M68HC12B Family

Data Sheet

M68HC12
Microcontrollers

M68HC12B
Rev. 9.1
07/2005

freescale.com

freescale™
semiconductor

Downloaded from Arrow.com.

Downloaded from Arrow.com.

M68HC12B Family

Data Sheet

To provide the most up-to-date information, the revision of our documents on the World Wide Web will be the most current. Your printed copy may be an earlier revision. To verify you have the latest information available, refer to:

http://freescale.com

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc., 2005. All rights reserved.

Downloaded from Arrow.com.

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Date	Revision Level	Description	Page Number(s)
June, 2001	2.0	Figure 1-7. BDM Tool Connector — Added NC (no connect) designator to pin 3	**29**
		Figure 18-16. BDM Tool Connector — Added NC designator to pin 3	**305**
		Table 14-2. Loop Mode Functions — Corrected table header, third column, from DDRS1 to DDS1	**195**
		WOMS bit description, fifth line, changed (via DDRS0/2) to (via DDS0/2)	**195**
		SSOE bit description, second line, changed DDRS7 to DDS7	**205**
		In the table notes following the SPC0 bit description, corrected bit designators from DDRS4, DDRS5, DDRS6, and DDRS7 to DDS4, DDS5, DDS6, and DDS7.	**205**
September, 2001	3.0	Table 13-3. Prescaler Selection — Added value column and updated prescale factors	**172**
		19.11 EEPROM Characteristics — Corrected minimum and maximum values for programming and erase times	**313**
April, 2002	4.0	Document type changed from Advance Information to Technical Data reflecting qualification.	N/A
		Figure 3-9. Condition Code Register (CCR) — Reset value for S bit corrected from U to 1	**62**
January, 2003	5.0	14.2.3.3 SCI Control Register 2 — Removed erroneous reference to Port S bit 3 in the definition for the transmitter enable bit (TE).	**197**
		Figure 14-20. Port S Data Register (PORTS) — Removed erroneous pin function for PS3 and PS2.	**208**
		Reformatted to meet publication standards	N/A
April, 2003	6.0	19.2 Maximum Ratings — Corrected maximum values for VDD, VDDA, VDDX, and VIn	307
		19.7 ATD Maximum Ratings — Corrected maximum values for VRH and VRL	310
		Figure 19-1. Programming Voltage Envelope — Corrected maximum values for VFP and VDD	315
May, 2003	7.0	19.12.1 Programming Voltage Supply Envelope — Added subsection for clarity.	315
		19.12.2 Example V_{FP} Protection Circuitry — Added subsection for clarity.	316
July, 2003	8.0	19.2 Maximum Ratings — Updated values	307
		19.7 ATD Maximum Ratings — Updated values	310
June, 2004	9.0	Table 13-3. Prescaler Selection — Corrected prescaler factor for values 6 and 7	183
July, 2005	9.1	Updated to meet Freescale identity guidelines.	Throughout

Downloaded from Arrow.com.

List of Chapters

Chapter 1 General Description 19

Chapter 2 Register Block 39

Chapter 3 Central Processor Unit (CPU) 59

Chapter 4 Resets and Interrupts 67

Chapter 5 Operating Modes and Resource Mapping 75

Chapter 6 Bus Control and Input/Output (I/O) 85

Chapter 7 EEPROM 93

Chapter 8 FLASH EEPROM 99

Chapter 9 Read-Only Memory (ROM) 111

Chapter 10 Clock Generation Module (CGM) 113

Chapter 11 Pulse-Width Modulator (PWM) 125

Chapter 12 Standard Timer (TIM) 141

Chapter 13 Enhanced Capture Timer (ECT) Module 159

Chapter 14 Serial Interface 191

Chapter 15 Byte Data Link Communications (BDLC) 213

Chapter 16 msCAN12 Controller 243

Chapter 17 Analog-to-Digital Converter (ATD) 277

Chapter 18 Development Support 289

Chapter 19 Electrical Specifications 307

Chapter 20 Mechanical Specifications 331

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Table of Contents

Chapter 1
General Description

1.1 Introduction . . . 19
1.2 Features . . . 20
1.3 Slow-Mode Clock Divider Advisory . . . 21
1.4 Block Diagrams . . . 22
1.5 Ordering Information . . . 24
1.6 Pinout and Signal Descriptions . . . 24
1.6.1 Pin Assignments . . . 24
1.6.2 Power Supply Pins . . . 24
1.6.2.1 V_{DD} and V_{SS} . . . 24
1.6.2.2 V_{DDX} and V_{SSX} . . . 24
1.6.2.3 V_{DDA} and V_{SSA} . . . 24
1.6.2.4 V_{RH} and V_{RL} . . . 24
1.6.2.5 V_{FP} (MC68HC912B32 and MC68HC912BC32 only) . . . 27
1.6.3 Signal Descriptions . . . 27
1.6.3.1 XTAL and EXTAL . . . 27
1.6.3.2 $\overline{\text{ECLK}}$ . . . 28
1.6.3.3 $\overline{\text{RESET}}$ . . . 28
1.6.3.4 $\overline{\text{IRQ}}$ . . . 28
1.6.3.5 $\overline{\text{XIRQ}}$ . . . 29
1.6.3.6 SMODN, MODA, and MODB . . . 29
1.6.3.7 BKGD . . . 29
1.6.3.8 ADDR15–ADDR0 and DATA15–DATA0 . . . 30
1.6.3.9 R/$\overline{\text{W}}$ . . . 30
1.6.3.10 $\overline{\text{LSTRB}}$ . . . 30
1.6.3.11 IPIPE1 and IPIPE0 . . . 30
1.6.3.12 $\overline{\text{DBE}}$ . . . 30
1.6.4 Port Signals . . . 32
1.6.4.1 Port A . . . 32
1.6.4.2 Port B . . . 33
1.6.4.3 Port E . . . 33
1.6.4.4 Port DLC . . . 34
1.6.4.5 Port CAN . . . 34
1.6.4.6 Port AD . . . 34
1.6.4.7 Port P . . . 35
1.6.4.8 Port T . . . 35
1.6.4.9 Port S . . . 35
1.6.5 Port Pullup, Pulldown, and Reduced Drive . . . 36

Downloaded from Arrow.com.

Chapter 2
Register Block

2.1 Introduction . . . 39
2.2 Registers . . . 40

Chapter 3
Central Processor Unit (CPU)

3.1 Introduction . . . 59
3.2 Programming Model . . . 59
3.3 CPU Registers . . . 60
3.3.1 Accumulators A and B . . . 60
3.3.2 Accumulator D . . . 60
3.3.3 Index Registers X and Y . . . 61
3.3.4 Stack Pointer . . . 61
3.3.5 Program Counter . . . 62
3.3.6 Condition Code Register . . . 62
3.4 Data Types . . . 63
3.5 Addressing Modes . . . 63
3.6 Indexed Addressing Modes . . . 64
3.7 Opcodes and Operands . . . 65

Chapter 4
Resets and Interrupts

4.1 Introduction . . . 67
4.2 Exception Priority . . . 67
4.3 Maskable Interrupts . . . 67
4.4 Latching of Interrupts . . . 70
4.5 Interrupt Control and Priority Registers . . . 70
4.5.1 Interrupt Control Register . . . 70
4.5.2 Highest Priority I Interrupt Register . . . 71
4.6 Resets . . . 71
4.6.1 Power-On Reset (POR) . . . 71
4.6.2 External Reset . . . 71
4.6.3 Computer Operating Properly (COP) Reset . . . 71
4.6.4 Clock Monitor Reset . . . 72
4.7 Effects of Reset . . . 72
4.7.1 Operating Mode and Memory Map . . . 72
4.7.2 Clock and Watchdog Control Logic . . . 72
4.7.3 Interrupts . . . 72
4.7.4 Parallel Input/Output (I/O) . . . 72
4.7.5 Central Processing Unit (CPU) . . . 72
4.7.6 Memory . . . 72
4.7.7 Other Resources . . . 73
4.8 Interrupt Recognition . . . 73

Downloaded from Arrow.com.

Chapter 5
Operating Modes and Resource Mapping

5.1 Introduction . . . 75
5.2 Operating Modes . . . 75
5.2.1 Normal Operating Modes . . . 76
5.2.1.1 Normal Expanded Wide Mode . . . 76
5.2.1.2 Normal Expanded Narrow Mode . . . 76
5.2.1.3 Normal Single-Chip Mode . . . 76
5.2.2 Special Operating Modes . . . 76
5.2.2.1 Special Expanded Wide Mode . . . 76
5.2.2.2 Special Expanded Narrow Mode . . . 76
5.2.2.3 Special Single-Chip Mode . . . 76
5.2.2.4 Special Peripheral Mode . . . 77
5.2.3 Background Debug Mode . . . 77
5.3 Internal Resource Mapping . . . 77
5.4 Mode and Resource Mapping Registers . . . 78
5.4.1 Mode Register . . . 78
5.4.2 Register Initialization Register . . . 80
5.4.3 RAM Initialization Register . . . 80
5.4.4 EEPROM Initialization Register . . . 81
5.4.5 Miscellaneous Mapping Control Register . . . 81
5.5 Memory Map . . . 83

Chapter 6
Bus Control and Input/Output (I/O)

6.1 Introduction . . . 85
6.2 Detecting Access Type from External Signals . . . 85
6.3 Registers . . . 85
6.3.1 Port A Data Register . . . 86
6.3.2 Port A Data Direction Register . . . 86
6.3.3 Port B Data Register . . . 87
6.3.4 Port B Data Direction Register . . . 87
6.3.5 Port E Data Register . . . 88
6.3.6 Port E Data Direction Register . . . 88
6.3.7 Port E Assignment Register . . . 89
6.3.8 Pullup Control Register . . . 91
6.3.9 Reduced Drive of I/O Lines . . . 92

Chapter 7
EEPROM

7.1 Introduction . . . 93
7.2 EEPROM Programmer's Model . . . 93
7.3 EEPROM Control Registers . . . 94
7.3.1 EEPROM Module Configuration Register . . . 94
7.3.2 EEPROM Block Protect Register . . . 95
7.3.3 EEPROM Test Register . . . 95
7.3.4 EEPROM Control Register . . . 96

Downloaded from Arrow.com.

Chapter 8
FLASH EEPROM

8.1 Introduction . . . 99
8.2 FLASH EEPROM Array . . . 99
8.3 FLASH EEPROM Registers . . . 99
8.3.1 FLASH EEPROM Lock Control Register . . . 100
8.3.2 FLASH EEPROM Module Configuration Register . . . 100
8.3.3 FLASH EEPROM Module Test Register . . . 100
8.3.4 FLASH EEPROM Control Register . . . 102
8.4 Operation . . . 103
8.4.1 Bootstrap Operation Single-Chip Mode . . . 103
8.4.2 Normal Operation . . . 103
8.4.3 Program/Erase Operation . . . 103
8.4.3.1 Read/Write Accesses During Program/Erase . . . 103
8.4.3.2 Program/Erase Verification . . . 104
8.4.3.3 Program/Erase Sequence . . . 104
8.5 Programming the FLASH EEPROM . . . 105
8.6 Erasing the FLASH EEPROM . . . 107
8.7 Program/Erase Protection Interlocks . . . 109
8.8 Stop or Wait Mode . . . 109
8.9 Test Mode . . . 109

Chapter 9
Read-Only Memory (ROM)

9.1 Introduction . . . 111
9.2 ROM Array . . . 111

Chapter 10
Clock Generation Module (CGM)

10.1 Introduction . . . 113
10.2 Block Diagram . . . 113
10.3 Register Map . . . 114
10.4 Clock Selection and Generation . . . 114
10.5 Slow Mode Divider . . . 116
10.6 Clock Functions . . . 116
10.6.1 Computer Operating Properly (COP) . . . 116
10.6.2 Real-Time Interrupt . . . 116
10.6.3 Clock Monitor . . . 116
10.7 Clock Registers . . . 117
10.7.1 Slow Mode Divider Register . . . 117
10.7.2 Real-Time Interrupt Control Register . . . 118
10.7.3 Real-Time Interrupt Flag Register . . . 119
10.7.4 COP Control Register . . . 119
10.7.5 Arm/Reset COP Timer Register . . . 120
10.8 Clock Divider Chains . . . 121

Downloaded from Arrow.com.

