46 shows the block's inputs and outputs in the default configuration. The entire list of available inputs and outputs is presented in alphabetical order in Table 68.This block does not apply for servo control IC, IRMCx143.

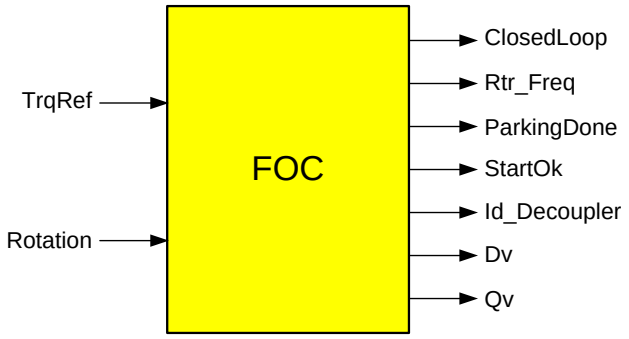


Figure 46. SENSORLESS_FOC Block

The SENSORLESS_FOC library block provides an interface to a subset of the IRMCx100-series motion peripheral registers. All registers that control and monitor the operation of the sensorless field orientation control are accessible through the SENSORLESS_FOC block (with customization). Note that many of these registers values can be calculated using the MCEWizard tool and are typically initialized only once at system startup from an 8051 or host application. It is normally not necessary to update these registers from within the MCE design and they are, therefore, rarely customized as inputs to the SENSORLESS_FOC block. These inputs are identified by the term "Config Input" in the "I/O" column of Table 68.

Each signal listed in the table corresponds directly to one of the motion peripheral registers described in Section 3.4 or, where noted, to a bit field within a motion peripheral register. The signal name is the same as the register or bit field name. The rightmost column of Table 68 provides a reference to the document section that describes the associated register.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Reference for detailed description and scaling
AngDel	Gain adjustment for current angle	Config Input	3.4.5
AngLim	Maximum limit on current angle phase	Config Input	3.4.5
AtanTau	Angle compensation for phase shift	Config Input	3.4.5
BrakeMaxSpeed	Maximum speed for reverse catch-spin startup	Config Input	3.4.10
CatchEnb	Enable catch spin (MtrCtrlBits_S, bit 0)	Config Input	3.4.1
CatchMaxSpeed	Maximum speed for forward catch-spin startup	Config Input	3.4.10
CatchTm	Sampling time of speed during catch-spin.	Config Input	3.4.10
ClosedLoop	Closed loop mode	Output	3.4.19
CsZcTimeout	Timeout when searching for current zero crossing during catch-spin	Config Input	3.4.10
Di	d-axis current	Output	3.4.21
DiagSelect	Setting of diagnostics mode (MtrCtrlBits 0 to 3)	Config Input	3.4.1
DisableFaults	Disable of fault handling	Config Input	3.4.1
Dv	D-axis command modulation index	Output	3.4.21
ExtFwdAngle	Angle sum into forward rotating angle	Input	3.4.14
ExtRevAngle	Angle sum into reverse rotating angle	Input	3.4.14
Ext_Flx_Alpha	External Alpha flux input	Input	3.4.14
Ext_Flx_Beta	External Beta flux input	Input	3.4.14
FaultFlags	Fault Status	Output	1.1.1
Flx_Alpha	Estimated motor flux of Alpha axis	Output	3.4.21
Flx_Beta	Estimated motor flux of Beta axis	Output	3.4.21
FlxAInit	Initial Alpha flux level	Config Input	3.4.5
FlxBInit	Initial Beta flux level	Config Input	3.4.5
Flx_M	Fundamental amplitude of Flx_Alpha	Output	3.4.21
FlxTau	Adjustment for flux estimator bandwidth	Config Input	3.4.5
FlxChkT	Time instant to check flux after start-up	Config Input	3.4.7
FlxThr_C	Flux threshold during catch spin	Config Input	3.4.10
FlxThrH	Upper threshold for flux check	Config Input	3.4.7
FlxThrL	Lower threshold for flux check	Config Input	3.4.7
FreqBW	Filter bandwidth for motor frequency	Config Input	3.4.5
I_Alpha	Alpha phase current	Output	3.4.21
I_Beta	Beta phase current	Output	3.4.21
IdRefExt	External d-axis motor current reference	Input	3.4.12
IdRef_C	d-axis command current	Output	3.4.21
Id_Decoupler	d-axis Current Decoupler output	Output	3.4.23
IfbkScl	Current gain	Config Input	3.4.9
IqRef_C	q-axis command current	Output	3.4.21
IregCompEnb	DC bus compensation (MtrCtrlBits_S, bit 4)	Config Input	3.4.1
IScl	Current gain scaler for flux estimator	Config Input	3.4.5
KpIreg	Proportional gain of q-axis current regulator	Config Input	3.4.3
KpIregD	Proportional gain of d-axis current regulator	Config Input	3.4.3
KxIreg	Integral gain of d- and q-axis current regulator	Config Input	3.4.3
KTorque	Gain used in open-loop startup mode	Config Input	3.4.6
L0	Apparent inductance of the motor	Config Input	3.4.5
LSlncy	Apparent saliency inductance of the motor	Config Input	3.4.5
MinSpd	Minimum allowed drive operating speed	Config Input	3.4.4
MtrSeqCtrl	Commands start and stop	Config Input	3.4.1

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Reference for detailed description and scaling
ParkingDone	Parking complete	Output	3.4.19
ParkingOne	First parking stage complete	Output	3.4.19
ParkAng	Second angle for parking stage	Config Input	3.4.7
ParkAng1	First angle for parking stage	Config Input	3.4.7
ParkI	DC current injection for parking stage	Config Input	3.4.7
ParkTm	Total parking time	Config Input	3.4.7
PllFreqLim	Frequency limit of the PLL integral gain output	Config Input	3.4.5
PllFreqLimCs	PLL frequency limit during catch spin	Config Input	3.4.10
PllKi	PLL tracking integral gain	Config Input	3.4.5
PllKp	PLL tracking proportional gain	Config Input	3.4.5
Qi	q-axis current feedback	Output	3.4.21
Qv	q-axis command modulation	Output	3.4.21
Rotation	Direction of rotation	Input	3.4.4
RotorAngle	Estimated rotor angle	Output	3.4.21
Rs	Motor per phase resistance	Config Input	3.4.5
Rtr_Freq	Estimated unfiltered rotor electrical frequency	Output	3.4.22
SearchAng	Rotor angle search command	Output	3.4.22
SequencerState	Reflects the FOC blocks main state	Output	1.1.1
SpeedScale	Internal speed scaling	Config Input	3.4.4
StartFail	Startup failed (FaultFlags, bit 7)	Output	1.1.1
StartOk	Startup succeeded	Output	3.4.19
ThetaStart	Initial rotor angle value for open-loop startup during catch-spin	Config Input	3.4.10
TrqRef	Command current input	Input	3.4.4
TwoPhsStatus	Indicates that two phase modulation is enabled	Output	3.4.19
UdFeedFwd	d-axis modulation feedforward	Input	3.4.14
UqFeedFwd	q-axis modulation feedforward	Input	3.4.14
Use2xFrqScl	Use 2x frequency scale (MtrCtrlBits, bit 5)	Config Input	3.4.1
Use4xFrqScl	Use 4x frequency scale (MtrCtrlBits_S, bit 1)	Config Input	3.4.1
Use2xMagScl	Use 2x flux boost for PLL (MtrCtrlBits_S, bit 2)	Config Input	3.4.1
Use4xMagScl	Use 4x flux boost for PLL (MtrCtrlBits_S, bit 3)	Config Input	3.4.1
UseExtFlux	Use external fluxes for PLL (MtrCtrlBits_S, bit 5)	Config Input	3.4.1
VdLim	d-axis current regulator output limit	Config Input	3.4.3
VFFreq	Volts/Hertz frequency for open-loop diagnostic	Config Input	3.4.16
VFGain	Volts per Hertz gain scaler for open-loop diagnostic	Config Input	3.4.16
VoltScl	Internal voltage scaling gain	Config Input	3.4.5
VqLim	q-axis current regulator output limit	Config Input	3.4.3
V_Alpha	Alpha motor phase voltage	Output	3.4.21
V_Beta	Beta motor phase voltage	Output	3.4.21
WeThr	Transition level for closed-loop operation	Config Input	3.4.7
Zero_Vec_Req	Forces output of current controllers to zero (zero modulation).	Config Input	3.4.16
ZeroVecTm	Duration of braking during reverse catch spin	Config Input	3.4.10
ZeroVoltBrake	Enable braking with alternating lower/upper inverter switches	Config Input	3.4.10

Table 68. SENSORLESS_FOC Available Inputs and Outputs

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The Inputs of the SENSORLESS_FOC block from the MCE program are used in the next PWM cycle, while the "Config Inputs" may have a latency of up to two PWM cycles depending on exactly when they are written. The outputs of the SENSORLESS_FOC block are valid immediately at the start of each PWM cycle and represent the values calculated during the previous cycle.

Figure 47 shows a block diagram of the sensorless field orientation control functions. There are three main components to the Sensorless FOC block. These components are Current Control, Rotor Position estimation and Startup Control.

3.3.1.1 Current Control

The current control system consists of two PI regulators, two Vector Rotators (forward and backward), a Current Decoupler, and a Clark transformation (3 → 2 phase).

Two Proportional plus Integral (PI) type current regulators with output limits and Anti-windup control are provided for torque and flux current control of motors. These two PI regulators operate in conjunction with a forward and a backward vector rotator to form a synchronously rotating frame current control system. The rotor magnet position is being chosen to be the frame of reference for the current control. This is done to achieve "Field-Orientation Control". The rotor position is supplied by an angle estimator.

The current controller receives current commands (d-q) from a Current decoupler. This decoupler calculates the optimum q-axis and d-axis (rotor magnet) command current to achieve maximum motor Torque per Ampere generation. The d-axis command current (IdRefExt) is an input to the Sensorless FOC. This flux current input can be fed from the d-axis output (Id_Decoupler) of the Current decoupler or can be obtained externally by another form of d-axis current control regime such as Field-weakening control. Both of these functions are implemented in the factory supplied MCE program.

Each current controller has an associated limit (VdLim, VqLim). VdLim operates normally, but the q-current current controller limit is calculated dynamically each PWM cycle based on the maximum modulation and the Vd modulation command. The minimum of this calculated value and VqLim is then applied as the limit on the output of the regulator.

The inputs to the forward Vector Rotator (converts dc to ac waveforms) are PWM modulation depths. The outputs of the Vector Rotator are fed to a Space Vector PWM (SVPWM) modulator for inverter firing control.

3.3.1.2 Rotor Position Estimation

The rotor position estimation consists of a flux estimator and an Angle-Frequency generator.

Rotor magnet position is calculated by an angle estimator which estimates rotor magnet position based on feedback current, estimated voltages (based on dc bus feedback voltage and modulation index) and motor parameters (inductance and resistance). The output of the flux estimator represents rotor magnet fluxes in Alpha-Beta (stationary orthogonal frame, u-phase aligned with Alpha) two-phase quantities. The Angle-Frequency generator computes position and frequency from the Alpha-Beta flux inputs. Users can also have the flexibility of using external fluxes (Ext_Flx_Alpha and Ext_Flx_Beta) instead of internally generated fluxes. This allows implementation of a Field-oriented control scheme using enhanced position sensing or flux sensing methods to achieve high performance low speed operations.

Another level of flexibility is provided with the option to completely replace the estimated rotor angle by external signals (ExtFwdAngle, ExtRevAngle).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

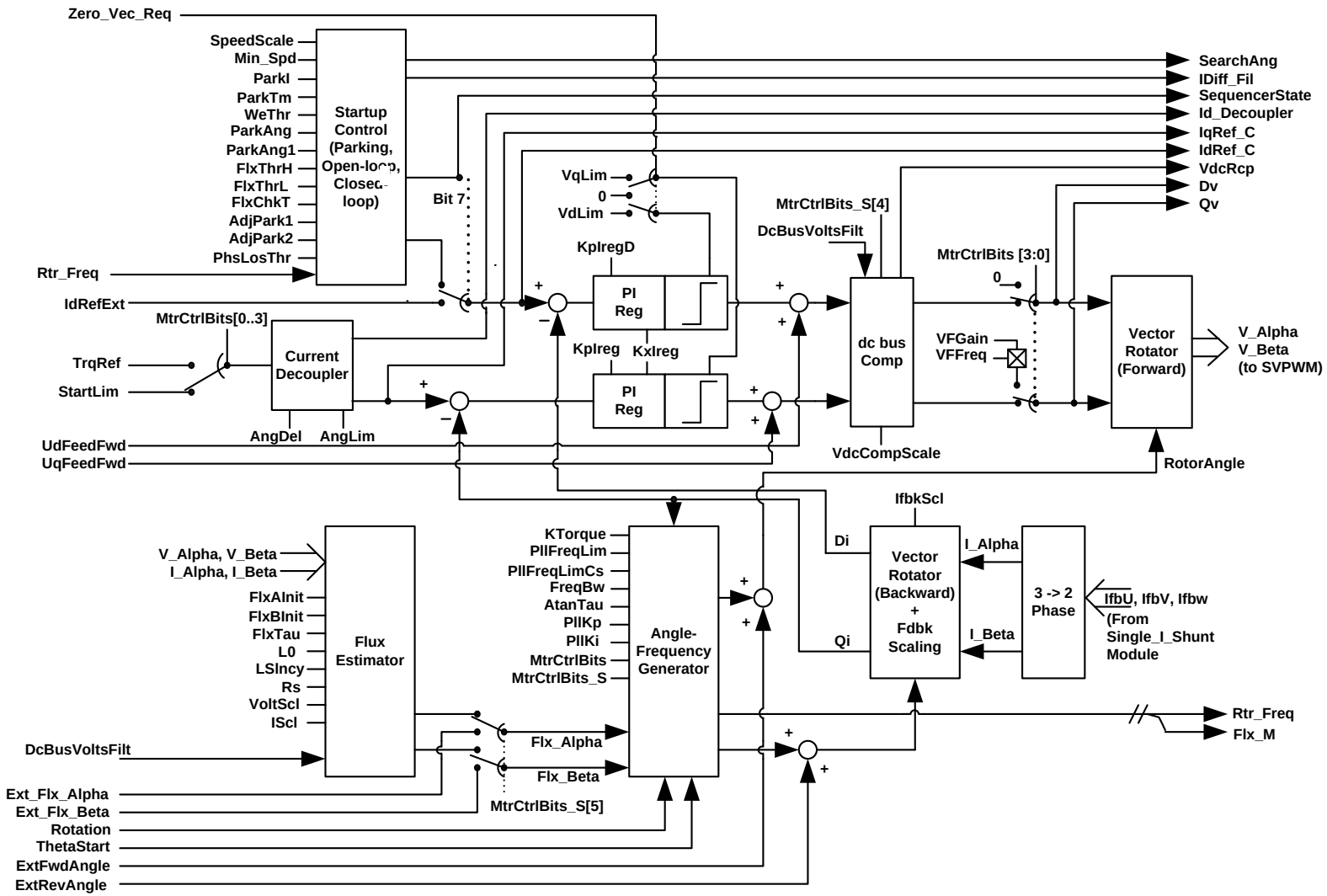


Figure 47. SENSORLESS_FOC Block Diagram

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.1.3 Startup Control

The motor control is performed without a shaft encoder (Sensorless). A special startup sequence is implemented to provide robust startup. The startup control block inside the Sensorless FOC block is used to assist drive startup with the ability to configure the controller dynamically to three unique operating states (Parking, Open-loop or Closed-loop). These three states are illustrated in Figure 48 and described below.

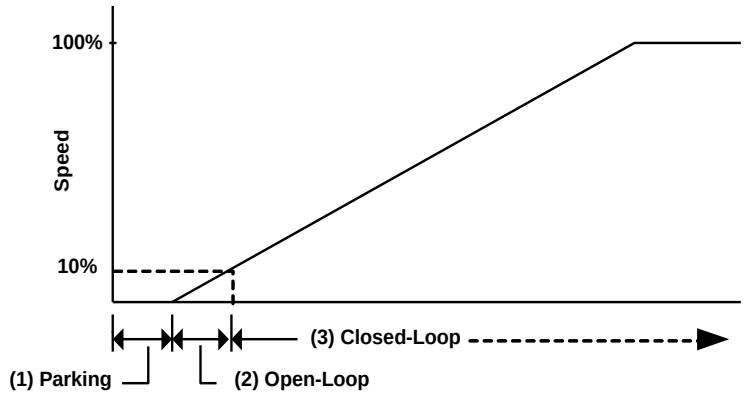


Figure 48. Drive Control Modes

State 1: Parking

The initial rotor angle is identified by forcing DC current into the motor and hence forcing the motor shaft to park at a certain prescribed angle. If catch spin is enabled then parking is replaced by zero vector braking at low and reverse speeds. If catch spin is enabled, then parking and open loop is eliminated at high forward speeds.

State 2: Open-loop

Immediately after Parking stage, the rotor angle is estimated in an open-loop fashion, which utilizes a simple motor-load mechanical model to estimate the rotor angle (estimate load characteristics). If the mismatch between the external load characteristics and the internal motor-load model is exceedingly large, start-up performance will suffer.

State 3: Closed-loop

Motor speed increases during start-up; the motor voltage also builds up due to the increase in speed. Useful information for rotor angle estimation can then be extracted from the motor voltage (estimated by using PWM modulation depth, DC bus voltage, and motor current). The drive will enter Closed-loop control mode as shown in Figure 48.

3.3.1.4 Sequencing

The FOC block has built-in sequencing which takes care of all state-handling related to running a motor. The user only needs to interact with the sequencer when start/stop commands are issued (MtrSeqCtrl, see section 3.4.1). A state diagram of the sequencing is shown in Figure 49. Blue type is the condition for sequencing state change.

Note that the variable sequencer_state shown in Figure 49 represents the actual system state and is equivalent to register SequencerState. A more detailed description of the catch spin states can be found in section 3.3.1.5.

3.3.1.5 Catch Spin

In some applications, the motor can be spinning prior to a real start of the system. IRMCx100 has algorithms that can handle both a forward and a reverse spinning rotor, i.e. catch spin support. State handling of this particular part of the controller is shown in Figure 50. Three groups of states/decision points are found in this figure, each group indicated by a color. The yellow, green and beige blocks deal with, respectively, initialization, forward catch spin and reverse catch spin. The algorithm will be discussed below with respect to these colors.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Initialization (yellow)
The main purpose of this part of the algorithm is to determine if catch spin is required and if so, whether in forward or reverse. First, if catch spin is not enabled (MtrCtrlBits_S[0] = 0) then the system proceeds to normal startup. If MtrCtrlBits_S[0] = 1 catch spin is initiated by setting i_d and i_q reference to zero and enabling the FOC. This will allow the current measurement to operate and the motor flux linkage to be estimated. However, since the requested current is 0, no torque will be provided.
The FOC is kept in closed loop for a period of time (CatchTm) after which all internal signals should reach steady state. Next, based on the absolute level of flux linkage, it is determined whether the motor is spinning at a speed which requires catch spin. At low speed (low flux linkage) the algorithm proceeds directly to normal startup with parking. If the flux linkage level is higher that the threshold (FluxThr_C) it means the motor is spinning at a speed that requires either forward or reverse catch spin—determined by comparing the target direction (TargetDir) to the actual direction.

Forward catch spin (green)
Once it has been established that the rotor is already spinning in the desired direction, it is simply a matter of bringing the FOC into normal closed loop operating mode. However, IRMCx100 makes a check on the motor speed before doing so. If the speed is higher than CatchMaxSpeed it is interpreted as the speed is too high for closed loop, a fault is generated (CsOverSpdFlt) and the system enters the fault state. It is left to the user what to do next – normally to retry startup.

Reverse Catch Spin (beige)
If the motor is spinning opposite the target direction, the algorithm proceeds to reverse catch spin. As with forward catch spin, a check is done on the speed. If it is too high, braking the motor may damage the drive. In this case a fault is generated (CsOverSpdFlt) and the system enters the fault state. It is left to the user what to do next.

If the speed is lower than BrakeMaxSpeed, braking is initiated with a choice between two types of braking – zero voltage brake (50% duty cycle) and zero vector brake (low side clamp). The braking method is set in ZeroVoltBrake.

Braking will continue for the time specified by ZeroVectTm, after which zero voltage brake is applied (regardless of ZeroVoltBrake) in order to engage the single shunt current reconstruction and observe the motor currents. A wait time of 100ms is provided for the system to reach steady state. Next, the algorithm starts to monitor i_α and i_β to determine whether the motor is still spinning or not. For a spinning motor, AC currents will be observed. On the other hand if no currents are observed, it is interpreted as a stopped motor which can be started as normal with parking (lower right part of Figure 50).

If AC currents are observed it means that, in spite of the braking, the rotor is still spinning. In this case there are two options – either the motor speed is too high for startup to be attempted or the rotor speed is low enough that it is safe to proceed to startup. If the later is true, a startup will be done without parking. Since the motor is spinning it does not make any sense to do parking.

To do startup without parking, the algorithm needs to search for the correct moment to initiate open loop startup. The algorithm waits for i_α to have a negative-to-positive going zero-crossing and, at the same time, i_β to be positive. In this case, the rotor angle is known and high starting torque can be provided.

If the rotor is still spinning after braking, it is possible that the speed is too high for startup to be attempted. If the flux linkage is greater than FluxThr_C it means the motor speed is too high and that catch spin has failed. In this case, CsOverSpdFlt is set and the system goes into a fault state. It is left to the user what to do next.

From any state except state_fault the system will go to state_stopped if a stop command is issued.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

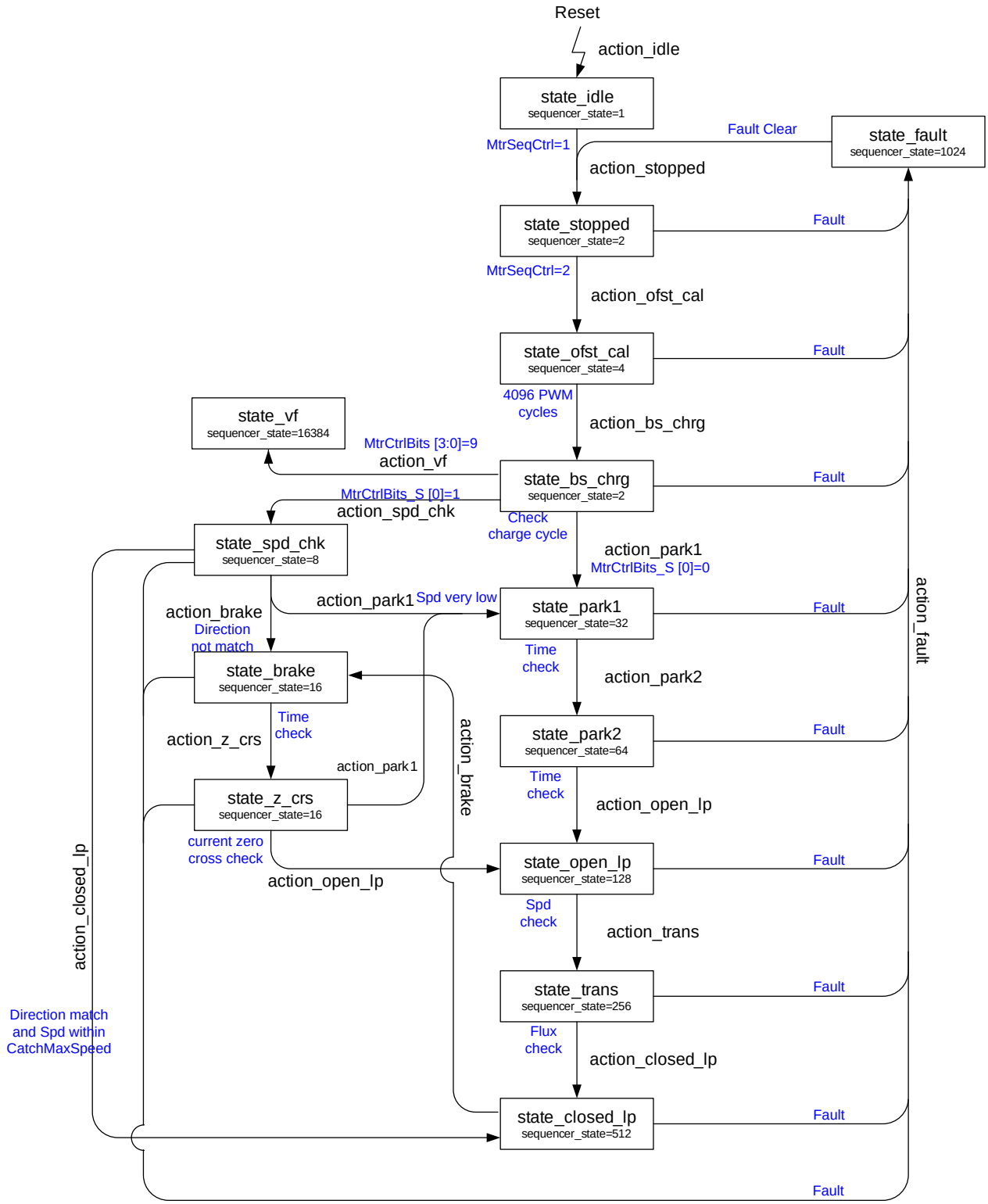


Figure 49. Sequencer of IRMCx100.

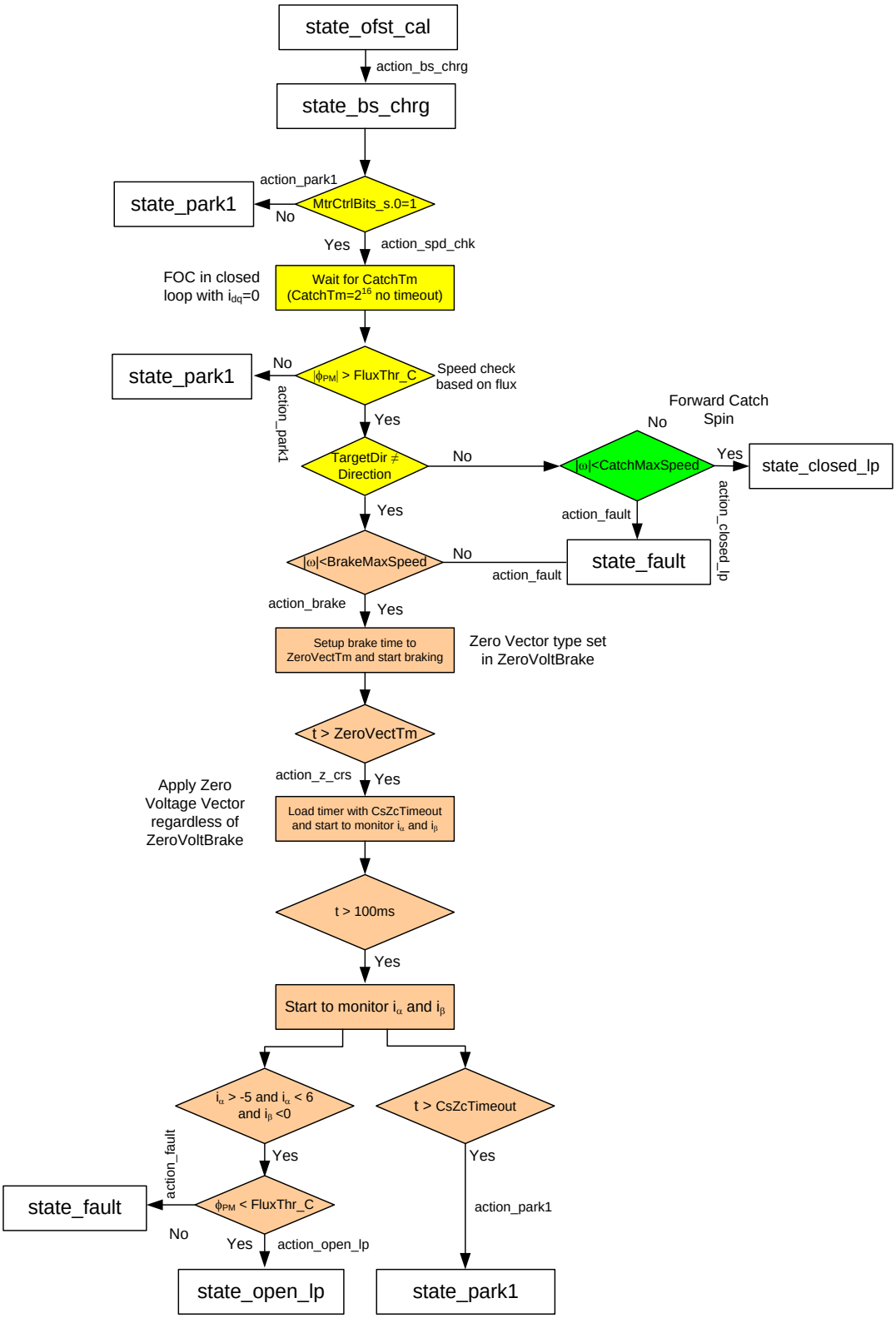


Figure 50. State Handling of Catch Spin Algorithm in IRMCx100

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.1.6 Diagnostic Modes

To facilitate development and troubleshooting, several diagnostic modes are available. The diagnostic mode can be selected using register MtrCtrlBits[0:3] (3.4.1).

Parking Diagnostic

When the parking diagnostic is selected, the sequencer will stay in the state_park2 indefinitely and will not transistion to state_open_lp. The parking current will be applied until the sequencer recives a stop command or an enabled fault occurs.

Start-up Diagnostic

In start-up diagnostic mode, the sequencer proceeds through stat_open_lp. When the threshold frequency (register WeThr) is reached, the sequencer returns to state_stopped (instead of proceeding to state_trans) and all outputs to the inverter are turned off.

Current Regulator Diagnostic

The current regulator diagnostic repeatedly applies a step command to the current regulator. The step command sequence has a 40ms period and 50% duty cycle. The amplitude of the step command is determined by register ParkI.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.2 CURRENT_MEAS

The inputs and outputs of the CURRENT_MEAS module can be customized using the CustomMotPer utility described in Section 5.6. Figure 51 shows the block's outputs in the default configuration. (There are no inputs in the default configuration.) The entire list of available inputs and outputs is presented in alphabetical order in Table 69.

The CURRENT_MEAS library block provides an interface to a subset of the IRMCx100-series motion peripheral registers. All registers that control and monitor the operation of the sensorless field orientation control are accessible through the CURRENT_MEAS block (with customization). Note that some of these registers values can be calculated using the MCEWizard tool and are typically initialized only once at system startup from an 8051 or host application. It is normally not necessary to update these registers from within the MCE design and they are, therefore, rarely customized as inputs to the CURRENT_MEAS block. These inputs are identified by the term "Config Input" in the "I/O" column of Table 69.

Each signal listed in the table corresponds directly to one of the motion peripheral registers described in Section 3.4 or, where noted, to a bit field within a motion peripheral register. The signal name is the same as the register or bit field name. The rightmost column of Table 69 provides a reference to the document section that describes the associated register.

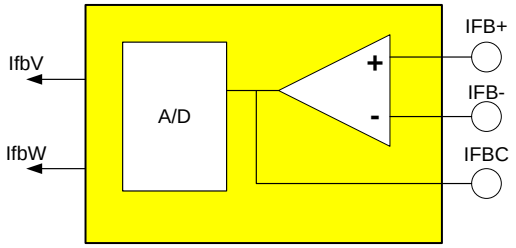


Figure 51. CURRENT_MEAS Block

Signal name	Description	I/O	Reference for detailed description and scaling
IfbOffset	Current feedback dc offset compensation	Output	3.4.23
IfbU	Reconstructed motor phase U current	Output	3.4.21
IfbV	Reconstructed motor phase V current	Output	3.4.21
IfbW	Reconstructed motor phase W current	Output	3.4.21
ScsSamples	Setup for adaptive current feedback sampling	Input	3.4.9
SHDelay	Hardware PWM gate propagation delay	Config Input	3.4.9
TCntMin2Phs	Minimum PWM pulse width for 2-phase modulation mode	Config Input	3.4.9
TCntMin3Phs	Minimum PWM pulse width for 3-phase modulation mode	Config Input	3.4.9

Table 69. CURRENT_MEAS available Inputs and Outputs

The outputs of the CURRENT_MEAS block are valid immediately at the start of each PWM cycle and represent the values calculated during the previous cycle.

3.3.2.1 Motor current sampling

A two-level inverter can produce eight possible basic voltage vectors; any desired (command) voltage vector can be formed by these eight vectors up to the inverter maximum output voltage limit (determined by the dc bus voltage level). In a PWM inverter drive system, the information of motor phase can be observed from the dc bus current when non-zero basic vectors are used. Each basic vector is assigned a specific time in a PWM cycle in order to generate the command voltage vector. However, if the time spent on any basic vector is not long enough, the motor current cannot be observed.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

The dc link current consists of high frequency (PWM switching) current pulses. These current pulses coupled with circuit layout parasitic and reverse recovery diode current cause ringing in the dc link feedback current. Therefore, minimum pulse constraints are inserted to allow reliable dc link current feedback sampling.

Figure 52 shows the insertion of minimum time constraints (TCntMin3Phs, TCntMin2Phs) and the sampling instances (S1 to S4) correspond to idealized PWM command gatings.

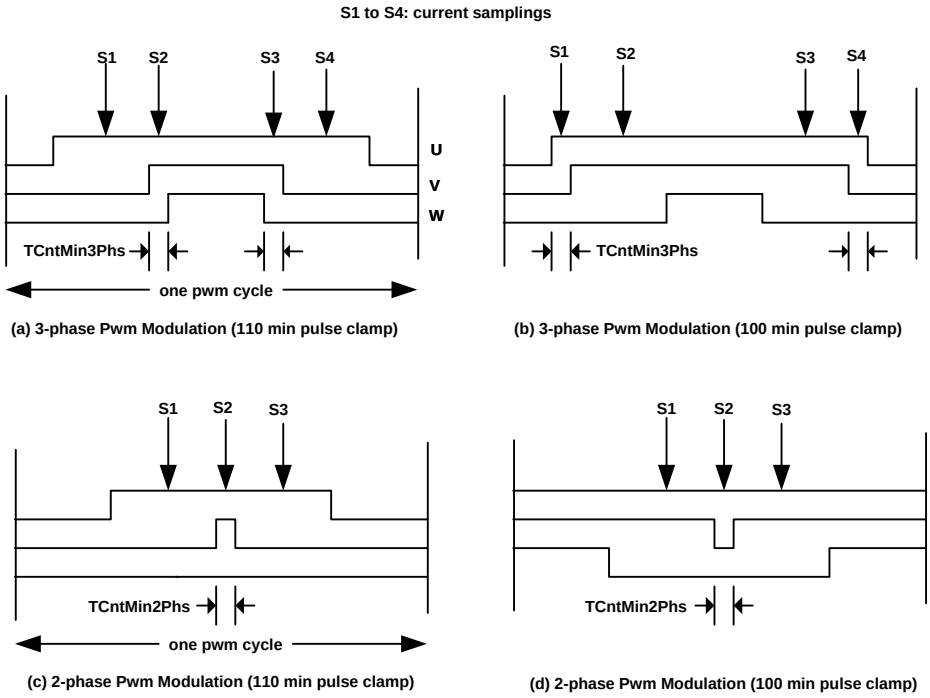


Figure 52. Single Current Shunt Registers (TCntMin2Phs, TCntMin3Phs)

When leg shunt measurement is used, sampling is not done during the active vectors, but in the middle of the zero vector as shown in Figure 53. To guarantee sufficient sampling time a Guard Band is introduced (register PwmGuardBand). The Guard Band ensures that the zero vector is applied for a certain minimum time which should be selected according to the ADC and analog circuit. Note that PwmGuardBand is applied both at the beginning and end of each PWM cycle. That means actual guard band will be twice the time set in PwmGuardBand. When leg shunt measurement is used, minimum pulse time (TCntMin) can be set to zero.