Chapter 11
Pulse-Width Modulator (PWM)

11.1 Introduction . . . 125
11.2 PWM Register Descriptions . . . 128
11.2.1 PWM Clocks and Concatenate Register . . . 128
11.2.2 PWM Clock Select and Polarity Register . . . 129
11.2.3 PWM Enable Register . . . 130
11.2.4 PWM Prescale Counter . . . 131
11.2.5 PWM Scale Register 0 . . . 131
11.2.6 PWM Scale Counter 0 Value . . . 131
11.2.7 PWM Scale Register 1 . . . 132
11.2.8 PWM Scale Counter 1 Value . . . 132
11.2.9 PWM Channel Counters 0–3 . . . 133
11.2.10 PWM Channel Period Registers 0–3 . . . 134
11.2.11 PWM Channel Duty Registers 0–3 . . . 135
11.2.12 PWM Control Register . . . 136
11.2.13 PWM Special Mode Register . . . 137
11.2.14 Port P Data Register . . . 137
11.2.15 Port P Data Direction Register . . . 138
11.3 PWM Boundary Cases . . . 138
11.4 Using the Output Compare 7 Feature to Generate a PWM . . . 138
11.4.1 PWM Period Calculation . . . 139
11.4.2 Equipment . . . 139
11.4.3 Code Listing . . . 140

Chapter 12
Standard Timer (TIM)

12.1 Introduction . . . 141
12.2 Timer Registers . . . 141
12.2.1 Timer Input Capture/Output Compare Select Register . . . 141
12.3 Block Diagram . . . 142
12.3.1 Timer Compare Force Register . . . 143
12.3.2 Output Compare 7 Mask Register . . . 143
12.3.3 Output Compare 7 Data Register . . . 143
12.3.4 Timer Count Register . . . 144
12.3.5 Timer System Control Register . . . 144
12.3.6 Timer Control Registers . . . 145
12.3.7 Timer Interrupt Mask Registers . . . 146
12.3.8 Timer Interrupt Flag Registers . . . 148
12.3.9 Timer Input Capture/Output Compare Registers . . . 149
12.3.10 Pulse Accumulator Control Register . . . 151
12.3.11 Pulse Accumulator Flag Register . . . 153
12.3.12 16-Bit Pulse Accumulator Count Register . . . 153
12.3.13 Timer Test Register . . . 154
12.3.14 Timer Port Data Register . . . 154
12.3.15 Data Direction Register for Timer Port . . . 155
12.4 Timer Operation in Modes . . . 155

Downloaded from Arrow.com.

12.5 Using the Output Compare Function to Generate a Square Wave 156
12.5.1 Sample Calculation to Obtain Period Counts 156
12.5.2 Equipment 156
12.5.3 Code Listing 156

Chapter 13
Enhanced Capture Timer (ECT) Module

13.1 Introduction 159
13.2 Basic Timer Overview 159
13.3 Enhanced Capture Timer Modes of Operation 159
13.3.1 IC Channels 160
13.3.1.1 Non-Buffered IC Channels 160
13.3.1.2 Buffered IC Channels 160
13.3.2 Pulse Accumulators 163
13.3.2.1 Pulse Accumulator Latch Mode 163
13.3.2.2 Pulse Accumulator Queue Mode 163
13.3.3 Modulus Down-Counter 163
13.4 Timer Registers 163
13.4.1 Timer Input Capture/Output Compare Select Register 166
13.4.2 Timer Compare Force Register 166
13.4.3 Output Compare 7 Mask Register 166
13.4.4 Output Compare 7 Data Register 167
13.4.5 Timer Count Registers 168
13.4.6 Timer System Control Register 168
13.4.7 Timer Control Registers 169
13.4.8 Timer Interrupt Mask Registers 171
13.4.9 Main Timer Interrupt Flag Registers 172
13.4.10 Timer Input Capture/Output Compare Registers 173
13.4.11 16-Bit Pulse Accumulator A Control Register 176
13.4.12 Pulse Accumulator A Flag Register 177
13.4.13 Pulse Accumulators Count Registers 178
13.4.14 16-Bit Modulus Down-Counter Control Register 179
13.4.15 16-Bit Modulus Down-Counter Flag Register 180
13.4.16 Input Control Pulse Accumulators Control Register 181
13.4.17 Delay Counter Control Register 181
13.4.18 Input Control Overwrite Register 182
13.4.19 Input Control System Control Register 182
13.4.20 Timer Test Register 183
13.4.21 Timer Port Data Register 184
13.4.22 Data Direction Register for Timer Port 184
13.4.23 16-Bit Pulse Accumulator B Control Register 185
13.4.24 Pulse Accumulator B Flag Register 185
13.4.25 8-Bit Pulse Accumulators Holding Registers 186
13.4.26 Modulus Down-Counter Count Registers 187
13.4.27 Timer Input Capture Holding Registers 187
13.5 Timer and Modulus Counter Operation in Different Modes 189

Downloaded from Arrow.com.

Chapter 14
Serial Interface

14.1 Introduction . . . 191
14.2 Serial Communication Interface (SCI) . . . 192
14.2.1 Data Format . . . 193
14.2.2 SCI Baud Rate Generation . . . 193
14.2.3 SCI Register Descriptions . . . 194
14.2.3.1 SCI Baud Rate Control Register . . . 194
14.2.3.2 SCI Control Register 1 . . . 195
14.2.3.3 SCI Control Register 2 . . . 197
14.2.3.4 SCI Status Register 1 . . . 198
14.2.3.5 SCI Status Register 2 . . . 199
14.2.3.6 SCI Data Register . . . 200
14.3 Serial Peripheral Interface (SPI) . . . 201
14.3.1 SPI Baud Rate Generation . . . 202
14.3.2 SPI Operation . . . 202
14.3.3 $\overline{SS}$ Output . . . 203
14.3.4 Bidirectional Mode (MOMI or SISO) . . . 204
14.3.5 SPI Register Descriptions . . . 204
14.3.5.1 SPI Control Register 1 . . . 204
14.3.5.2 SPI Control Register 2 . . . 205
14.3.5.3 SPI Baud Rate Register . . . 206
14.3.5.4 SPI Status Register . . . 207
14.3.5.5 SPI Data Register . . . 207
14.4 Port S . . . 208
14.4.1 Port S Data Register . . . 208
14.4.2 Port S Data Direction Register . . . 208
14.4.3 Pullup and Reduced Drive Register for Port S . . . 209
14.5 Serial Character Transmission using the SCI . . . 210
14.5.1 Equipment . . . 210
14.5.2 Code Listing . . . 210
14.6 Synchronous Character Transmission using the SPI . . . 211
14.6.1 Equipment . . . 211
14.6.2 Code Listing . . . 211

Chapter 15
Byte Data Link Communications (BDLC)

15.1 Introduction . . . 213
15.2 Features . . . 213
15.3 Functional Description . . . 213
15.4 BDLC Operating Modes . . . 214
15.4.1 Power Off Mode . . . 215
15.4.2 Reset Mode . . . 215
15.4.3 Run Mode . . . 215
15.5 Power-Conserving Modes . . . 215
15.5.1 BDLC Wait and CPU Wait Mode . . . 216
15.5.2 BDLC Stop and CPU Wait Mode . . . 216

Downloaded from Arrow.com.

15.5.3 BDLC Stop and CPU Stop Mode . . . 217
15.6 Loopback Modes . . . 217
15.6.0.1 Digital Loopback Mode . . . 217
15.6.0.2 Analog Loopback Mode . . . 217
15.7 BDLC MUX Interface . . . 218
15.7.1 Rx Digital Filter . . . 218
15.7.1.1 Operation . . . 218
15.7.1.2 Performance . . . 219
15.7.2 J1850 Frame Format . . . 219
15.7.2.1 SOF — Start-of-Frame Symbol . . . 219
15.7.2.2 Data — In-Message Data Bytes . . . 220
15.7.2.3 CRC — Cyclical Redundancy Check Byte . . . 220
15.7.2.4 EOD — End-of-Data Symbol . . . 220
15.7.2.5 IFR — In-Frame Response Bytes . . . 220
15.7.2.6 EOF — End-of-Frame Symbol . . . 220
15.7.2.7 IFS — Interframe Separation Symbol . . . 220
15.7.2.8 BREAK — Break . . . 221
15.7.2.9 IDLE — Idle Bus . . . 221
15.7.3 J1850 VPW Symbols . . . 221
15.7.3.1 Logic 0 . . . 222
15.7.3.2 Logic 1 . . . 222
15.7.3.3 Normalization Bit (NB) . . . 222
15.7.3.4 Break Signal (BREAK) . . . 222
15.7.3.5 Start-of-Frame Symbol (SOF) . . . 223
15.7.3.6 End-of-Data Symbol (EOD) . . . 223
15.7.3.7 End-of-Frame Symbol (EOF) . . . 223
15.7.3.8 Inter-Frame Separation Symbol (IFS) . . . 223
15.7.3.9 Idle . . . 223
15.7.4 J1850 VPW Valid/Invalid Bits and Symbols . . . 223
15.7.4.1 Invalid Passive Bit . . . 224
15.7.4.2 Valid Passive Logic 0 . . . 224
15.7.4.3 Valid Passive Logic 1 . . . 224
15.7.4.4 Valid EOD Symbol . . . 224
15.7.4.5 Valid EOF and IFS Symbols . . . 225
15.7.4.6 Idle Bus . . . 225
15.7.4.7 Invalid Active Bit . . . 226
15.7.4.8 Valid Active Logic 1 . . . 226
15.7.4.9 Valid Active Logic 0 . . . 226
15.7.4.10 Valid SOF Symbol . . . 226
15.7.4.11 Valid BREAK Symbol . . . 227
15.7.5 Message Arbitration . . . 227
15.8 BDLC Protocol Handler . . . 228
15.8.1 Protocol Architecture . . . 228
15.8.2 Rx and Tx Shift Registers . . . 228
15.8.3 Rx and Tx Shadow Registers . . . 228
15.8.4 Digital Loopback Multiplexer . . . 229

Downloaded from Arrow.com.

15.8.5 State Machine . . . 229
15.8.5.1 4X Mode . . . 229
15.8.5.2 Receiving a Message in Block Mode . . . 229
15.8.5.3 Transmitting a Message in Block Mode . . . 230
15.8.5.4 J1850 Bus Errors . . . 230
15.8.5.5 Summary . . . 231
15.9 BDLC Registers . . . 231
15.9.1 BDLC Control Register 1 . . . 231
15.9.2 BDLC Control Register 2 . . . 233
15.9.3 BDLC State Vector Register . . . 237
15.9.4 BDLC Data Register . . . 239
15.9.5 BDLC Analog Roundtrip Delay Register . . . 240
15.9.6 Port DLC Control Register . . . 241
15.9.7 Port DLC Data Register . . . 242
15.9.8 Port DLC Data Direction Register . . . 242

Chapter 16
msCAN12 Controller

16.1 Introduction . . . 243
16.2 External Pins . . . 243
16.3 Message Storage . . . 243
16.3.1 Background . . . 244
16.3.2 Receive Structures . . . 245
16.3.3 Transmit Structures . . . 246
16.4 Identifier Acceptance Filter . . . 247
16.5 Interrupts . . . 250
16.5.1 Interrupt Acknowledge . . . 250
16.5.2 Interrupt Vectors . . . 250
16.6 Protocol Violation Protection . . . 251
16.7 Low-Power Modes . . . 251
16.7.1 msCAN12 Sleep Mode . . . 252
16.7.2 msCAN12 Soft-Reset Mode . . . 253
16.7.3 msCAN12 Power-Down Mode . . . 254
16.7.4 Programmable Wakeup Function . . . 254
16.8 Timer Link . . . 254
16.9 Clock System . . . 254
16.10 Memory Map . . . 257
16.11 Programmer's Model of Message Storage . . . 257
16.11.1 Message Buffer Organization . . . 257
16.11.2 Identifier Registers . . . 258
16.11.3 Data Length Register . . . 260
16.11.4 Data Segment Registers . . . 261
16.11.5 Transmit Buffer Priority Register . . . 261
16.12 Programmer's Model of Control Registers . . . 262
16.12.1 msCAN12 Module Control Register 0 . . . 262
16.12.2 msCAN12 Module Control Register 1 . . . 263
16.12.3 msCAN12 Bus Timing Register 0 . . . 264

Downloaded from Arrow.com.

16.12.4 msCAN12 Bus Timing Register 1 . . . 265
16.12.5 msCAN12 Receiver Flag Register . . . 266
16.12.6 msCAN12 Receiver Interrupt Enable Register . . . 268
16.12.7 msCAN12 Transmitter Flag Register . . . 269
16.12.8 msCAN12 Transmitter Control Register . . . 270
16.12.9 msCAN12 Identifier Acceptance Control Register . . . 270
16.12.10 msCAN12 Receive Error Counter . . . 271
16.12.11 msCAN12 Transmit Error Counter . . . 272
16.12.12 msCAN12 Identifier Acceptance Registers . . . 272
16.12.13 msCAN12 Identifier Mask Registers . . . 273
16.12.14 msCAN12 Port CAN Control Register . . . 275
16.12.15 msCAN12 Port CAN Data Register . . . 275
16.12.16 msCAN12 Port CAN Data Direction Register . . . 276

Chapter 17
Analog-to-Digital Converter (ATD)

17.1 Introduction . . . 277
17.2 Functional Description . . . 277
17.3 ATD Registers . . . 279
17.3.1 ATD Control Register 0 . . . 279
17.3.2 ATD Control Register 1 . . . 279
17.3.3 ATD Control Register 2 . . . 279
17.3.4 ADT Control Register 3 . . . 280
17.3.5 ATD Control Register 4 . . . 281
17.3.6 ATD Control Register 5 . . . 282
17.3.7 ATD Status Registers . . . 284
17.3.8 ATD Test Registers . . . 285
17.3.9 Port AD Data Input Register . . . 286
17.3.10 ATD Result Registers . . . 286
17.4 ATD Mode Operation . . . 287
17.5 Using the ATD to Measure a Potentiometer Signal . . . 287
17.5.1 Equipment . . . 287
17.5.2 Code Listing . . . 287

Chapter 18
Development Support

18.1 Introduction . . . 289
18.2 Instruction Queue . . . 289
18.3 Background Debug Mode (BDM) . . . 290
18.3.1 BDM Serial Interface . . . 290
18.3.2 Enabling BDM Firmware Commands . . . 292
18.3.3 BDM Commands . . . 293
18.3.4 BDM Registers . . . 295
18.3.5 BDM Instruction Register . . . 296
18.3.5.1 Hardware Command . . . 296
18.3.5.2 Firmware Command . . . 297

Downloaded from Arrow.com.

18.3.6 BDM Status Register . . . 298
18.3.7 BDM Shifter Register . . . 298
18.3.8 BDM Address Register . . . 299
18.3.9 BDM CCR Holding Register . . . 299
18.4 Breakpoints . . . 299
18.4.1 Breakpoint Modes . . . 300
18.4.1.1 SWI Dual Address Mode . . . 300
18.4.1.2 BDM Full Breakpoint Mode . . . 300
18.4.1.3 BDM Dual Address Mode . . . 300
18.4.2 Breakpoint Registers . . . 301
18.4.2.1 Breakpoint Control Register 0 . . . 301
18.4.2.2 Breakpoint Control Register 1 . . . 302
18.4.2.3 Breakpoint Address Register High . . . 303
18.4.2.4 Breakpoint Address Register Low . . . 303
18.4.2.5 Breakpoint Data Register High . . . 304
18.4.2.6 Breakpoint Data Register Low Byte . . . 304
18.5 Instruction Tagging . . . 304

Chapter 19
Electrical Specifications

19.1 Introduction . . . 307
19.2 Maximum Ratings . . . 307
19.3 Functional Operating Range . . . 308
19.4 Thermal Characteristics . . . 308
19.5 5.0 Volt DC Electrical Characteristics . . . 309
19.6 Supply Current . . . 310
19.7 ATD Maximum Ratings . . . 310
19.8 ATD DC Electrical Characteristics . . . 311
19.9 Analog Converter Operating Characteristics . . . 312
19.10 ATD AC Operating Characteristics (Operating) . . . 313
19.11 EEPROM Characteristics . . . 313
19.12 FLASH EEPROM Characteristics . . . 314
19.12.1 Programming Voltage Supply Envelope . . . 315
19.12.2 Example V_{FP} Protection Circuitry . . . 316
19.13 Pulse-Width Modulator Characteristics . . . 319
19.14 Control Timing . . . 320
19.15 Peripheral Port Timing . . . 325
19.16 Multiplexed Expansion Bus Timing . . . 326
19.17 Serial Peripheral Interface (SPI) Timing . . . 328

Chapter 20
Mechanical Specifications

20.1 Introduction . . . 331
20.2 80-Pin Quad Flat Pack (Case 841B-02) . . . 332

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 1
General Description

1.1 Introduction

The MC68HC912B32, MC68HC12BE32 and MC68HC(9)12BC32, are 16-bit microcontroller units (MCUs) composed of standard on-chip peripherals. The multiplexed external bus can also operate in an 8-bit narrow mode for interfacing with single 8-bit wide memory in lower-cost systems. There is a slight feature set difference between the four pin-for-pin compatible devices as shown in Table 1-1.

Table 1-1. M68HC12B Series Feature Set Comparisons

Features	MC68HC912B32	MC68HC12BE32	MC68HC912BC32	MC68HC12BC32
CPU12	X	X	X	X
Multiplexed bus	X	X	X	X
32-Kbyte FLASH electrically erasable, programmable read-only memory (EEPROM)	X		X	
32-Kbyte read-only memory (ROM)		X		X
768-byte EEPROM	X	X	X	X
1-Kbyte random-access memory (RAM)	X	X	X	X
Analog-to-digital (A/D) converter	X	X	X	X
Standard timer module (TIM)	X		X	X
Enhanced capture timer (ECT)		X		
Pulse-width modulator (PWM)	X	X	X	X
Asynchronous serial communications interface (SCII)	X	X	X	X
Synchronous serial peripheral interface (SPI)	X	X	X	X
J1850 byte data link communication (BDLC)	X	X		
Controller area network module (CAN)			X	X
Computer operating properly (COP) watchdog timer	X	X	X	X
Slow mode clock divider	X	X	X	X
80-pin quad flat pack (QFP)	X	X	X	X
Single-wire background debug mode (BDM)	X	X	X	X

Downloaded from Arrow.com.

1.2 Features

Features include:

- 16-bit CPU12:
  - Upwardly compatible with the M68HC11 instruction set
  - Interrupt stacking and programmer's model identical to the M68HC11
  - 20-bit arithmetic logic unit (ALU)
  - Instruction queue
  - Enhanced indexed addressing
  - Fuzzy logic instructions
- Multiplexed bus:
  - Single chip or expanded
  - 16-bit by 16-bit wide or 16-bit by 8-bit narrow modes
- Memory:
  - 32-Kbyte FLASH electrically erasable, programmable read-only memory (EEPROM) with 2-Kbyte erase-protected boot block — MC68HC912B32 and MC68HC912BC32 only
  - 32-Kbyte ROM — MC68HC12BE32 and MC68HC12BC32 only
  - 768-byte EEPROM
  - 1-Kbyte random-access memory (RAM) with single-cycle access for aligned or misaligned read/write
- 8-channel, 10-bit analog-to-digital converter (ATD)
- 8-channel standard timer module (TIM) — MC68HC912B32 and MC68HC(9)12BC32 only:
  - Each channel fully configurable as either input capture or output compare
  - Simple pulse-width modulator (PWM) mode
  - Modulus reset of timer counter
- Enhanced capture timer (ECT) — MC68HC12BE32 only:
  - 16-bit main counter with 7-bit prescaler
  - Eight programmable input capture or output compare channels; four of the eight input captures with buffer
  - Input capture filters and buffers, three successive captures on four channels, or two captures on four channels with a capture/compare selectable on the remaining four
  - Four 8-bit or two 16-bit pulse accumulators
  - 16-bit modulus down-counter with 4-bit prescaler
  - Four user-selectable delay counters for signal filtering
- 16-bit pulse accumulator:
  - External event counting
  - Gated time accumulation
- Pulse-width modulator (PWM):
  - 8-bit, 4-channel or 16-bit, 2-channel
  - Separate control for each pulse width and duty cycle
  - Programmable center-aligned or left-aligned outputs

Downloaded from Arrow.com.

- Serial interfaces:
  - Asynchronous serial communications interface (SCI)
  - Synchronous serial peripheral interface (SPI)
  - J1850 byte data link communication (BDLC), MC68HC912B32 and MC68HC12BE32 only
  - Controller area network (CAN), MC68HC(9)12BC32 only
- Computer operating properly (COP) watchdog timer, clock monitor, and periodic interrupt timer
- Slow-mode clock divider
- 80-pin quad flat pack (QFP)
- Up to 63 general-purpose input/output (I/O) lines
- Single-wire background debug mode (BDM)
- On-chip hardware breakpoints

1.3 Slow-Mode Clock Divider Advisory

Current versions of the M68HC12B-series devices include a slow-mode clock divider feature. This feature is fully described in Chapter 10 Clock Generation Module (CGM). The register that controls this feature is located at $00E0. Older device mask sets do not support the slow-mode clock divider feature. This register address is reserved in older devices and provides no function.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC912B32 include: G96P, G86W, and H91F.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC12BE32 include: H54T and J38M.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC(9)12BC32 include: J15G.

Downloaded from Arrow.com.