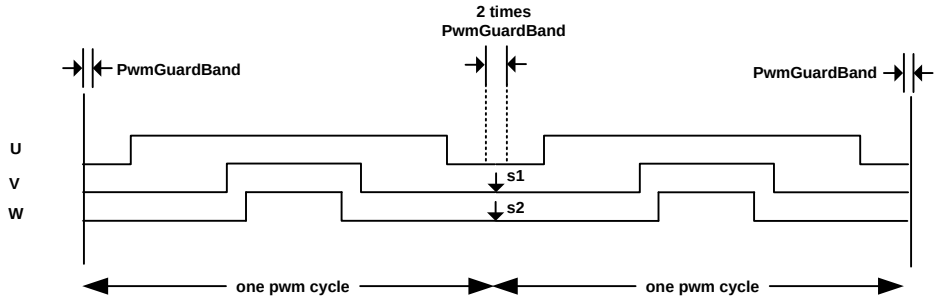


Figure 53. Leg Shunt Register (PwmGuardBand).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.3 DC_BUS_VOLTAGE

The DC_BUS_VOLTAGE module is shown in Figure 54. Its outputs are listed in Table 70. The block has no inputs.

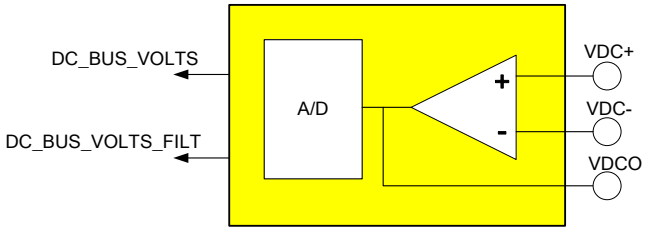


Figure 54. DC_BUS_VOLTAGE Block

Signal name	Function name	I/O	Type
DC_BUS_VOLTS	DC bus voltage	Output	16 bit, signed integer
DC_BUS_VOLTS_FILT	Filtered DC bus voltage	Output	16 bit, signed integer

Table 70. DC_BUS_VOLTAGE Outputs

The outputs of the DC_BUS_VOLTAGE block are valid immediately at the start of each PWM cycle and represent the values calculated during the previous cycle. Output DC_BUS_VOLTS corresponds to the DcBusVolts register and output DC_BUS_VOLTS_FILT corresponds to the DcBusVoltsFilt register (both described in Section 3.4.20).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.4 A_D – A/D Converter and A/D Compensation

In this section, analog channels of IRMCx100 series is explained.IRMCK182 and IRMCF183 have fewer analog channels when compared to IRMCx171; however they both follow the same sampling structure as IRMCx171.Some channels have dedicated functions while others are provided for general purpose use. Access to the general-purpose converted outputs is provided through the A_D library blocks. The block also performs A/D compensation, when enabled, based on reference voltage inputs to certain pins, as described in Section 3.3.4.4. The outputs of the IRMCx171 ,IRMCF188 and IRMCx143 A_D blocks are listed in Table 71. The A_D blocks have no input signals, but there are several configuration parameters required for A/D compensation. Note that AIN0 is used for DC Bus feedback, so that AO0 is the same as DC_BUS_VOLTS (see Section 3.3.3).

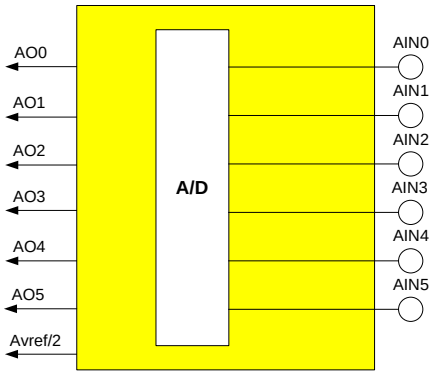


Figure 55. A_D_171 Block

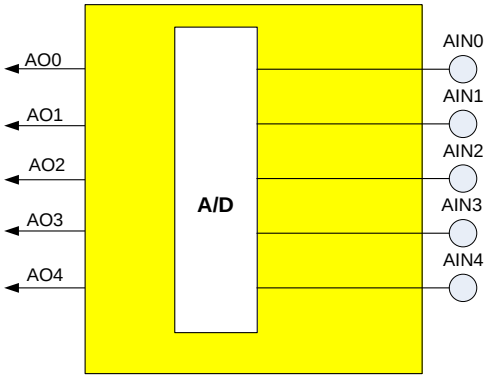


Figure 56. A_D_143 Block

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

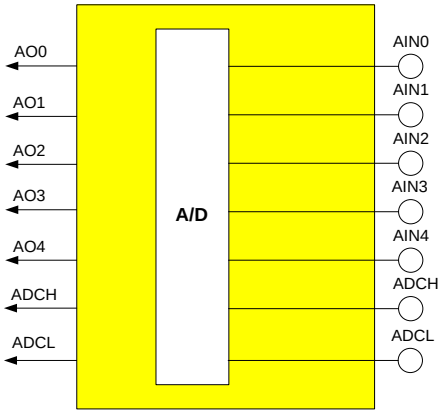


Figure 57. A_D_188 Block

Signal name	Function name	I/O	Type
AO0	A/D converter raw output for input AIN0	Output	12 bit, signed integer
AO1	A/D converter raw output for input AIN1	Output	12 bit, signed integer
AO2	A/D converter raw output for input AIN2	Output	12 bit, signed integer
AO3	A/D converter raw output for input AIN3	Output	12 bit, signed integer
AO4	A/D converter raw output for input AIN4	Output	12 bit, signed integer
AO5[1]	A/D converter raw output for input AIN5	Output	12 bit, signed integer
Avref/2	Filtered A/D converter output for input AIN6 (AVREF/2)	Output	12 bit, signed integer
ADCH	A/D converter raw or compensated output for input ADCH (IRMCF188 only)	Output	12 bit, signed integer
ADCL	A/D converter raw or compensated output for input ADCL (IRMCF188 only)	Output	12 bit, signed integer

[1]Only valid when single shunt current measurement is used for IRMCx171.

Table 71. A_D_100 Outputs

Conversion time for each channel is a maximum of 2.0 microseconds. Unlike a traditional A/D converter in a microcontroller, the conversion process and associated timing of sample/hold and multiplexer are automated by internal firmware logic. This is due to the fact that there is a dedicated analog input channels for current feedback which require specific timing combined with a sample/hold.

All analog circuitry is supplied by AVDD (1.8V) and AGND. The reference for the ADC, AVREF, is 1.2V

IRMCx171 and IRMCx143 input circuit and application connections are shown in Figure 57.

The current sensing channels contain operational amplifiers and dedicated sample/hold circuits. The operational amplifier can be used for adjusting analog input voltage developed across a shunt resister. The operational amplifier is powered by 1.8V (AVDD) and the output common mode voltage of the operational amplifier should be mapped to 0 - 1.2V. The typical application connection is a differential mode amplifier, which can be realized using the external resistors and capacitor as shown in Figure 57.

Two parallel sample/hold circuits are designed to capture two motor phase chopped current signals by synchronizing to the PWM switching pattern. It is designed to hold for a maximum 10 microseconds and is able to charge to half of the common mode voltage (0.6V) within 250 nanoseconds. When single shunt current measurement is used the sample/hold switch is closed and opened at the center point of a new active voltage vector, as shown in Figure 58. In the case of leg shunt measurement, only one of the sample/holds is used

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Besides the current sensing channel, the device provides general purpose analog inputs with a 50mV – 1.2V input range (For application margin, 100mV~1100mV input range is recommended and guaranteed). These are the channels accessed through the A_D block. The A/D data update rate is synchronous to the Motor PWM carrier frequency. Each of the unbuffered A/D channels is updated once every five cycles in all devices. See the description of Scheduling of General Purpose A/D Channels in Section 3.3.4.3 for more information.

The firmware contains an A/D compensation function designed to decrease the feedback error by using two analog channels as reference voltages. The reference channels are used to calculate gain and feedback correction parameters. Section 3.3.4.4 below describes the A/D compensation function in detail.

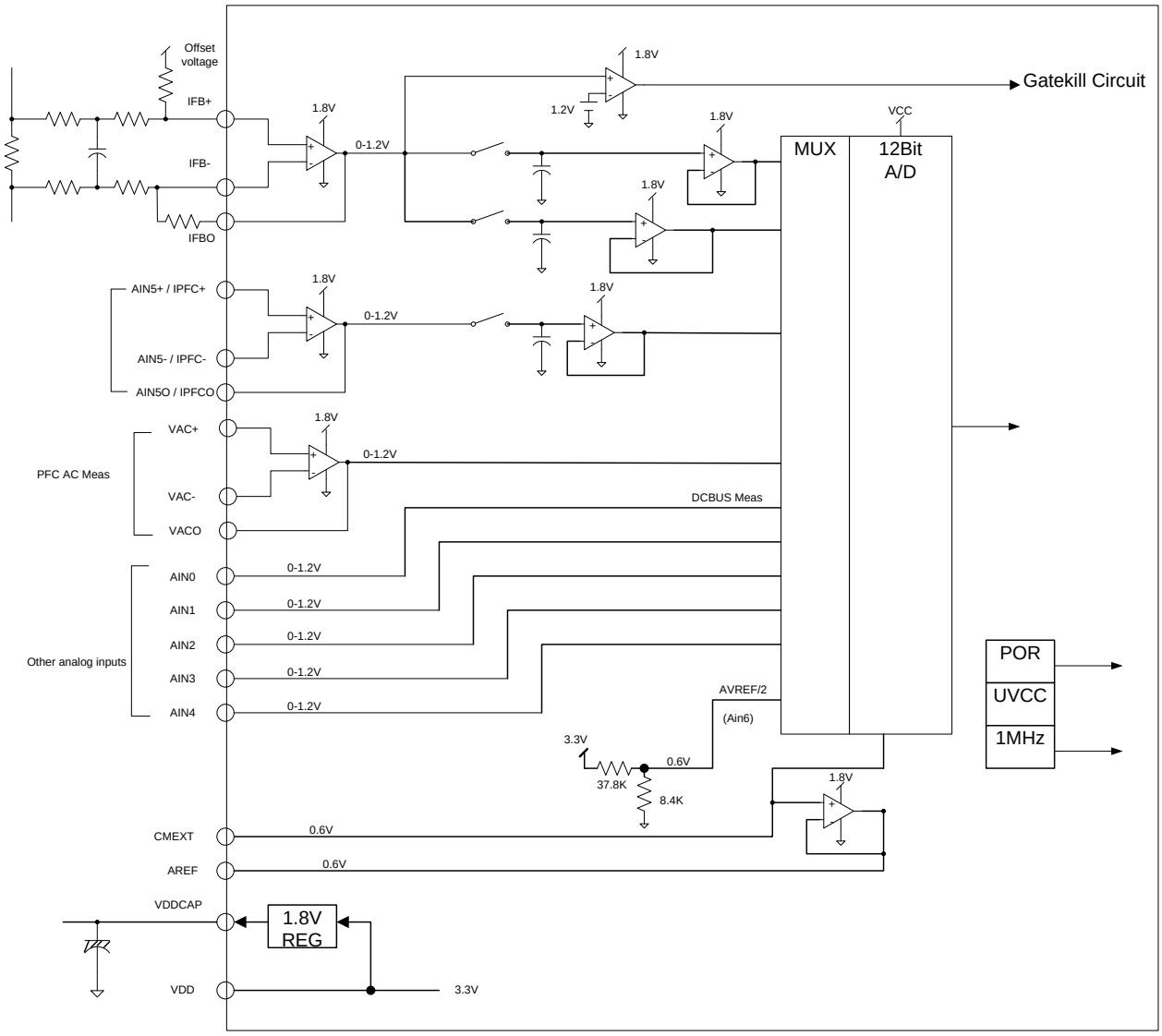


Figure 57. IRMCx100 Series A/D Converter Structure

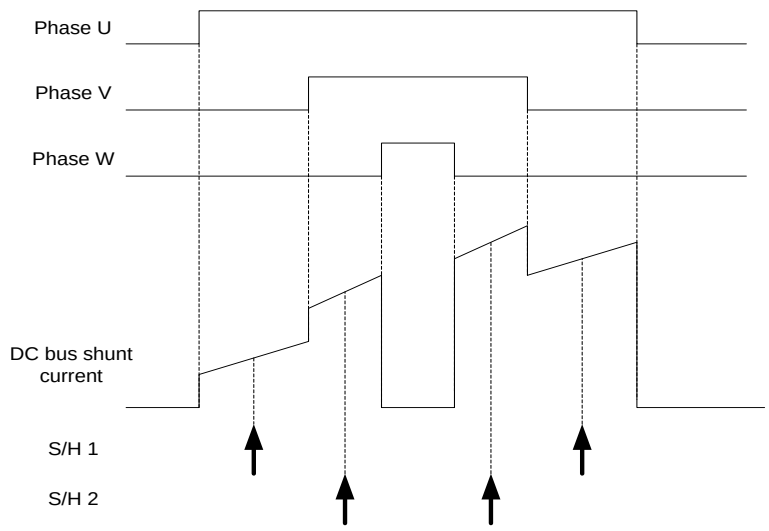


Figure 58. Single Shunt Current Sense Timing

3.3.4.1 AIN5

AIN5 can be used for two purposes in IRMCx171. The first option is as a general purpose A/D channel operated in the same way as AIN2 – AIN4. The second option is as Leg Shunt current measurement channel. AIN5 is not available in IRMCF188 because it is dedicated for a specific purpose (IPFC).

<u>When used as a general purpose channel AIN5, follow these guidelines:</u>
1. Disable AIN5 opamp in the HWCFG register, see section 2.4.5.
2. Select single shunt current measurement (IfbSelect=0, see section 3.4.9)
3. Read converted result in register AIN5.

<u>When used for leg shunt current measurement, follow these guidelines:</u>
1 Enable AIN5 opamp in HWCFG register, see section 2.4.5.
2. Select leg shunt current measurement (IfbSelect=1, see section 3.4.9)
3. Read converted result from register Ifb_V (see section 3.4.21).

If leg shunts are used register AIN5 is not valid. Register Ifb_V will output the V-phase motor current regardless of the current measurement technique.

3.3.4.2 AIN6 (Avref/2)

Ain6 (Avref/2) is an internally driven analog signal designed to provide the offset of the A/D converter. As shown in Figure 57, the 3.3V supply is divided down to 0.6V to provide the AIN6 (AVREF/2) signal. The filtered A/D output is available as register Avref/2. See the description of Avref/2 in Section 1.1.1 for more information.

3.3.4.3 Scheduling of General Purpose A/D Channels

The general purpose A/D channels are scheduled according to Figure 59 in IRMCx100. AIN0 and AIN1 are sampled and made available at every PWM cycle, while only one of AIN2 to AIN6 (AVREF/2) is sampled and converted each cycle. A total of 5 PWM cycles is required to complete sampling of all channels. The indices ‘m’ and ‘n’ refer to time-discrete sample numbers in the figure below, and t_{sw} is the PWM period.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

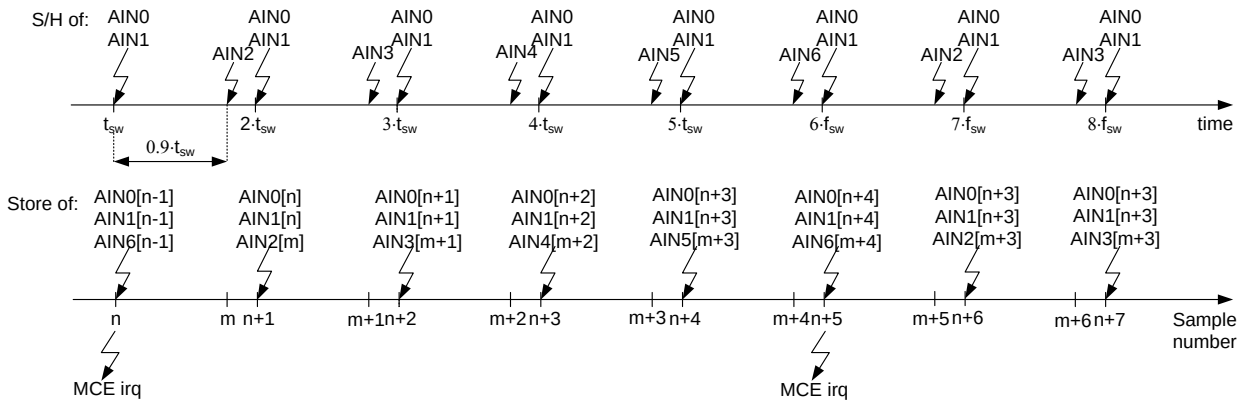


Figure 59. Scheduling of General Purpose A/D Channel

When the sequence AIN2 to AIN6 (AVREF/2) has been sampled and converted, an MCE interrupt is generated to inform the 8051 that new data are ready.

With this setup, the following sampling/conversion frequencies will exist (f_{sw} is the inverter switching frequency):

Channel	Sampling/conversion frequency	Notes
AIN0	f_{sw}	
AIN1	f_{sw}	
AIN2	$f_{sw}/5$	
AIN3	$f_{sw}/5$	
AIN4	$f_{sw}/5$	
AIN5	$f_{sw}/5$	IRMCx171 only
AIN6 (AVREF/2)	$f_{sw}/5$	IRMCx171 only

3.3.4.4 A/D Compensation Function

The firmware contains an automatic A/D compensation function. This function requires the use of two analog channels to provide the reference voltages for the compensation, with input pins for each part and recommended voltages as listed in the table below. Additionally, the MCEWizard calculates values for registers AdRefH and AdRefL which correspond to the ideal A/D converted value of each reference voltage.

Reference Input	Recommended Voltage	IRMCx171, IRMCx143
High Reference	about 1100mV	AIN3
Low Reference	about 100mV	AIN4

After the first MCE configuration (MtrSeqCtrl=1), the firmware will accumulate the raw ADC result of AIN3 & Ain4 for 4096 times (Motor PWM cycles), then take the average value to filter the A/D result.
By using AdRefH & AdRefL (ideal AD results of AIN3 & Ain4 inputs) the firmware can calculate the real AD gain and offset information, which is stored as AdGainx1024 & AdOfst. These values become the K & B, respectively for 2-point AD compensation.

Figure 60 below shows how the AD correction parameters are calculated by the firmware based on the AD converted result and the ideal AD converted value of the reference voltages. The correction factors can be read by the designer in registers AdGainx1024 and AdOfst, described in detail in Section 3.4.25.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

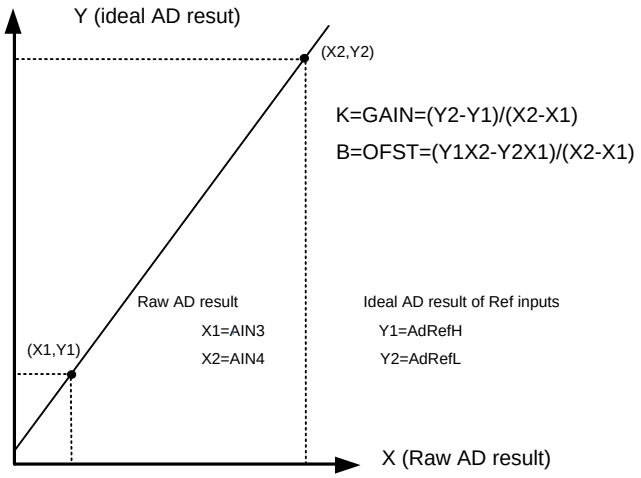


Figure 60. Calculation of A/D correction parameters

The AD compensation is actually is the computation of Y=K*X + B, where Y is the compensated AD result output and X is the raw AD result input.

Please note that AD compensation is only avaibale 4096 PWM cyles after the first MCE configuration (8051 set to MtrSeqCtrl=1 after power on). Before setting MtrSeqCtrl=1, please make sure AdRefH & AdRefL as well as all of other MCE registers are correctly configured. During the 4096 motor PWM cycles, when MCE is calculating the AD offset and gain parameters, it is better for the 8051 not to read any AD results. It is recommended to wait longer than PWM period * 4096 before reading AD channels from 8051.

When this function is enabled, all of the outputs of the the A/D blocks are corrected using the formula below. The A/D compensation function is enabled by default. To disable A/D compensation, set bit 9 of the MtrCtrlBits_S register to 1.

There is also an A/D compensation fault which indicates that the correction values are outside of a reasonable range. If this fault occures, the reference voltages or the values of registers AdRefH and AdRefL are incorrect, or that the IC has an abnormally large AD conversion error. More information about the A/D compensation fault can be found in Section 3.3.6.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.5 Low Loss Space Vector PWM

The Low Loss Space Vector PWM (LOWLOSS_SVPWM) module accepts modulation index commands and generates the appropriate gating waveforms for each PWM cycle. The inputs (Alpha-Beta modulation depth) of the Space Vector modulator are normally connected internally to the output of the SENSORLESS_FOC module. However, users can utilize the LOWLOSS_SVPWM without SENSORLESS_FOC. When register UserVabEn is set to 1, the LOWLOSS_SVPWM module accepts user generated modulation index commands (U_Alpha and U_Beta).

The inputs and outputs of the LOWLOSS_SVPWM module can be customized using the CustomMotPer utility described in Section 5.6. Figure 61 shows the block's inputs in the default configuration (the default configuration has no outputs). The entire list of available inputs and outputs is presented in alphabetical order in Table 72.

The LOWLOSS_SVPWM library block provides an interface to a subset of the IRMCx100-series motion peripheral registers. All registers that control and monitor the operation of the LOWLOSS_SVPWM module are accessible through the LOWLOSS_SVPWM block (with customization). Note that many of these registers values can be calculated using the MCEWizard tool and are typically initialized only once at system startup from an 8051 or host application. It is normally not necessary to update these registers from within the MCE design and they are, therefore, rarely customized as inputs to the LOWLOSS_SVPWM block. These inputs are identified by the term "Config Input" in the "I/O" column of Table 72.

Each signal listed in the table corresponds directly to one of the motion peripheral registers described in Section 3.4 or, where noted, to a bit field within a motion peripheral register. The signal name is the same as the register or bit field name. The rightmost column of Table 72 provides a reference to the document section where the associated register is described.

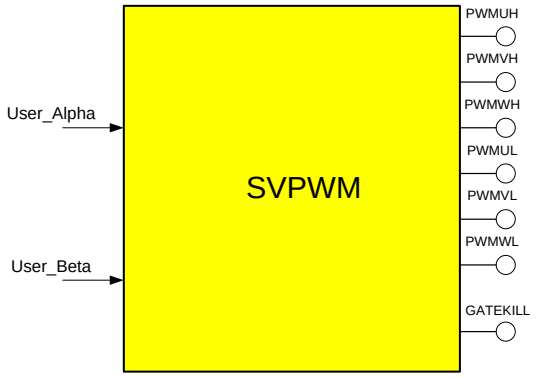


Figure 61. Low Loss SVPWM Block

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Reference for detailed description and scaling
GCChargePD	Number of Gate Pre-charge pulses	Config Input	3.4.2
GCChargePW	Gate Pre-charge pulse width	Config Input	3.4.2
GCChargeT	Gate Pre-charge duration	Config Input	3.4.2
Pwm2HiThr	3-phase to 2-phase PWM high threshold	Config Input	3.4.2
Pwm2LowThr	3-phase to 2-phase PWM low threshold	Config Input	3.4.2
PwmDeadTm	Inverter blanking time	Config Input	3.4.2
PwmGuardBand	Guard Band	Config Input	3.4.2
PwmPeriodConfig	PWM carrier period configuration	Config Input	3.4.2
PWMUH	U phase high side output (pwm_lines, bit 6)	Output	1.1.1
PWMUL	U phase low side output (pwm_lines, bit 7)	Output	1.1.1
PWMVH	V phase high side output (pwm_lines, bit 8)	Output	1.1.1
PWMVL	V phase low side output (pwm_lines, bit 9)	Output	1.1.1
PWMWH	W phase high side output (pwm_lines, bit 10)	Output	1.1.1
PWMWL	W phase low side output (pwm_lines, bit 11)	Output	1.1.1
User_Alpha	User Alpha modulation index	Input	3.4.11
User_Beta	User Beta modulation index	Input	3.4.11
User_U	User U-phase duty ratio control	Input	3.4.11
UserVabEn	SVPWM modulation input selector	Input	3.4.11
UserVuvwn	PWM pattern selector	Input	3.4.11
User_V	User V-phase duty ratio control	Input	3.4.11
User_W	User W-phase duty ratio control	Input	3.4.11
TwoPhsEnb	Enable 2-phase modulation (TwoPhsCtrl, bit 0)	Config Input	3.4.2
TwoPhsType	2-phase modulation type (TwoPhsCtrl, bits 1-2)	Config Input	3.4.2

Table 72. LOWLOSS_SVPWM Available Inputs and Outputs

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

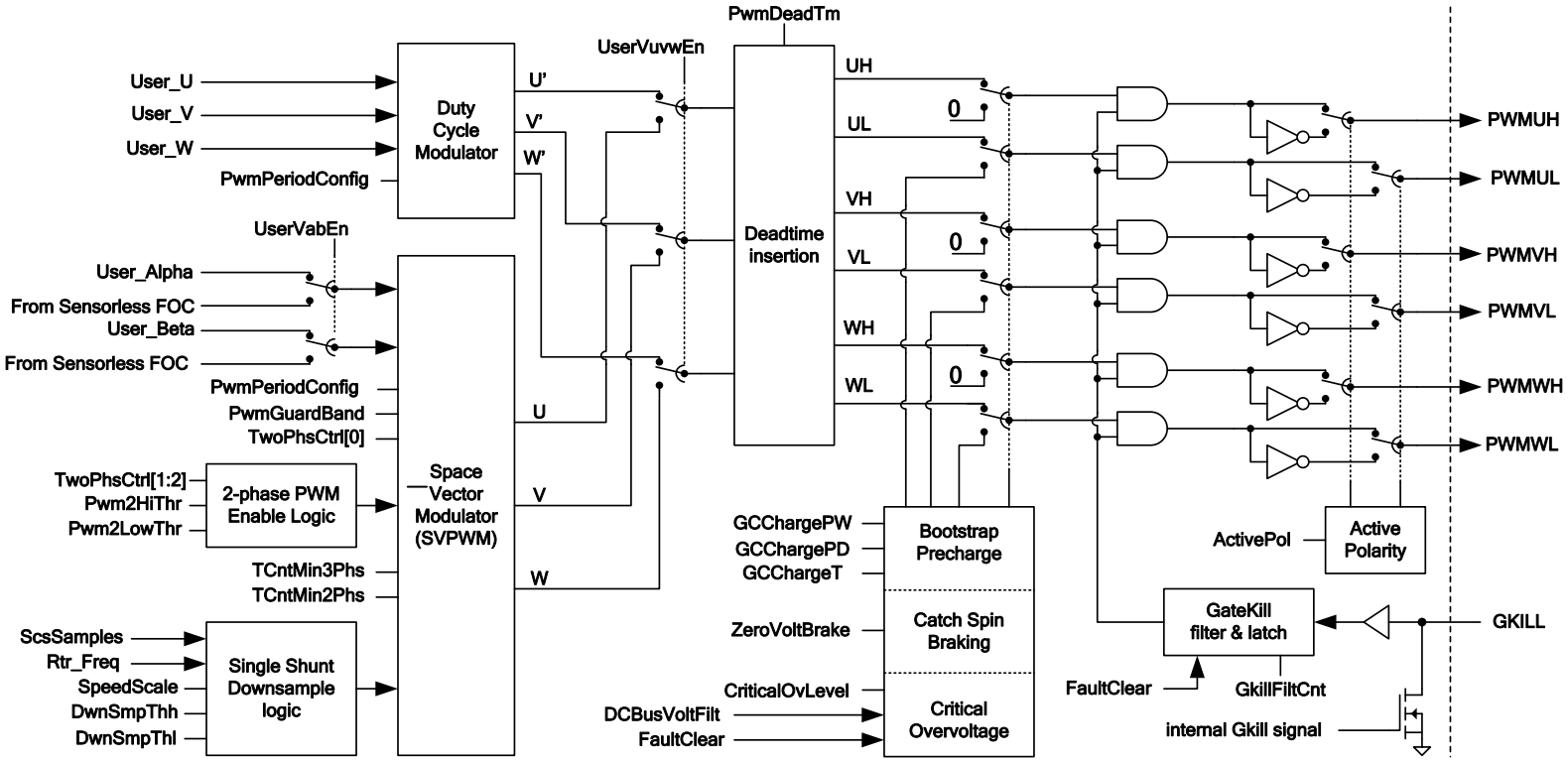


Figure 62. SVPWM Internal Block Diagram

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.5.1 SVPWM Transfer Characteristics

A three-phase two-level inverter with dc link configuration can have eight possible switching states, which generates the output voltage of the inverter. Each inverter switching state generates a voltage Space Vector (V1 to V6 active vectors, V7 and V8 zero voltage vectors) in the Space Vector plane, as shown in Figure 63. The magnitude of each active vector (V1 to V6) is 2/3 Vdc (dc bus voltage).

The user modulation inputs User_Alpha and User_Beta are related to the modulation depth by:

$$Umag = \sqrt{(User_Alpha^2 + User_Beta^2)}$$

The maximum achievable modulation in the linear operating range occurs when modulation (Umag) reaches 2355. Under such circumstance, the voltage vector touches the unit circle (Figure 63). The corresponding inverter line rms voltage is $Vdc/\sqrt{2}$ (Vdc – dc bus voltage). It is best to operate the PWM inverter in the linear range to minimize current harmonics.

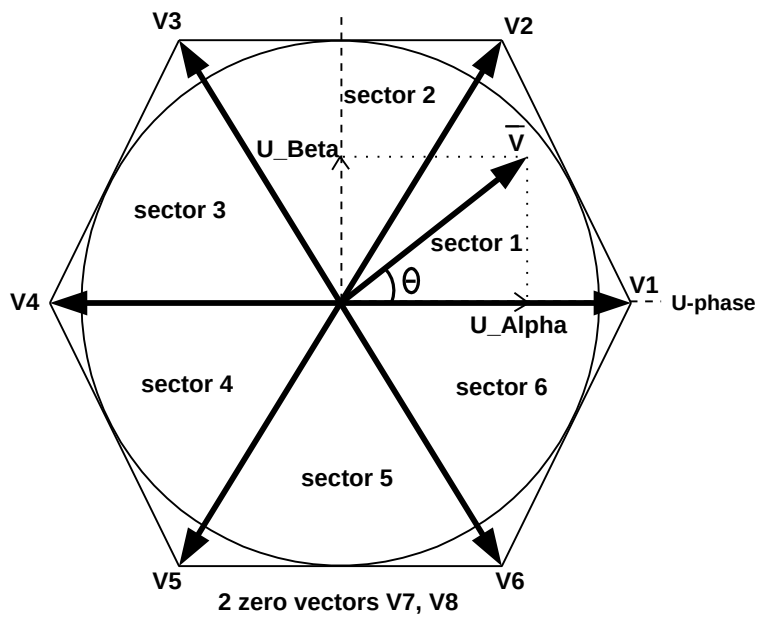


Figure 63. Space Vector Diagram

Over modulation occurs when modulation Umag > Mod_Pk. This corresponds to the condition where the voltage vector in Figure 63 increases beyond the unit circle. Under such circumstance, the Space Vector PWM algorithm will rescale the magnitude of the voltage vector to fit within the Hexagon limit. The magnitude of the voltage vector is restricted within the Hexagon; however, the phase angle (θ) is always preserved. The transfer gain of the PWM modulator is reduced and becomes non-linear in the over modulation region. Voltage vector rescaling is illustrated in Figure 64.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

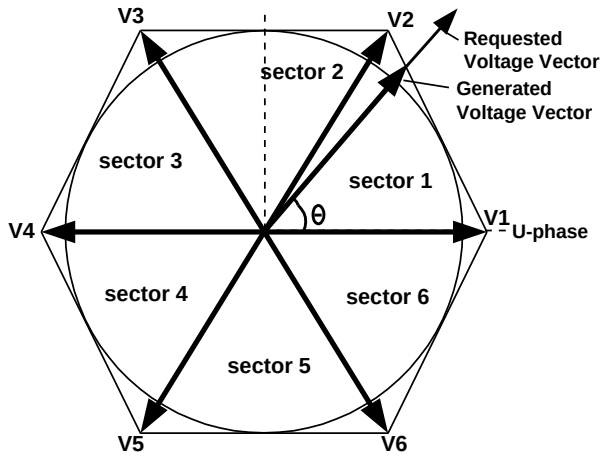


Figure 64. Voltage Vector Rescaling

3.3.5.2 Deadtime Insertion Logic

Blanking time is inserted to avoid shoot through between the top and bottom devices of the same inverter leg. Register PwmDeadTm specifies the amount of inverter blanking time (dead time).

The deadtime insertion logic chops off the high side commanded volt*seconds by the amount of deadtime and adds the same amount of volt*seconds to the low side signal. Thus, it eliminates the complete high side turn on pulse if the commanded volt*seconds is less than the programmed deadtime. It is recommended to set the minimum pulse width (TCntMin3Phs/TCntMin2Phs) greater than the deadtime (PWMDeadTm).

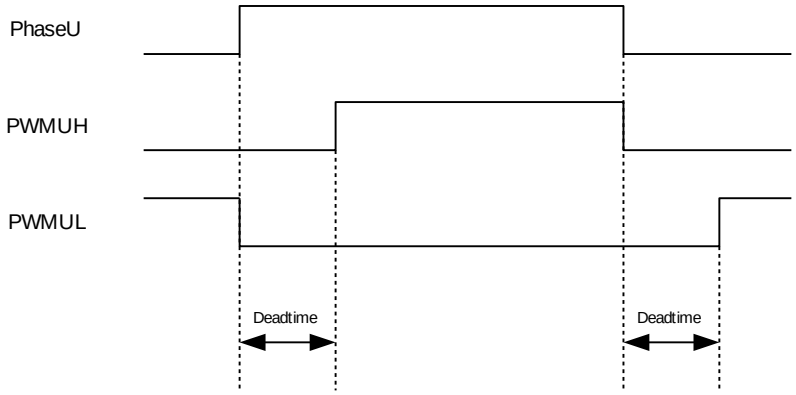


Figure 65. Deadtime Insertion

This function inserts the programmed deadtime between the high and low side gate signals within a phase, as shown in Figure 65. The deadtime register, named **PwmDeadTm** is also double buffered to allow "on the fly" deadtime change and control while PWM logic is inactive.

3.3.5.3 Three-Phase and Two-Phase Modulation

Three-phase and Two-phase Space Vector PWM modulation options are provided.