1.4 Block Diagrams

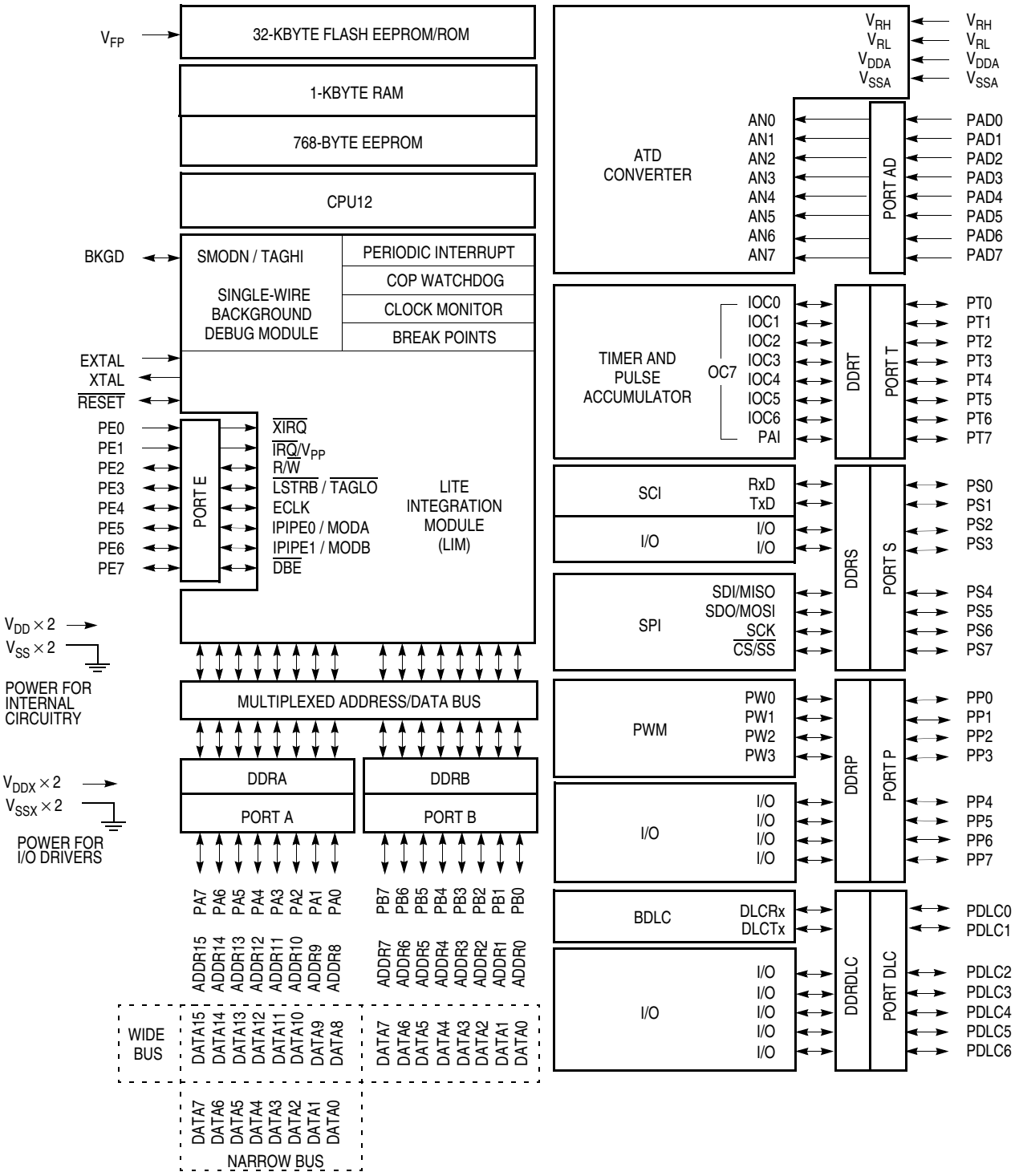


Figure 1-1. Block Diagram for MC68HC912B32 and MC68HC12BE32

Downloaded from Arrow.com.

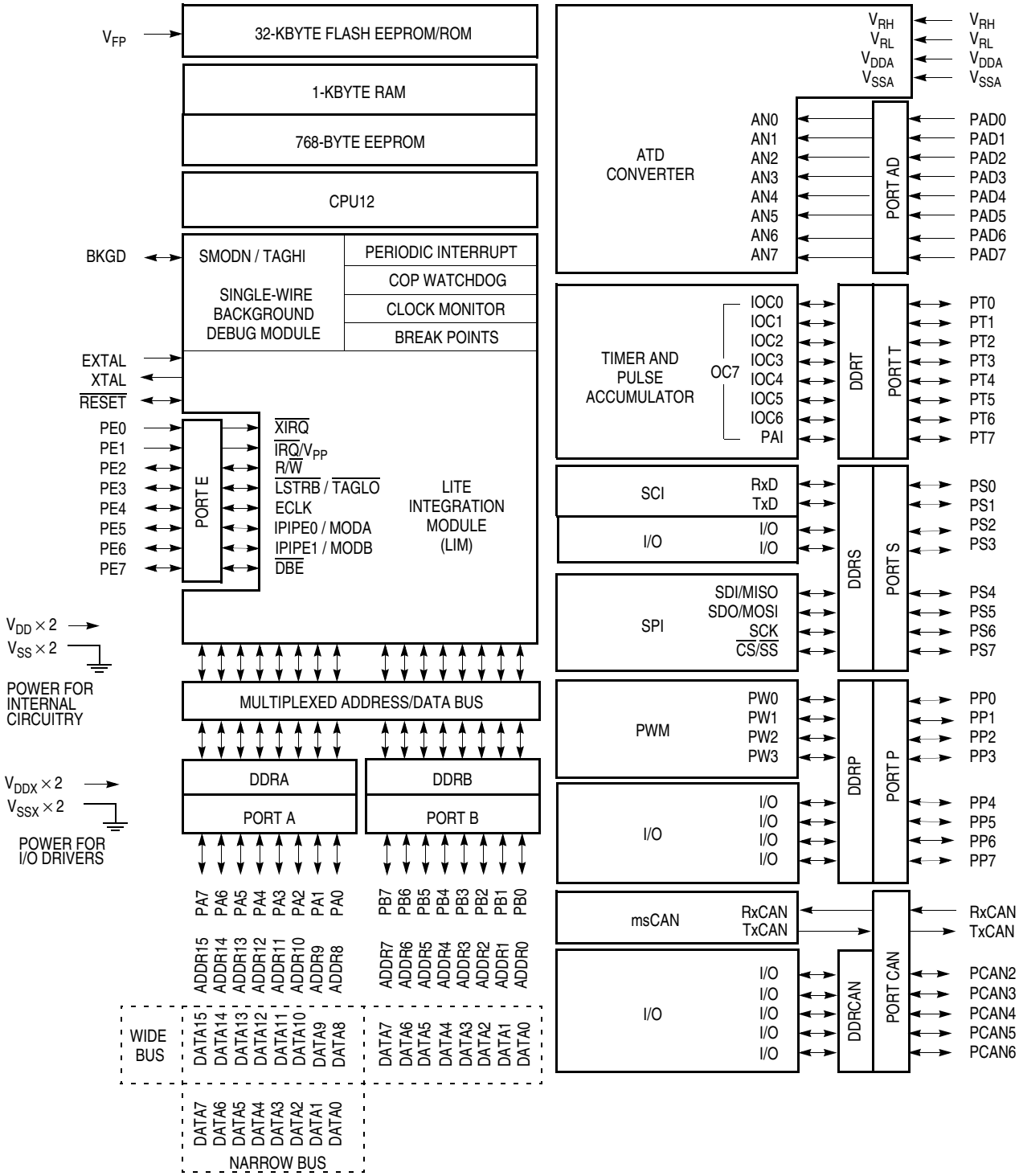


Figure 1-2. Block Diagram for MC68HC(9)12BC32

Downloaded from Arrow.com.

1.5 Ordering Information

The M68HC12B-series devices are available in 80-pin quad flat pack (QFP) packaging and are shipped in 2-piece sample packs, 84-piece trays, or 420-piece bricks.

Operating temperature range, package type, and voltage requirements are specified when ordering the specific device.

Documents to assist in product selection are available from the Freescale Literature Distribution Center or your local Freescale sales offices.

Product selection guides can also be found on the worldwide web at this URL:
http://freescale.com

Evaluation boards, assemblers, compilers, and debuggers are available from Freescale and from third-party suppliers. An up-to-date list of products that support the M68HC12 Family of microcontrollers can be found on the worldwide web at this URL:
http://freescale.com

1.6 Pinout and Signal Descriptions

1.6.1 Pin Assignments

The MCU is available in an 80-pin quad flat pack (QFP). Figure 1-3 and Figure 1-4 show the pin assignments. Most pins perform two or more functions, as described in the 1.6.3 Signal Descriptions.

1.6.2 Power Supply Pins

The MCU power and ground pins are described here and summarized in Table 1-2.

1.6.2.1 V_{DD} and V_{SS}

V_{DD} and V_{SS} are the internal power supply and ground pins. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded.

1.6.2.2 V_{DDX} and V_{SSX}

V_{DDX} and V_{SSX} are the external power supply and ground pins. Because fast signal transitions place high, short-duration current demands on the power supply, use bypass capacitors with high-frequency characteristics and place them as close to the MCU as possible. Bypass requirements depend on how heavily the MCU pins are loaded.

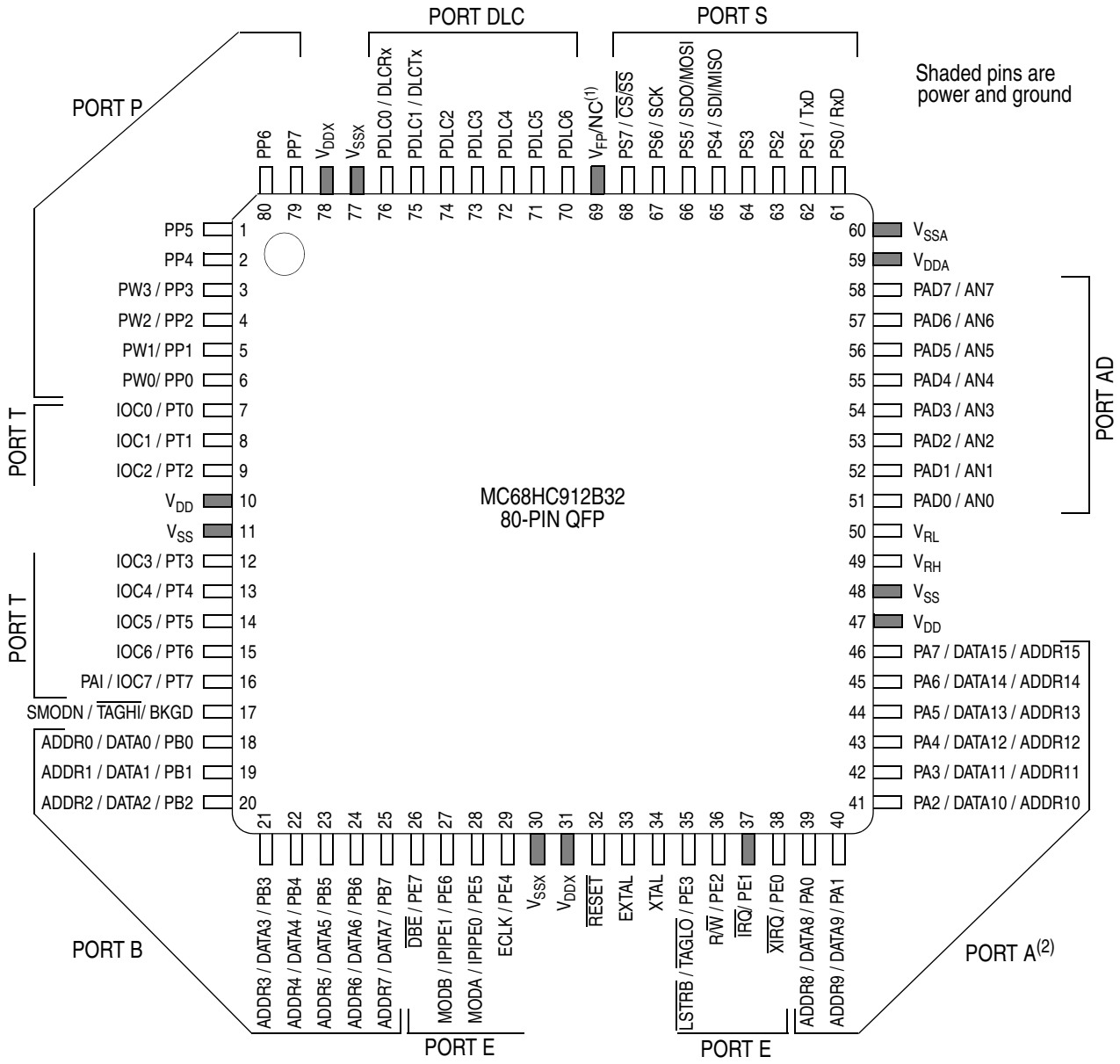
1.6.2.3 V_{DDA} and V_{SSA}

V_{DDA} and V_{SSA} are the power supply and ground pins for the analog-to-digital converter (ATD). This allows the supply voltage to be bypassed independently.

1.6.2.4 V_{RH} and V_{RL}

V_{RH} and V_{RL} are the reference voltage pins for the ATD.

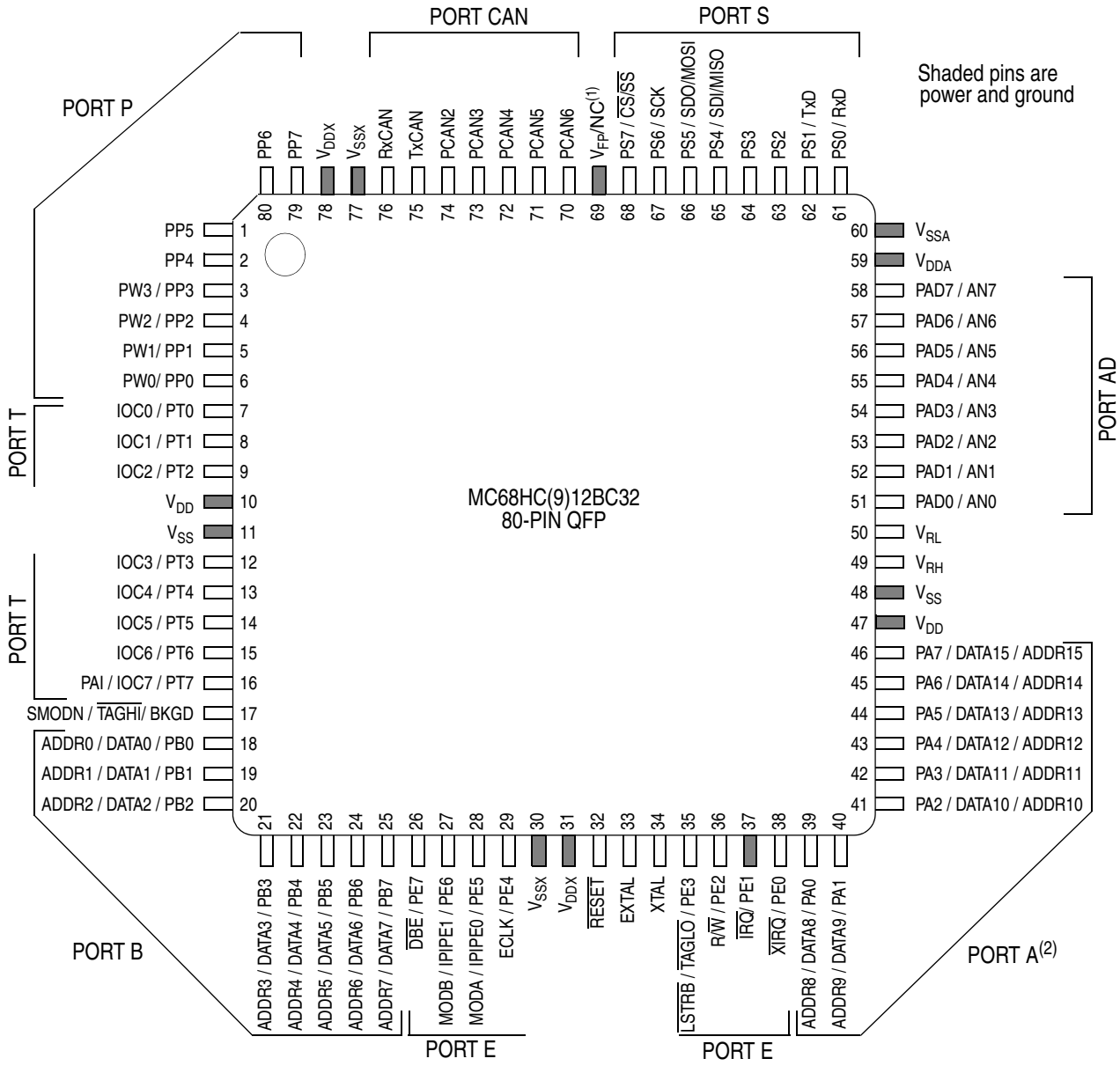
Downloaded from Arrow.com.



Notes:
1. Pin 69 is an NC (no connect) on the MC68HC12BE32.
2. In narrow mode, high and low data bytes are multiplexed in alternate bus cycles on port A.

Figure 1-3. Pin Assignments for MC68HC912B32 and MC68HC12BE32 Devices

Downloaded from Arrow.com.



Notes:
1. Pin 69 is an NC (no connect) on the MC68HC12BC32.
2. In narrow mode, high and low data bytes are multiplexed in alternate bus cycles on port A.

Figure 1-4. Pin Assignments for MC68HC(9)12BC32 Devices

Downloaded from Arrow.com.

1.6.2.5 V_{FP} (MC68HC912B32 and MC68HC912BC32 only)

V_{FP} is the FLASH EEPROM programming voltage and supply voltage during normal operation for the MC68HC912B32 and MC68HC912BC32 only.

Table 1-2. Power and Ground Connection Summary

Mnemonic	Pin Number	Description
V_{DD}	10, 47	Internal power and ground
V_{SS}	11, 48	
V_{DDX}	31, 78	External power and ground supply to pin drivers
V_{SSX}	30, 77	
V_{DDA}	59	Operating voltage and ground for the ATD; allows the supply voltage to be bypassed independently
V_{SSA}	60	
V_{RH}	49	Reference voltages for the analog-to-digital converter
V_{RL}	50	
V_{FP}	69	Programming voltage for the FLASH EEPROM and required supply for normal operation — MC68HC912B32 and MC68HC912BC32 only. Pin 69 is a no connect (NC) on the MC68HC12BE32 and MC68HC12BC32.

1.6.3 Signal Descriptions

The MCU signals are described here and summarized in Table 1-3.

1.6.3.1 XTAL and EXTAL

XTAL and EXTAL are the crystal driver and external clock input pins. They provide the interface for either a crystal or a CMOS compatible clock to control the internal clock generator circuitry. Out of reset the frequency applied to EXTAL is twice the desired E-clock rate. All the device clocks are derived from the EXTAL input frequency.

XTAL is the crystal output. The XTAL pin must be left unterminated when an external CMOS compatible clock input is connected to the EXTAL pin. The XTAL output is normally intended to drive only a crystal. The XTAL output can be buffered with a high-impedance buffer to drive the EXTAL input of another device.

NOTE

In all cases, take extra care in the circuit board layout around the oscillator pins. Load capacitances shown in the oscillator circuits include all stray layout capacitances. Refer to Figure 1-5 and Figure 1-6 for diagrams of oscillator circuits.

Downloaded from Arrow.com.

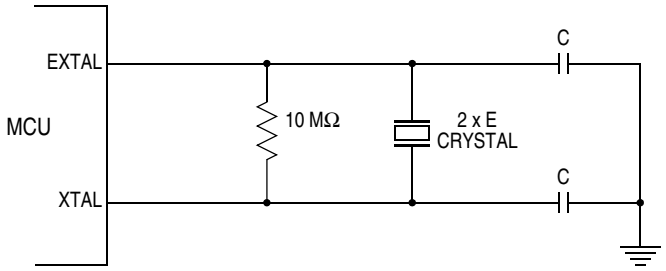


Figure 1-5. Common Crystal Connections

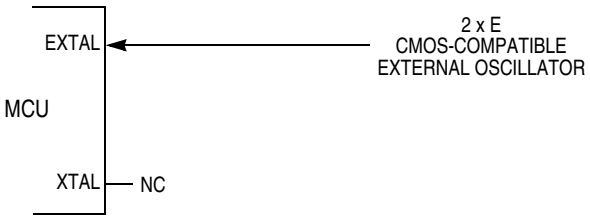


Figure 1-6. External Oscillator Connections

1.6.3.2 ECLK

ECLK is the output connection for the internal bus clock and is used to demultiplex the address and data and is used as a timing reference. ECLK frequency is equal to one half the crystal frequency out of reset.

In normal single-chip mode, the E-clock output is off at reset to reduce the effects of radio frequency interference (RFI), but it can be turned on if necessary.

In special single-chip mode, the E-clock output is on at reset but can be turned off.

In special peripheral mode, the E clock is an input to the MCU.

All clocks, including the E clock, are halted when the MCU is in stop mode. It is possible to configure the MCU to interface to slow external memory. ECLK can be stretched for such accesses.

1.6.3.3 $\overline{\text{RESET}}$

An active-low, bidirectional control signal, $\overline{\text{RESET}}$ is an input to initialize the MCU to a known startup state. It also acts as an open-drain output to indicate that an internal failure has been detected in either the clock monitor or COP watchdog circuit. The MCU goes into reset asynchronously and comes out of reset synchronously. This allows the part to reach a proper reset state even if the clocks have failed, while allowing synchronized operation when starting out of reset.

It is possible to determine whether a reset was caused by an internal source or an external source. An internal source drives the pin low for 16 cycles; eight cycles later, the pin is sampled. If the pin has returned high, either the COP watchdog vector or clock monitor vector is taken. If the pin is still low, the external reset is determined to be active and the reset vector is taken. Hold reset low for at least 32 cycles to assure that the reset vector is taken in the event that an internal COP watchdog timeout or clock monitor fail occurs.

1.6.3.4 $\overline{\text{IRQ}}$

$\overline{\text{IRQ}}$ is the maskable external interrupt request pin. It provides a means of applying asynchronous interrupt requests to the MCU. Either falling edge-sensitive triggering or level-sensitive triggering is program

Downloaded from Arrow.com.

selectable (interrupt control register, INTCR). $\overline{IRQ}$ is always configured to level-sensitive triggering at reset. When the MCU is reset, the $\overline{IRQ}$ function is masked in the condition code register.