The Volt-sec generated by the two PWM schemes is identical under the same modulation depth input. However with two-phase modulation the switching losses can be reduced significantly at higher switching frequencies (>10KHz). Figure 66 shows the switching pattern for PWM cycle when the voltage vector is inside sector 1.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

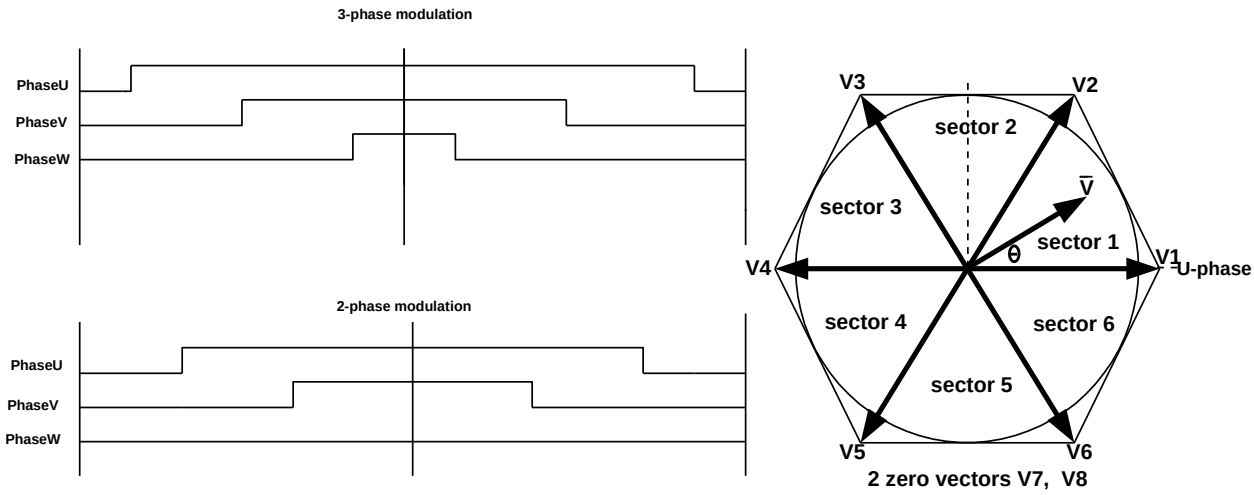
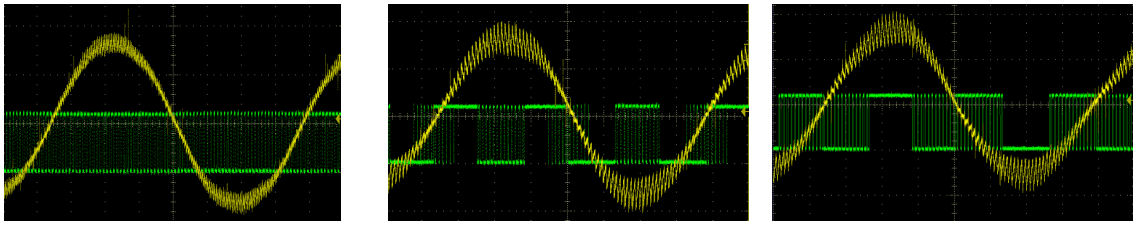


Figure 66. Three-Phase and Two-Phase Modulation

Three types of 2-phase modulation schemes are provided. Register TwoPhsCtrl specifies the selection. The figure below illustrates the inverter Pole voltage and motor current of various types of PWM schemes. Type 1 two-phase PWM is designed to minimize current ripple, while Type 2 is designed to minimize losses. Both Type 1 and Type 2 modulation can only be used with single shunt current sensing. Type 3 two-phase modulation (not shown) only uses the low side zero vector (000) in all sectors so that it can be enabled for either leg shunt or single shunt current sensing situations.



(a) Three-phase PWM (b) Two-phase (type 1) PWM (c) Two-phase (type 2) PWM

Figure 67. Types of Space Vector PWM

When 2-phase modulation is enabled, the SVPWM algorithm permits transitioning from 3-phase to 2-Phase SVPWM when the motor speed exceeds speed threshold Pwm2HiThr. 3-phase SVPWM will resume when motor speed drops below speed threshold Pwm2LowThr.

3.3.5.4 PWM Pre-Charge Control

Zero vector (low side devices on) is normally applied for the initial turn-on of the PWM inverter output. If a Bootstrap gate driver is used, the Bootstrap capacitors (u, v, w phase) will all be charged simultaneously. In some applications, the bootstrap capacitor charging current can be significantly higher than the motor rated current. This will cause a nuisance Itrip as soon as inverter firing comes on. The Bootstrap capacitor charging current can be limited by the built-in Pre-charge control function.

Instead of turning on all low side devices simultaneously for a prolonged duration, the gate Pre-charge control will schedule an alternating (u, v, w phase) charging sequence with programmable (GCChargePW) charging pulse duration. Figure 68 illustrates the pre-charge sequence and the corresponding dc link current. This current represents the

charging current of the bootstrap capacitors (U, V, W phases). As can be seen from this figure, the charging current reduces significantly after two charging cycles (U→V→W).

Please note that during PWM pre-charge, the Motor PWM frequency is changed according to GCChargePW& GCChargePD instead of PwmPeriodConfig .
For IRMCF183 applications, the PFC frequency may change since the PFC frequency is calculated based on the Motor PWM frequency and PFC/Motor PWM Rate, which means GCChargePW& GCChargePD are not allowed to change in MCE Wizard.

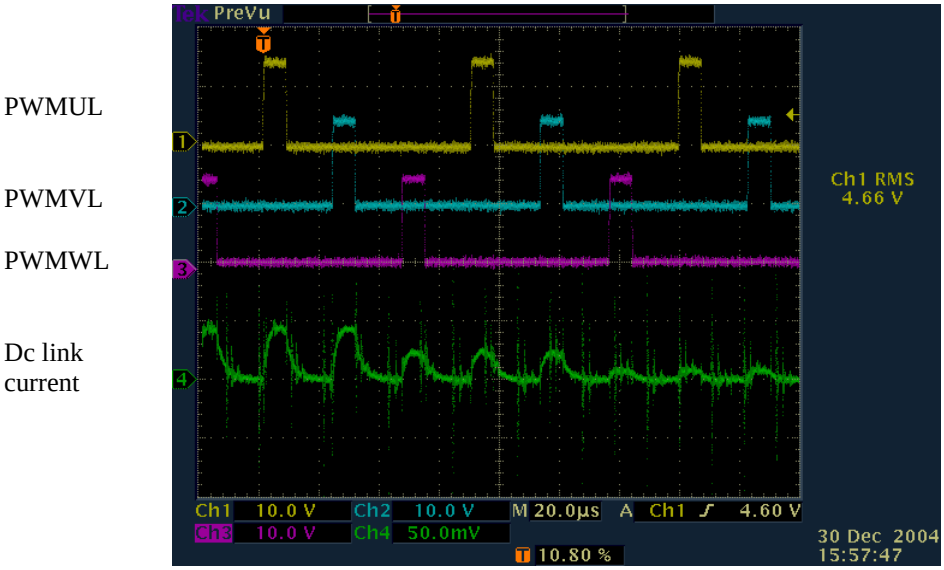


Figure 68. Bootstrap Pre-Charge Sequence

As shown in Figure 69, register GCChargePW controls the charging pulse width while register GCChargePD controls the spacing between u, v and w phase charging. The spacing (GCChargePD) between consecutive charging is specified as number of charge pulses. The total number of pulses for all 3 phases is specified by GCChargeT, i.e. each phase will have GCChargeT/3 charge pulses.

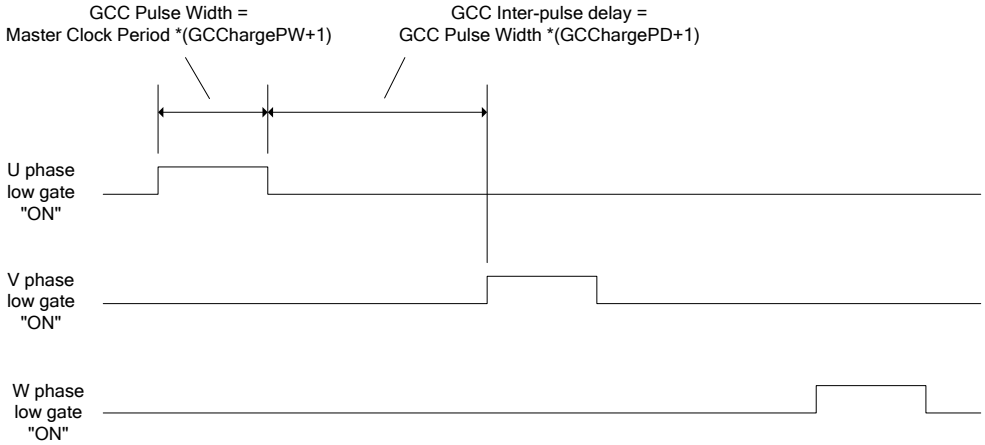


Figure 69. Timing of Bootstrap Capacitor Charging

3.3.5.5 Gatekill

Besides high speed operational amplifiers for current measurement, IRMCx100 provides analog circuits for over current protection, i.e. Gatekill generation. The user has the choice of enabling/disabling the internal circuit and

furthermore, an external Gatekill signal may be applied as well. How the Gatekill circuit interacts with the PWM generation is illustrated in Figure 70. See section 2.4.5 for details on configuring the Gatekill protection logic.

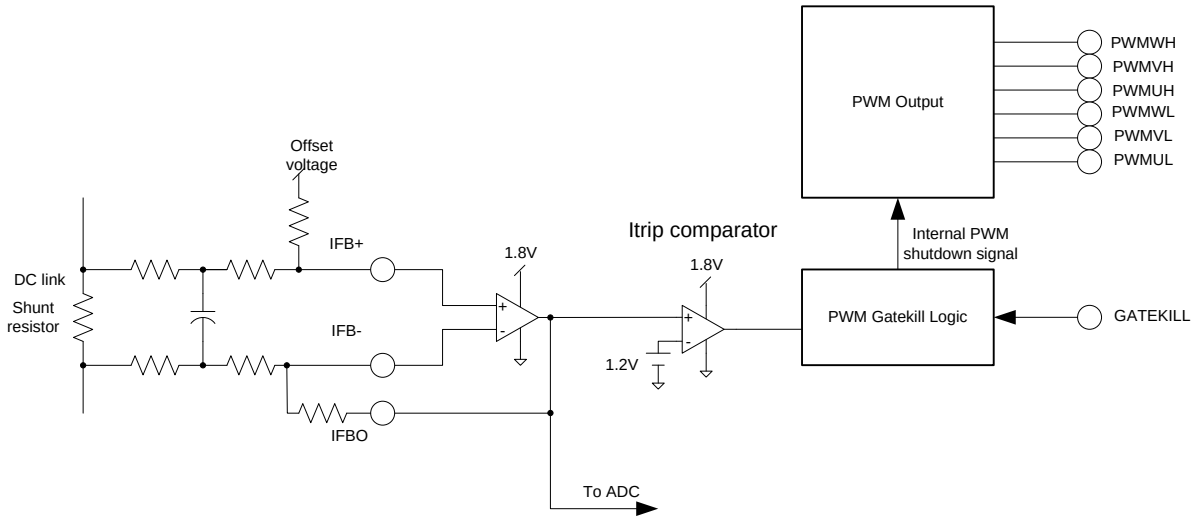


Figure 70. Internal Gatekill Generation

The comparator it self is implemented with hysteresis to provide noise immunity. The upper threshold is 1.22V and the lower is 1.1V. Gain of the current amplifier should be designed to give an appropriate shut down limit.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.6 FAULTS Block

The FAULTS block provides outputs associated with the FaultFlags motion peripheral register defined in Section 1.1.1.

Figure 71 shows the FAULTS block and its outputs are listed in Table 73. All outputs are Boolean values, with 0 indicating no fault condition and 1 indicating that the fault condition is present.

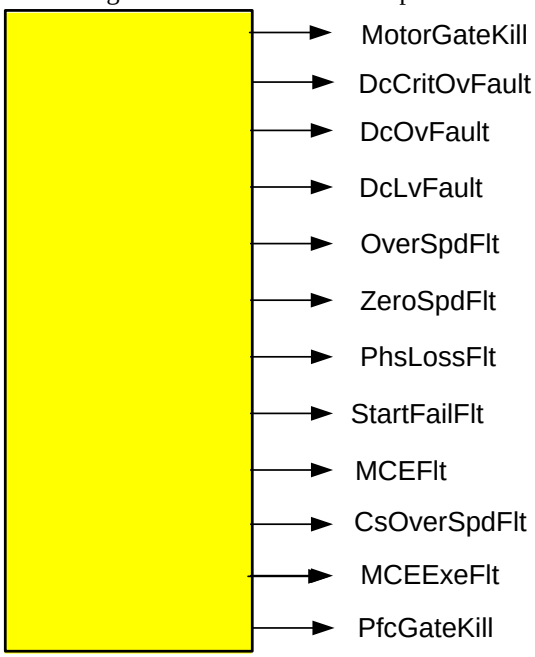


Figure 71. FAULTS Block

Signal name	Function name	I/O	Type
MotorGateKill	Motor Gate kill fault (FaultFlags, bit 0)	Output	Boolean
DcCritOvFault	dc bus critical overvoltage fault (FaultFlags, bit 1)	Output	Boolean
DcOvFault	dc bus over voltage fault (FaultFlags, bit 2)	Output	Boolean
DcLvFault	dc bus under voltage fault (FaultFlags, bit 3)	Output	Boolean
OverSpdFlt	Motor over speed fault (FaultFlags, bit 4)	Output	Boolean
ZeroSpdFlt	Motor zero speed fault (FaultFlags, bit 5)	Output	Boolean
PhsLossFlt	Motor phase loss fault (FaultFlags, bit 6)	Output	Boolean
StartFailFlt	Start fail fault (FaultFlags, bit 7)	Output	Boolean
MCEFlt	MCE-generated fault (FaultFlags, bit 8)	Output	Boolean
CsOverSpdFlt	Catch spin overspeed fault (FaultFlags, bit 9)	Output	Boolean
MCEExeFlt	MCE execution fault (FaultFlags, bit 10)	Output	Boolean
PfcGateKill	Pfc Gate kill fault (FaultFlags, bit 11)	Output	Boolean
ADCompFlt	A/D Compensation Fault	Output	Boolean

Table 73. FAULTS Block Outputs for IRMCx100 Series

The IRMCx100 series monitors motor feedback information and trips a fault condition if it detects a critical operating parameter out of range. Fault conditions can be detected by motion hardware, the Sensorless FOC block in motion firmware or through logic in the MCE Simulink design. Regardless of the source, application software handles all faults in the same way, as described below.

When any fault condition occurs, an interrupt is generated from the MCE to the 8051 (see Section 4.6 for details) and the fault is reported in the FaultFlags register. If the motor is running and the fault is not disabled (through the

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

DisableFaults register, Section 3.4.1), the motor is stopped and the fault latched. Once the motor has been stopped due to a fault condition, it cannot be restarted until the fault is cleared. To clear a fault condition, write "1" to the FaultClear register (Section 3.4.1).

When a disabled fault condition occurs, the fault is reported in FaultFlags and an interrupt is generated to the 8051, but the motor is not stopped. The fault will appear in FaultFlags until the condition which triggered the fault disappears, i.e. the fault does not latch. Fault types GateKill and CritOvFault cannot be disabled.

Table 74 summarizes the faults detected by the IRMCx100 series.

Fault	Bit in FaultFlags	Generated by	Can be disabled	Reference to Detailed Information
MotorGateKill	0	motion hardware	No	For information on gatekill fault generation, refer to the description of the Gatekill Configuration register in Section 2.4.5.
DcCritOvFault	1	motion firmware	No	A dc bus critical over voltage fault is generated if dc bus voltage exceeds the threshold defined by register CriticalOvLevel (Section 3.4.14).
DcOvFault	2	motion firmware	Yes	A dc bus over voltage fault is generated if dc bus voltage exceeds the threshold defined by register DcBusOvLevel (Section 3.4.14).
DcLvFault	3	motion firmware	Yes	A dc bus under trip voltage fault is generated if dc bus voltage falls below the threshold defined by register DcBusLvLevel (Section 3.4.14).
OverSpdFlt	4	motion firmware	Yes	An OverSpdFlt is generated if the motor speed exceeds the motor maximum speed.
ZeroSpdFlt	5	motion firmware	Yes	A zero speed fault is generated if motor speed falls below half of the value of minimum drive speed for a time duration of more than two seconds. Refer to the MinSpd register in Section 3.4.4 for more information
PhsLossFlt	6	motion firmware	Yes	Phase loss detection is configured using the registers of the Phase Loss Detect write register group, Section 3.4.8. See also the description of the IDiff_Fil register in Section 3.4.23.
StartFailFlt	7	motion firmware	Yes	Start failure detection is configured using the registers of the Startup write register group, Section 3.4.7.
MCEFlt	8	MCE Simulink design	Yes	An MCE fault can be generated by writing to the MceFault register (Section 3.4.1) or using the MCE_FAULT motion peripheral block (Section 3.3.7).
CsOverSpdFlt	9	motion firmware	Yes	CsOverSpdFlt is generated during catch spin if the motor speed is too high to proceed with catchspin
MCEExeFlt	10	motion hardware	Yes	An MCE execution overrun fault occurs if the MCE firmware does not complete its PWM-cycle processing before the next PWM pulse.
PfcGateKill	11	PFC hardware	No	PFC Gatekill fault occurs if the PFCGKILL pin is pulled low.
ADCompFlt	12	motion firmware	Yes	AD Compensation fault occurs if the (absolute value) of the offset is > 100mV or the AD gain is >±15% different than the ideal value.

Table 74. Summary of Fault Handling

What protection functions to run is highly dependent on the system state. For example, it is not relevant to check for Over Speed fault during parking.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Below showns how each protection fuction is related to system state.

Fault	Signal	Active					
		Stopped	Gate charge	Parking	Starting	Closed loop	Catch spin
Motor Gatekill	HW signal	Yes	Yes	Yes	Yes	Yes	Yes
Critical OV	DcBusVoltsFilt	Yes	Yes	Yes	Yes	Yes	Yes
Over voltage	DcBusVoltsFilt	Yes	Yes	Yes	Yes	Yes	Yes
Under voltage	DcBusVoltsFilt	Yes	Yes	Yes	Yes	Yes	Yes
MCE fault	MceFault	Yes	Yes	Yes	Yes	Yes	Yes
Exe fault	-------	Yes	Yes	Yes	Yes	Yes	Yes
Over speed	Rtr_Freq	No	No	No	No	Yes	No
Zero speed	Rtr_Freq	No	No	No	No	Yes	No
Phase loss	Iw	No	No	Yes	No	No	No
Start fail	Flx_M	No	No	No	No	Yes	No
Catch spin OS	Flx_M	No	No	No	No	No	Yes
PFC Gatekill	HW signal	Yes	Yes	Yes	Yes	Yes	Yes
ADCompFlt	AdOfst, AdGainx1024	***Only active at initialization stage. Not checked after first MCE Configuration is done.***					

Table 75. Relationship between System State and Active Fault Functions

3.3.6.1 Fault State Control

The IRMCx100 offers two ways of handling faults. The first one will automatically shut down the motor (and PFC) if one of the predefined limits is exceeded and at the same time notify the 8051 of the condition by generating a fault interrupt. The fault type can be read in the register FaultFlags or in the Simulink design. The second way of handling faults involves the register DisableFaults. Here one or more faults may be disabled. In case one of the predefined limits is exceeded in this mode the motor will continue running but an interrupt is generated and the type of fault condition is reported in FaultFlags and in the Fault block. It is up to the 8051 to decide what action should be taken – if any. Note that Motor Gatekill, PFC Gatekill and Critical Overvoltage cannot be disabled. The state handling of fault algorithm is shown in Figure 72.

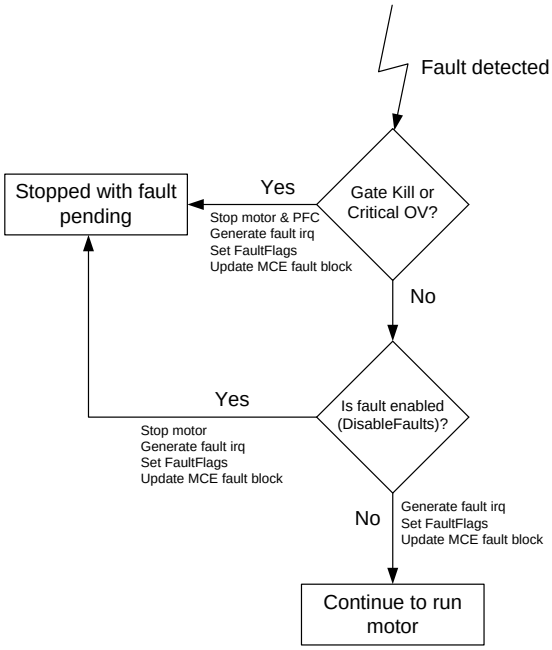


Figure 72. State Control of Fault Handling

It is possible for multiple faults to be present at the same time and they are all handled with the same priority. Before faults can be cleared and the motor restarted the fault condition that caused the fault must be gone (monitoring continues after stopping the motor).

3.3.7 MCE_FAULT Generator

The MCE_FAULT block is used to generate a fault condition from within the MCE design. Figure 73 shows the block. It has a single input, as described in Table 76. The Boolean input corresponds directly to the MceFault register (Section 3.4.1).

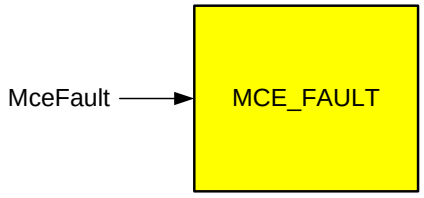


Figure 73. MCE_FAULT Block

Signal name	Description	I/O	Type
MceFault	MCE Fault condition	Input	Boolean 0 = No fault condition; 1 = Generate fault

Table 76. MCE_FAULT Block Inputs

3.3.8 PFC_PWM

This block generates the PFC PWM control pulses to the gate driver based on the duty cycle command produced in the PFC control loop, which is implemented in the MCE. Meanwhile, with a built-in PWM Blanking function, it provides protection against a nuisance over-current fault situation when the AC input voltage becomes higher than the DC bus voltage under normal full-mode PFC. The PFC_PWM block also provides the flexibility of designing the circuit to operate as either full-mode boost PFC or partial-mode high-frequency boost PFC (IR patented). This block does not apply for servo control IC, IRMCx143.

The PFC_PWM library block provides an interface to a subset of the IRMCF188 motion peripheral registers that control and monitor the operation of the PFC_PWM module. Figure 74 shows the PFC_PWM block and Table 77 lists its inputs and outputs.

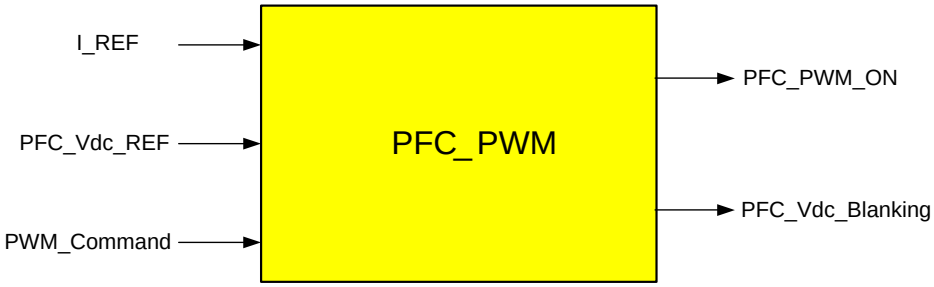


Figure 74. PFC_PWM Block

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Signal name	Description	I/O	Type
PWM_Command	PFC PWM duty cycle command, used for PFC PWM generation and IPFC_AD timer setup	Input	16 bit, unsigned integer
I_REF	Reference command of PFC current loop, used for PFC current blanking	Input	12 bit, unsigned integer
PFC_Vdc_REF	PFC Vdc Ref from Voltage loop, used for PFC DCBUS Blanking	Input	16 bit, unsigned integer
PFC_PWM_ON	PFC PWM active status	Output	Boolean, 0 = PFC PWM output disabled 1 = PFC PWM output enabled
PFC_Vdc_Blanking	PFC DCBUS voltage blanking status	Output	Boolean, 0 = Vac lower than DC bus voltage, PFC PWM can be normal output. 1 = Vac higher than DC bus voltage, PFC PWM need blanking.

Table 77. PFC_PWM Inputs and Outputs

The internal block diagram in Figure 74 shows the two main components of the PFC PWM block: PWM Generation and PWM Blanking. This diagram shows all the motion peripheral registers that are associated with the operation of the PFC PWM block. Those that are accessible through the PFC_PWM library block are shown in **bold** type. Those that are written from a host or 8051 application are shown in *italic* type.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

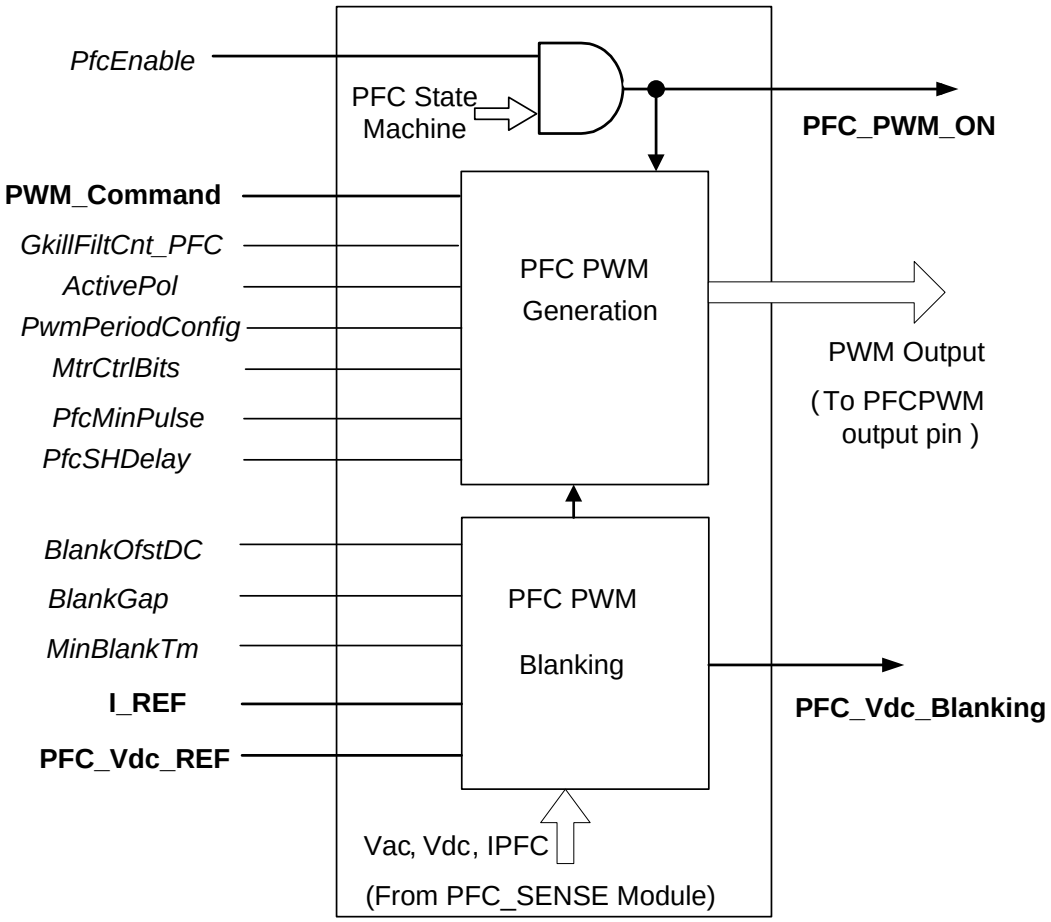


Figure 75. PFC_PWM Internal Block Diagram

3.3.8.1 PFC PWM Generation

The generation of the PWM output is based on a comparison between the duty cycle command PWM_Command and an up counter PWM carrier. The PWM carrier frequency is set up by the Motor register PwmPeriodConfig and the Motor/PFC PWM ratio, which is defined in bits 6 – 8 of MtrCtrlBits. The PFC PWM can be setup at the first MCE configure after power on and updated only at the end of the PWM Pre-charge step. This means that the PFC Frequency cannot be updated at any time.

A Sync pulse is generated based on the set up of the PFC PWM carrier frequency. At every Sync pulse, the latest PWM_Command that is produced in the control loop is latched into the PWM Generation block. Prior to the arrival of the next Sync pulse, this latched PWM_Command signal is compared with the up counter PWM carrier. When the latched PWM_Command is higher than the carrier, the PWM Output is high; when the latched PWM_Command is lower than the carrier, the PWM Output is low.

The PFC current sample instant is set to the middle of the PWM pulse by adding the hardware propagation delay (please refer to Section 1.1.1, register PfcSHDelay for details).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

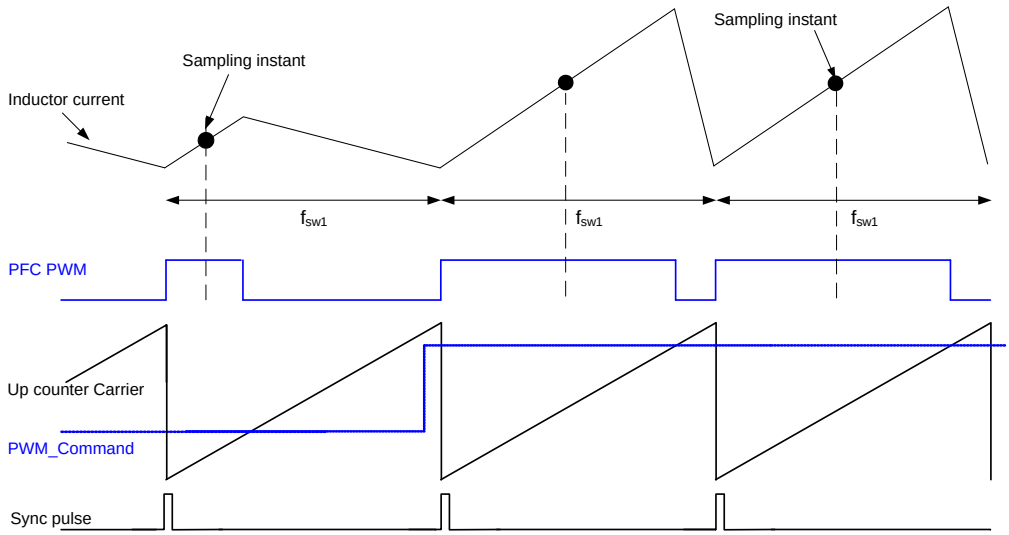


Figure 76. Generation of PFC PWM Output and ADC Timing

When a PFC GateKill occurs, the PWM Output is disabled. Meanwhile, the PWM Output is enabled/disabled by the combination of PFCEnable input and the Vdc_blanking signal, which is the output the PWM_Blanking block: when PFCEnable = 1 and Vdc_blanking = 0, enable PWM; when PFCEnable = 0 or Vdc_blanking = 1, disable PWM.

3.3.8.2 PFC PWM Blanking

The full-mode boost PFC operation requires that the DC bus voltage must be higher than the peak of the AC input voltage. However, in partial PFC mode, during input voltage transients or a sudden large increase in load, the peak of the AC input voltage can be higher than the DC bus voltage. If the PWM switching continues, the boost inductor could go to saturation because its volt-seconds cannot be balanced. Consequently, high current can be generated and cause a nuisance over-current fault. The PWM_Blanking block provides protection by generating an output PFC_PWM_ON signal.

There are two types of PFC PWM blanking inside the PFC PWM blanking module.

Voltage Blanking

This block instantly compares the DC bus voltage Vdc and AC input voltage Vac_in every motor PWM cycle. Register BlankOfstDc provides offset and adjustment for the comparison and MinBlankTm provides the minimal blanking time. If Vac_in > Vdc – BlankOfstDc, the PFC_Vdc_Blanking flag is set and then the PFC PWM is disabled for a minimum time defined by MinBlankTm, which is typically 0.5 msec.

Current Blanking

Besides DCBUS voltage blanking, there is PFC current blanking, checked at every PFC PWM cycle before setup of the PFC current A/D sample time. PFC current blanking compares the instantaneous PFC current and the PFC current reference and disables PFC PWM for the current PWM cycle if the PFC current is higher than PFC current reference with a gap (BlankGap). During PFC current blanking PWM cycle, the PFC current sampling instant will be set to the center of the PFC PWM cycle.

The PFC voltage and current Blanking feature provides fast and smooth transitions between the PWM enable and disable modes, which provides the flexibility of designing the circuit to operate as either full-mode boost PFC or partial-mode high-frequency boost PFC.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.9 PFC_SENSE

The PFC_SENSE module provides the ADC feedback signals that are used for the PFC control loop. Figure 77 shows the PFC_SENSE block and Table 78 lists its outputs.

The PFC_SENSE library block provides an interface to a subset of the IRMCx100 motion peripheral registers. All registers associated with the operation of the PFC_SENSE module are accessible through the PFC_SENSE block. Each signal listed in the table corresponds directly to one of the motion peripheral registers described in Section 3.4.24. This block does not apply for servo control IC, IRMCx143.

The register name associated with the signal is shown in the "Associated Register" column of the table. The rightmost column of the table provides a reference to the document section where the register is described. Each of the output signals can be traced through MCEDesigner.

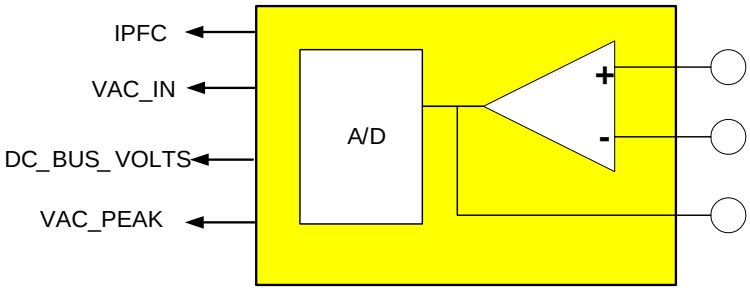


Figure 77. PFC_SENSE Block

Signal name	Description	I/O	Associated register	Reference for detailed description and scaling
IPFC	A/D feedback signal of AC input current, processed data	Output	IPFC	1.1.1
VAC_IN	A/D feedback signal of AC input voltage, absolute value	Output	VAC	1.1.1
DC_BUS_VOLTS	dc bus voltage feedback	Output	DcBusVoltsFilt	1.1.1
VAC_PEAK	Peak AC input voltage during last 100 msec	Output	VacPeak	1.1.1

Table 78. PFC_SENSE Outputs

The PFC_SENSE block provides the ADC feedback signals that are used for the PFC control loop, in 12-bit digital counts: DC bus voltage, AC input current (IPFC), AC input voltage (VAC) and AC peak voltage for the last 100 msec (VacPeak). For PFC control designs, VAC_IN generates the half-wave sinusoidal reference (rectified AC voltage) for the current control loop. The AC input current (IPFC) is provided as corrected current feedback (register IPFC).