This pin is always an input and can always be read. In special modes, it can be used to apply external EEPROM V_{PP} in support of EEPROM testing. External V_{PP} is not needed for normal EEPROM program and erase cycles. Because the $\overline{IRQ}$ pin is also used as an EEPROM programming voltage pin, there is an internal resistive pullup on the pin.

1.6.3.5 $\overline{XIRQ}$

$\overline{XIRQ}$ is the non-maskable external interrupt pin. It provides a means of requesting a non-maskable interrupt after reset initialization. During reset, the X bit in the condition code register (CCR) is set and any interrupt is masked until MCU software enables it. Because the $\overline{XIRQ}$ input is level sensitive, it can be connected to a multiple-source wired-OR network. This pin is always an input and can always be read. There is an active pullup on this pin while in reset and immediately out of reset. The pullup can be turned off by clearing the PUPE bit in the pullup control register (PUCR). $\overline{XIRQ}$ is often used as a power loss detect interrupt.

When $\overline{XIRQ}$ or $\overline{IRQ}$ are used with multiple interrupt sources ($\overline{IRQ}$ must be configured for level-sensitive operation if there is more than one source of $\overline{IRQ}$ interrupt), each source must drive the interrupt input with an open-drain type of driver to avoid contention between outputs. There must also be an interlock mechanism at each interrupt source so that the source holds the interrupt line low until the MCU recognizes and acknowledges the interrupt request. If the interrupt line is held low, the MCU recognizes another interrupt as soon as the interrupt mask bit in the MCU is cleared, normally upon return from an interrupt.

1.6.3.6 SMODN, MODA, and MODB

SMODN, MODA, and MODB are the mode-select signals. Their state during reset determines the MCU operating mode. After reset, MODA and MODB can be configured as instruction queue tracking signals IPIPE0 and IPIPE1. MODA and MODB have active pulldowns during reset.

The SMODN pin can be used as BKGD or $\overline{TAGHI}$ after reset.

NOTE

To aid in mode selection, refer to Figure 1-8 and Figure 1-9. These schematics are provided as suggestive layouts only.

1.6.3.7 BKGD

BKGD is the single-wire background mode pin. It receives and transmits serial background debugging commands. A special self-timing protocol is used. The BKGD pin has an active pullup when configured as input; BKGD has no pullup control. Currently, the tool connection configuration shown in Figure 1-7 is used.

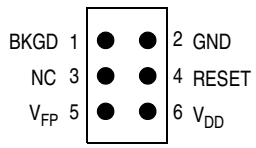


Figure 1-7. BDM Tool Connector

Downloaded from Arrow.com.

1.6.3.8 ADDR15–ADDR0 and DATA15–DATA0

ADDR15–ADDR0 and DATA15–DATA0 are the external address and data bus pins. They share functions with general-purpose I/O ports A and B. In single-chip operating modes, the pins can be used for I/O; in expanded modes, the pins are used for the external buses.

In expanded wide mode, ports A and B multiplex 16-bit data and address buses. The PA7–PA0 pins multiplex ADDR15–ADDR8 and DATA15–DATA8. The PB7–PB0 pins multiplex ADDR7–ADDR0 and DATA7–DATA0.

In expanded narrow mode, ports A and B are used for the 16-bit address bus. An 8-bit data bus is multiplexed with the most significant half of the address bus on port A. In this mode, 16-bit data is handled as two back-to-back bus cycles, one for the high byte followed by one for the low byte. The PA7–PA0 pins multiplex ADDR15–ADDR8, DATA15–DATA8, and DATA7–DATA0. The state of the address pin should be latched at the rising edge of E. To allow for maximum address setup time at external devices, a transparent latch should be used.

1.6.3.9 R/$\overline{W}$

R/$\overline{W}$ is the read/write pin. In all modes, this pin can be used as input/output (I/O) and is a general-purpose input with an active pullup out of reset. If the read/write function is required, it should be enabled by setting the RDWE bit in the port E assignment register (PEAR). External writes are not possible until enabled.

1.6.3.10 $\overline{LSTRB}$

$\overline{LSTRB}$ is the low-byte strobe pin. In all modes, this pin can be used as I/O and is a general-purpose input with an active pullup out of reset. If the strobe function is required, it should be enabled by setting the LSTRE bit in the PEAR register. This signal is used in write operations and so external low-byte writes are not possible until this function is enabled. This pin is also used as $\overline{TAGLO}$ in special expanded modes and is multiplexed with the $\overline{LSTRB}$ function.

1.6.3.11 IPIPE1 and IPIPE0

IPIPE1 and IPIPE0 are the instruction queue tracking pins. Their signals are used to track the state of the internal instruction execution queue. Execution state is time-multiplexed on the two signals.

1.6.3.12 $\overline{DBE}$

$\overline{DBE}$ is the data bus enable signal. It is an active-low signal that is asserted low during E-clock high time. $\overline{DBE}$ provides separation between output of a multiplexed address and the input of data. When an external address is stretched, $\overline{DBE}$ is asserted during what would be the last quarter cycle of the last E-clock cycle of stretch. In expanded modes, this pin is used to enable the drive control of external buses during external reads only. Use of the $\overline{DBE}$ is controlled by the NDBE bit in the PEAR register. $\overline{DBE}$ is enabled out of reset in expanded modes. This pin has an active pullup during and after reset in single-chip modes.

Downloaded from Arrow.com.

Table 1-3. Signal Description Summary

Pin Name	Pin Number	Description
PW3–PW0	3–6	Pulse-width modulator channel outputs
ADDR7–ADDR0 DATA7–DATA0	25–18	External bus pins share function with general-purpose I/O ports A and B. In single-chip modes, the pins can be used for I/O. In expanded modes, the pins are used for the external buses.
ADDR15–ADDR8 DATA15–DATA8	46–39	
IOC7–IOC0	16–12, 9–7	Pins used for input capture and output compare in the timer and pulse accumulator subsystem
PAI	16	Pulse accumulator input
AN7–AN0	58–51	Analog inputs for the analog-to-digital conversion module
$\overline{\text{DBE}}$	26	Data bus control and, in expanded mode, enables the drive control of external buses during external reads
MODB, MODA	27, 28	State of mode select pins during reset determines the initial operating mode of the MCU. After reset, MODB and MODA can be configured as instruction queue tracking signals IPIPE1 and IPIPE0 or as general-purpose I/O pins.
IPIPE1, IPIPE0	27, 28	
ECLK	29	E-clock is the output connection for the external bus clock. ECLK is used as a timing reference and for address demultiplexing.
$\overline{\text{RESET}}$	32	An active low bidirectional control signal, $\overline{\text{RESET}}$ acts as an input to initialize the MCU to a known startup state and an output when COP or clock monitor causes a reset.
EXTAL	33	Crystal driver and external clock input pins. On reset all the device clocks are derived from the EXTAL input frequency. XTAL is the crystal output.
XTAL	34	
$\overline{\text{LSTRB}}$	35	Low byte strobe (0 = low byte valid), in all modes this pin can be used as I/O. The low strobe function is the exclusive-NOR of A0 and the internal $\overline{\text{SZ8}}$ signal. The $\overline{\text{SZ8}}$ internal signal indicates the size 16/8 access.
$\overline{\text{TAGLO}}$	35	Pin used in instruction tagging
R/$\overline{\text{W}}$	36	Indicates direction of data on expansion bus; shares function with general-purpose I/O; read/write in expanded modes
$\overline{\text{IRQ}}$	37	Maskable interrupt request input provides a means of applying asynchronous interrupt requests to the MCU. Either falling edge-sensitive triggering or level-sensitive triggering is program selectable (INTCR register).
$\overline{\text{XIRQ}}$	38	Provides a means of requesting asynchronous non-maskable interrupt requests after reset initialization
BKGD	17	Single-wire background interface pin is dedicated to the background debug function. During reset, this pin determines special or normal operating mode.
TAGHI	17	Pin used in instruction tagging
DLCRx/RxCAN(1)	76	BDLC receive pin
DLCTx/TxCAN(1)	75	BDLC transmit pin
$\overline{\text{CS}}$/$\overline{\text{SS}}$	68	Slave-select output for SPI master mode; input for slave mode or master mode
SCK	67	Serial clock for SPI system
SDO/MOSI	66	Master out/slave in pin for serial peripheral interface
SDI/MISO	65	Master in/slave out pin for serial peripheral interface
TxD0	62	SCI transmit pin
RxD0	61	SCI receive pin

1. The RxCAN and TxCAN designations are for the MC68HC(9)12BC32 only.

Downloaded from Arrow.com.

1.6.4 Port Signals

The MCU incorporates eight ports which are used to control and access the various device subsystems. When not used for these purposes, port pins may be used for general-purpose I/O. In addition to the pins described here, each port consists of:

- A data register which can be read and written at any time
- With the exception of port AD and PE1–PE0, a data direction register which controls the direction of each pin

After reset, all port pins are configured as input. (Refer to Table 1-4 for a summary of the port signal descriptions.)

Table 1-4. Port Description Summary

Port Name	Pin Numbers	Data Direction DD Register (Address)	Description
Port A PA7–PA0	46–39	In/Out DDRA ($0002)	Port A and port B pins are used for address and data in expanded modes. The port data registers are not in the address map during expanded and peripheral mode operation. When in the map, port A and port B can be read or written anytime. DDRA and DDRB are not in the address map in expanded or peripheral modes.
Port B PB7–PB0	25–18	In/Out DDRB ($0003)	
Port AD PAD7–PAD0	58–51	In	Analog-to-digital converter and general-purpose I/O
Port DLC/PCAN(1) PDLC6–PDLC0 PCAN6–PCAN2	70–76	In/Out DDRDLC ($00FF)	Byte data link communication (BDLC) subsystem and general-purpose I/O
Port E PE7–PE0	26–29, 35–38	PE1–PE0 In PE7–PE2 In/Out DDRE ($0009)	Mode selection, bus control signals, and interrupt service request signals; or general-purpose I/O
Port P PP7–PP0	79, 80, 1–6	In/Out DDRP ($0057)	General-purpose I/O. PP3–PP0 are used with the pulse-width modulator when enabled.
Port S PS7–PS0	68–61	In/Out DDRS ($00D7)	Serial communications interface and serial peripheral interface subsystems and general-purpose I/O
Port T PT7–PT0	16–12, 9–7	In/Out DDRT ($00AF)	General-purpose I/O when not enabled for input capture and output compare in the timer and pulse accumulator subsystem

1. Port DLC applies to the MC68HC912B32 and MC68HC12BE32 and PCAN to the MC68HC(9)12BC32.

1.6.4.1 Port A

Port A pins are used for address and data in expanded modes. The port data register is not in the address map during expanded and peripheral mode operation. When it is in the map, port A can be read or written at anytime.

The port A data direction register (DDRA) determines whether each port A pin is an input or output. DDRA is not in the address map during expanded and peripheral mode operation. Setting a bit in DDRA makes

Downloaded from Arrow.com.

the corresponding bit in port A an output; clearing a bit in DDRA makes the corresponding bit in port A an input. The default reset state of DDRA is all 0s.

When the PUPA bit in the PUCR register is set, all port A input pins are pulled up internally by an active pullup device. This bit has no effect if the port is being used in expanded modes as the pullups are inactive.

Setting the RDPA bit in the reduced drive register (RDRIV) causes all port A outputs to have reduced drive levels. RDRIV can be written once after reset and is not in the address map in peripheral mode. Refer to Chapter 6 Bus Control and Input/Output (I/O).

1.6.4.2 Port B

Port B pins are used for address and data in expanded modes. The port data register is not in the address map during expanded and peripheral mode operation. When it is in the map, port B can be read or written at anytime.

The port B data direction register (DDRB) determines whether each port B pin is an input or output. DDRB is not in the address map during expanded and peripheral mode operation. Setting a bit in DDRB makes the corresponding bit in port B an output; clearing a bit in DDRB makes the corresponding bit in port B an input. The default reset state of DDRB is all 0s.

When the PUPB bit in the PUCR register is set, all port B input pins are pulled up internally by an active pullup device. This bit has no effect if the port is being used in expanded modes because the pullups are inactive.

Setting the RDPB bit in register RDRIV causes all port B outputs to have reduced drive levels. RDRIV can be written once after reset. RDRIV is not in the address map in peripheral mode. Refer to Chapter 6 Bus Control and Input/Output (I/O).

1.6.4.3 Port E

Port E pins operate differently from port A and B pins. Port E pins are used for bus control signals and interrupt service request signals. When a pin is not used for one of these specific functions, it can be used as general-purpose I/O. However, two of the pins, PE1 and PE0, can be used only for input, and the states of these pins can be read in the port data register even when they are used for $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$.

The PEAR register determines pin function, and the data direction register (DDRE) determines whether each pin is an input or output when it is used for general-purpose I/O. PEAR settings override DDRE settings. Because PE1 and PE0 are input-only pins, only DDRE7–DDRE2 have effect. Setting a bit in the DDRE register makes the corresponding bit in port E an output; clearing a bit in the DDRE register makes the corresponding bit in port E an input. The default reset state of DDRE is all 0s.

When the PUPE bit in the PUCR register is set, PE7, PE3, PE2, and PE0 are pulled up. PE7, PE3, PE2, and PE0 are active pulled-up devices, while PE1 is always pulled up by means of an internal resistor.

Port E and DDRE are not in the map in peripheral mode or in expanded modes when the EME bit in the MODE register is set.

Setting the RDPE bit in register RDRIV causes all port E outputs to have reduced drive level. RDRIV can be written once after reset. RDRIV is not in the address map in peripheral mode. Refer to Chapter 6 Bus Control and Input/Output (I/O).

Downloaded from Arrow.com.

1.6.4.4 Port DLC

The MC68HC912B32 and MC68HC12BE32 contain the port DLC.

Byte data link communications (BDLC) pins can be configured as general-purpose I/O port DLC. When BDLC functions are not enabled, the port has seven general-purpose I/O pins, PDLC6–PDLC0. The port DLC control register (DLCSCR) controls port DLC function. The BDLC function, enabled with the BDLCEN bit, takes precedence over other port functions.

The port DLC data direction register (DDRDLC) determines whether each port DLC pin is an input or output. Setting a bit in DDRDLC makes the corresponding pin in port DLC an output; clearing a bit makes the corresponding pin an input. After reset, port DLC pins are configured as inputs.

When the PUPDLC bit in the DLCSCR register is set, all port DLC input pins are pulled up internally by an active pullup device.

Setting the RDPDLC bit in register DLCSCR causes all port DLC outputs to have reduced drive level. Levels are at normal drive capability after reset. RDPDLC can be written anytime after reset. Refer to Chapter 15 Byte Data Link Communications (BDLC).

1.6.4.5 Port CAN

The MC68HC(9)12BC32 contains the port CAN.

The port CAN has five general-purpose I/O pins, PCAN[6:2]. The msCAN12 receive pin, RxCAN, and transmit pin, TxCAN, cannot be configured as general-purpose I/O on port CAN.

The msCAN data direction register (DDRCAN) determines whether each port CAN pin PCAN[6:2] is an input or output. Setting a bit in DDRCAN makes the corresponding pin in port CAN an output; clearing a bit makes the corresponding pin an input. After reset, port CAN pins PCAN[6:2] are configured as inputs.

When a read to the port CAN is performed, the value read from the most significant bit (MSB) depends on the MSB, PCAN7, of the port CAN data register, PORTCAN, and the MSB of DDRCAN: it is 0 if DDRCAN7 = 0 and is PCAN7 if DDRCAN7 = 1.

When the PEUCAN bit in the port CAN control register (PCTLCAN) is set, port CAN input pins PCAN[6:2] are pulled up internally by an active pullup device.

Setting the RDRCAN bit in register PCTLCAN causes the port CAN outputs PCAN[6:2} to have reduced drive level. Levels are at normal drive capability after reset. RDRCAN can be written anytime after reset. Refer to Chapter 16 msCAN12 Controller.

1.6.4.6 Port AD

Port AD provides input to the analog-to-digital subsystem and general-purpose input. When analog-to-digital functions are not enabled, the port has eight general-purpose input pins, PAD7–PAD0. The ADPU bit in the ATD control register 2 (ATDCTL2) enables the A/D function.

Port AD pins are inputs; no data direction register is associated with this port. The port has no resistive input loads and no reduced drive controls. Refer to Chapter 17 Analog-to-Digital Converter (ATD).

Downloaded from Arrow.com.

1.6.4.7 Port P

The four pulse-width modulation channel outputs share general-purpose port P pins. The PWM function is enabled with the PWM enable register (PWEN). Enabling PWM pins takes precedence over the general-purpose port. When pulse-width modulation is not in use, the port pins may be used for general-purpose I/O.

The port P data direction register (DDRP) determines pin direction of port P when used for general-purpose I/O. When DDRP bits are set, the corresponding pin is configured for output. On reset, the DDRP bits are cleared and the corresponding pin is configured for input.

When the PUPP bit in the PWM control register (PWCTL) register is set, all input pins are pulled up internally by an active pullup device. Pullups are disabled after reset.

Setting the RDPP bit in the PWCTL register configures all port P outputs to have reduced drive levels. Levels are at normal drive capability after reset. The PWCTL register can be read or written anytime after reset. Refer to Chapter 11 Pulse-Width Modulator (PWM).

1.6.4.8 Port T

This port provides eight general-purpose I/O pins when not enabled for input capture and output compare in the timer and pulse accumulator subsystem. The TEN bit in the timer system control register (TSCR) enables the timer function. The pulse accumulator subsystem is enabled with the PAEN bit in the pulse accumulator control register (PACTL).

The port T data direction register (DDRT) determines pin direction of port T when used for general-purpose I/O. When DDRT bits are set, the corresponding pin is configured for output. On reset the DDRT bits are cleared and the corresponding pin is configured for input.

When the PUPT bit in the timer mask register 2 (TMSK2) is set, all input pins are pulled up internally by an active pullup device. Pullups are disabled after reset.

Setting the RDPT bit in the TMSK2 register configures all port T outputs to have reduced drive levels. Levels are at normal drive capability after reset. The TMSK2 register can be read or written anytime after reset. For the MC68HC912B32 and MC68HC(9)12BC32, refer to Chapter 12 Standard Timer (TIM). For the MC68HC12BE32, refer to Chapter 13 Enhanced Capture Timer (ECT) Module.

1.6.4.9 Port S

Port S is the 8-bit interface to the standard serial interface consisting of the serial communications interface (SCI) and serial peripheral interface (SPI) subsystems. Port S pins are available for general-purpose parallel I/O when standard serial functions are not enabled.

Port S pins serve several functions depending on the various internal control registers. If WOMS bit in the SCI control register 1 (SC0CR1) is set, the P-channel drivers of the output buffers are disabled for bits 0–1 (2–3). If SWOM bit in the SP0CR1 register is set, the P-channel drivers of the output buffers are disabled for bits 4–7 (wired-OR mode). The open drain control affects both the serial and the general-purpose outputs. If the RDPSx bits in the PURDS register are set, the appropriate port S pin drive capabilities are reduced. If PUPSx bits in the port S pullup, reduced drive register (PURDS) are set, the appropriate pullup device is connected to each port S pin which is programmed as a general-purpose input. If the pin is programmed as a general-purpose output, the pullup is disconnected from the pin regardless of the state of the individual PUPSx bits.

Downloaded from Arrow.com.

1.6.5 Port Pullup, Pulldown, and Reduced Drive

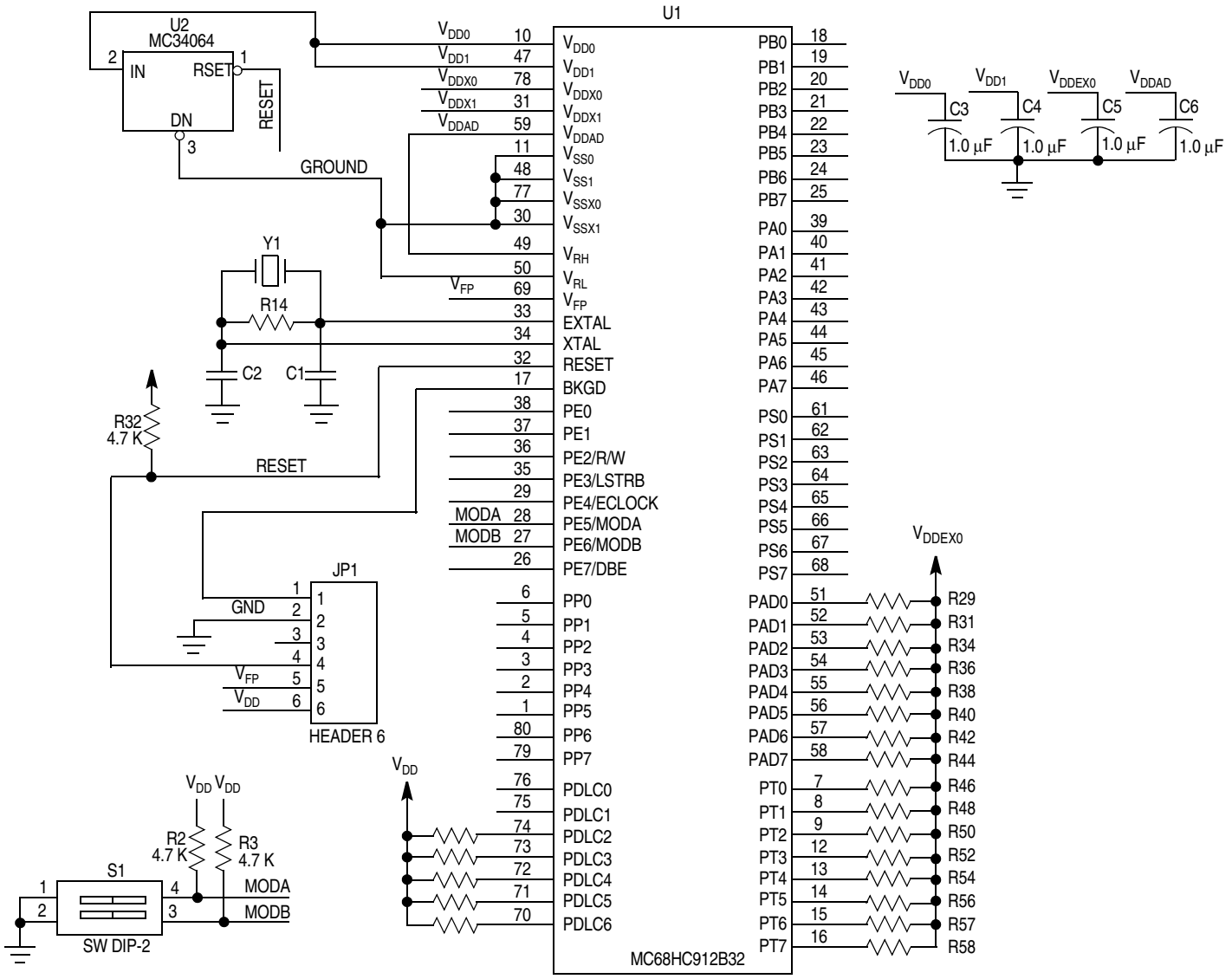
MCU ports can be configured for internal pullup. To reduce power consumption and RFI, the pin output drivers can be configured to operate at a reduced drive level. Reduced drive causes a slight increase in transition time depending on loading and should be used only for ports which have a light loading. Table 1-5 summarizes the port pullup default status and controls.