PFC sensing is synchronized with PFC PWM pulse generation; the ADC sampling point of IPFC is the middle of PFC PWM ON period and the ADC sampling point of VAC occurs at the beginning of the PFC PWM cycle. With this built-in feature, the average value of the PFC inductor current in every switching cycle is sampled and converted to a digital signal for the control loop execution, and the switching noise has minimum influence on the ADC process. For detailed ADC timing, refer to Figure 76.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.3.10 PFC_AC_STATUS

The PFC_ AC_STATUS module provides AC input voltage status that is used for PFC state machine to disable/enable the PFC according to the AC input voltage, if Bit8 in MtrCtrlBits_S is set to 0. Figure 78 shows the PFC_AC_STATUS block and Table 79 lists its outputs. This block does not apply for servo control IC, IRMCx143.

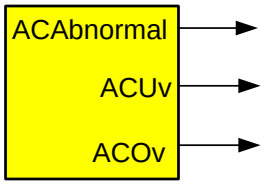


Figure 78. PFC_ AC_STATUS Block

Signal name	Description	I/O	Associated register	Reference for detailed description and scaling
ACAbnormal	AC input voltage abnormal flag. This bit is set to 1 if PFC AC input is abnormal, and set to 0 after PFC AC input is normal for at least AcOKTimeThr.	Output	AcStatus Bit 0	1.1.1
ACUv	AC input under voltage flag. This bit is set to 1 if the AC voltage is lower than AcUvThr for at least AcUvTimeThr.	Output	AcStatus Bit 1	1.1.1
ACOv	AC input Over voltage flag. This bit is set to 1 if the AC voltage is higher than AcOvThr for at least AcOvTimeThr.	Output	AcStatus Bit 2	1.1.1

Table 79 . PFC_AC_STATUS Outputs

AC Over voltage and under voltage detection timing and signal generation are shown in Figure 79.

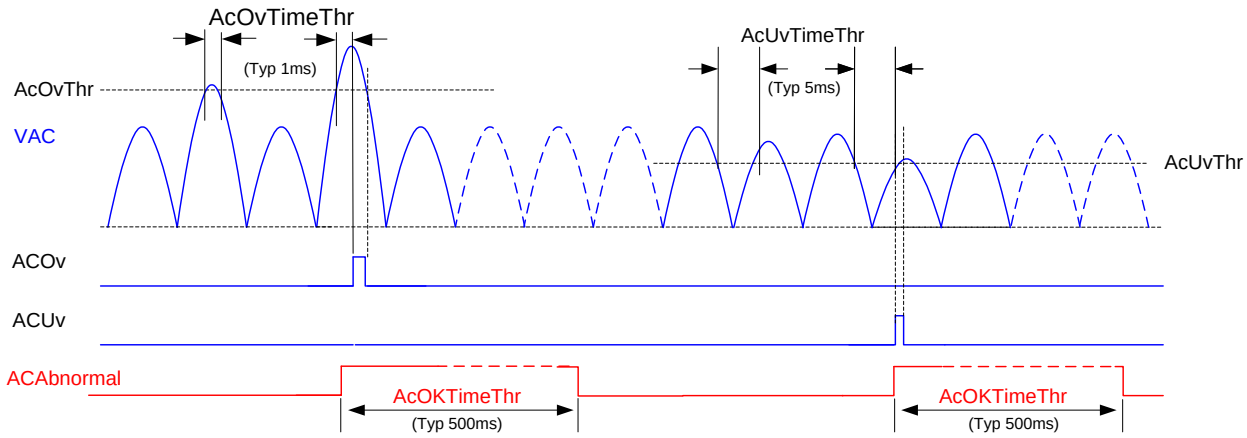


Figure 79. AC Over/Under Voltage Detection

If MtrCtrlBits_S bit 8 = 0 then PFC is disabled while the ACAbnormal bit = 1, regardless of the setting of PFC control registers.

The PFC_ AC_STATUS library block provides an interface to the bits of register AcStatus, described in Section 1.1.1.

3.3.11 PFC_MOTOR_MISC

The PFC_ MOTOR_MISC module provides RMS measurement results in engineering units for PFC power, AC voltage and current, motor current and DC voltage, so that the application code can use it for system control. Figure 80 shows the PFC_ MOTOR_MISC block and Table 80 lists its outputs. The calculation of these values can be disabled with MtrCtrlBits_S bit 7. This block does not apply for servo control IC, IRMCx143.

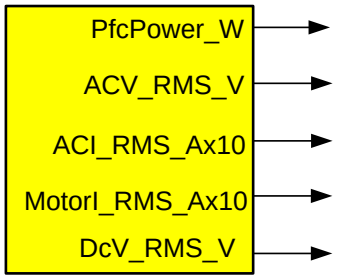


Figure 80. PFC_MOTOR_MISC

Signal name	Description	I/O	Associated register
PfcPower_W	PFC input power, 1count =1W PfcPower_W = ACI_RMS_Ax10 * ACV_RMS_V/10	Output	PfcPower_W
ACV_RMS_V	Ac RMS voltage, 1count =1V $\text{ACV_RMS_ V} = \frac{100 * \sqrt{\frac{\sum_{0}^{N} Vac^2}{N}}}{PfcVFBKScalx100}$	Output	AcV_RMS_V
ACI_RMS_Ax10	Ac RMS Current, 1count =0.1A $\text{ACI_RMS_Ax10} = \frac{100 * \sqrt{\frac{\sum_{0}^{n} IPFC^2}{N}}}{PfcIFBKScalx10}$	Output	AcI_RMS_Ax10
MotorI_RMS_Ax10	Motor RMS Current, 1count =0.1A $\text{MotorI_RMS_Ax10} = \frac{\lvert I_U \rvert + \lvert I_V \rvert + \lvert I_W \rvert}{2.7 * MtrIFBKScalx10}$	Output	MotorI_RMS_Ax10
DcV_RMS_V	DC RMS voltage, 1count =1V $\text{DcV_RMS_V} = \frac{DcBusVoltsFilt * 100}{VdcFBKScalx100}$	Output	DcV_RMS_V

Table 80. PFC_MOTOR_MISC Outputs

3.4 Motion Peripheral Registers

This section provides detailed descriptions of the motion peripheral registers.

The motion peripheral registers are divided into two types:

- Motion hardware registers (MHRs) are defined in hardware (RTL) and are memory mapped for access from both the MCE and the 8051. On the MCE memory bus, these registers are mapped to the address range 0xE80 – 0xFFF. On the 8051 memory bus, the registers are mapped to the range 0xF100 – 0xF7FF.
- Motion firmware registers (MFRs) are defined in the MCE firmware and reside in shared data RAM. On the MCE memory bus, these registers are in the address range 0x000 – 0x7FF. On the 8051 memory bus, they appear in the address range 0x8000 – 0x8FFF.

The 8051 address of motion hardware register can be obtained from the MCE address using the following equation:

8051 address = (MCE address – 0xE80) * 2 + 0xF100

The 8051 address of a motion firmware register can be obtained from the MCE address using the following equation:

8051 address = MCE address * 2 + 0x8000

In the MCEDesigner tool, offsets for MHRs are defined as the 8051 address divided by two (equivalent to the MCE address + 0x6A00). Offsets for MFRs are given as the MCE address.

Motion peripheral registers (both MHRs and MFRs) range in size from 16 bits to 32 bits (although in some cases fewer than 16 bits are used). On the 8051 memory bus, the registers appear in little endian byte ordering (see Section 1.3), low-order byte first. For 16- and 32-bit registers, the 8051 address given is for the low-order (first) byte. On the 16-bit MCE bus, registers larger than 16 bits appear in little endian word order, low-order word first and the MCE address given is for the low-order (first) word. The MCEDesigner tool defines all registers as 16 bits, so in that context 32-bit registers are divided into two parts: a low word and a high word.

Motion firmware registers reside in RAM and can be read and written in the same manner as ordinary RAM variables. When accessing MFRs from the 8051 processor, the coherent read/write mechanism described in Section 4.4.1 should always be used. The MCE initializes data RAM to zero as one of the first tasks after the MCE starts running. Reset values shown for MFRs in the following descriptions assume that MCE data RAM has been initialized to zero.

Motion hardware registers are initialized by hardware at power up and reset. These registers do not reside in RAM but they are memory mapped so they can be directly read and written similarly to motion firmware registers, with the following restrictions:

- Write registers can be read and written, but reserved or undefined bits may read back as zero regardless of the value written.
- Read registers are read only, and values written to those addresses are ignored.

Certain motion firmware and motion hardware registers require initialization to a specific value, where noted in the detailed descriptions below. In these cases, the registers should be initialized before beginning execution of the MCE processor.

From an MCE design, registers are read and written through connections to Motion Peripheral blocks. (Block inputs correspond to write registers and block outputs correspond to read registers.)

The sections below categorize the registers into functional groups and describe the purpose and format of each register. Motion hardware registers are shown with yellow shading and motion firmware registers are shown with brown shading.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.1 System Write Register Group

MtrCtrlBits

Address:	*0x82E8 (8051)* *0x174 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *with bit field definitions*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: These parameters (MtrCtrlBits & MtrCtrlBits_S) specify the configuration of various motor and PFC control functions. These bit-packed parameters serve as software jumpers for turning on and off certain motor and PFC control functions. Certain bits in this parameter provide internal scaling adjustment such that data scaling can be managed within the specified range (16-bit signed). For instance: Bit 2 and 3 of MtrCtrl Bits_S (flux attenuation jumpers for PLL) are used to scale the inputs (Flx_Alpha and Flx_Beta of Figure 82) of the PLL such that the PLL gains (PllKp and PllKi of Figure 82) can always stay within range for different operating conditions (frequency range). These two bits are configured by MCEWizard.

The AIN1 Function Select field in MtrCtrlBits applies to IRMCx143 and IRMCx188 only and defines the use of the AIN1 input. When this bit is set to 0, AIN1 is used as a dedicated VAC feedback input so that motor leg shunt current sampling is possible in the PFC application. An external OP amp is required for this configuration. When the bit is set to 1, AIN1 is used as a general-purpose A/D input. For IRMCx171, this bit is not used and should be set to zero.

MtrCtrlBits[6:8], available for IRMCF188 only, define the PWM frequency ratio between Motor and PFC. Available PWM rates are 1:2, 1:3, 1:4 and 1:5. Please set to 0 for IRMCx171.

Bits 0 – 3 **DiagSelect**
- 0000 No diagnostic enabled
- 0001 Enable parking diagnostic
- 0010 Enable start-up diagnostic
- 0101 Enable current regulator diagnostic
- 1001 Enable Volts/Hz diagnostic

Bit 4 **AIN1 Function Select**
- 0 AIN1 is used as a dedicated VAC feedback input
- 1 AIN1 is used as a general-purpose A/D input

Bit 5 **Use2xFrqScl**
- 0 Do not use 2x frequency scale
- 1 Use 2x frequency scale

Bit 6 – 8 Motor/PFC PWM rate select [1]

Bit 8	Bit 7	Bit 6	Motor : PFC Ratio
0	0	0	1:2
x	x	1	1:3
x	1	0	1:4
1	0	0	1:5

(x means don't care)

Bits 9 – 15 Unused; set to zero

[1]Note: PFC PWM can be setup at first MCE configurate after power on and updated at the end of PWM Pre-charge step, which means it cannot be updated at any time. In other words, the PFC PWM rate can be updated in Motor stop state, but the real PFC frequency change only happens at motor start (updated at end of PWM Pre-charge step).

MtrCtrlBits_S

Address:	*0x82EA (8051)* *0x175 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *with bit field definitions*	*Reset value:*	*0*

Scaling or Notation: See description.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Description: See description in MtrCtrlBits. Bits 7& 8 apply to IRMCF188 only. Please set to 0 for other IRMCx100 Ics.

Bit 0 **CatchEnb**
- 0 Disable catch spin function
- 1 Enable catch spin function

Bit 1 **Use4xFrqScl (see** Figure 82**)**
- 0 Do not use 4x frequency scale
- 1 Use 4x frequency scale

Bit 2 **Use2xMagScl (see** Figure 82**) (Note:** this bit only applies when bit 3 = 0)
- 0 Attenuate 16x flux amplitude.
- 1 Attenuate 8x flux amplitude.

Bit 3 **Use4xMagScl (see** Figure 82**)**
- 0 Attenuate 8x or 16x (depends on Bit 2) flux amplitude for PLL inputs
- 1 Attenuate 4x flux amplitude for PLL inputs

Bit 4 **IregCompEnb**
- 0 Disable dc bus compensation for current regulators
- 1 Enable dc bus compensation for current regulators

Bit 5 **UseExtFlux**
- 0 Use internal fluxes for PLL
- 1 Use external fluxes (Ext_Flx_Alpha, Ext_Flx_Beta, Section 3.4.14) for PLL

Bit 6 **SkipOffset** [1]
- 0 Preform current feedback offset calibration before start of the motor. Offset calibration is done by calculating the average of 4096 samples of the current feedback.
- 1 Skip current feedback offset calibration. The offset value calculated at last start up is used. If no offset has been calculated a default value of 2048 is used.

Bit 7 **PFCPOWER_CALC_Disable**
- 0 Enable PFC power, AC voltage and current, motor current and DC voltage calculation.
- 1 Disable PFC power, AC voltage and current, motor current and DC voltage calculation.

Bit 8 **AC_FAULT_CHK_Disable**
- 0 Enable AC over/under voltage check.
- 1 Disable AC over/under voltage check, PFC will run regardless AC status.

Bit 9 **ADCompDisable**
- 0 Enable AD Compensation function.
- 1 Disable AD Compensation function.

Bits 10 - 15 Unused; set to zero

[1]Note: SkipOffset=1 is needed when the allowed Motor start time is too short for offset calibration, or the Motor is started in catch spin mode. Please make sure there is at least one normal motor start with SkipOffset=0 before starting the motor with skip offset mode so that the current feedback calibration is done correctly.

FaultClear

Address:	*0x8346 (8051)* *0x1A3 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 1 = Clear all faults; 0 = no action.

Description: Writing 1 to this register clears all faults. Once clear has been done, the register must be set back to 0 for new faults to be registered.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MceFault

Address:	*0xF176 (8051)* *0xEF6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 1 = Generate MCE fault condition; 0 = no action.

Description: This register is used to generate an MCE fault condition. An MCE fault is cleared by writing to the FaultClear register, above. Generally this register is used only as an input to the MCE_FAULT motion peripheral block described in Section 3.3.7.

DisableFaults

Address:	*0x83BA (8051)* *0x1DD (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input with bit field definitions*	*Reset value:*	*0*

Scaling or Notation: For each bit, 1 – Ignore the associated fault; 0 – enable processing of the associated fault.

Description: This register specifies disabling of fault handling as follows:

Bit 0	Reserved. Must be set to “0”.
Bit 1	Reserved. Must be set to “0”.
Bit 2	Dc bus over voltage fault
Bit 3	Dc bus low voltage fault
Bit 4	Over speed fault
Bit 5	Zero speed fault
Bit 6	Phase loss fault
Bit 7	Start Fail fault
Bit 8	MCE fault
Bit 9	Catch Spin overspeed fault
Bit 10	MCE Execution fault
Bit 11	Reserved. Must be set to “0”.
Bit 12	AD Compensation Fault
Bits 13 – 15	Reserved. Must be set to “0”.

When a fault is disabled (bit set to “1”), the fault condition is ignored and the motor keeps running. However, even when a fault is disabled, its occurrence causes an interrupt to be generated (see Section 4.6) and is reported in the FaultFlags register, until the condition that caused the fault disappears, so that it can be handled by the application software.

MtrSeqCtrl

Address:	*0x827A (8051)* *0x13D (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This register is the command control register. It controls the system state with the following values:

1	go to idle state
2	start drive in FOC mode or diagnostic mode selected by MtrCtrlBits
4	stop drive
8	enter reverse catch spin sequence

The command ‘1’ is only valid after power up. Once the system has left the idle state it cannot go back. Idle state is used to halt the system while configuration parameters are assigned.

The command ‘8’ causes the sequencer to transistion from state_closed_lp to state_brake as shown in Figure 49. Therefore, this command is ignored for all states except state_closed_lp.

A typical start up sequence is:
1. Assign all configuration parameters and wait until done.
2. Set MtrSeqCtrl to 1. The system goes from idle to stopped.
3. When time to start the motor, set MtrSeqCtrl to 2.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

SysClk

Address:	*0x83E0 (8051)* *0x1F0 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1228*	*Reset value:*	*0*

Scaling or Notation: System clock = SysClk $\cdot 10^5$, where System clock is in Hz

Description: This register informs the MCE firmware about the system clock rate. The register is configured by MCEWizard.

Example: If the system clock is 100MHz, SysClk should be set to 1000,

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.2 PWM Configuration Write Register Group

GCChargePD

Address:	*0x81D2 (8051)* *0x0E9 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: Number of charge pulse spacing = GCChargePD.

Description: This parameter specifies the number of pulse widths between charging pulses for the Bootstrap Pre-charge function.

A zero vector (low side devices on) is normally applied for the initial turn-on of the PWM inverter output. If a Bootstrap gate driver is used, the Bootstrap capacitors (for u, v, w phase) will all be charged simultaneously, which can lead to a nuisance trip. This charging current can be limited by the built-in Pre-charge control function.

Instead of charging all low side devices simultaneously, the gate Pre-charge control will schedule an alternating (u, v, w phase) charging sequence with programmable charging pulse duration (register GCChargePW).

Parameter GCChargePW controls the charging pulse width while parameter GCChargePD controls the delay between u, v and w phase charging. The delay (GCChargePD) between consecutive charging is specified as a number of charge pulse widths. GCChargeT sets the total number of charging pulses. Figure 69 illustrates this relationship.

GCChargePW

Address:	*0x81D0 (8051)* *0x0E8 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: Gate Precharge duration = $\frac{GCChargePW}{SysClk}$ [usec]

where $SysClk$ is the system clock frequency in MHz.

Description: This parameter specifies the Gate Pre-charge duration. Zero vector (low side devices on) is normally applied for the initial turn-on of the PWM inverter output. If the Bootstrap gate driver is used, the Bootstrap capacitors (u, v, w phase) will all be charged simultaneously. This charging current can be limited by the built-in Pre-charge control function.

Instead of charging all low side devices simultaneously, the gate Pre-charge control will schedule an alternating (u, v, w phase) charging sequence with programmable charging pulse duration.

Parameter GCChargePW controls the charging pulse duration while parameter GCChargePD controls the spacing between u, v and w phase charging. Figure 69 illustrates this relationship.

GCChargeT

Address:	*0x8324 (8051)* *0x192 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: Number of charge pulse = GCChargeT

Description: This parameter specifies the number of Gate Pre-charge pulses. The stated number will be equally split between the 3 phases. For example, GCCharge=30 will give 10 charge pulses in each phase.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

PwmDeadTm

Address:	*0xF10E (8051)* *0xE87 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: Inverter blanking time = PwmDeadTm / SysClk [usec]
where SysClk is the system clock frequency in MHz.

Description: Inverter blanking time for avoiding shoot through between high side and low side devices. The blanking time setting is power device dependent. In some applications, power device switching is intentionally slowed down to reduce EMI noise. Hence inverter blanking time is extended to accommodate slower switching profile.

PwmGuardBand

Address:	*0x8328 (Motor1)* *0x194 (Motor2)*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: Guard band duration = PwmGuardBand / SysClk [usec]
where SysClk is the system clock frequency in MHz.

Description: This parameter provides a guard band such that PWM switching at high modulation cannot migrate into the beginning and end of a PWM cycle. The guard band insertion can improve feedback noise immunity for signals sampled near the beginning and end of a PWM cycle, as in leg shunt current feedback configuration. The guard band duration is independent of PWM carrier frequency.

Note: Guard band insertion will reduce the maximum achievable inverter output voltage.

PwmPeriodConfig

Address:	*0x81D4 (8051)* *0x0EA (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 16383*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the number of digital counts of the SVPWM counter for half of a pwm cycle. The resolution of pwm is affected by the system clock (SysClk) being used. And this parameter is directly related to the system clock and the inverter switching frequency being chosen. The relationship is given by:

$$PwmPeriodConfig = \frac{SysClk \cdot 1000}{2 \cdot FreqPwm} - 1$$

where:
SysClk is the system clock in MHz and
FreqPwm is the inverter switching frequency in KHz.

Pwm2HiThr

Address:	*0x8314 (8051)* *0x18A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Speed threshold = $MaxRpm \times Pwm2HiThr / 128$ [rpm]
where MaxRpm is the maximum application speed in rpm. It is a required entry ("Max RPM") in MCEWizard.

Description: This parameter defines the upper speed threshold for switching from 3-phase to 2-phase PWM. The transition between 3-phase and 2-phase PWM is done using a hysteresis band on speed.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Pwm2LowThr

Address:	*0x8316 (8051)* *0x18B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Speed threshold = $MaxRpm \times Pwm2LowThr / 128$ [rpm]
where MaxRpm is the maximum application speed in rpm. It is a required entry ("Max RPM") in MCEWizard.

Description: This parameter defines the lower speed threshold for switching from 2-phase to 3-phase PWM. The transition between 3-phase and 2-phase PWM is done using a hysteresis band on speed.

TwoPhsCtrl

Address:	*0x82D0 (8051)* *0x168 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input with bit field definitions*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the setup for 3-phase or 2-phase modulation. If TwoPhsCtrl[0] = 1, then inverter PWM will transition from 3-phase to 2-phase PWM modulation whenever the absolute motor speed exceeds the threshold specified by register Pwm2HiThr and transitions back to 3-phase mode whenever the motor speed falls below the threshold specified by register Pwm2LowThr. Modulation type is determined as follows:

Bit 0	**TwoPhsEnable**	
	0	Use 3-phase SVPWM
	1	Use 2-phase SVPWM
Bits 1 - 2	**TwoPhsType**[1]	
	0x	use 2-phase modulation type 1
	10	use 2-phase modulation type 2
	11	use 2-phase modulation type 3
Bits 3 - 15		Not used; should be set to 0.

[1]If leg shunt current sensing is selected and TwoPhsCtrl[0] = 1, then modulation will always be type 3.

GkillFiltCnt

Address:	*0xF1D6 (8051)* *0xEEB (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0x0013*

Scaling or Notation: 1 = 1 / SysClk [usec],
where SysClk is the system clock frequency in MHz.

Description: Parameter GkillFiltCnt defines the number of system clock cycles that the Motor Gatekill signal (IC input pin GATEKILL) must persist before a latched fault (pwm shut down) is generated. This is done to avoid nuisance motor drive shut down due to noise.

Bits 0 – 7	GkillFiltCnt value (0 – 255).
Bits 8 – 15	Not used; should be set to 0.

ActivePol

Address:	*0xF110 (8051)* *0xE88 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter controls active polarity of PWM output as follows:

Bit 0	**Motor GateSenLow**	
	0	active high for low side switches
	1	active low for low side switches

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Bits 1 – 2	Not used; should be set to 0	
Bit 3	**Motor GateSenHigh**	
	0	active high for high side switches
	1	active low for high side switches
Bits 4	Not used; should be set to 0	
Bits 5	**PFC GateSen**	
	0	active high for high side switches
	1	active low for high side switches
Bits 6 – 15	Not used; should be set to 0	

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.3 Torque Loop Configuration Write Register Group

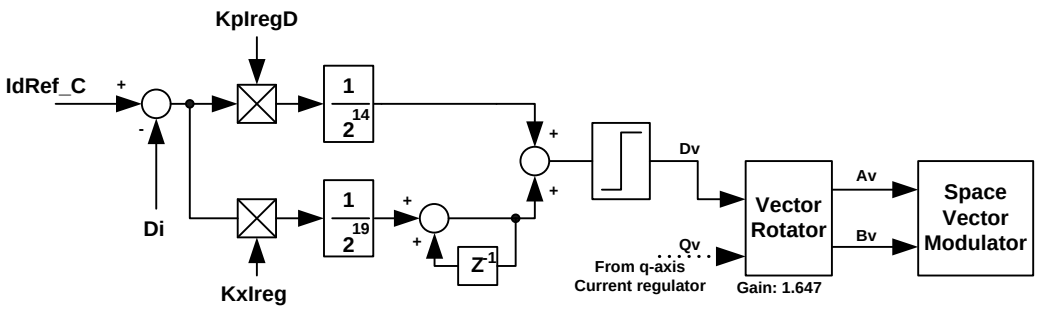
KpIreg

Address:	*0x826C (8051)* *0x136 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: % modulation index = KpIreg × 0.01748 × IqErr / RatedMotorAmps [%]
Where: *IqErr* is absolute current error (abs(command – feedback)) of q-axis in Amps

Description: This parameter specifies the proportional gain of the q-axis current regulator. The parameter relates current error to modulation index. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $Vdc / \sqrt{2}$.

The d-axis channel current regulator gain scaling is shown in the figure below. Q-axis current regulator structure is identical (replace gain KpIregD with KpIreg) to the d-axis. This figure illustrates the gain scaling starting from current error (IdRef_C – Di) up to the input of the Space Vector Modulator. Note: Only gain scalings are illustrated in the figure, antiwindup, dc bus compensation and preservation of numerical truncation are not shown.



KpIregD

Address:	*0x8360 (8051)* *0x1B0 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: % modulation index = KpIregD × 0.01748 × IdErr / RatedMotorAmps [%]
Where: *IdErr* is absolute current error (abs(command – feedback)) of d-axis in Amps

Description: This parameter specifies the proportional gain of the d-axis current regulator. The parameter relates current error to modulation index. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $Vdc / \sqrt{2}$. (See also KpIreg for internal scaling).

KxIreg

Address:	*0x8152 (8051)* *0x0A9 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the integral gain of the d-axis and q-axis current regulator. The parameter relates current error-second (current error integration) to modulation index. The scaling depends on the current regulator execution rate which is directly related to the pwm frequency.

Scaling: Time duration = $\dfrac{1430 \times 2^{19} \times PwmPeriod}{Ierr \times KxIreg}$ [sec.]

where Ierr is the current error in digital count (4095 = rated motor current) and PwmPeriod is the pwm period in sec. The above scaling defines the time required to reach full (100%) modulation

(Dv or Qv) when a fixed current error (Ierr) is present. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding theoretical rms motor line to line voltage at 100% modulation is $Vdc/\sqrt{2}$. As can be seen from this scaling, when PwmPeriod increases (lower pwm switching frequency), KxIreg has to be increased to maintain integral gain strength (time duration to reach 100% modulation). (see KpIreg for internal scaling diagram).

VdLim

Address:	*0x826E (8051)* *0x137 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *-1430 – 1430*	*Reset value:*	*0*

Scaling or Notation: 1430 = 100 [% modulation]

Description: This parameter specifies the d-axis current regulator output limit. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $Vdc / \sqrt{2}$.

VqLim

Address:	*0x8174 (8051)* *0x0BA (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *-1430 – 1430*	*Reset value:*	*0*

Scaling or Notation: 1430 = 100 [% modulation]

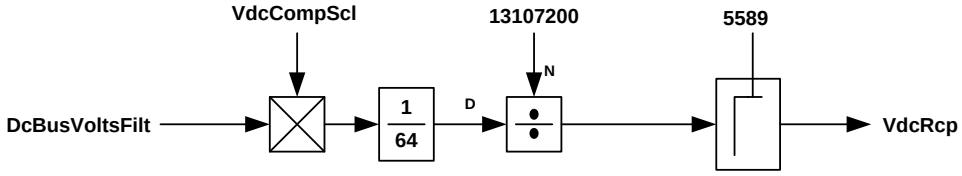
Description: This parameter specifies the q-axis current regulator output limit. 100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line to line voltage at 100% modulation is $Vdc / \sqrt{2}$. This limit provides an upper bound to the dynamic VqLim described in Section 3.3.1.1.

VdcCompScale

Address:	*0x82CE (8051)* *0x167 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See Figure 81.

Description: This parameter is used in DC bus compensation. An appropriate value is calculated by the Wizard



VdcCompScl is a scaler

VdcCompScl = 3200 * 64 / (VdcNom * VdcScl)

where:

VdcNom - the nonimal dc bus voltage in volts.
(Parameter Configurator: "Nominal Vdc")

VdcScl - dc bus scaling in digital counts per volt.
(Parameter Configurator: "dc bus Scale")

Figure 81. Calculation of VdcRcp

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.4 Velocity Control Write Register Group

Rotation

Address:	*0x83D2 (8051)* *0x19E (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 1 = positive rotation; 0 = negative rotation
Description: This parameter defines the direction of rotation.

TrqRef

Address:	*0x8292 (8051)* *0x149 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-16384 – 16383*	*Reset value:*	*0*

Scaling or Notation: 4095 = 100 [% rated motor Amps]
Description: Command current input to inner-loop regulator. In practice, this signal is fed by the output of a speed regulator.

MinSpd

Address:	*0x8344 (8051)* *0x1A2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *1 – 255*	*Reset value:*	*0*

Scaling or Notation: Minimum speed $= \frac{MinSpd \times MaxRPM}{2048}$ [rpm]
where MaxRPM is the maximum application speed, entered in MCEWizard.
Description: This parameter defines the minimum permissible drive operating speed. It is used to detect a zero speed trip fault and also for minimum command speed clamping (in MCE application software). A zero speed fault will be generated if motor speed falls below half of the value of minimum drive speed for a time duration of more than two seconds. The check for zero speed fault can be disabled by setting bit 5 of register DisableFaults (Section 3.4.1). Note that the maximum value of MinSpd represents about 12.5% of the maximum motor speed.

SpeedScale

Address:	*0x8318 (8051)* *0x18C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description
Description: This parameter set the internal MCE speed scaling. It is calculated by MCEWizard

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.5 Closed Loop Angle Estimator Write Register Group

PllFreqLim

Address:	*0x8254 (8051)* *0x12A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the frequency limit of the PLL integral gain output. The relationship between the actual frequency in Hz and this parameter is given by:

$$A = \frac{PllFreqLim \cdot FreqPwm \cdot FreqScl}{2^{13}} [Hz]$$

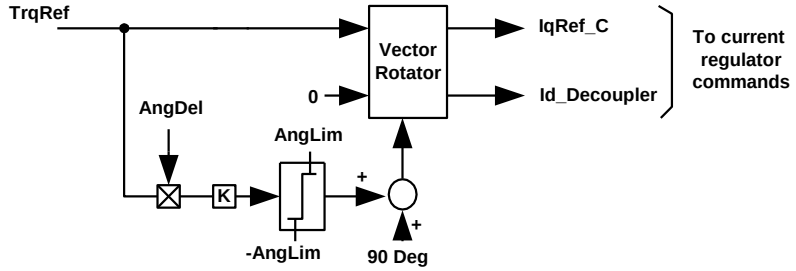
where:
A is the actual frequency in Hz;
FreqPwm is the inverter pwm frequency in Hz; and
FreqScl is the frequency scaler, determined as:
if (Use4xFreqScl = 1) FreqScl = 4
else if (Use2xFreqScl = 1) FreqScl = 2
else FreqScl = 1
(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1)

AngDel

Address:	*0x80B8 (8051)* *0x05C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Angle advancement = AngDel × 0.35156 × I_Motor / Rated Motor Amps [Deg]
Where I_Motor is the operating motor current in Amps

Description: This parameter provides gain adjustment for current angle advancement. The current angle advancement is added to a fixed defaulted phase (90 Deg) and the rotor angle to form the relative phasing of the current vector. Current angle advancement is required for Permanent Magnet motor with rotor saliency (Interior Permanent Magnet Motors). A value of zero represents zero angle advancement and therefore the current vector is placed at 90 degrees with respect to the rotor angle. Details on angle advancement function are given in the Application Developer's Guide under the Interior Permanent Magnet Motor Control section. Diagram below shows the implementation of the angle advancement function and the related controller parameters.



AngLim

Address:	*0x80E2 (8051)* *0x071 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: 1 = 0.17578 [Deg.]

Description: This parameter provides the maximum limit on the current angle phase advancement specified by register AngDel. Details on angle advancement function are given in the Application Developer's Guide. (See also AngDel.)

AtanTau

Address:	*0x814A (8051)* *0x0A5 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides angle compensation (frequency dependent) for the phase shift introduced by flux integration time constant (register FlxTau; see description below). Inside the Sensorless FOC module (Figure 47 Angle Frequency generator), frequency (Rtr_Freq) is multiplied by a time constant (AtanTau) to form a compensating angle. This angle represents the phase shift introduced by the non-ideal flux integrators (low pass filter). Pure (ideal) integrator cannot be used due to dc offset problem. The flux integration time constant is an entry of the iMotion MCEWizard. Typical range of integrator time constant is in the range of 0.01 to 0.025 sec.

Scaling: Time constant = $\frac{A\tan Tau \times PwmPeriod}{2 \times PI \times FreqScl}$ [sec.]

where FreqScl is determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1.)

FlxAInit

Address:	*0x80D0 (8051)* *0x068 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-128 – 127*	*Reset value:*	*0*

Scaling or Notation: 78 = 100 [% rated flux]

Description: This parameter specifies the initial Alpha flux level right after parking stage of motor startup. The initial Alpha flux level depends on the parking angle (ParkAng). MCEWizard calculates this parameter based on the parking angle setup.

FlxBInit

Address:	*0x80D2 (8051)* *0x069 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-128 – 127*	*Reset value:*	*0*

Scaling or Notation: 78 = 100 [% rated flux]

Description: This parameter specifies the initial Beta flux level right after parking stage of motor startup. The initial Beta flux level depends on the parking angle (ParkAng). MCEWizard calculates this parameter based on the parking angle setup.

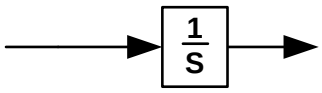
This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

FlxTau

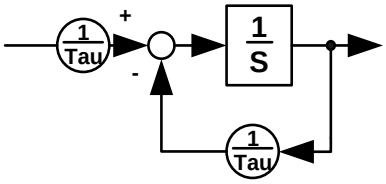
Address:	*0x80D4 (8051)* *0x06A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 8191*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: Motor flux is calculated by integration of estimated voltages. Pure (ideal) integrator cannot be used due to dc offset problem. The integration is done using non-ideal integrator (low pass filter) as shown in the diagram below. The flux integration time constant (Tau) is an entry of the iMotion MCEWizard. Typical range of non-ideal integrator time constant is in the range of 0.01 to 0.025 sec.



Ideal Integrator



Non-ideal integrator

This parameter provides the adjustment for the flux estimator bandwidth. FlxTau is inversely proportional to the "Flux estimator time constant" entered in MCEWizard. The relationship of the Flux estimator time constant and FlxTau is given by:

$$\text{Flux estimator time constant} = \frac{2^{18} \times PwmPeriod}{FlxTau} - PwmPeriod \quad \text{[sec.]}$$

where FlxTm is the Flux estimator time constant,
PwmPeriod = 1/(PWM switching frequency).
FlxTm and PwmPeriod are in seconds.

FreqBW

Address:	*0x8386 (8051)* *0x1C3 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: 1 = 1/ (8192 · PwmPeriod) [rad/sec]
Where PwmPeriod = 1/(PWM switching frequency).

Description: This parameter specifies the filter (first order) bandwidth for estimated motor frequency. This filtered motor frequency is applied to correct the phase shift introduced by the flux integration time constant (register FlxTau). The phase correction is done by taking arctan of filtered frequency x integrator time constant Tau. Since the flux integrator is a first order low pass filter, its phase shift can be easily compensated by arctan of frequency x Tau

IScl

Address:	*0x80B6 (8051)* *0x05B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the current gain scaler for the flux estimator. MCEWizard calculates this parameter. This current scaling is used inside the flux estimator.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

L0

Address:	*0x80E8 (8051)* *0x074 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the apparent inductance of the motor and it is used by the flux estimator. It is proportional to:

$$\frac{Ld + Lq}{2}$$

where Ld and Lq are the d and q axis motor inductance.

The scaling between the actual inductance in Henry and this parameter is formulated in MCEWizard.

LSlncy

Address:	*0x80E4 (8051)* *0x072 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the apparent saliency inductance of the motor. It is proportional to:

$$\frac{Lq - Ld}{2}$$

where Ld and Lq are the d and q axis motor inductance. Typically, Lq/Ld ≈ 1 for Surface PM motors and 1.2 < Lq/Ld < 2.5 for Interior Permanent Magnet motors.

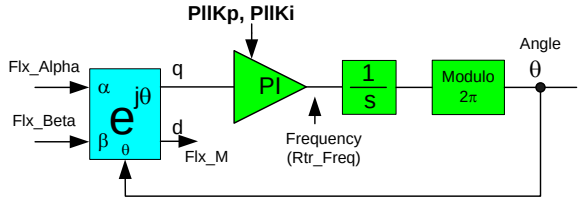
The scaling between the actual inductance in Henry and this parameter is formulated in MCEWizard.

PllKi

Address:	*0x810E (8051)* *0x087 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 8191*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the Angle Frequency Generator (Figure 47) tracking integral gain. The Angle Frequency Generator is mainly a phase lock loop (PLL) device. The diagrams below show both a simplified and the detailed PLL architecture (Figure 82). As can be seen in this diagram, PLLKi relates internal PLL tracking error (q) to frequency (Rtr_Freq). A larger value of PllKi will increase tracking bandwidth at the expense of increasing speed or frequency ripple.



Simplified block diagram of a Flux PLL

MCEWizard calculates this gain based on the selected PLL bandwidth.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

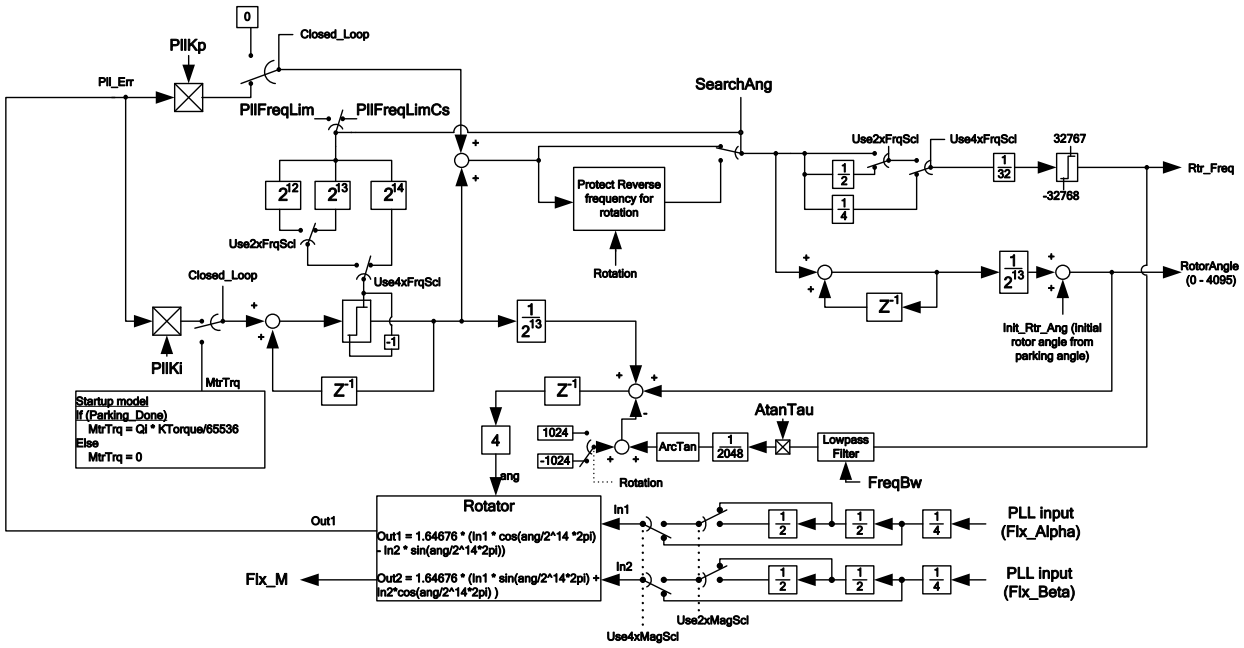


Figure 82. Detail Scaling of PLL

PllKp

Address:	*0x810C (8051)* *0x086 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 8191*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the Angle Frequency Generator tracking proportional gain. The Angle Frequency Generator is mainly a phase lock loop (PLL) device. A larger value of PllKp will increase tracking bandwidth at the expense of increasing speed or frequency ripple. (See Figure 82.)

MCEWizard calculates this gain based on the selected PLL bandwidth.

Rs

Address:	*0x80E6 (8051)* *0x073 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the motor per phase equivalent (motor + cable) resistance at 25 Deg C.

The scaling between the actual motor resistance and this parameter depends on drive voltage and current scaling. The relationship between the actual resistance in ohms and Rs is formulated in MCEWizard.

VoltScl

Address:	*0x80CC (8051)* *0x066 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the gain scaler for translating voltage to internal voltage scaling of the rotor angle estimator. MCEWizard calculates this parameter.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.6 Open Loop Angle Estimator Write Register Group

KTorque

Address:	*0x80D8 (8051)* *0x06C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the motor mechanical model gain used in Open-loop startup mode. KTorque relates motor developed torque to drive acceleration. This gain plays an important role in robust startup. The acceleration scaling is given by:

$$\text{Drive acceleration} = \frac{KTorque \times IMotor \times FreqPwm^2}{RatedMotorAmps \times 2^{29}} \quad \text{[Hz/sec]}$$

where: $FreqPwm$ is the inverter switching frequency in Hz.

$IMotor$ is the motor current in Amps

For instance: At rated motor Amps (IMotor = RatedMotorAmps) and 10KHz inverter pwm frequency, setting KTorque = 100 will yield a 18.63 Hz/sec acceleration rate. At 50% rated motor Amps, the acceleration will reduce by 50%.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.7 Startup Angle Estimator Write Register Group

ParkAng1

Address:	*0x8284 (8051)* *0x142 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

ParkAng

Address:	*0x8286 (8051)* *0x143 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: 64 = 90 [Deg]

Description: These parameters specify the angle to be used in the parking stage of startup. During parking, two parking angles are used. This angle is between motor U-phase and the applied current vector. The drive will first use ParkAng1 for a short duration (25% of total parking duration); thereafter, the parking angle will switch to ParkAng to complete the parking duration.

ParkI

Address:	*0x8280 (8051)* *0x140 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description

Description: This parameter specifies the amount of dc current injection during startup parking stage.

Note: Scaling of ParkI and actual W-phase motor current is given by.

$Iw = ParkI/ 2^8 \times \sqrt{2} \times$ rated motor Amp $\times \cos(ParkAngle - 60^o)$ [peak Amps]

ParkTm

Address:	*0x8188 (8051)* *0x0C4 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Parking Time duration = ParkTm $\times$ 0.01568627 [sec.]

Description: This parameter specifies the total parking duration.

The maximum parking duration that can be set directly using this register is four seconds. It is possible to manually configure an extended parking duration by forcing the drive into parking mode. This is done by enabling the Parking Diagnostic through bit field DiagSelect of register MtrCtrlBits (see Section 3.4.1). The Parking Diagnostic overrides control of parking duration using the ParkTm register.

The following example illustrates this procedure. In the example, parking time is extended to ten seconds by activating the Parking Diagnostic for ten seconds (steps 1 – 4) and then resuming normal drive operation with zero parking time (steps 5 – 7).

1. DiagSelect field of MtrCtrlBits = 1 (enable Parking Diagnostic).
2. Start drive.
3. Delay ten seconds.
4. Stop drive.
5. DiagSelect field of MtrCtrlBits = 0 (disable Parking Diagnostic).
6. ParkTm = 0 (zero parking time since parking is already established).
7. Start drive.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

WeThr

Address:	*0x8148 (8051)* *0x0A4 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the transition level (frequency) from Open-loop to Closed-loop mode operation. The scaling of WeThr is related to internal frequency scaling of the drive by:

$$SwFreq = WeThr \times FreqScl \times FreqPwm / 2^{20} \quad [Hz]$$

where:
SwFreq is the desired switch over frequency in Hz (typically 5 to 10% rated motor electrical frequency);
FreqPwm is the inverter pwm frequency in Hz; and
FreqScl is the frequency scaler, determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively (see 3.4.1).

FlxThrH

Address:	*0x828E (8051)* *0x147 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Upper Flux threshold = $\frac{FlxThrH \times 128}{RatedFlxCounts \times 1.647 \times M}$ x 100 [% rated flux]

where: RatedFlxCounts = 5000 (default by MCEWizard)

```
if (Use4xMagScl = 1) M = 4
Elseif (Use2xMagScl = 1) M = 2
Else M = 1
```

Description: In the Sensorless FOC block, a start fail detection signal (see section 3.4.19) is provided for startup failure detection. This signal can be used by a master motor control sequencer to carry appropriate actions (for instance: startup retry) upon drive startup failure. A successful startup detection is determined by comparing two flux threshold levels (FlxThrH and FlxThrL) and the calculated motor flux (Flx_M). This parameter specifies the upper flux threshold level for determining a successful drive startup.

FlxThrL

Address:	*0x828C (8051)* *0x146 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Lower Flux thershold = $\frac{FlxThrL \times 64}{RatedFlxCounts \times 1.647 \times M}$ x 100 [% rated flux]

where: RatedFlxCounts = 5000 (default by MCEWizard)

```
if (Use4xMagScl = 1) M = 4
Elseif (Use2xMagScl = 1) M = 2
Else M = 1
```

Description: In the Sensorless FOC block, a start fail detection signal (see section 3.4.19) is provided for startup failure detection. This signal can be used by a master motor control sequencer to carry appropriate actions (for instance: startup retry) upon drive startup failure. A successful startup detection is determined by comparing two flux threshold levels (FlxThrH and FlxThrL) and the calculated motor flux (Flx_M). This parameter specifies the lower flux threshold level for determining a successful drive startup.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

FlxChkT

Address:	*0x83C2 (8051)* *0x1E1 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: Startup flux sampling instance = FlxChkT × 1.966 [msec.]

Description: In the Sensorless FOC block, a successful start detection is determined by comparing two flux band levels (FlxThrH and FlxThrL) and the calculated motor flux (Flx_M). The comparison is done at a certain time duration (FlxChkT) after open-loop startup is accomplished. This parameter specifies the sampling instant of motor flux (Flx_M) for determination of a successful startup. FlxChkT is measured from the time Closed-loop is activated. This sampling delay is required to ensure a valid flux establishment in the drive.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.8 Phase Loss Detect Write Register Group

AdjPark1

Address:	*0x8372 (8051)* *0x1B9 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

AdjPark2

Address:	*0x8374 (8051)* *0x1BA (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: These two parameters specify the Phase loss detection current gain.

During parking stage of drive startup, W-phase motor current is compared against anticipated current levels to determine whether a phase loss (connection between inverter and motor) is presented.

Since current regulation is enforced during parking, the motor current will track command current under normal circumstances. If the current error is larger than 50% of the expected current, a phase loss condition is generated. The command current levels are computed using the parking current (ParkI, see Section 3.4.7). The value of ParkI is multiplied by a scaler of either AdjPark1 (first stage of parking) or AdjPark2 (second stage of parking) for proper current comparison between command and actual w-phase current. (See section 3.4.22 register IDiff_Fil for phase loss detection diagram.) Phase loss is evaluated at the end of each parking stage.

In MCEWizard, these current gain scalers are calculated such that proper scaling is applied to parking current (ParkI) for comparison to the scaled feedback current Iw (see section 3.4.22 register IDiff_Fil for more details). Phase loss detection requires sufficient ParkI level to get a valid fault. ParkI is recommended to be at least 30% of rated current for reliable phase loss detection. Additionally, if ParkTm is too short then the parking current may not have stabilized enough to get a valid phase loss detection.

Phase loss detection can be disabled through bit 6 in register DisableFaults (Section 3.4.1).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.9 Current feedback Write Register Group

IfbkScl

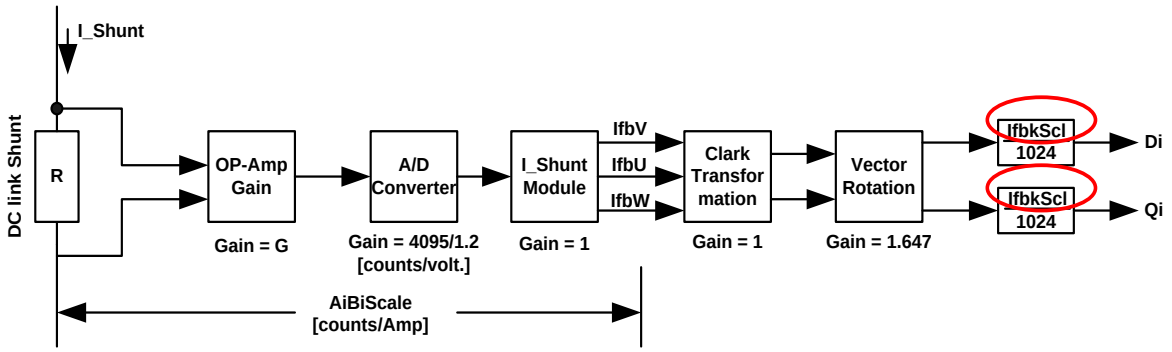
Address:	*0x8150 (8051)* *0x0A8 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-16384 – 16383*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides current gain such that 4095 digital counts of d-axis or q-axis current represents rated motor current. IfbkScl is calculated in MCEWizard and is a function of motor rated Amps and analog current scaling (how many amps per volt of A/D input). The following block diagram shows the motor current feedback path of the iMotion controller IC and the involvement of this gain parameter (IfbkScl). The current feedback path involves Shunt resistor, analog signal conditioning (OP-Amp), A/D converter, Single shunt current reconstruction, Clark transformation and Vector rotator. Each component of the feedback path has its associated gain scaling as shown in the figure. The scaling of IfbkScl is calculated such that at I_Shunt = Rated motor Amps, the value of current magnitude ($\sqrt{Qi^2 + Di^2}$) is equal to 4095 digital counts. For instance: if Di (d-axis current) is regulated to zero (SPM motors) value, Qi will be equal to 4095 when I_Shunt = Rated motor Amps. As shown in the diagram below, based on the target objective of obtaining 4095 digital counts for rated motor current, the current feedback gain (IfbkScl) can be calculated by:

$$\text{IfbkScl} = \frac{4095 \times 1024}{1.647 \times AiBiScale \times RatedMotorAmps \times \sqrt{2}}$$

Where: RatedMotorAmps is in rms Amps. This calculation for the current feedback gain is embedded in the iMotion MCEWizard.



AiBiScale = R x G x 4095 / 1.2 [counts/Amp]

Qi = (IfbkScl / 1024) x 1.647 x AiBiScale x I_Shunt [counts]

Where:
Di and Qi are the d and q current in digital count.
I_Shunt is the dc link current in Amps.
G is the analog amplifier gain. This amplifier is inside the Control IC.
R is the shunt resistance value in ohms.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

IfbSelect

Address:	*0x81DE(8051)* *0x0EF (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 0 = use single shunt current feedback
1 = use leg shunt current feedback

Description: This register selects between single- and leg shut current feedback. If leg shunt is enabled, make sure to enable OP amp 5 in the HWCFG register, see section 2.4.5.

ScsSamples

Address:	*0x82D8 (8051)* *0x16C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *with bit field definitions*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the setup for adaptive current feedback sampling. In the region of minimum pulse constraint (registers TCntMin3Phs or TCntMin2Phs, described below), the frequency of feedback sampling can be reduced in order to minimize the occurrence of minimum pulse clamping. This is done to suppress audible noise for noise-sensitive applications.

Bits 0 – 3 **FbkSampleRate**
This value provides setup for feedback sampling rate in the region of minimum pulse constraint, as follows:

Feedback sample rate = FbkSampleRate + 1 [Pwm cycles/feedback]

Bits 4 – 7 **GainAtten**
This value provides setup for current regulator gain attenuation, as follows:

$$\text{Gain attenuation} = \frac{1}{2^{\text{GainAtten}}}$$

Bits 8 – 15 Unused; set to 0.

For instance, a value of 35 (00100011b) in ScsSamples implies four Pwm cycles/feedback and four times current regulator gain attenuation.

In practice, the gain attenuation should be set equal to the number of Pwm cycles/feedback. It is preferred to use a minimum possible value of Pwm cycles/feedback to satisfy an application in terms of audible noise. The default setting of ScsSamples is 0 (1 PWM cycle/feedback).

SHDelay

Address:	*0x8048 (8051)* *0x024 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *1 – 102*	*Reset value:*	*0*

Scaling or Notation: 1 = 1 / SysClk [usec.]
where SysClk is the system clock frequency in MHz.

Description: This parameter specifies the hardware PWM gate propagation delay. It is measured from IC gating output to the actual turn-on of the power-switching device. It is used by the CURRENT_MEAS module to schedule current sampling instants. In practice, the total PWM gate propagation delay is dominated by the gate driver IC.

SHDelay applies to both single shunt and leg shunt measurement but should include inverter dead time only in single shunt configuration. MCEWizard correctly configures this parameter.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

TCntMin2Phs

Address:	*0x831A (8051)* *0x18D (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: 1 = 1 / SysClk [usec.]
where SysClk is the system clock frequency in MHz.

Description: This parameter specifies the minimum PWM pulse width to be used when 2-phase modulation mode is allowed (by setting register TwoPhsCtrl to appropriate value; see Section 3.4.2).

Figure 52 (c) and (d) illustrate minimum pulse clamping of a 2-phase modulation scheme.

TCntMin3Phs

Address:	*0x8042 (8051)* *0x021 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: 1 = 1 / SysClk [usec.]
where SysClk is the system clock frequency in MHz.

Description: This parameter specifies the minimum PWM pulse width to be used in 3-phase modulation mode. Figure 52 (a) and (b) illustrate minimum pulse clamping of a 3-phase modulation scheme.

DwnSmpThh

Address:	*0x82DC (8051)* *0x16E (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Scaling or Notation: See decription.

Description: This register provides motor speed above which the noise reduction scheme turns off.

$$A = \frac{DwnSmpThh \cdot FreqPwm \cdot FreqScl \cdot 2 \cdot 60}{2^{14} \cdot p}[rpm]$$

where:
p is number of motor poles
A is the actual motor speed in rpm;
FreqPwm is the inverter pwm switching frequency in Hz; and
FreqScl is the frequency scaler, determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1.)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

DwnSmpThl

Address:	*0x82DA (8051)* *0x16D (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Scaling or Notation: See decription.

Description: This register provides motor speed below which the noise reduction scheme turns on.

$$A = \frac{DwnSmpThl \cdot FreqPwm \cdot FreqScl \cdot 2 \cdot 60}{2^{14} \cdot p}[rpm]$$

where:
p is number of motor poles
A is the actual motor speed in rpm;
FreqPwm is the inverter pwm switching frequency in Hz; and
FreqScl is the frequency scaler, determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1.)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.10 Catch Spin Write Register Group

PllFreqLimCs

Address:	*0x8256 (8051)* *0x12B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the frequency limit of the PLL integral gain output during Catch-Spin mode. The relationship between the actual frequency in Hz and this parameter is given by:

$$A = \frac{PllFreqLimCs \cdot FreqPwm \cdot FreqScl}{2^{13}} [Hz]$$

where:
A is the actual frequency in Hz;
FreqPwm is the inverter pwm frequency in Hz; and
FreqScl is the frequency scaler, determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1.)

ZeroVecTm

Address:	*0x8298 (8051)* *0x14C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This register determines the duration of the braking during catch-spin startup. The actual duration of braking is equal to:

$$Break\ Time = \frac{ZeroVecTm \cdot 8}{FreqPwm}$$

where:
FreqPwm is the inverter pwm frequency in Hz.

BrakeMaxSpeed

Address:	*0x829A (8051)* *0x14D (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This register determines the maximum speed at which reverse catch-spin startup is attempted. The actual maximum braking speed is equal to:

$$Braking\ Speed = \frac{BrakeMaxSpeed \cdot Max_RPM}{64} [rpm]$$

where:
Max_RPM is maximum motor speed in rpm as entered in MCEWizard.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

CatchMaxSpeed

Address:	*0x83B4 (8051)* *0x1DA (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This register determines the maximum speed at which forward catch-spin startup is attempted. The actual maximum forward catching speed is equal to:

$$Catch\ Speed = \frac{CatchMaxSpeed \cdot Max_RPM}{64}[rpm]$$

where:
Max_RPM is maximum motor speed in rpm as entered in MCEWizard.

ZeroVoltBrake

Address:	*0x83B6 (8051)* *0x1DB (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 0 = brake with zero vector
1 = brake with zero voltage

Description: This register enables braking with zero voltage by alternating upper and lower switches during reverse catch-spin startup. (Default is braking with zero vector by shorting lower inverter switches.)

CatchTm

Address:	*0x8282 (8051)* *0x141 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This register determines the time instant at which speed is sampled during catch-spin startup. The actual time instant T_speed_sample is equal to:

$$T_speed_sample = \frac{CatchTm}{FreqPwm}[\text{sec}]$$

where:
FreqPwm is the inverter pwm frequency in Hz.

ThetaStart

Address:	*0x8330 (8051)* *0x198 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4096*	*Reset value:*	*0*

Scaling or Notation: 4096 = 360 electrical degrees.

Description: This register determines the initial rotor angle value for open-loop startup during catch-spin startup.

FlxThr_C

Address:	*0x83BE (8051)* *0x1DF (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: 255 = 100% rated flux

Description: This register determines the minimum flux linkage at which catch spin will be engaged. If flux linkage is lower that FlxThr_C the motor will be started with regular parking and open-loop acceleration.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

CsZcTimeout

Address:	*0x82FE (8051)* *0x17F (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: 1 = 8/FreqPwm [sec]

Description: This register determines the timeout value when searching for current zero crossing after braking. If zero crossing is found before timeout, the motor is started without parking. If a zero crossing can not be found before timeout, it is interpreted as a non-rotating motor and startup is done with parking.

3.4.11 User Control Write Register Group

UserVabEn

Address:	*0x8378 (8051)* *0x1BC (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 0 = Select Sensorless FOC to drive SVPWM input modulation
1 = Select User_Alpha and User_Beta to drive SVPWM input modulation

Description: This parameter selects the SVPWM modulation input.

UserVuvwEn

Address:	*0x837A (8051)* *0x1BD (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 0 = Select gating control from SVPWM
1 = Select gating control from duty ratio modulator

Description: PWM pattern selector. Setting UserVuvwEn to 1 selects gating pattern generation from duty ratio control. The built-in SVPWM will be bypassed and registers User_U, User_V and User_W are used instead.

User_Alpha

Address:	*0x837C (8051)* *0x1BE (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Scaling or Notation: 2355 = 100% modulation

Description: This parameter provides the User Alpha (align with U phase) modulation index.

The Space Vector modulator is normally driven by a modulation index generated by the Sensorless FOC. When register UserVabEn = 1, users can bypass the Sensorless FOC and drive Alpha and Beta modulation directly to the inputs of the SVPWM. 100% modulation provides inverter line rms output voltage of $Vdc / \sqrt{2}$.

User_Beta

Address:	*0x837E (8051)* *0x1BF (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Scaling or Notation: 2355 = 100% modulation

Description: This parameter provides the User Beta (Orthogonal to U phase) modulation index.

The Space Vector modulator is normally driven by a modulation index generated by the Sensorless FOC. When register UserVabEn = 1, users can bypass the Sensorless FOC and drive Alpha and Beta modulation directly to the inputs of the SVPWM. 100% modulation provides inverter line rms output voltage of $Vdc / \sqrt{2}$.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

User_U

Address:	*0x8380 (8051)* *0x1C0 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: 1 = 1/PwmPeriodConfig [duty ratio]

Description: This parameter provides the User U-phase duty ratio control.

The inverter-gating pattern is normally generated by the Space Vector Modulator (SVPWM). By setting register UserVuvwEn = 1, users can bypass SVPWM and directly control u-phase gating duty ratio (symmetrically centered) via User_U.

The duty ratio is given by:

$$Duty\ Ratio = \frac{User_U}{PwmPeriodConfig}$$

where PwmPeriodConfig is the value of register PwmPeriodConfig (see Section 3.4.2).

User_V

Address:	*0x8382 (8051)* *0x1C1 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: 1 = 1/PwmPeriodConfig [duty ratio]

Description: This parameter provides the User V-phase duty ratio control.

The inverter-gating pattern is normally generated by the Space Vector Modulator (SVPWM). By setting register UserVuvwEn = 1, users can bypass SVPWM and directly control v-phase gating duty ratio (symmetrically centered) via User_V.

The duty ratio is given by:

$$Duty\ Ratio = \frac{User_V}{PwmPeriodConfig}$$

where PwmPeriodConfig is the value of register PwmPeriodConfig (see Section 3.4.2).

User_W

Address:	*0x8384 (8051)* *0x1C2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: 1 = 1/PwmPeriodConfig [duty ratio]

Description: This parameter provides the User W-phase duty ratio control.

The inverter-gating pattern is normally generated by the Space Vector Modulator (SVPWM). By setting register UserVuvwEn = 1, users can bypass SVPWM and directly control w-phase gating duty ratio (symmetrically centered) via User_W.

The duty ratio is given by:

$$Duty\ Ratio = \frac{User_W}{PwmPeriodConfig}$$

where PwmPeriodConfig is the value of register PwmPeriodConfig (see Section 3.4.2).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.12 Field Weakening Control Write Register Group

IdRefExt

Address:	*0x8364 (8051)* *0x1B2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-16384 – 16383*	*Reset value:*	*0*

Scaling or Notation: 4095 = 100 [% rated motor current]

Description: This parameter specifies the command d-axis motor current in normal operation (excluding parking). The purpose of d-axis current control is to obtain optimal torque per ampere and field-weakening control.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.13 Protection Write Register Group

CriticalOvLevel

Address:	*0x8274 (8051)* *0x13A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: Critical Overvoltage Level = CriticalOvLevel ×16 / VdcScl [volts].
where VdcScl is the dc bus scale (entered as "DC Bus Feedback Scaling" in the MCEWizard) in counts per volt.

Description: Detection level for Critical Overvoltage. If this threshold is exceeded, all low side switches are clamped (zero-vector-braking) to protect the drive and to brake the motor. The zero-vector is held until fault is cleared.

DcBusOvLevel

Address:	*0x8276 (8051)* *0x13B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: Dc bus over voltage trip level = DcBusOvLevel ×16 / VdcScl [volts]
where VdcScl is the dc bus scale (entered as "DC Bus Feedback Scaling" in the MCEWizard) in counts per volt.

Description: This parameter defines the dc bus over voltage trip level. A dc bus over voltage fault will be generated if dc bus voltage exceeds this threshold. Refer to the description of the FaultFlags register in Section 1.1.1 for more information.

DcBusLvLevel

Address:	*0x8278 (8051)* *0x13C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: Dc bus under voltage trip level = DcBusLvLevel ×16 / VdcScl [volts]
where VdcScl is the dc bus scale (entered as "DC Bus Feedback Scaling" in the MCEWizard) in counts per volt.

Description: This parameter defines the dc bus under voltage trip level. A dc bus under trip voltage fault will be generated if dc bus voltage falls below this threshold. Refer to the description of the FaultFlags register in Section 1.1.1 for more information.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.14 External Signals Write Register Group

ExtFwdAngle

Address:	*0x8396 (8051)* *0x1CB (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 1024 = 90 [Degs]

Description: This parameter provides an angle sum into the forward rotating angle (can be set to zero by gain adjustment) of the current regulator. Users have the flexibility of providing their own angle calculation for improved controller performance. The ability to employ an external angle also provides the flexibility of implementing Field-Oriented Control for various types of rotating field motors.

ExtRevAngle

Address:	*0x838E (8051)* *0x1C7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 1024 = 90 [Degs]

Description: This parameter provides an angle sum into the reverse rotating angle (can be set to zero by gain adjustment) of the current regulator. Users have the flexibility of providing their own angle calculation for improved controller performance. The ability to employ an external angle also provides the flexibility of implementing Field-Oriented Control for various types of rotating field motors.

Ext_Flx_Alpha

Address:	*0x839C (8051)* *0x1CE (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Ext_Flx_Beta

Address:	*0x839A (8051)* *0x1CD (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-32768 – 32767*	*Reset value:*	*0*

Scaling or Notation: 5000 = rated motor flux [Volt-sec]

Description: The Sensorless FOC block calculates estimated fluxes (Alpha and Beta pair) internally. These internally generated fluxes can be replaced by external fluxes (Ext_Flx_Alpha and Ext_Flx_Beta). This allows users to substitute appropriate fluxes for implementing Field-Oriented Control of various types of rotating field motors. Setting UseExtFlux = 1 selects these external fluxes. (UseExtFlux is a bit field of register MtrCtrlBits_S; see Section 3.4.1.)

UdFeedFwd

Address:	*0x81E4 (8051)* *0x0F2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-1430 –1430*	*Reset value:*	*0*

Scaling or Notation: 1430 = 100 [% modulation]

Description: This parameter provides d-axis modulation feedforward. This signal sums into the output of the d-axis current regulator.

100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line voltage at 100% modulation is $Vdc / \sqrt{2}$.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

UqFeedFwd

Address:	*0x839E (8051)* *0x1CF (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed input* *-1430 – 1430*	*Reset value:*	*0*

Scaling or Notation: 1430 = 100 [% modulation]

Description: This parameter provides q-axis modulation feedforward. This signal sums into the output of the q-axis current regulator.

100% modulation corresponds to the maximum achievable value of the SVPWM linear range. The corresponding rms motor line voltage at 100% modulation is $Vdc / \sqrt{2}$.

3.4.15 AD Correction Write Register Group

AdRefH

Address:	*0x82A8 (8051)* *0x154 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 4095 = 1.2V reference voltage

Description: This parameter represents the ***ideal*** analog-to-digital conversion of the high reference voltage applied for the purposes of AD compensation. The recommended voltage to apply to the external AD channel is about 1100mV. This parameter is configured by MCEWizard.

This register corresponds to the voltage at pin AIN3 of IRMCx171 and IRMCx143.

AdRefL

Address:	*0x82AA (8051)* *0x155 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 4095 = 1.2V reference voltage

Description: This parameter represents the *ideal* analog-to-digital conversion of the low reference voltage applied for the purposes of AD compensation. The recommended voltage to apply to the external AD channel is about 100mV. This parameter is configured by MCEWizard.

This register corresponds to the voltage at pin AIN4 of IRMCx171 and IRMCx143.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.16 Miscellaneous Signals Write Register Group

VFGain

Address:	*0x82BE (8051)* *0x15F (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the Volts per Hertz gain scaler for the purpose of the open-loop diagnostic. In this diagnostic mode, the command target frequency (VFFreq; see Section 3.4.16) is multiplied by VFGain to generate a modulation index, thereby maintaining a constant Volts/Hz ratio under constant dc bus operation. This mode of operation generates prescribed inverter voltages without requiring current feedbacks. Therefore, it can be used to drive passive load or Induction motor load for troubleshooting hardware (feedback and PWM) related issues.

The scaling relationship is given by:

Modulation Index = VFGain $\times$ VFFreq / 2^8 [digital count of modulation]

where 1430 represents 100% Modulation Index (VLine = $Vdc/\sqrt{2}$) and VFFreq is the frequency command (the value of register VFFreq, as described in Section 3.4.16).

VFFreq

Address:	*0x82BC (8051)* *0x15E (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 16384*	*Reset value:*	*0*

Scaling or Notation: Drive Frequency = VFFreq * 0.01552583 [Hz]

Description: This parameter specifies the Volts per Hertz frequency command for the purpose of the open-loop diagnostic. In this diagnostic mode, the command target frequency (VFFreq) is multiplied by VFGain to generate a modulation index, thereby maintaining a constant Volts/Hz ratio under constant dc bus operation. This mode of operation generates prescribed inverter voltages without requiring current feedbacks. Therefore, it can be used to drive passive load or Induction motor load for troubleshooting hardware (feedback and PWM) related issues.

Zero_Vec_Req

Address:	*0x8326 (8051)* *0x193 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean input* *0 – 1*	*Reset value:*	*0*

Scaling or Notation: 0 = Current regulator output limits (d and q) resumes normal value;
1 = Current regulator output limits (d and q) are set to zero

Description: This control bit commands the Sensorless controller to issue zero output voltage (zero modulation). When this register value is set high, the outputs of the current regulators (d-q) are clamped to zero. When the register value is set low, the current regulator output limits resume their normal values, determined by the settings of registers VdLim and VqLim.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.17 PFC Control Write Register Group

PfcEnable

Address:	*0x819C (8051)* *0x0CE (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1*	*Reset value:*	*0*

Scaling or Notation: 1 = Start PFC PWM
0 = Stop PFC PWM

Description: This parameter acts as the highest level switch of PFC control. Set PfcEnable = 1 to start the PFC PWM. However, the PFC PWM may be disabled depending on the PFC state machine's status, which is determined by system status, AC input voltage and DC bus voltage.

GkillFiltCnt_PFC

Address:	*0xF1D8 (8051)* *0xEED (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 255*	*Reset value:*	*0x0013*

Scaling or Notation: 1 = 1 / SysClk [usec],
where SysClk is the system clock frequency in MHz.

Description: Parameter GkillFiltCnt_PFC defines the number of system clock cycles that the PFC Gatekill signal (IC input pin PFC GATEKILL) must persist before a latched fault (PFC pwm shut down) is generated. This is done to avoid nuisance PFC drive shut down due to noise.

Bits 0 – 7 GkillFiltCnt value.
Bits 8 – 15 Not used; should be set to 0.

Iref

Address:	*0x832C (8051)* *0x196 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 4095 = Maximum IPFC [Amps]

Description: This parameter should receive the reference command from the PFC current loop. This signal is normally obtained from the PFC multiplier that produces the reference command to the current regulator. It is the responsibility of the hardware designer to ensure that maximum IPFC is less than an Iref value of 4095.

PFCVdcRef

Address:	*0x8252 (8051)* *0x129 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides the DCBUS reference to the PFC voltage blanking function. This signal is normally obtained from the reference command to the voltage loop. The PFC blanking block will compare Vac and Vdc to determine the status of the PfcVdcBlanking flag.

BlankOfstDc

Address:	*0x81AA (8051)* *0x*0D5 *(MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: 1 count = (1 / 4095) * Maximum Vdc [Volt]

Description: This parameter defines the hysteresis gap for the PFC voltage blanking function. The PWM will turn off when Vac > Vdc /2 – BlankOfstDc – BlankGap. It is the responsibility of the hardware designer to ensure that Maximum Vdc corresponds to an ADC count of 4095. A hysteresis gap of 12V is recommended.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

BlankGap

Address:	*0x81AC (8051)* *0x*0D6 *(MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter defines the hysteresis gap for PFC current blanking check and AC over voltage check to avoid sensitive detection. It corresponds to the hardware noise level: for higher hardware noise a bigger gap is needed. For a good PCB layout with IPFC and motor current noise less than 50 counts, BlankGap should be set to around 50 counts.

MinBlankTm

Address:	*0x832A (8051)* *0x195 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 127*	*Reset value:*	*0*

Scaling or Notation: 1 count = Motor PWM cycle

Description: This parameter provides a minimum PFC voltage blanking time after the PFC_Vdc_Blanking flag is set as described in Section 3.3.8.2. (Typical time is 0.5 ~ 1 msec.) The minimum blanking time is given by:

Blanking Time = MinBlankTm * Motor_PWM period

PfcPwmCommand

Address:	*0x827C (8051)* *0x13E (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 65535*	*Reset value:*	*0*

Scaling or Notation: Duty Cycle % = (PfcPwmCommand / 65535) * 100

Description: This parameter specifies the PFC PWM duty cycle. This value is normally generated from the PFC control loop, which is implemented in the MCE microprocessor.

PfcSHDelay

Address:	*0x832E (8051)* *0x197 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: 1 = 1 / SysClk [usec]
where SysClk is the system clock frequency in MHz. (A count of 100 gives a 1 usec delay with a 100 MHz system clock.)

Description: This parameter specifies the PFC hardware PWM gate propagation delay. It is measured from IC gating output to the actual turn-on of the power-switching device. It is used by the PFC_SENSE module to schedule current sampling instants. In practice, the total PFC PWM gate propagation delay is dominated by the PFC gate driver IC and IGBT turn on delay. (Typical time is 0.26 usec for the IRMCS1043 reference design kit.)

PfcMinPulse

Address:	*0x81A8 (8051)* *0x0D4 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 1023*	*Reset value:*	*0*

Scaling or Notation: 1 count = 1 SysClk period. (A count of 100 gives a 1 usec pulse width with a 100 MHz system clock.)

Description: This parameter specifies the minimum PFC PWM width. A PWM command less than PfcMinPulse will be ignored. (Typical time is 1 usec for the IRMCS1043 reference design kit.)

AcOvThr

Address:	*0x8270 (8051)* *0x138 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 –4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter specifies the AC over voltage check threshold for the AC Voltage monitoring function.
AcOvThr = AC OV threshold * 1.414 * AC voltage sensing scale.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

AcOvTimeThr

Address:	*0x82EE (8051)* *0x177 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 2047*	*Reset value:*	*0*

Scaling or Notation: 1 count = Motor PWM cycle
Description: This parameter specifies the overvoltage time threshold for the AC Voltage monitoring function. The default time is set to 1 msec in MCEWizard.

AcUvThr

Address:	*0x8272 (8051)* *0x139 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.
Description: This parameter specifies the AC under voltage check threshold for the AC Voltage monitoring function.
AcUvThr = AC UV threshold * 1.414 * AC voltage sensing scale.

AcUvTimeThr

Address:	*0x82ec (8051)* *0x176 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 2047*	*Reset value:*	*0*

Scaling or Notation: 1 count = Motor PWM cycle
Description: This parameter specifies the under voltage time threshold for AC under voltage check. The default time is set to about 25% of a grid period in MCEWizard.

AcOKTimeThr

Address:	*0x82F0 (8051)* *0x178 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 8191*	*Reset value:*	*0*

Scaling or Notation: 1 count = Motor PWM cycle
Description: This parameter specifies the time delay for the PFC to resume from an AC under voltage or over voltage condition. The delay is set to 500 msec in MCEWizard.
Delay time = Motor PWM period * AcOKTimeThr.

MtrIFBKScalx10

Address:	*0x8246 (8051)* *0x123 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: Motor current feedback hardware scale * 10
Description: This parameter is defined as 10 times the Motor current feedback hardware scale. For example, MtrIFBKScalx10 = 1356 if the motor current feedback hardware scale is 135.64 cts/A.

PfcVFBKScalx100

Address:	*0x834E (8051)* *0x1A7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: Vac feedback hardware scale * 100
Description: This parameter is defined as 100 times the Vac feedback hardware scale. For example, PfcVFBKScalx100 = 827 if Vac feedback hardware scale is 8.27 cts/V.

PfcIFBKScalx10

Address:	*0x83AE (8051)* *0x1D7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: IPFC feedback hardware scale * 10
Description: This parameter is defined as 10 times the PFC current feedback hardware scale. For example, PfcIFBKScalx10 = 2098 if PFC current feedback hardware scaling is 209.8 cts/A.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

VdcFBKScalx100

Address:	*0x8354 (8051)* *0x1AA (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 –32767*	*Reset value:*	*0*

Scaling or Notation: Vdc feedback hardware scale * 100

Description: This parameter is defined as 100 times the DCBUS feedback hardware scale. For example, VdcFBKScalx100 = 827 if DCBUS voltage feedback hardware scaling is 8.27 cts/V.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.18 System Read Register Group

pwm_lines

Address:	*0xF234 (8051)* *0xF1A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *with bit field definitions*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides internal PWM gating bits.

Bits 0 – 5	Unused	
Bit 6	**PWMUH**	U phase high side output
Bit 7	**PWMUL**	U phase low side output
Bit 8	**PWMVH**	V phase high side output
Bit 9	**PWMVL**	V phase low side output
Bit 10	**PWMWH**	W phase high side output
Bit 11	**PWMWL**	W phase low side output
Bit 12	**PWMPFC**	PFC gating (IRMCF188 only)
Bits 13 – 15	Unused	

DcBusVolts (AIN0)

Address:	*0x82F4 (8051)* *0x17A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

AIN1

Address:	*0x82F6 (8051)* *0x17B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

AIN2

Address:	*0x82F8 (8051)* *0x17C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

AIN3

Address:	*0x82FA (8051)* *0x17D (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

AIN4

Address:	*0x82FC (8051)* *0x17E (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

AIN5

Address:	*0x8334 (8051)* *0x19A (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

Scaling or Notation: 0 – 1.2V maps to 0 – 4095 digital counts

Description: A/D converter raw output or compensated output for IC pin inputs AIN0 – AIN5. Bit9 of MtrCtrlBits_S controls if these registers provide the raw ADC value or the compensated value if the firmware has the AD compensation function.
If leg shunt current measurement is used, AIN5 will not be valid (It will be a downsampled version of the V-phase current). For more information, refer to section 3.3.4. Please note that AIN5 is only avialble for IRMCx171.

Avref/2

Address:	*0x8332 (8051)* *0x199 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*2048*

Scaling or Notation: 0 – 1.2V maps to 0 – 4095 digital counts

Description: Avref/2 holds a lowpass filtered signal that corresponds to half the ADC reference at A/D converter channel AIN6 (AVREF/2). The signal is not accessible on any pins in IRMCx171 and

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

IRMCx143, but is created internally as shown in Figure 83. The nominal value of Avref/2 is 2048 digital counts corresponding to 0.6V.
The filter used on AIN6 (AVREF/2) is a first order lowpass with the following difference equation:

$$y[n] = \frac{1}{2^9}\left(u[n] - y[n-1]\right) + y[n-1]$$

The bandwidth is determined by:

$$\alpha = \frac{f_{sw}}{5 \cdot \left(2^5 - 1\right)} \left[rad / \sec\right]$$

At system initialization the expected value of the offset measurement (2048) is assigned to the recursive part of the filter. Before using Avref/2, be sure to wait until the filter has settled. It is recommended to wait at least 5 τ (where τ = 1/α). For more information, refer to section 3.3.4.

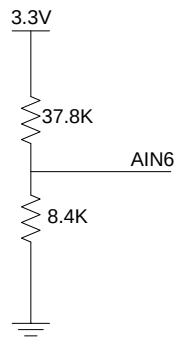


Figure 83. Measurement of AIN6 (AVREF/2).

SequencerState

Address:	*0x8182 (8051)* *0x0C1 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 512*	*Reset value:*	*undefined*

Scaling or Notation: See description.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Description: This register reflects the main state of the FOC block. See sequencer diagram in

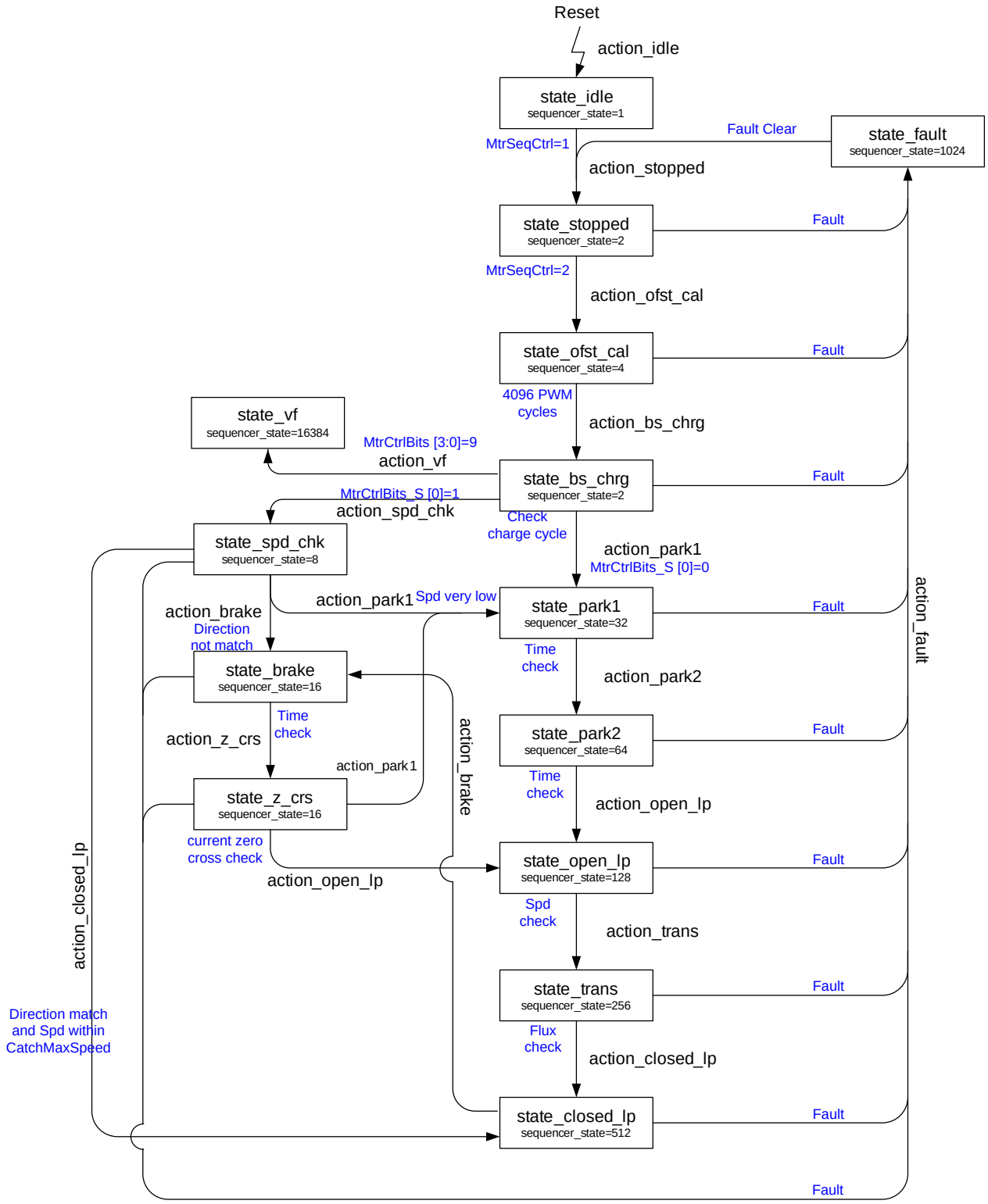


Figure 49.

1	idle state
2	stopped and bootstrap capacitor precharge
4	measure offset current

8	check speed (in catch-spin)
16	braking (in catch-spin)
32	first stage parking
64	second stage parking
128	open loop startup
256	transition from open to closed loop mode
512	closed loop mode
1024	fault pending
16384	open-loop v/f control

FaultFlags

Address:	*0x83DC (8051)* *0x1EE (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *with bit field definitions*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides drive fault status. Most faults are handled by a fault handling routine operating at the PWM inverter switching frequency with the exception of Gate Kill faults. Gate Kill and Critical Overvoltage faults are handled within the Faults module and will instantly initiate inverter and regulator shutdown. The FaultFlags register indicates currently pending fault conditions. The FaultClear register is used to reset fault conditions.

For all bit fields defined below, a value of 1 indicates that the corresponding fault condition is pending.

Bit 0	**MotorGateKill**	Motor gatekill fault
Bit 1	**DcCritOvFault**	Critical overvoltage fault
Bit 2	**DcOvFault**	Dc bus overvoltage fault
Bit 3	**DcLvFault**	Dc bus undervoltage fault
Bit 4	**OverSpdFlt**	Motor overspeed fault
Bit 5	**ZeroSpdFlt**	Zero speed fault
Bit 6	**PhsLossFlt**	Phase loss fault
Bit 7	**StartFailFlt**	Start fail fault
Bit 8	**MCEFlt**	The MCE has generated a fault condition
Bit 9	**CsOverSpdFlt**	Catch Spin overspeed fault
Bit 10	**MCEExeFlt**	MCE execution fault[1]
Bit 11	**PFCGateKill**	PFC gatekill fault
Bit 12	**ADCompFlt**	AD Compensation fault[2]
Bits 13 – 15	Unused (reserved; always zero)	

[1]MCE execution overrun fault. This fault occurs if the MCE firmware does not complete its PWM-cycle processing before the next PWM pulse.

[2] AD compensation fault only checked at power-on MCE configuration stage, which means 4096 Motor PWM cycles after the first time MtrSeqCtrl=1.

SwFaults

Address:	*0x82D6 (8051)* *0x16B (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *with bit field definitions*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register is derived from FaultFlags by the following bitwise logical operation:

$$SwFaults = FaultFlags \cdot \overline{DisableFaults}$$

SwFaults is cleared by FaultClear. For bit field definition, refer to Faultflags.

Version

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Address:	*0x8320 (8051)* *0x190 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 65535*	*Reset value:*	*See description*

Scaling or Notation: See description.
Description: This register provides a version number for the MCE firmware.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.19 System Status Read Register Group

StartOk

Address:	*0x83C4 (8051)* *0x1E2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = Startup in progress or drive stopped
1 = Startup has succeeded (cleared whenever drive stops)

Description: This register indicates a successful startup based on the motor flux. The motor flux (Flx_M) is checked to be between the upper and lower thresholds (FlxThrH and FlxThrL) at the transition from SequencerState from 256 to 512.

ClosedLoop

Address:	*0x8112 (8051)* *0x089 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = Closed-loop mode is not enabled
1 = Closed-loop mode is enabled

Description: This register tells whether or not the system is in closed loop.

ParkingDone

Address:	*0x80CE (8051)* *0x067 (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = Parking stage is not complete.
1 = Parking done; Parking stage has been accomplished.

Description: This register provides status of the parking sequence.

ParkingOne

Address:	*0x80B4 (8051)* *0x05A (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = First stage of Parking is not complete
1 = First stage (25% of the total parking duration) of Parking has been accomplished

Description: This register provides status of the parking sequence.

StartFail

Address:	*0x83DC(8051)* *0x1EE (MCE)*[1]	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = No startup failure
1 = Startup has failed (latched until drive restart occurs).

Description: This register indicates a failed startup based on the motor flux. The motor flux (Flx_M) is checked to be outside the threshold range (FlxThrH – FlxThrL) at the transition from SequencerState from 256 to 512.

[1] This status bit is derived from FaultFlags (see section 1.1.1). Here the address of FaultFlags is listed. However, only bit 7 of the above address is used for StartFail.

TwoPhsStatus

Address:	*0x82D0(8051)* *0x168 (MCE)*[1]	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*undefined*

Scaling or Notation: 0 = Two phase modulation is not enabled
1 = Two phase modulation is enabled

Description: This register is se when two phase modulations is enabled.

1 This status bit is derived from TwoPhsCtrl (see section 3.4.2). Here the address of TwoPhsCtrl is listed. However, only bit 0 of the address above is used for TwoPhsStatus

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.20 DC Bus Voltage Read Register Group

DcBusVoltsRaw

Address:	*0x817A (8051)* *0x0BD (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides raw ADC result of dc bus voltage feedback. This signal is synchronized to PWM cycle data transfer.

Scaling: 1 = 1 / VdcScl [Volts]

where VdcScl is the dc bus scaling in digital counts per volt. VdcScl relates the actual voltage to the raw A/D counts. VdcScl is an entry ("DC Bus Feedback Scaling") in MCEWizard.

DcBusVoltsFilt

Address:	*0x8070 (8051)* *0x038 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides filtered (0.492 msec time constant) and **AD compensated** dc bus voltage feedback. This is the filtered version of the raw dc bus voltage feedback (see register DcBusVolts, above).

Scaling: 1 = 1 / VdcScl [Volts]

where VdcScl is the dc bus scaling in digital counts per volt. VdcScl relates the actual voltage to the raw A/D counts. VdcScl is an entry ("DC Bus Feedback Scaling") in MCEWizard.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.21 FOC Diagnostic Data Read Register Group

Di

Address:	*0x8398 (8051)* *0x1CC (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-16384 – 16383*	*Reset value:*	*undefined*

Scaling or Notation: 4095 = rated motor current [Amps]

Description: This register provides flux current (d-axis) feedback. This signal is reconstructed from single current shunt current feedback.

Dv

Address:	*0x83A0 (8051)* *0x1D0 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-2048 – 2047*	*Reset value:*	*undefined*

Scaling or Notation: 1430 = 100 [% modulation]

Description: This register provides d-axis command modulation index (output of d-axis current regulator).

Flx_Alpha

Address:	*0x8392 (8051)* *0x1C9 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: 5000 = 100 [% rated flux]

Description: This register provides estimated motor flux of Alpha axis. This signal is calculated by the flux estimator. At speeds less than 5% rated, the estimated flux amplitude will start to decrease gradually. This is a consequence of the transfer characteristics of the flux estimator to reject dc offset.

Flx_Beta

Address:	*0x8394 (8051)* *0x1CA (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: 5000 = 100 [% rated flux]

Description: This register provides estimated motor flux of Beta axis. This signal is calculated by the flux estimator. At speeds less than 5% rated, the estimated flux amplitude will start to decrease gradually. This is a consequence of the transfer characteristics of the flux estimator to reject dc offset.

Flx_M

Address:	*0x8138 (8051)* *0x09C (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 8191*	*Reset value:*	*undefined*

Scaling or Notation: % rated flux = $\dfrac{Flx_M \times 16}{RatedFlxCounts \times 1.647 \times M}$ x 100 [%]

where: RatedFlxCounts = 5000 (default of MCEWizard)
if (Use4xMagScl = 1) M = 4
elseif (Use2xMagScl = 1) M = 2
else M = 1

Description: This signal represents the fundamental flux amplitude. (See Figure 82 for the generation of Flx_M.)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

I_Alpha

Address:	*0x8190 (8051)* *0x0C8 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: Scaling = Current Feedback Scaling [digital counts/Amps peak] as output by MCEWizard.
Description: This register provides Alpha phase current of the Alpha-Beta orthogonal frame (Alpha aligned with u-phase).
The scaling of this variable depends on the analog gain setup of the current feedback path ("Current Amp. gain" setting in MCEWizard). See Section 3.4.9.

I_Beta

Address:	*0x8192 (8051)* *0x0C9 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: Scaling = Current Feedback Scaling [digital counts/Amps peak] as output by MCEWizard.
Description: This register provides Beta phase current of the Alpha-Beta orthogonal frame

The scaling of this variable depends on the analog gain setup of the current feedback path ("Current Amp. gain" setting in MCEWizard). See Section 3.4.9.

IdRef_C

Address:	*0x814C (8051)* *0x0A6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-16384 – 16383*	*Reset value:*	*undefined*

Scaling or Notation: 4095 = 100 [% rated motor current]
Description: This register provides d-axis command current. This is the current command being used by the d-axis current regulator.

IqRef_C

Address:	*0x814E (8051)* *0x0A7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-16384 – 16383*	*Reset value:*	*undefined*

Scaling or Notation: 4095 = 100 [% rated motor current]
Description: This register provides q-axis command current. This is the current command being used by the q-axis current regulator.

Qi

Address:	*0x80D6 (8051)* *0x06B (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-16384 – 16383*	*Reset value:*	*undefined*

Scaling or Notation: 4095 = 100 [% rated motor Amps]
Description: This register provides torque current (q-axis) current feedback.

Qv

Address:	*0x83A2 (8051)* *0x1D1 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-2048 – 2047*	*Reset value:*	*undefined*

Scaling or Notation: 1430 = 100 [% modulation]
Description: This register provides q-axis command modulation (output of q-axis current regulator).

RotorAngle

Address:	*0x8134 (8051)* *0x09A (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-2048 – 2047*	*Reset value:*	*undefined*

Scaling or Notation: 1024 = 90 [Deg.]
Description: This is the estimated rotor angle. It is used for the Field-Oriented control reference frame.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

V_Alpha

Address:	*0x8336 (8051)* *0x19B (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides Alpha motor phase voltage. This signal is constructed by dc bus voltage feedback and modulation index (input of SVPWM).

This signal gives the modulation depth, where 100% modulation = 2355. So the inverter line rms output voltage is $\frac{V_Alpha}{2355} \cdot \frac{Vdc}{\sqrt{2}}$

V_Beta

Address:	*0x8338 (8051)* *0x19C (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides Beta motor phase voltage. This signal is constructed by dc bus voltage feedback and modulation index (input of SVPWM).

This signal gives the modulation depth, where 100% modulation = 2355. So the inverter line rms output voltage is $\frac{V_Alpha}{2355} \cdot \frac{Vdc}{\sqrt{2}}$

IfbU

Address:	*0x818A (8051)* *0x0C5 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides reconstructed motor phase U current (offset eliminated). This current is calculated from the dc bus link current feedback, sampled on every motor PWM cycle.

The scaling of this variable depends on the analog gain setup of the current feedback path ("Current Amp. gain" entry in MCEWizard). The Current Feedback Scaling as output by MCEWizard specifies the scaling in digital counts/Amps peak for this variable.

IfbV

Address:	*0x818C (8051)* *0x0C6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides reconstructed motor phase V current (offset eliminated). This current is calculated from the dc bus link current feedback, sampled on every motor PWM cycle.

The scaling of this variable depends on the analog gain setup of the current feedback path ("Current Amp. gain" entry in MCEWizard). The Current Feedback Scaling as output by MCEWizard specifies the scaling in digital counts/Amps peak for this variable.

IfbW

Address:	*0x818E (8051)* *0x0C7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides reconstructed motor phase W current (offset eliminated). This current is calculated from the dc bus link current feedback, sampled on every motor PWM cycle.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

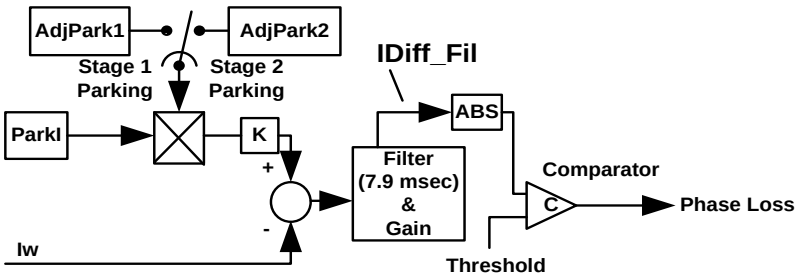
The scaling of this variable depends on the analog gain setup of the current feedback path ("Current Amp. gain" entry in MCEWizard). The Current Feedback Scaling as output by MCEWizard specifies the scaling in digital counts/Amps peak for this variable.

IDiff_Fil

Address:	*0x83E4 (8051)* *0x1F2 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*undefined*

Scaling or Notation: Parking current error = IDiff_Fil × / IfbScl [Amps peak]
where IfbScl is the Current Feedback Scaling [digital counts/Amps peak] as output by MCEWizard

Description: This register provides the parking current error for Phase Loss detection. This signal represents the absolute current error between anticipated w-phase current (calculated from parking current and the actual w-phase feedback current. During parking, if the anticipated w-phase current should match up with the actual w-phase current at the end of the parking duration. If the absolute difference is larger than half the expected current, phase loss is assumed. This register is output for diagnostic purposes. The diagram below shows how IDiff_Fil is calculated.



3.4.22 Velocity Status Read Register Group

Rtr_Freq

Address:	*0x813A (8051)* *0x09D (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-32768 – 32767*	*Reset value:*	*undefined*

Scaling or Notation: See description.

Description: This register provides estimated unfiltered rotor electrical frequency.

$$A = Rtr_Freq \times FreqPwm \times FreqScl / 2^{20} \text{ [Hz]}$$

where:
A is the actual frequency in Hz;
FreqPwm is the inverter pwm switching frequency in Hz; and
FreqScl is the frequency scaler, determined as:

```
if ( Use4xFreqScl = 1 ) FreqScl = 4
else if ( Use2xFreqScl = 1 ) FreqScl = 2
else FreqScl = 1
```

(Use4xFreqScl and Use2xFreqScl are bit fields of the MtrCtrlBits_S and MtrCtrlBits registers, respectively. See Section 3.4.1.)

SearchAng

Address:	*0x8118 (8051)* *0x08C (MCE)*	*Size:*	*16 bits*	*Range:*	*Boolean output* *0 or 1*	*Reset value:*	*0*

Scaling or Notation: 0 = Skip searching for rotor angle;
1 = Search for rotor angle

Description: The Sensorless FOC block set this bit to indicate it is searching for the rotor angle before enabling torque. This is done to ensure smooth start into a spinning motor. In practice, the motor control sequencer will issue this control bit at the very beginning of the start command if the Catch-Spin function is enabled (CatchEnb bit in register MtrCtrlBits_S; see Section 3.4.1).

3.4.23 Current Feedback Offset Read Register Group

Id_Decoupler

Address:	*0x8300 (8051)* *0x180 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-16384 – 16383*	*Reset value:*	*undefined*

Scaling or Notation: 4095 = 100 [% rated motor current]

Description: This register provides d-axis Current Decoupler output. The Current Decoupler provides the optimal current angle for maximum torque per ampere control. In practice, IdRefExt input (see Section 3.4.12) should include Id_Decoupler.

IfbOffset

Address:	*0x82D2 (8051)* *0x169 (MCE)*	*Size:*	*16 bits*	*Range:*	*Signed output* *-4096 –4095*	*Reset value:*	*undefined*

Scaling or Notation: See description

Description: This register provides current feedback dc offset compensation. The entry “Ai Bi scale” in the MCEWizard specifies the scaling in digital counts/Amps peak for this variable. Note, the current measurement channel is supposed to be biased by AVREF/2. This parameter holds any offset that deviates from AVREF/2.
In case of leg shunt measurement, IfbOffset is the average offset of the two current channels.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.24 Protection Read Register Group

VdcRcp

Address:	*0x8388 (8051)* *0x1C4 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 5589*	*Reset value:*	*0*

Scaling or Notation: 4096 = Nominal dc bus

Description: This signal represents the reciprocal of dc bus voltage feedback. It can be used for current regulator dc bus compensation. If dc bus compensation is enabled (bit IregCompEnb of register MtrCtrlBits_S; see Section 3.4.1), the current regulator will use this signal to perform dc bus compensation. See Figure 81 for further information.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.25 AD Correction Read Register Group

AdGainx1024

Address:	*0x82A6 (8051)* *0x153 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: 1024 = Ideal AD gain

Description: This correction factor is the calculated AD gain (relative to the ideal gain x 1024). This parameter is calculated from the ideal (AdRefH & AdRefL) and actual AD result of the reference voltages at AIN3 & AIN4 for IRMCx171 and IRMCx143 as shown in Figure 60.

Figure 84 below shows how the MCE does the AD compensation based on AdGainx1024 and AdOfst.

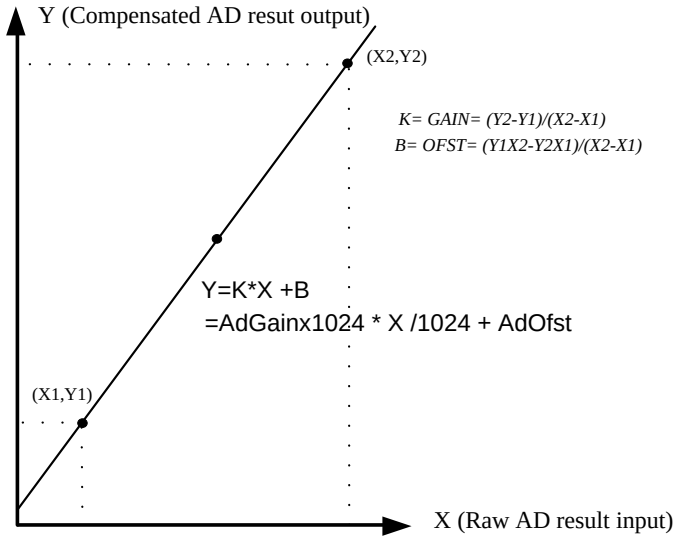


Figure 84. Implementation of AD Compensation

AdOfst

Address:	*0x82A4 (8051)* *0x152 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *-1024 - 1023*	*Reset value:*	*0*

Scaling or Notation: 1024 = 0.3V

Description: This correction factor is the calculated AD offset. This parameter is calculated from the ideal (AdRefH & AdRefL) and actual AD result of the reference voltages at AIN3 & AIN4 for IRMCx171 and IRMCx143 as shown in Figure 84 above.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

3.4.26 PFC Status Read Register Group

VAC

Address:	*0x817C (8051)* *0x0BE (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 2047*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides the A/D feedback signal of AC input voltage, as an absolute value, or rectified half-wave AC input voltage.

Scaling: 1 = 1 / VacScl [Volts]

where VacScl is the AC line feedback scaling in digital counts per volt. VacScl relates the actual voltage to the raw A/D counts. AC line Feedback scaling is an entry ("AC Line Feedback Scaling") in MCEWizard.

VacVoltsFilt

Address:	*0x80DC (8051)* *0x06E (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 2047*	*Reset value:*	*0*

Scaling or Notation: Same as VAC.

Description: This parameter provides the filtered AC input voltage, using a low pass filter with a bandwidth one quarter of the motor PWM frequency. VacVoltsFilt is used for Vac peak voltage detection and over/under voltage checking.

IPFC

Address:	*0x8376 (8051)* *0x1BB (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides the A/D feedback signal of AC input current.

.

The scaling of this variable depends on the analog gain setup of the PFC current feedback path ("Current Amp. gain" entry in MCEWizard). The Current Feedback Scaling as output by MCEWizard specifies the scaling in digital counts/Amps peak for this variable.

VacPeak

Address:	*0x827E (8051)* *0x13F (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: Normally the DC bus and the AC voltage feedback scaling are the same. This parameter provides the Vac peak voltage, which is used for the DCBUS reference in partial PFC mode and for VAC RMS calculation when PFC is OFF. This parameter is updated every 100 msecs. In partial PFC mode the DCBUS Ref should be slightly lower than the Vac peak voltage to optimize the tradeoff between total harmonic distortion and PFC switching loss. For this reason, VacPeak – VdcHyst is used as the partial PFC DCBUS reference.

The output range of VacPeak is [1.25 * UvLevel, 0.96875 * OvLevel], which is limited by the MCE to ensure that the partial PFC DCBUS reference does not cause a DC over or under voltage fault.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

PfcPwmOn

Address:	*0x819E (8051)* *0x0CF (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 1*	*Reset value:*	*0*

Scaling or Notation: 1 = PFC PWM is active;
0 = PFC PWM output is disabled.

Description: This parameter is an output signal of the PFC_PWM block's blanking module which enables or disables the PFC PWM output. PfcPwmOn is 1only when PFCEnable is 1 and all PFC-related parameters are set as required for PFC PWM to be active (checked by PFC state machine).

When PfcPwmOn is 0, PFC PWM output is disabled regardless of the value of PFCEnable. PfcPwmOn is 0 if any one of following conditions is true:

1. PFC command is stop
2. PWM has been disabled by the PFC voltage blanking function
3. PWM command is less than PFC minimum pulse
4. AC input is over or under voltage

This output can be used in an MCE design or 8051 application code to enable, disable or reset the PFC control loop (voltage, current, etc.).

PfcVdcBlanking

Address:	*0x81AE (8051)* *0x0D7 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 1*	*Reset value:*	*0*

Scaling or Notation: 1 = PFC PWM is disabled because instantaneous Vac higher than DC bus
0 = Vdc is higher than Vac instantaneous value.

Description: This parameter indicates when PWM is disabled because of PFC voltage blanking. This flag is updated every *motor* PWM cycle.

AcStatus

Address:	*0x80B2 (8051)* *0x059 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 5*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides status of AC input as detected by the AC voltage monitoring function. Bit definition of AcStatus:

Bit 0: 1 = PFC AC input abnormal.
0 = PFC AC input normal for at least AcOKTimeThr.

Bit 1: PFC AC input UV flag: This bit is set to 1 if the AC voltage is lower than AcUvThr for at least AcUvTimeThr.

Bit 2: PFC AC input OV flag: This bit is set to 1 if the AC voltage is higher than AcOvThr for at least AcOvTimeThr.

VacOfst

Address:	*0x836C (8051)* *0x1B6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides the A/D offset of the AC input voltage feedback channel. The MCE calculates the difference between the Vac & Vdc A/D channels and saves the difference as VacOfst at the motor current offset correction stage.

IPfcOfst

Address:	*0x81A0 (8051)* *0x0D0 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned output* *0 – 4095*	*Reset value:*	*0*

Scaling or Notation: See description.

Description: This parameter provides the A/D offset digital count of the AC input current feedback channel. The MCE calculates the offset at the first power-on initialization, and again at the motor current offset correction if the PFC is stopped before the motor is started.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MotorI_RMS_Ax10

Address:	*0x8250 (8051)* *0x128 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: 1 count = 0.1A
Description: This parameter is the measured RMS Motor current.

AcV_RMS_V

Address:	*0x8362 (8051)* *0x1B1 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: 1 count = 1V
Description: This parameter is the measured RMS AC input voltage.

AcI_RMS_Ax10

Address:	*0x83AC (8051)* *0x1D6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: 1 count = 0.1A
Description: This parameter is the measured RMS AC input current.

DcV_RMS_V

Address:	*0x8356 (8051)* *0x1AB (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 –32767*	*Reset value:*	*0*

Scaling or Notation: 1 count = 1V
Description: This parameter is the measured RMS DCBUS voltage.

PfcPower_W

Address:	*0x834C (8051)* *0x1A6 (MCE)*	*Size:*	*16 bits*	*Range:*	*Unsigned input* *0 – 32767*	*Reset value:*	*0*

Scaling or Notation: 1 count = 1W
Description: This parameter is the measured PFC input power.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4 8051 / MCE Interface

This section describes the methods by which the 8051 processor and MCE communicate and the resources they share. These include:

- OTP (IRMCK1xx) or flash (IRMCF1xx) Memory
- Shared RAM
- MCE processor registers
- MCE motion peripheral configuration register interface
- Interrupts from the MCE to the 8051

4.1 Memory Map

The 8051 is an 8-bit processor and memory is addressed in 8-bit bytes, whereas the MCE is a 16-bit processor and memory is addressed in 16-bit words. Both processors have separate program and data address spaces; it is allowable for address ranges in program space to overlap ranges in data space. In 8051 address space, the non-volatile memory (OTP or flash) is mapped to program space, while MCE program RAM, data RAM and the motion hardware registers (MHRs) are mapped to data space. In MCE address space, both program and data RAM are based at address 0x000. MCE program RAM is mapped to program space, with data RAM and the motion hardware registers (MHRs) mapped into data space.

Figure 85 illustrates the 8051 and MCE address maps for the IRMCK1xx. OTP memory is shown in yellow. Figure 86 shows the 8051 and MCE address maps for the IRMCF1xx. Flash memory is shown in yellow. In both figures, note that the 8051 program and data address spaces have overlapping address ranges.

In both figures, the MCE program is shown in pink. This 12K-byte area of memory is mapped contiguously in MCE program address space, but in the 8051 data address space, it's divided into an 8K-byte segment and a 4K-byte segment.

Shared data RAM is shown in blue and the MHR area is shown in green. These areas are mapped to both 8051 and MCE data address spaces.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

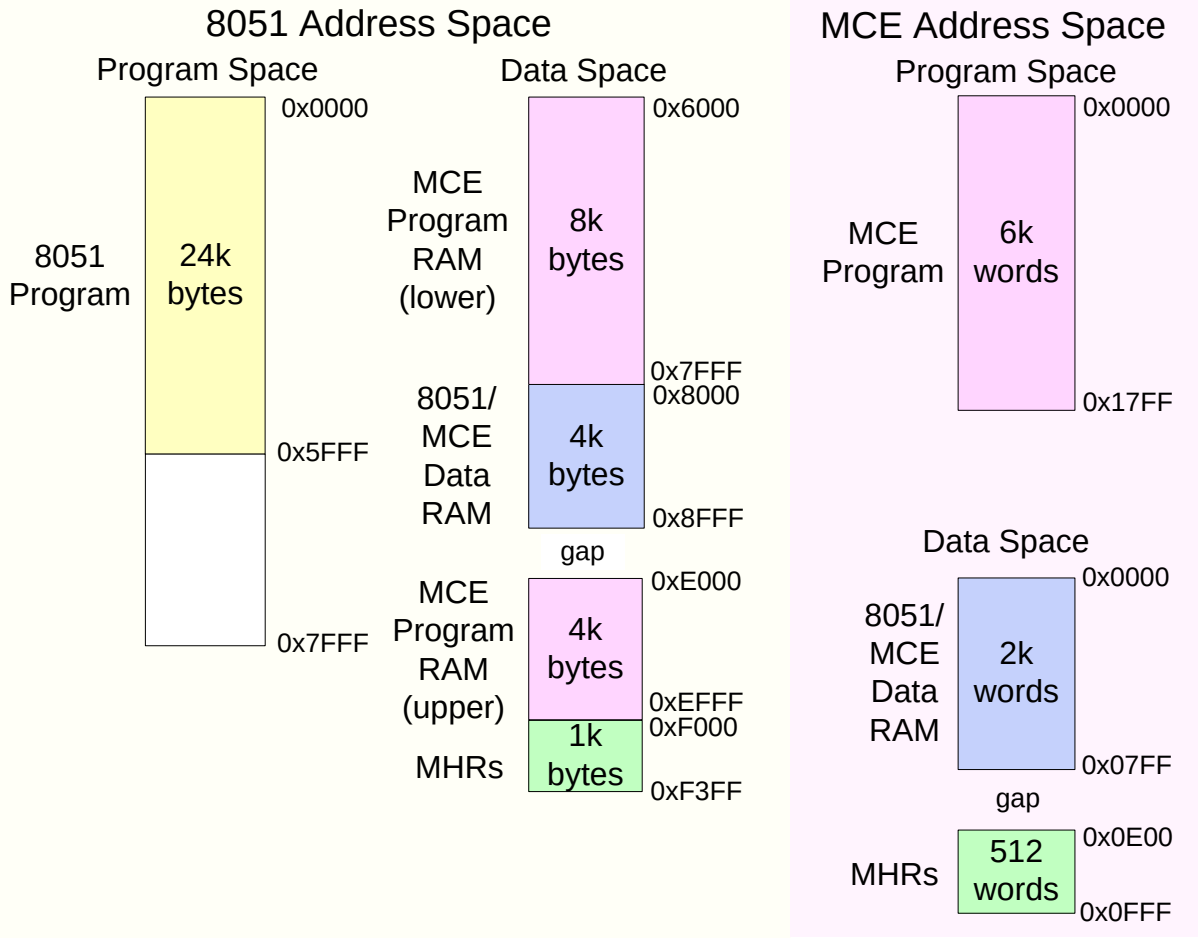


Figure 85. Memory Map of IRMCK100 Series

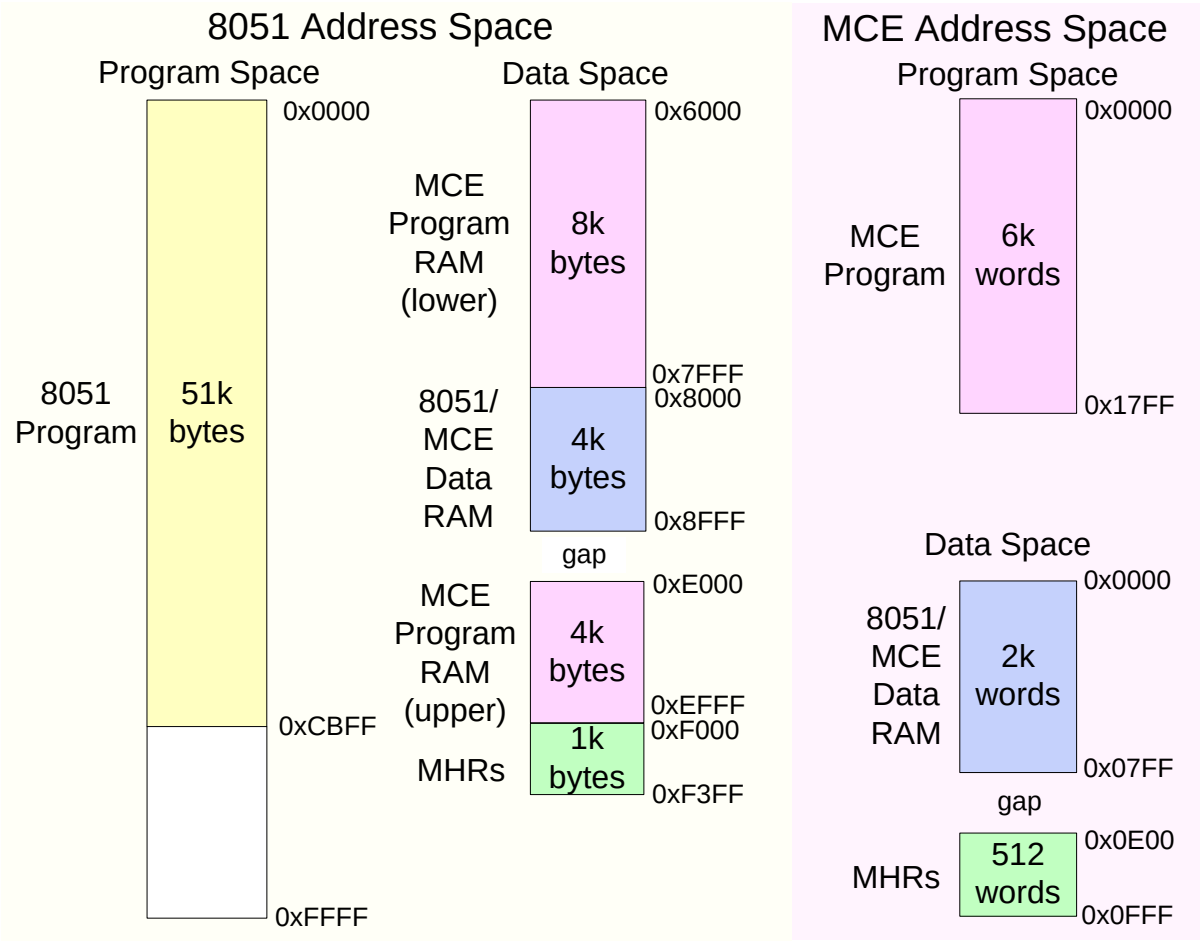


Figure 86. Memory Map of IRMCF100 Series

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4.2 IRMCK1xx OTP Memory

The 32K-byte OTP memory contains both 8051 and MCE programs, and is divided into two areas with the 8051 program contained in the first section and the MCE program contained in the second, The boundary between the two areas is dynamic and can vary from 20K bytes to 24K bytes. This allows a maximum size of 24K bytes for the 8051 program (limiting MCE program size to 8K bytes) or a maximum size of 12K bytes for the MCE program (limiting the 8051 program to 20K bytes). The total program size (8051 + MCE) cannot exceed 32K bytes. For example, if the MCE program is 10K bytes, the 8051 program cannot exceed 22K bytes.

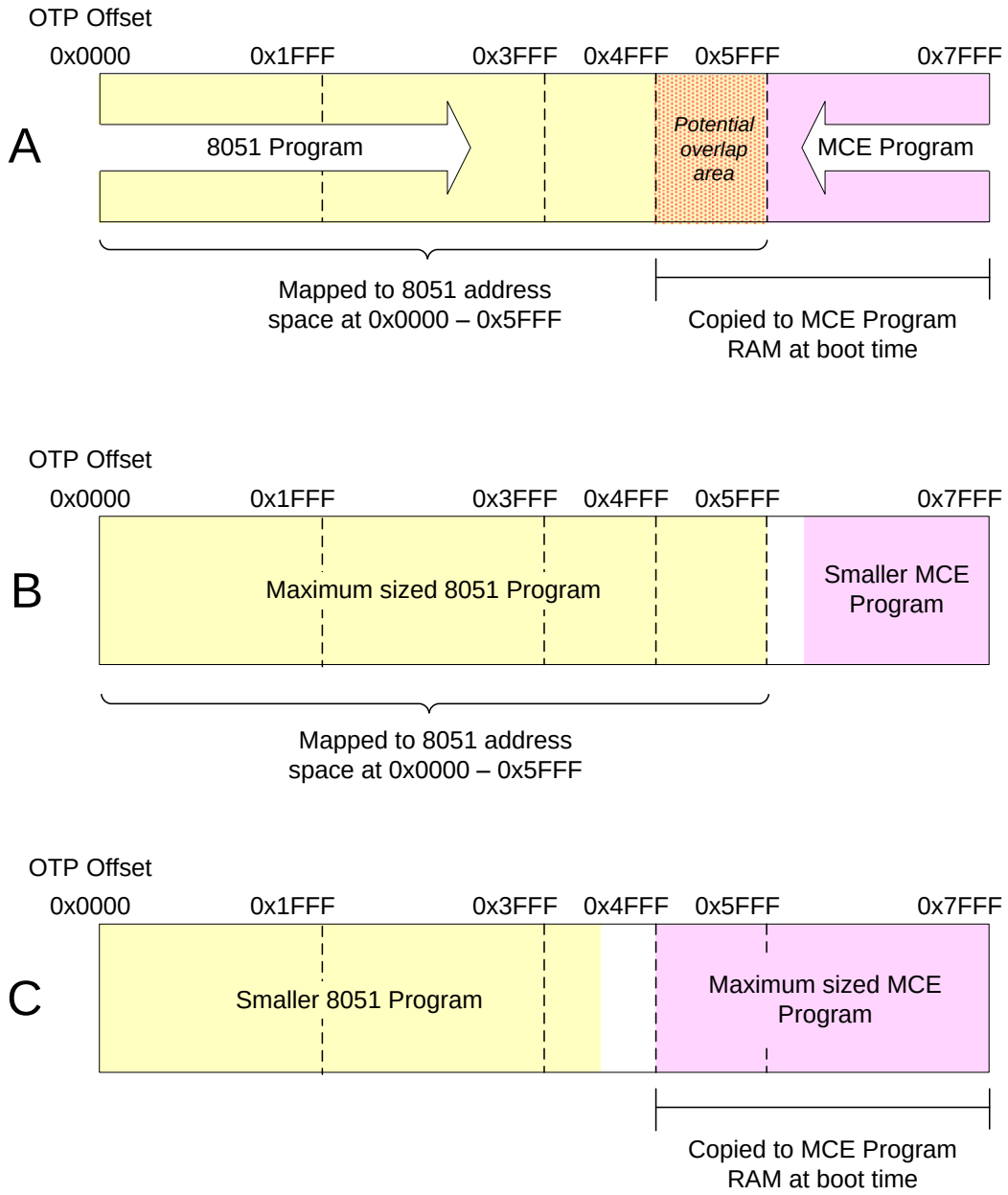


Figure 87. OTP Memory Usage for IRMCK100 Series

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Figure 87 illustrates how OTP memory is used. Diagram A shows that the 8051 program begins at the base of OTP memory and extends toward higher offsets. MCE program begins at the top of OTP memory and extends toward lower offsets. The size of the 8051 program is limited to a maximum of 24K bytes. The last 12K bytes of OTP are copied to MCE program RAM at boot time, limiting the MCE program size to a maximum of 12K bytes. The diagram shows the potential overlap of 8051 program and MCE program at OTP offsets 0x5000 – 0x5FFF. The user must ensure that such an overlap does not occur. Diagram B shows an 8051 program using the maximum 24K-byte portion of OTP memory. This limits the MCE program to the remaining 8K bytes. Diagram C shows an MCE program using the entire 12K byte area that is copied to MCE program RAM at boot time. This limits the 8051 program to the first 20K bytes. Diagrams B and C both show valid allocations of OTP memory.

The details of OTP memory usage are as follows:

- In 8051 program address space, OTP memory is mapped to addresses 0x0000 – 0x7FFF.
- The first 24k bytes of OTP can contain user application program for 8051 and the 8051 program counter starts at address 0x0000. The 8051 processor executes its program directly from OTP memory.
- The last 8K bytes of OTP are reserved for the MCE application program and this content is not normally accessed by the 8051 CPU. (This boundary is enforced by the iMotion software tools but is not a hardware requirement. It is possible for the 8051 application program to exceed 24K bytes in size if the MCE program is smaller than 8K bytes.)
- The MCE processor executes its program from RAM, not directly from OTP memory. The MCE cannot access the OTP memory.
- At power-up, internal boot loader hardware automatically copies the MCE program from the upper section of OTP memory into the MCE program RAM area. Hardware loads the user MCE program backward from address 0x7FFE at the top of OTP to the base of MCE program RAM, so that the byte at offset 0x7FFE in OTP is loaded to the byte at offset 0x0000 in MCE RAM, the byte at offset 0x7FFD in OTP is loaded into the byte at offset 0x0001 in RAM, and so on. (OTP address 0x7FFF is reserved for OTP protection support.) Regardless of the actual MCE program size, the boot loader hardware always copies the entire maximum 12K-byte MCE program from OTP to MCE program RAM.
- For security, the MCE program is automatically encrypted by the MCE compiler and stored in OTP memory in encrypted form. The program is decrypted automatically as it is copied to MCE program RAM during the boot process. The 8051 does not have read access to MCE program RAM, so the MCE program cannot be read once it has been decrypted.
- MCE program RAM is contiguous in MCE program address space, but in the 8051 data address space it is divided into two areas: the first 8K bytes are mapped to addresses 0x6000 – 0x7FFF and the last 4K bytes are mapped to addresses 0xE000 – 0xEFFF. The internal boot loader hardware copies the last 4K bytes of program in the same backward fashion as the first 8K bytes, as shown below:

Offset in OTP	copied to	8051 Address
0x7FFE		0x6000
0x7FFD		0x6001
…		…
0x6000		0x7FFE
0x5FFF		0x7FFF
0x5FFE		0xE000
0x5FFD		0xE001
…		…
0x5000		0xEFFE
0x4FFF		0xEFFF

- Note that MCE program RAM area cannot be used for 8051 local data RAM. The 8051 has only write access to MCE program RAM.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4.3 IRMCF1xx Flash Memory

The 64K-byte flash memory contains both 8051 and MCE programs, and is divided into two areas with the 8051 program contained in the first section and the MCE program contained in the second. The boundary between the two areas is fixed at offset 0xCBFF and the upper 1K bytes (beginning at offset 0xFC00) are reserved. This allows a maximum size of 51K bytes for the 8051 program and a maximum size of 12K bytes for the MCE program.

Figure 88 illustrates how flash memory is used. The 8051 program begins at the base of flash memory and extends toward higher offsets. MCE program begins near the top of flash memory and extends toward lower offsets. The upper 1K bytes of flash memory are reserved. The entire flash memory is mapped to 8051 program address space. The 12K bytes of MCE program are copied from flash to MCE program RAM at boot time. 8051 and MCE program ranges do not overlap in flash as they do in OTP memory.

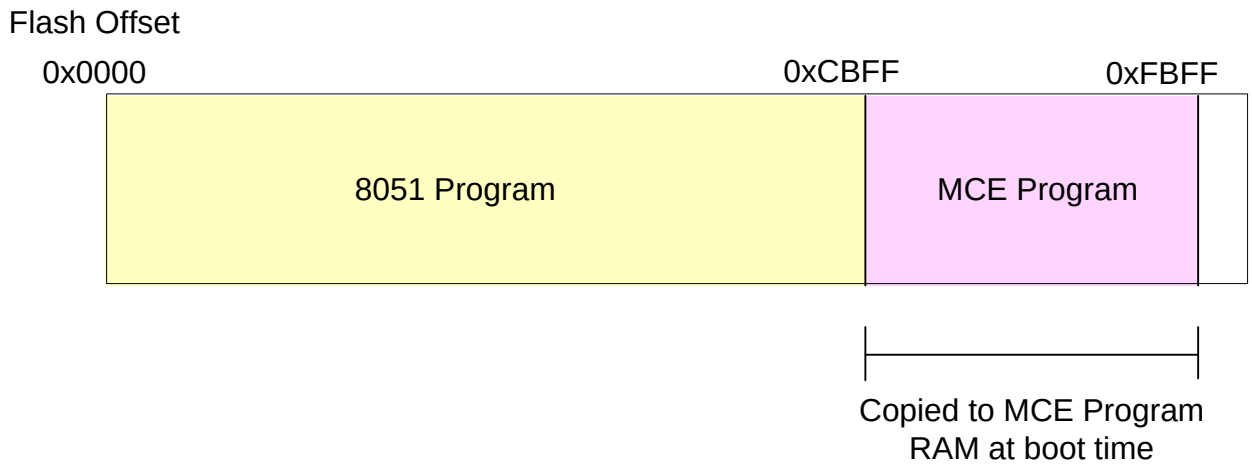


Figure 88. Flash Memory Usage for IRMCF100 Series

The details of flash memory usage are as follows:

- In 8051 program address space, the flash memory is mapped to program address range 0x0000 – 0xFFFF.
- The first part of flash memory contains the user application program for 8051 and the 8051 program counter starts at address 0x0000. The 8051 processor executes its program directly from OTP memory.
- Address ranges 0xCBFF – 0xFFFF are reserved for the MCE application program and this content is not normally accessed by the 8051 CPU. (This boundary is enforced by the iMotion software tools but is not a hardware requirement. It is possible for the 8051 application program to exceed 51K bytes in size if the MCE program is smaller than 12K bytes.)
- The MCE processor executes its program from RAM, not directly from flash memory. The MCE cannot access the flash memory.
- At power-up, internal boot loader hardware automatically copies the MCE program from the upper section of flash memory into the MCE program RAM area. Hardware loads the user MCE program backward from address 0xFBFE near the top of flash to the base of MCE program RAM, so that the byte at offset 0xFBFE in flash is loaded to the byte at offset 0x0000 in MCE RAM, the byte at offset 0xFBFD in flash is loaded into the byte at offset 0x0001 in RAM, and so on. (Flash address range 0xFBFF – 0xFFFD is unused and addresses 0xFFFE – 0xFFFF are reserved for flash protection support.) Regardless of the actual MCE program size, the boot loader hardware always copies the entire maximum 12K-byte MCE program from flash to MCE program RAM.
- For security, the MCE program is automatically encrypted by the MCE compiler and stored in flash memory in encrypted form. The program is decrypted automatically as it is copied to MCE program RAM during the boot process. The 8051 does not have read access to MCE program RAM, so the MCE program cannot be read once it has been decrypted.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

- MCE program RAM is contiguous in MCE address space, but in the 8051 address space it is divided into two areas of data space: the first 8K bytes are mapped to addresses 0x6000 – 0x7FFF and the last 4K bytes are mapped to addresses 0xE000 – 0xEFFF. The internal boot loader hardware copies the last 4K bytes of program in the same backward fashion as the first 8K bytes, as shown below:

Offset in flash	copied to	8051 Address
0xFBFE		0x6000
0xFBFD		0x6001
…		…
0xDC00		0x7FFE
0xDBFF		0x7FFF
0xDBFE		0xE000
0xDBFD		0xE001
…		…
0xCC00		0xEFFE
0xCBFF		0xEFFF

- Note that MCE program RAM area cannot be used for 8051 local data RAM. The 8051 has only write access to MCE program RAM.

4.4 The Shared Data RAM

The IRMCx100 series contains 4K bytes of shared data RAM, all of which is accessible to both the 8051 processor and the MCE. From the 8051 processor, the shared RAM is accessed as external data RAM in the address range 0x8000 – 0x8FFF. The shared RAM is used for:

- MCE processor registers (see Section 4.5)
- MCE firmware data RAM, including motion firmware registers and internal variables
- MCE application data RAM, including MCE registers and internal variables
- 8051 data RAM

The MCE processor registers occupy a small fixed area at the beginning of shared RAM at addresses 0x8000 – 0x801F. The boundaries between the remaining areas are not fixed. The MCE compiler determines MCE firmware data usage dynamically and allocates space for the MCE application from unused areas. The MCE compiler reserves the address range 0x8600 – 0x8FFF for the 8051 application.

4.4.1 Reading and Writing Shared RAM

The MCE is a 16-bit processor. All of its memory accesses are on 16-bit boundaries and it always reads and writes 16-bit words. However, the 8051 is an 8-bit processor and reads and writes only 8-bit words. This could potentially lead to race hazard when the two processors access shared RAM, since the 8051 requires two memory accesses to read or write a 16-bit word. The 8051 could potentially read the first byte of a word and the MCE could modify the word before the 8051 reads the second byte, resulting in corrupted data.

To prevent data corruption in shared RAM when reading and writing 16-bit values, special hardware assistance is implemented using an extension register defined as a group of 8051 special function registers (SFRs). Whenever an 8051 application reads or writes a 16-bit or 32-bit value that is shared with the MCE processor, it should use this special “coherent data” mechanism to ensure data integrity. This mechanism need not be used for 8-bit accesses or when the 8051 is accessing data in shared RAM that is not shared with the MCE.

The coherent data mechanism also allows 32-bit reads and writes to be performed from the 8051 processor without data corruption. However, all MCE shared data is defined as 16-bit values so this operation is not described here.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

MCE COHERENT DATA (MCECD0, MCECD1, MCECD2, MCECD3)

Address:	*0xE9 (MCECD0),* *0xEA (MCECD1),* *0xEB (MCECD2,* *0xEC (MCECD3)*	*Not Bit Addressable*	*Reset value:*	*00000000b*

Since the MCE uses 16-bit addressing and the 8051 uses 8-bit addressing, all MCE data are aligned at even addresses in 8051 memory space. The MCE stores data in little-endian (Intel) byte ordering, so within each 16-bit value, the low-order byte is at the lower (even) 8051 address and the high-order byte is located at the upper (odd) 8051 address.

To write a 16-bit value to shared RAM at address *N*, use the following procedure:

1. Disable interrupts if necessary to prevent other 8051 accesses to shared memory while the operation is in progress.
2. Write the low-order byte of data to register MCECD2.
3. Write the high-order byte of data to the odd address (*N* + 1).
4. Re-enable interrupts if they were disabled at step 1.

The procedure for reading a 16-bit value from shared RAM depends on whether the address to be read is at a longword (32-bit) boundary. An address is at a longword boundary if it is evenly divisible by four. When all values are aligned at even addresses (as they are in MCE shared memory), an easy way to check for a longword boundary is to test bit 1 of the address. If bit 1 is zero, the address is at a longword boundary.

Use the following procedure to read a 16-bit value from shared RAM at address *N*:

1. Disable interrupts if necessary to prevent other 8051 accesses to shared memory while the operation is in progress.
2. Read the low-order byte from the even address (*N*).
3. If the address *N* is at a longword boundary, read the high-order byte from register MCECD1. Otherwise, read the high-order byte from register MCECD3.
4. Re-enable interrupts if they were disabled at step 1.

4.4.2 Arbitration

RAM arbitration is required because both processors (the 8051 and the MCE) can attempt simultaneous access to the shared RAM. In general, the arbiter gives the MCE sequencer precedence over the 8051. Since the 8051 program and internal data RAM are not shared the 8051 can generally execute instructions without restriction. 8051 instruction execution is held off only when the instruction accesses shared RAM and an MCE RAM access is pending or in progress. The MCE, on the other hand, fetches its instructions from shared RAM and stores all of its data there. In addition, the MCE performs many time critical operations that cannot be delayed. MCE instruction execution is held off only if an 8051 RAM access is already in progress or if the 8051 has been stalled for more than five clock cycles due to continued MCE RAM access.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4.5 MCE Processor Registers

The MCE processor registers are mapped to the shared RAM at addresses 0x8000 – 0x801F. Each register is 16 or 32 bits and appear in memory in little-endian byte order (low-order byte at the lowest address). The registers are defined as shown in Table 81 below.

Register	Size (bits)	MCE address (hex)	8051 address (hex)	Description
FLAGS	16	000	8000	Status bits, including MCE interrupt status
PC	16	001	8002	MCE program counter
ACC	16	002	8004	MCE accumulator, lower 16 bits
ACC_EXT	16	003	8006	MCE accumulator, upper 16 bits
INDR0_ADDR	16	004	8008	Used as a pointer for indirect reads/writes
INDR1_ADDR	16	005	800A	Used as a pointer for indirect reads/writes
INDR0_DATA	32	006	800C	Used as access portal for indirect data (from data RAM)
INDR1_DATA	32	008	8010	Used as access portal for indirect data (from data RAM)
INDR2_DATA	32	00A	8014	Used as access portal for indirect data (from MCE program RAM)
CTRL	16	00C	8018	Control bits, used to stop and enable the processor

Table 81. MCE Processor Registers

Most of the registers are intended for the exclusive use of the MCE processor. The 8051 application may perform the following operations on these registers:

- Read and write the high-order byte of the FLAGS register, which indicates and controls MCE interrupt status.
- Write to the control register to start or halt the MCE processor.
- While the MCE processor is halted, write to other registers to initialize their values. In particular, the 8051 must initialize the MCE program counter (PC) before starting the MCE processor.

The bits of the FLAGS register are defined as follows:

Bits 0 – 7	Internal MCE processor status flags
Bit 8	MCE interrupt pending, motor PWM cycle
Bit 9	*Reserved MCE interrupt*
Bit 10	*Reserved MCE interrupt*
Bit 11	MCE interrupt pending, A/D conversion complete
Bits 12 – 14	*Reserved MCE interrupts*
Bit 15	Internal MCE processor status flag

Refer to Section 4.6.1 for more information on MCE interrupts.

The bits of the CTRL register are defined as follows:

Bit 0	Used to control MCE execution. Write “0” to halt the processor or “1” to run.
Bit 1	Indicates MCE status while running. “0” indicates that the processor is currently running; “1” indicates that the processor is suspended until the next SYNC pulse.
Bit 2	*Reserved*
Bits 3 – 4	Indicate MCE execution status. A binary value of “00” indicates that the processor is running; a binary value of “01” indicates that the processor has been halted. Other values are reserved.
Bits 5 – 15	*Unused*

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

4.6 Interrupts from the MCE to the 8051

Three interrupt sources are defined for communication between the MCE and the 8051:

- The MCE interrupt, which is generated from the MCE motion firmware to inform the 8051 of various events related to PWM-cycle processing.
- The SYNC interrupt, which is generated from the MCE motion hardware to signal the 8051 that a SYNC pulse has occurred.
- The fault interrupt, which is generated from the MCE motion hardware when any fault occurs.

4.6.1 MCE Interrupt

The MCE motion firmware generates an MCE interrupt at each of the following events:

- At the completion of a full cycle of A/D conversion of all the analog channels, effectively every 5 PWM cycles as shown in Figure 59. In this case, bit 11 of the MCE FLAGS register is set.
- On each motor PWM cycle after the motor control calculations are completed. In this case, bit 8 of the MCE FLAGS register is set. The timing is indicated in Figure 89.

To clear an MCE interrupt, the 8051 application must write a “0” value to the corresponding interrupt bit in the MCE FLAGS register. (See Section 4.5 for MCE interrupt bit definitions in the FLAGS register.)

4.6.2 SYNC Interrupt

The SYNC interrupt is a periodic event signal generated by the MCE. Its timing is illustrated in Figure 89. This is the most important signal used for synchronization between the 8051 (CPU side) and the MCE (motion control side). An 8051 application software task that needs to pass commands to the MCE and/or receive updated data from the MCE may require specific synchronization with the MCE. This is due to the fact that MCE computation is initiated and triggered by the SYNC pulse at every PWM carrier frequency period. It is also true that six PWM outputs to the power device gate drive will occur at exactly one clock moment of the system clock (SYSCLK) at the beginning of the SYNC event. If synchronization is not implemented and the 8051 application software writes multiple data items to the MCE via the shared RAM, it is possible that some of the data are written in the previous MCE scan period while the rest of data are written in the current MCE scan period. Therefore, 8051 application software should use the SYNC signal for synchronization to insure that multiple data items are updated or read coherently within a particular scan period.

For IRMCF188, where the motor and PFC can be set to different PWM frequencies, the SYNC interrupt occurs only on the global SYNC pulse (the slower of the two). For example, if the PFC is configured for a PWM frequency of 20 KHz and the motor is configured for a PWM frequency of 10 KHz, the SYNC interrupt will occur at a frequency of 10 KHz.

No action is necessary to clear a SYNC interrupt after servicing it.

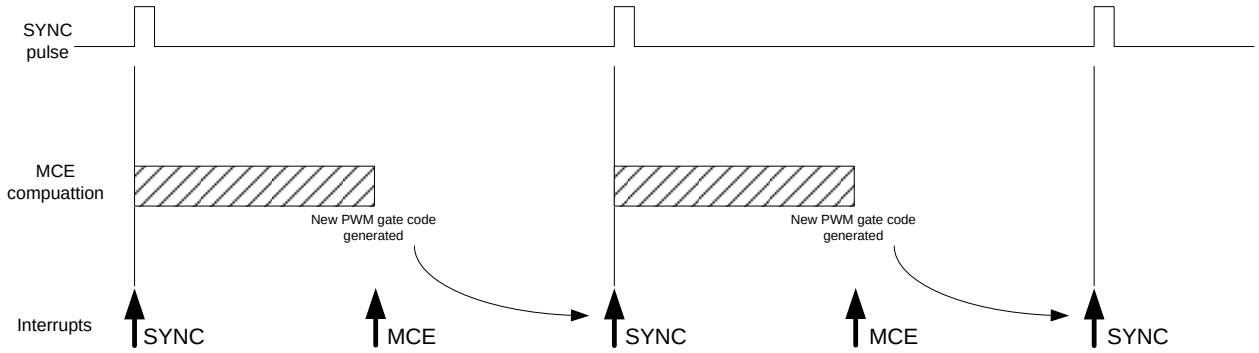


Figure 89. Timing of Sync and MCE Computation

4.6.3 Fault Interrupt

The fault interrupt is generated from the MCE motion hardware when any fault occurs, whether or not the fault has been disabled using the DisableFaults register (Section 3.4.1). Refer to Section 3.3.6 for a description of fault conditions. Note that the MCE fault described in that section is not related to the MCE interrupt types described above (Section 4.6.1). An MCE fault condition generates a fault interrupt, not an MCE interrupt.

To clear the fault interrupt, pending fault conditions must be cleared by writing to the FaultClear register (Section 3.4.1).

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

5 The MCE Development Process

This section describes how to realize an MCE design using MATLAB's Simulink graphical user interface. Motion control blocks provided by International Rectifier in the form of a Simulink library represent the available IRMCx100 series functions.

The MCE development environment consists of the following components:

- A library of graphically-represented Simulink control blocks to be used in the design of a motor control system.
- The MCE compiler, which analyzes the Simulink design and generates a corresponding file that is executed by the MCE processor on the IRMCx100 series.
- MCEDesigner, which provides a graphical user interface to the IRCMK100 series to allow download of the MCE executable file, control of MCE operation, and analysis of system function and performance. MCEDesigner is described in a separate document.

The MCE development tools software distribution is organized beneath a main directory named `MCE Compiler`. The main directory contains three subdirectories: `Simulink Library`, which contains the Simulink library blocks; `Matlab`, which contains the MATLAB scripts that implement the graphical interfaces described in Sections 5.4 and 5.5; and `bin`, which contains the executable files and linkable object files for the MCE compiler.

The modules of the Simulink library are grouped into seven main categories, with a library model file in the `Simulink Library` directory for each category. These are:

- Configuration
- Registers
- Control
- Math
- Tools
- Motion Peripherals
- Designs

Simulink library files have a `.mdl` filename extension (same as Simulink model files). For example, the Math library file is named `Math.mdl`.

The MCE development tools are designed to operate with MATLAB version 7.4 and Simulink version 6.6. They should operate properly with newer versions, but may not function correctly with older versions of MATLAB and Simulink.

5.1 MCE Design Generation

A Simulink model (`.mdl`) file defines a graphical Simulink model, or design, using a proprietary syntax in text format. The basic elements of the definition syntax are Systems, Blocks, Ports and Lines. A System is a functional collection of Blocks and Lines. A Block is an individual design component or a representation of a subsystem. Ports define the inputs and outputs of a Block or a System, and Lines are the connections between Blocks. Using a Block to represent a subsystem enables the creation of a hierarchical, or layered, design.

The MCE compiler analyzes the graphical elements defined in a model file to generate MCE instructions to implement the represented design. The compiler processes a model file that represents a complete IRMCx100 series system.

The MCE compiler analyzes a Simulink model file and uses information in the database to determine inputs and outputs for each Block and an execution sequence for the Blocks. It then creates an MCE executable file and the following optional output files:

- A register map file that can be imported into MCEDesigner so host read and write registers defined in the design can be accessed through MCEDesigner at runtime.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

- A header file in C source code format that defines the host read and write registers so they can be accessed from an 8051 application resident on the IRMCx100 series.

5.2 Creating an MCE Design Using Simulink

This section describes how to create, test and compile MCE designs in the MATLAB/Simulink environment.

Before You Start

The very first time you use the MCE design tools with MATLAB, you need to create a MATLAB search path for MCE so that MATLAB knows where to find the MCE Libraries and utilities. To set the search path, you'll need to know the location of the main MCE directory within your iMOTION software installation. (The default path is `C:\Program Files\iMOTION\MCE Compiler`, but a different location can be selected during installation.) If you're not sure where the software is installed on your computer, open an MS-DOS command prompt window and type the following command:

```
echo %MCEBASE%
```

This command displays the full pathname of the MCE base directory.

To set the search path, start MATLAB and select *Set Path…* from the *File* menu. In the *Set Path* dialog box, click the *Add Folder…* button and browse for the main MVP directory. Click *OK* in the *Browse for Folder* dialog box and then click *Save* in the *Set Path* dialog box. Click *Close* to close the dialog box. (If you don't click *Save* before you click *Close*, you'll need to add the search path again next time you run MATLAB.)

Step 1.
Start MATLAB, and in the MATLAB command window, type `mceinit100` to open the MCE Simulink Libraries. Open the standard libraries supplied with Simulink by typing `simulink` in the command window.

Step 2.
Create a new Simulink model file with the appropriate MCE subsystem hierarchy. The easiest way to do this is to make a copy of the model file template171.mdl in the main MCE directory and open it in MATLAB. If you want to create your own MCE model template, refer to the description in Section 5.8.

Step 3.
Compose the design of each control loop subsystem within your model. You can drag and drop blocks from the MCE libraries into the control loop subsystems. (Do not add blocks to the top level or the PWM subsystems.) Use Simulink's graphical design features to arrange, size and connect the blocks appropriately. To document your design you can add annotations and, if you wish, assign a descriptive name to each line and block. Refer to Section 5.3 for more information about the MCE library blocks and other design components.

Step 4.
Customize your read and write register blocks. Write register blocks define configurable parameters that you want to be able to set through the host interface at runtime. Read register blocks define output values that you want to be able to view through the host interface. To customize a register block, double click it. In the ***Parameters*** section of the ***Mask Parameters*** dialog box, follow the prompts to enter the desired values. This information is exported to MCEDesigner.

Step 5.
When you are satisfied with your Simulink design, it's time to run the compiler. This procedure is detailed in Section 5.4.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

5.3 Simulink MCE Design Components

This section describes the components of an MCE Simulink design. Most of your design components will be taken from the MCE library, but some components of the standard Simulink library are also used.

5.3.1 The MCE Library

The main window of MCE Simulink library is shown in Figure 90.

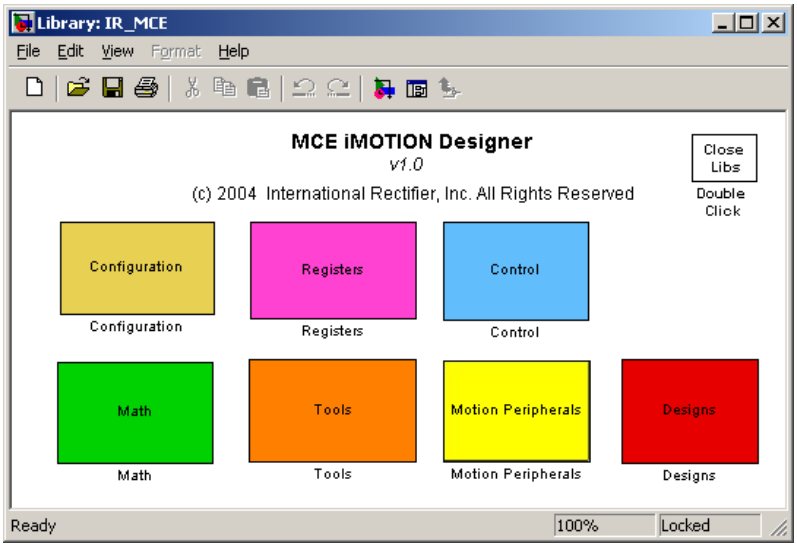


Figure 90. MCE Simulink Library

There are seven library groups, described below.

5.3.1.1 Configuration

The Configuration group contains the Configure PWM and Configure Control Loop blocks that are used in the formation of the MCE hierarchical design for a complete system. If you create your system design using the MCE design template file `template.mdl`, these blocks are already included at the appropriate locations in the subsystem hierarchy. (See Section 5.8 for more information.)

5.3.1.2 Registers

The Registers group contains host read and write register blocks, which you can use in any of your control loop subsystems. If you want to define a configurable parameter that can be set from the MCEDesigner tool (or other host interface) or from an 8051 application, drag a write register block into your design and connect its output to the input of the appropriate module(s) that will use the configurable parameter. If you want to monitor a module output from MCEDesigner or an 8051 application, drag a read register block into your design and connect the module output to it.

5.3.1.3 Control

The Control group contains the special-function motion control blocks that are used to implement your motion control algorithms. You can drag these blocks into any of your control loop subsystems.

5.3.1.4 Math

The Math group contains general-purpose math blocks that you can use in any of your control loop subsystems.

5.3.1.5 Tools

The Tools group contains the MCE Compiler block, which you can add to your design to simplify access to the MCE compiler (see Section 5.4 for more information). The Tools group also includes a Host Register Summary block, which you can add to your design and use to view and modify the read and write host register blocks you've included

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

in your design (see Section 5.5 for more information) and a tool that allows you to customize the inputs and outputs of certain motion peripheral blocks (described in Section 5.6).

5.3.1.6 Motion Peripherals

The Motion Peripherals group contains the special-function motion peripheral blocks that can be included in your control loop subsystems.

5.3.1.7 Designs

The Designs group contains sample designs shipped with the product, as well as the system template design that you can copy and use as a basis for your system designs. You can add your custom system designs to this library group if you wish.

5.3.2 Standard Simulink Library Components

The standard Simulink library components described below can be included in your design. Enter `simulink` in the MATLAB command window to open the Simulink library.

5.3.2.1 Enabled Subsystem

Use this block to create PWM and control loop subsystems for your system design. If you start with the MCE design template file `template.mdl`, the appropriate subsystem blocks are already present in the design. Refer to Section 5.8 for more information about the use of the Enabled Subsystem block in the MCE design hierarchy.

5.3.2.2 Constant

Use this block to define a constant value as an input to a block in any of your control loop subsystems. Double click the constant block to set a value for the constant.

5.3.2.3 Scope

If you want a module output in a control loop subsystem to have the capability of being traced (using MCEDesigner's trace monitor feature), drag a Scope block into your design and connect the module output to it. The name you assign to the Scope block will be used in MCEDesigner so you can recognize the trace item.

5.3.2.4 Goto and From

If you need to connect elements in two different subsystems of your design, you can use a Goto block at the source of the signal and a From block at the destination. To avoid cluttering your diagram with long and circuitous lines, you can also use Goto and From blocks to connect elements at distant points within the same subsystem.

After dragging a Goto into your design, double click it to set its parameters. Set the tag field to a unique name, which is used to match the Goto with one or more From blocks. Set tag visibility to "global" if any matching From blocks are in other subsystems or "local" if all matching From blocks are in the same subsystem as the Goto. (Visibility type "scoped" is not used.) Double click each From block to set its goto tag. This tag identifies the matching Goto block and must match the tag you specified in the Goto block.

Note: When a block's input is obtained from another subsystem (using a global Goto and From), The compiler cannot guarantee that the block providing the data will execute before the block using the data. Depending on the order of execution of the subsystems (which can depend on product type and configuration), the input data may be current (if the providing subsystem runs first) or one cycle old (if the providing subsystem has not yet run during the current PWM period).

5.3.2.5 Unit Delay

You can use the Unit Delay block to introduce a signal delay of one or more PWM cycles. In certain situations, a delay is required to identify a feedback signal (an input data value obtained from a previous cycle). For example, suppose an output of block A is used as an input to block B and an output from block B is used as an input to block A. Both inputs cannot be generated on the current cycle since one block must execute before the other. A Unit Delay block must be inserted in one of the two paths (between block A's output and block B's input or between block B's

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

output and block A's input) to identify which signal is obtained from a previous cycle. The compiler uses this information to sequence the blocks correctly.

After dragging a Unit Delay block into your design, double click it to set its parameters. The initial condition defines the value of the signal used for the initial cycles until stored values (from previous cycles) are available. The sample time defines the number of cycles to delay. (Note that the MCE Compiler's use of the sample time parameter differs from Simulink's definition.)

5.4 The MCE Compiler

Before You Start

The MCE compiler uses the Simulink model file as input. If your design is open in Simulink when you run the compiler, be sure to save your changes before running the compiler.

The Tools group of the MCE Simulink library contains a block called "MCE Compiler". You can also access `the compiler` by copying that block into your design and double-clicking it. If you start with one of the reference designs or the MCE design template file `template.mdl`, the MCE Compiler block is already present at the top level. When you double-click the MCE Compiler block the input screen appears as shown in Figure 91. (The "Use previously selected firmware option" checkbox is not displayed the first time you compile a design.)

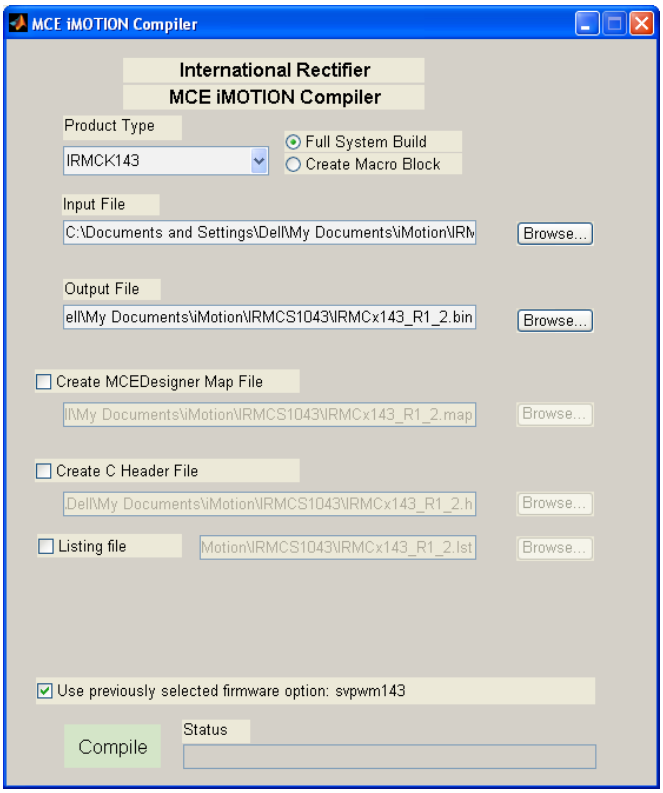


Figure 91. MCE Compiler Input Screen

Step 1.
To compile a complete system design, click the "Full System Build" radio button. Select optional output files:

- If you want the compiler to generate a register map file for use with MCEDesigner, check "create MCEDesigner Map File".
- If you want the compiler to generate C-language register definitions in a header file for use with your 8051 application, check "create C Header File".

Step 2.
Select your product type from the pulldown menu.

Step 3.
Enter the pathname of your Simulink model file in the "Upload Design file (.mdl)" edit box, or browse for the file by clicking the browse button to the right of the edit box. The location where the output files are saved can also be modified.

Step 4.
Check the "Generate listing?" checkbox if you want the compiler to generate an output text file that lists the order of block execution and all the block connections within your design. You can use this information as an aid in testing and verifying your design.

Step 5
If the "Use previously selected firmware option" checkbox is displayed, leave it checked unless you want the compiler to link with a different firmware version than you used the last time you compiled.

Step 6.
The compiler checks your model file for a Model Info block and selects a firmware option as described in Section 5.7. If there are multiple options are available and the "Use previously selected firmware option" is not checked (or not displayed), the compiler displays the Firmware Selection window similar to the example shown in Figure 92. Select the firmware option you want to use and click the **Continue** button.

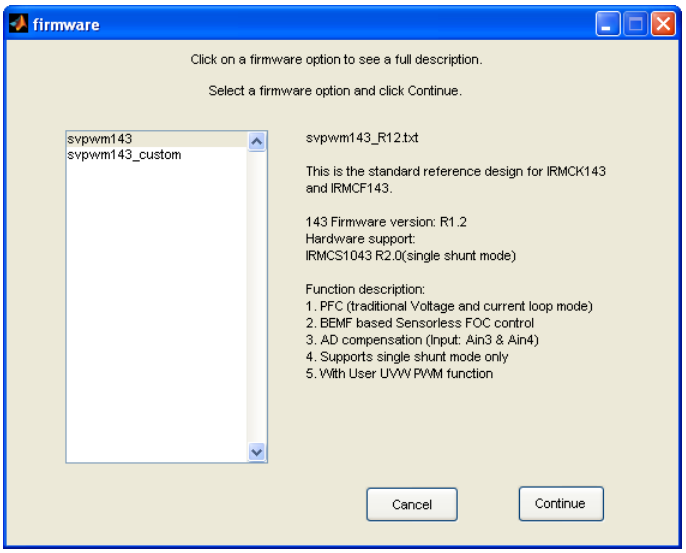


Figure 92. Example Firmware Selection Window

When compilation is complete, the MCE Compiler results window appears. An example is shown in Figure 93. To verify that the compiler linked with the firmware option you intended, look for the message "Opening system code object file…" in the results window and verify that the firmware name is correct.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

If the results window is not displayed, check the "Status" information at the bottom of the main window for an error message. Many errors are caused by a mismatch between selected product type and firmware option. Check to be sure you selected the proper product type for your design.

Time Estimates

The compiler output includes execution time estimates (in system clock cycles) for each control loop as well as the total size of the MCE program and data. You should review this information carefully. The compiler displays a warning message if your code and/or data is too large to fit in the available memory. However, the compiler cannot warn you if the execution time of your control loops is too long, because the time available for control loop execution depends on the PWM frequencies configured at run time. Enter the time estimates into MCE Wizard to check that the execution time is not too long.

Note that the compiler produces worst-case time estimates based on cycle counts for all MCE instructions it generates, including those that may be executed only under certain conditions. The execution time estimates documented for each block in Section 3 of this document are more accurate and provide a range of cycle counts when execution time varies depending on conditions. For this reason, the compiler's execution time estimate will generally exceed the estimate you would obtain by summing the documented execution times for each block in the design.

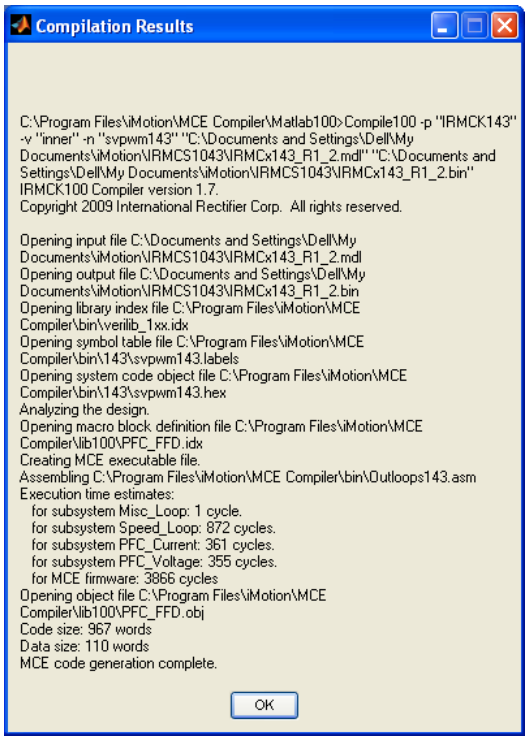


Figure 93. MCE Compiler Results Example

5.5 The Host Register Summary Utility

The Tools group of the MCE Simulink library contains a block called "Host Register Summary". This utility allows you to view a list of the host read and write registers in your design. To use it, you must first drag the Host Register Summary block into your design. If you start with the MCE design template file `template171.mdl`, the Host Register Summary block is already present at the top level.

Once you have added the block to your design, double-click the block to display a summary of your host read and write registers. If you click on a register in the list, you can view and modify the register settings.

The main window of the Host Register Summary utility is shown in Figure 94.

The list box in the top section of the main window lists the full "path" of all the registers in your design. The path identifies the model name and the subsystem in which the register is defined in addition to the register name. In the example, the path of the selected register is "Sample/Motor1/Speed Loop/TargetDir". This means that the model name is "Sample," the register is defined in PWM subsystem "Motor1" and control loop subsystem "Speed Loop." The register name is "TargetDir."

The detailed information in the lower section of the window shows the settings defined for the register that's selected in the list box. (Just click on a register to select it.) You can modify any of the settings except the register type (read or write). Changes take effect as soon as they are entered.

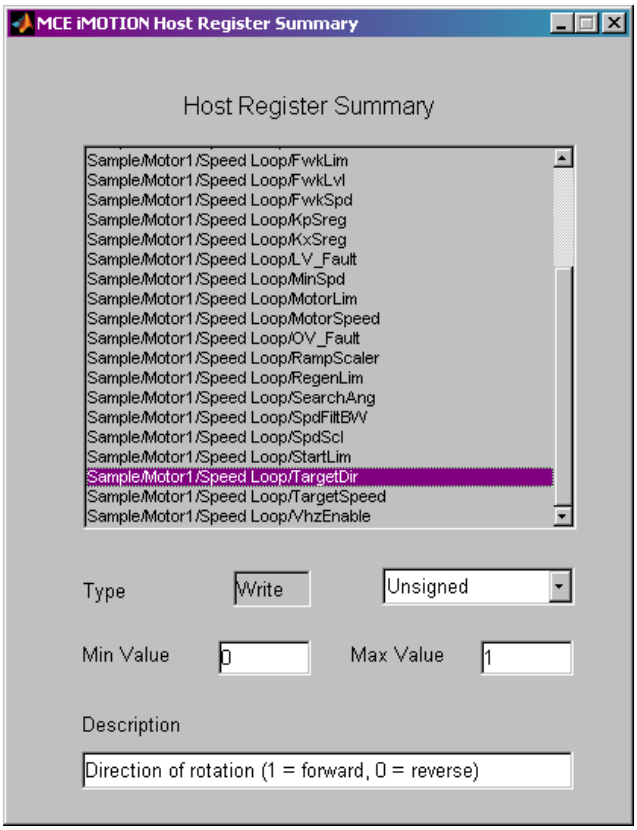


Figure 94. The Host Register Summary Utility

5.6 Customizing Motion Peripheral Library Blocks

The CustomMotPer tool allows you to modify the inputs and outputs of certain motion peripheral library blocks. You can add and remove inputs and outputs selecting from lists of available signals.

To customize a motion peripheral block, first drag it from the library into your design. Then drag the CustomMotPer block from the Tools library into your design and double-click it.

When you double-click the CustomMotPer block, it starts the Customize Motion Peripheral Block GUI, as shown in Figure 95. The GUI has a single screen, at the top of which is a pull-down list of the customizable blocks in your design. Once you've selected the block you want to customize, the current-defined inputs for the block are shown in the list on the left-hand side of the window and the currently-defined outputs are shown on the right.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

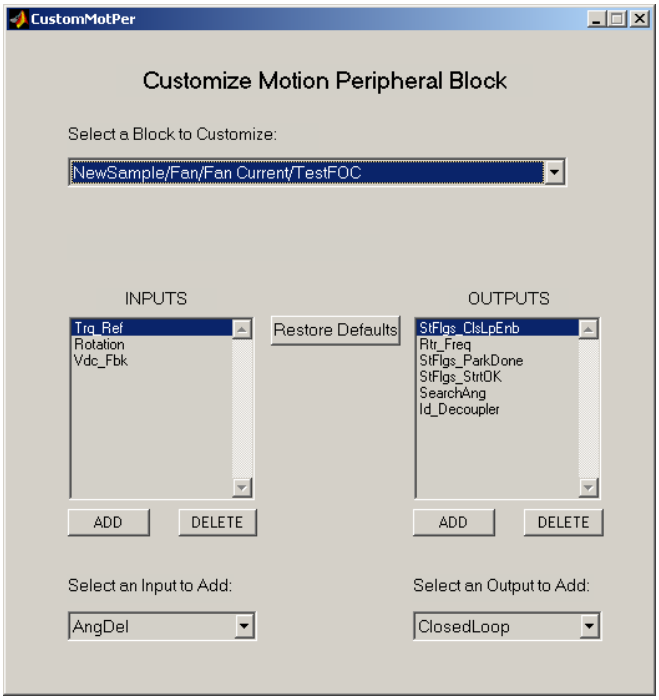


Figure 95. The CustomMotPer Utility

Summary of the display:

- The pull-down list labeled "Select a Block to Customize" lets you choose any one of the customizable blocks in the design that's currently open in Simulink.
- The Inputs and Outputs list boxes show the inputs and outputs (respectively) that are currently defined for the selected block.
- The pull-down list labeled "Select an Input to Add" lets you choose from a list of inputs available for addition to the selected block.
- The pull-down list labeled "Select an Output to Add" lets you choose from a list of outputs available for addition to the selected block.
- Click the ADD button after selecting an input or output from the appropriate "available" list.
- Click the DELETE button after selecting an existing input or output.
- Click the Restore Defaults button to restore the entire block (inputs and outputs) to the standard default settings (as defined in the Motion Peripherals library).
- When you click DELETE or Restore Defaults, a confirmation message with CANCEL and OK buttons is displayed in red in the upper portion of the window. Click the CANCEL button to abort the operation or OK to proceed.

To delete an existing input or output:
In the Inputs or Outputs list box, click on the item you want to delete and then click the DELETE button. In the upper part of the window, click the red OK button to confirm the operation.

To add a new input or output:
Select an available input or output from the appropriate pull-down list. Click the ADD button to add the new input/output.

To restore the default inputs and outputs:
Click the Restore Defaults button. In the upper part of the window, click the red OK button to confirm the operation. This restores all inputs and outputs to the default configuration. (You can't restore only inputs or only outputs.)

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Once you've customized a block in your design, you can copy it to another location in the design (if the block is intended to be used once for each motor) or drag it into another design. For blocks that can be used once for each motor, you can customize each usage of the block with different inputs and outputs.

5.7 Providing Information about Compatible Firmware Options

By default, the MCE Compiler links your system design with the standard MCE firmware for the product type you specify when you compile. If your design interfaces with a different version of the firmware or can support more than one firmware version, you can provide a list of compatible firmware options in a Model Info block (a component of the standard Simulink library) and the compiler uses the option information to link with the appropriate firmware at compile time.

How to Specify Firmware Options in the Model Info Block:
The Model Info block allows you to enter descriptive text in a free format. If you want to add one or more firmware options to the Model Info block for the compiler's use, you must use a defined format for the list so the compiler can recognize the information. Enter the firmware option information as follows: Below any descriptive information and variables you would like to include, enter a line that reads "Compatible Firmware:" (be sure to include the colon). Then on the following lines, list the names of the compatible firmware options, one option on each line. The firmware name is the object filename without the file extension. For example, the standard firmware object file for the IRMCx143 is svpwm143.hex, so the firmware name would be entered as "svpwm143". An example model info block listing two firmware options is shown in
.

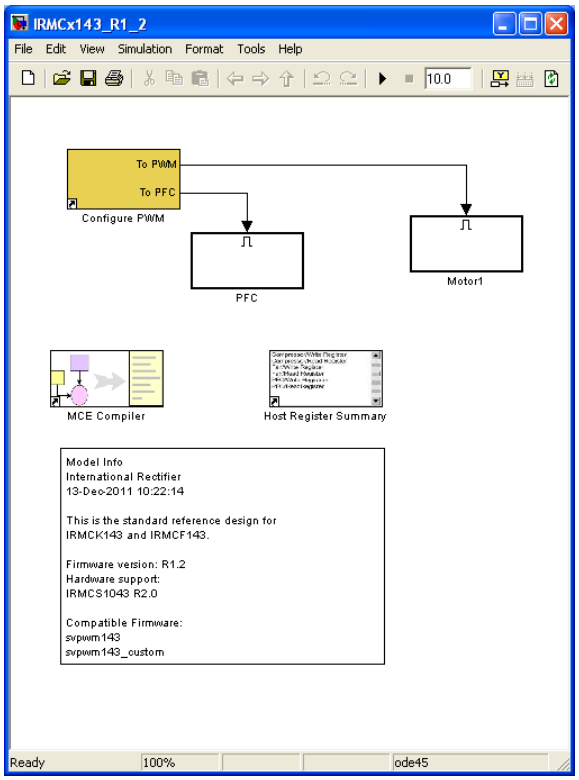


Figure 96. Example Model Info Block Showing Two Firmware Options

Where to Find the Firmware Files:
The MCE Compiler is located within the installation (typically Program Files\iMotion) in the folder MCE Compiler\bin. The firmware object files are located in device subfolders underneath MCE Compiler\bin, where each subfolder is named according to the last three digits of an IRMCx1xx part number (171, 143, etc.). When you run the

MCE Compiler (see Section 5.4) you specify a product type and the compiler uses firmware only from the corresponding device subfolder.

How the MCE Compiler Chooses a Firmware Option:
After you click the Compile button the MCE Compiler reads the specified model file and selects a firmware option as follows:

- The compiler uses the default firmware for the specified product type if:
  - There is no Model Info block in the design;
  - There is a Model Info block but it does not contain a "Compatible Firmware:" list; or
  - There is a "Compatible Firmware:" list but no matching firmware files are found in the device subfolder for the selected product type.
- If the "Compatible Firmware:" list contains only one entry and an object file matching that entry is found in the appropriate device subfolder, the compiler uses the firmware option specified in the list.
- If the "Compatible Firmware:" list contains multiple entries with matching object files in the appropriate device subfolder, the compiler displays a new window listing the valid firmware options and allows you to select the desired option.
- If the saved compilation history (docompile.bat file in the MATLAB working directory) specifies a firmware option, the compiler uses the previously-selected option unless you uncheck the "Use previously selected firmware option" checkbox.

5.8 MCE Design Hierarchical Format

This section describes the hierarchical structure of a complete MCE system and provides instructions for creating a new MCE model template

The MCE design hierarchy has the structure shown in Figure 97.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

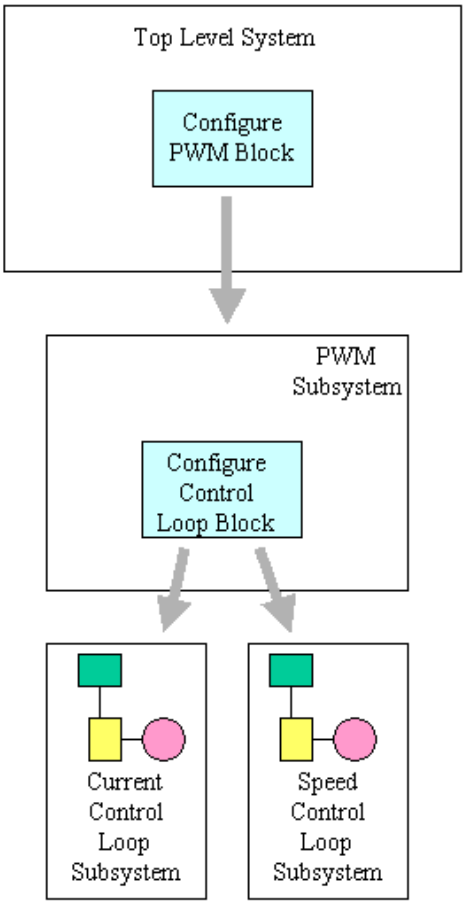


Figure 97. MCE Design Hierarchy

The top level of the system design contains a ***Configure PWM*** block and one PWM subsystem block, which is implemented using a standard Simulink ***Enabled Subsystem*** block. The Configure PWM block has one output. The Configure PWM block output must be connected to the Enable input of the PWM subsystem block. There are no other blocks or connections at the top level of the design.

The PWM subsystem contains a ***Configure Control Loop*** block and two control loop subsystem blocks, which are implemented using standard Simulink ***Enabled Subsystem*** blocks. The Configure Control Loop block has three outputs, labeled ***Current***, ***Speed*** and ***Voltage***. Each control loop subsystem is identified by connecting the appropriate Configure Control Loop block output to the Enable input of a Configure Control Loop subsystem block. In a standard PWM subsystem, the Current and Speed outputs are connected and the Voltage output is left unconnected. There are no other blocks or connections at the top level of the PWM subsystem.

The procedure described below can be used to create an empty MCE design in the correct hierarchical format.

Step 1.
Create a new (empty) Simulink model. (From the MATLAB ***File*** menu, select ***New*** and then ***Model***.) Right click in the new window and select ***Model Properties***. On the ***Summary*** tab of the ***Model Properties*** dialog, you can enter a text description of the design and save your name as its creator.

Step 2.
From the Configuration group of the MCE library, drag a ***Configure PWM*** block into the model. From the standard Simulink library's Subsystems group, drag an ***Enabled Subsystem*** block into the model. Connect the output of the

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

Configure PWM block to the Enable input of the subsystem block. Double click the label under the subsystem block to enter a name of your choice for the PWM subsystem.

Step 3.
Double click the PWM subsystem to open it. Delete the default input and output ports and the line that connects them. From the Configuration group of the MCE library, drag a ***Configure Control Loop*** block into the model. From the standard Simulink library's Subsystems group, drag two ***Enabled Subsystem*** blocks into the model. Connect the Current and Speed outputs of the Configure Control Loop block to the Enable input of each of the subsystem blocks. Double click the label under each subsystem block to enter a name of your choice for the control loop subsystem.

Step 4.
The hierarchical structure is now complete, and you can begin designing your motion control algorithms by adding and connecting MCE library blocks in each of the control loop subsystems.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

6 The 8051 Development Process

The IRMCx100 series includes an 8051 microprocessor that can be used for motor control applications as well as more general applications such as an interface to a user control panel.

To support MCE development, International Rectifier provides the MCEDesigner tool, which allows a user to control and monitor the operation of the MCE by reading and writing host registers. MCEDesigner has two components: a user interface running on a PC and a helper application or "agent" running on the 8051 microprocessor. The agent software is supplied as an executable image that is stored in external EEPROM and loaded to IRMCx100-series program RAM at power-up.

To assist the development of custom application software for the 8051, International Rectifier provides a number of source-code programming examples, such as interrupt handlers, UART driver and interface to the MCE shared RAM and RTL registers.

6.1 The Keil and FS2 Tools

It is strongly recommended that you use the following tools for 8051 software development:

- Keil Software PK51 Professional Developer's Kit (includes 8051 compiler, assembler, linker, debugger and uVision integrated development environment)
- First Silicon Solutions (FS2) ISA-M8051EW In-Target System Analyzer for the Mentor Graphics M8051EW Microprocessor Core. This product includes a debug pod and device driver that provides an interface between the Keil debugger and the IRMCx100 series IC for debugging.

This document and all 8051 software provided by IR for the IRMCx100 series assumes that you are using these tools. Refer to the IRMCx100 Software Developer's Guide for more information about using the Keil and FS2 tools.

6.1.1 Software Installation

Install the Keil tools first, then FS2 according to the instructions provided with the installation media. When requested to select options for FS2 setup, leave all settings at the default values.

Once you have completed the installation, run uVision and open a uVision project. Click on the Debug tab in the project settings window (Options for Target…). Click the Use radio button in the top right-hand section of the display and select Fs2/Keil ISA-M8051EW Driver from the pull-down menu. If you don't see that option in the menu, the Fs2 driver is not installed properly.

6.1.2 The Keil Compiler

The Keil optimizing Cx51 compiler includes a number of enhancements specifically for the 8051 processor and embedded programming environments. It is recommended that you read the Cx51 Compiler User's Guide carefully.

The following tips may ease your software development effort:

- While debugging, use a fairly low level of compiler optimization, such as level 3. Using high optimization levels can give unexpected results in the debugger (breakpoints not hit when expected, for example). When your code is working properly, increase the optimization level and verify operation again.
- The compiler supports several types of pointers. Avoid using generic (three-byte) pointers, which produce larger and slower code and can be confusing when viewing memory locations with the debugger.
- The compiler supports several memory models. Due to the extensive external RAM included in the IRMCx100 series, the large model should be used.
- The compiler supports special "sfr" and "sbit" keywords for accessing byte- and bit-addressable SFRs. See the IR sample file IRMCx1xx.h for SFR definitions using these keywords.
- The special "interrupt" keyword must be used when defining an interrupt service routine.

This document is the property of International Rectifier and may not be copied or distributed without expressed consent.

- The 8051 has very little stack space and the compiler does not normally store function parameters and local variables on the stack. For this reason, the special "reentrant" keyword must be used to define functions that may be reentrant or called recursively.

6.1.3 Debugging

Debugging 8051 code on the IRMCx100 series requires connection of the Fs2 pod to the JTAG connector on the development board. Once the pod is connected, use the following procedure to prepare the software for a debug session. The steps must be executed in the order shown below.

1. Using the MCEProgrammer tool, program OTP or flash memory with the 8051 application to be debugged.
2. On the host PC, run the Keil uVision application.
3. Apply power to the Fs2 pod.
4. Apply power to the development board.
5. From the uVision Debug menu, select Start/Stop Debug Session.
6. Wait for the load process to complete. A progress bar is shown in the lower left corner of the uVision window. Note that, since the 8051 program is preprogrammed to OTP or flash memory, the software is not actually downloaded to the target processor during this step.
7. Set breakpoints as desired and then select Go from the Debug menu (or click the Go toolbar button).

Restrictions while debugging:

- The single step operation cannot be used unless all interrupts are disabled.
- After stopping at a breakpoint, you must disable or delete the breakpoint before continuing unless all interrupts are disabled.
- Be sure that the source and object files in your uVision development directory exactly match the executable image that you programmed to OTP or flash memory. Any differences will result in unpredictable behavior in the uVision debugger.

When you have finished debugging, select Stop Running from the Debug menu (or click the Stop toolbar button) and then select Start/Stop Debug Session from the Debug menu. Do not power off the development board or the Fs2 pod until you've terminated the uVision debug session.

International
IOR Rectifier

IR WORLD HEADQUARTERS: 233 Kansas St., El Segundo, California 90245, Tel: (310) 252-7105
http://www.irf.com *Data and specifications subject to change without notice. 12/21/2011*

Sales Offices, Agents and Distributors in Major Cities Throughout the World.

Trademarks of Infineon Technologies AG
µHVIC™, µIPM™, µPFC™, AU-ConvertIR™, AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, CoolDP™, CoolGaN™, COOLiR™, CoolMOS™, CoolSET™, CoolSiC™, DAVE™, DI-POL™, DirectFET™, DrBlade™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, GaNpowIR™, HEXFET™, HITFET™, HybridPACK™, iMOTION™, IRAM™, ISOFACE™, IsoPACK™, LEDrivIR™, LITIX™, MIPAQ™, ModSTACK™, my-d™, NovalithIC™, OPTIGA™, OptiMOS™, ORIGA™, PowIRaudio™, PowIRStage™, PrimePACK™, PrimeSTACK™, PROFET™, PRO-SIL™, RASIC™, REAL3™, SmartLEWIS™, SOLID FLASH™, SPOC™, StrongIRFET™, SupIRBuck™, TEMPFET™, TRENCHSTOP™, TriCore™, UHVIC™, XHP™, XMC™

Trademarks updated November 2015

Other Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2015-03-25

Published by

Infineon Technologies AG

81726 München, Germany

© 2016 Infineon Technologies AG.
All Rights Reserved.

Do you have a question about this document?

Email: erratum@infineon.com

Document reference

ifx1

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie") .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.