Table 1-5. Port Pullup, Pulldown, and Reduced Drive Summary

Port Name	Resistive Input Loads	Enable Bit			Reduced Drive Control Bit		
		Register (Address)	Bit Name	Reset State	Register (Address)	Bit Name	Reset State
Port A	Pullup	PUCR ($000C)	PUPA	Disabled	RDRIV ($000D)	RDPA	Full drive
Port B	Pullup	PUCR ($000C)	PUPB	Disabled	RDRIV ($000D)	RDPB	Full drive
Port E PE7, PE3, PE2, PE1, PE0	Pullup	PUCR ($000C)	PUPE	Enabled	RDRIV ($000D)	RDPE	Full drive
Port E PE4	None	—			RDRIV ($000D)	RDPE	Full drive
Port E PE6, PE5	Pulldown	Enabled during reset			—	—	—
Port P	Pullup	PWCTL ($0054)	PUPP	Disabled	PWCTL ($0054)	RDPP	Full drive
Port S PS1–PS0	Pullup	PURDS ($00DB)	PUPS0	Disabled	PURDS ($00DB)	RDPS0	Full drive
Port S PS3–PS2	Pullup	PURDS ($00DB)	PUPS1	Disabled	PURDS ($00DB)	RDPS1	Full drive
Port S PS7–PS4	Pullup	PURDS ($00DB)	PUPS2	Disabled	PURDS ($00DB)	RDPS2	Full drive
Port T	Pullup	TMSK2 ($008D)	PUPT	Disabled	TMSK2 ($008D)	RDPT	Full drive
Port DLC/PCAN[1]	Pullup	DLCSCR ($00FD)	DLCPUE	Disabled	DLCSCR ($00FD)	DLCRDV	Full drive
Port AD	None	—			—		
BKGD	Pullup	—	—	Enabled	—	—	Full drive

1. Port DLC applies to the MC68HC912B32 and MC68HC12BE32 and PCAN to the MC68HC(9)12BC32.

Downloaded from Arrow.com.



NOTE: This figure provides a suggested schematic only.

Figure 1-8. Basic Single-Chip Mode Schematic

Downloaded from Arrow.com.

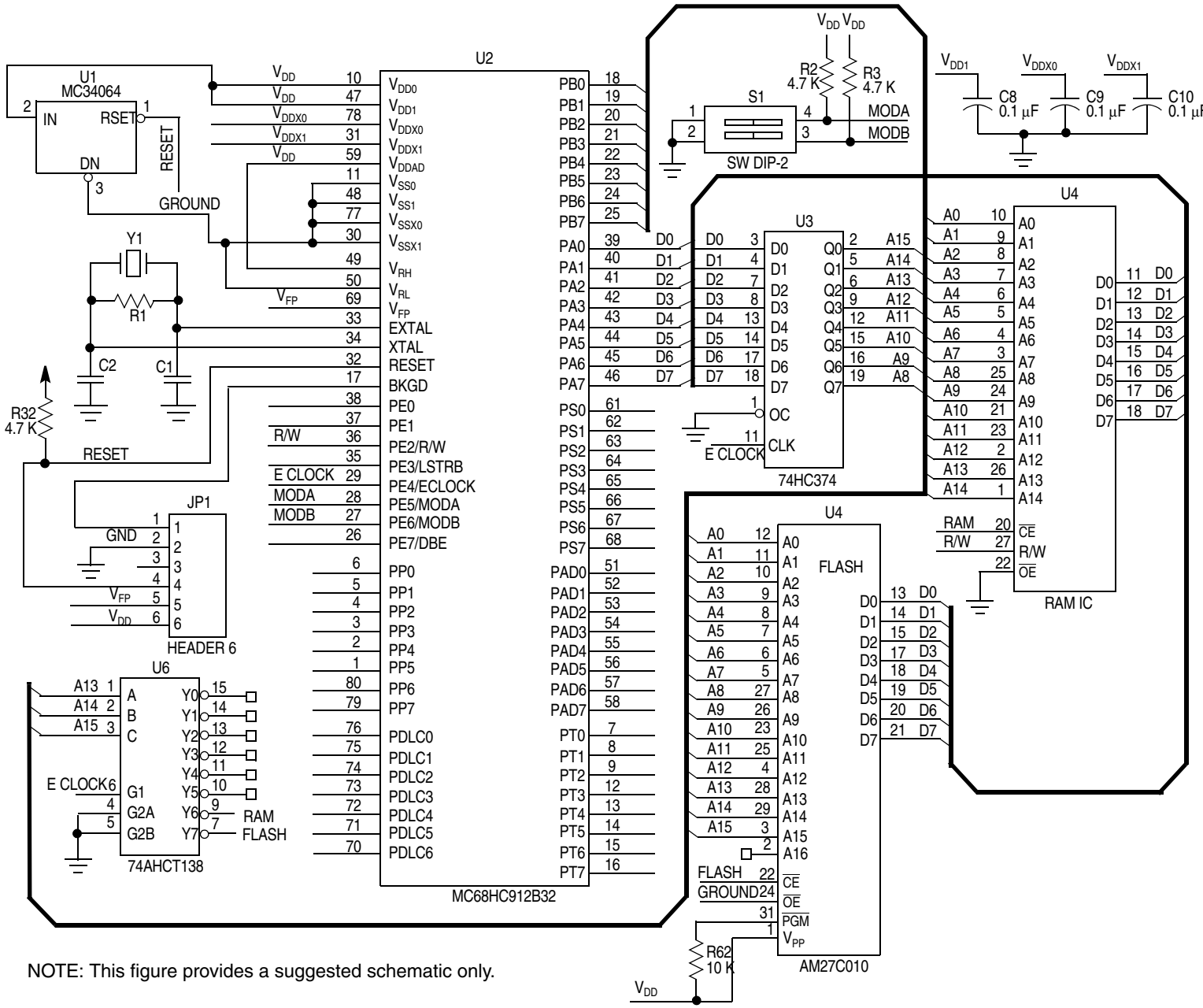


Figure 1-9. RAM Expansion Schematic with FLASH in Narrow Mode

Downloaded from Arrow.com.

Chapter 2
Register Block

2.1 Introduction

The register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space by manipulating bits REG15–REG11 in the register initialization register (INITRG). INITRG establishes the upper five bits of the register block's 16-bit address. The register block occupies the first 512 bytes of the 2-Kbyte block.

Default addressing (after reset) is indicated in Figure 2-1. For additional information, refer to Chapter 5 Operating Modes and Resource Mapping.

NOTE

In expanded and peripheral modes, these registers are not in the map:

- Port A data register, PORTA
- Port B data register, PORTB
- Port A data direction register, DDRA
- Port B data direction register, DDRB

In peripheral mode or in expanded modes w*ith the emulate port E bit (EME)* set, these registers are not in the map:

- Port E data register, PORTE
- Port E data direction register, DDRE

In peripheral mode, these registers are not in the map:

- Mode register, MODE
- Pullup control register, PUCR
- Reduced drive register, RDRIV

Downloaded from Arrow.com.

2.2 Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PORTA) See page 86.	Read: Write:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
		Reset:	U	U	U	U	U	U	U	U
$0001	Port B Data Register (PORTB) See page 87.	Read: Write:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
		Reset:	U	U	U	U	U	U	U	U
$0002	Data Direction Register A (DDRA) See page 86.	Read: Write:	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
		Reset:	0	0	0	0	0	0	0	0
$0003	Data Direction Register B (DDRB) See page 87.	Read: Write:	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
		Reset:	0	0	0	0	0	0	0	0
$0004	Reserved		R	R	R	R	R	R	R	R
↓										
$0007	Reserved		R	R	R	R	R	R	R	R
$0008	Port E Data Register (PORTE) See page 88.	Read: Write:	PE7	PE6	PE5	PD4	PD3	PD2	PD1	PD0
		Reset:	0	0	0	0	0	0	0	0
$0009	Data Direction Register E (DDRE) See page 88.	Read: Write:	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	0	0
		Reset:	0	0	0	0	1	0	0	0
$000A	Port E Assignment Register (PEAR) See page 89.	Read: Write:	NDBE	CGMTE	PIPOE	NECLK	LSTRE	RDWE	0	0
		Reset:	1	0	0	1	0	0	0	0
$000B	Mode Register (MODE) See page 78.	Read: Write:	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	0	EME
		Reset:	0	0	0	1	1	0	0	1
$000C	Pullup Control Register (PUCR) See page 91.	Read: Write:	0	0	0	PUPE	0	0	PUPB	PUPA
		Reset:	0	0	0	1	0	0	0	0
$000D	Reduced Drive Register (RDRIV) See page 92.	Read: Write:	0	0	0	0	RDPE	0	RDPB	RDPA
		Reset:	0	0	0	0	0	0	0	0
$000E	Reserved		R	R	R	R	R	R	R	R
$000F	Reserved		R	R	R	R	R	R	R	R

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 1 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0010	RAM Initialization Register (INITRM) See page 80.	Read:	RAM15	RAM14	RAM13	RAM12	RAM11	0	0	0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$0011	Register Initialization Register (INITRG) See page 80.	Read:	REG15	REG14	REG13	REG12	REG11	0	0	MMSWAI
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0012	EEPROM Initialization Register (INITEE) See page 81.	Read:	EE15	EE14	EE13	EE12	0	0	0	EEON
		Write:								
		Reset:	0	0	0	1	0	0	0	1
$0013	Miscellaneous Mapping Control Register (MISC) See page 82.	Read:	0	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	MAPROM	ROMON
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0014	Real-Time Interrupt Control Register (RTICTL) See page 118.	Read:	RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	Real-Time Interrupt Flag Register (RTIFLG) See page 119.	Read:	RTIF	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0016	COP Control Register (COPCTL) See page 119.	Read:	CME	FCME	FCM	FCOP	DISR	CR2	CR1	CR0
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$0017	Arm/Reset COP Timer Register (COPRST) See page 120.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	Reserved		R	R	R	R	R	R	R	R
↓										
$001D	Reserved		R	R	R	R	R	R	R	R
$001E	Interrupt Control Register (INTCR) See page 70.	Read:	IRQE	IRQEN	DLY	0	0	0	0	0
		Write:								
		Reset:	0	1	1	0	0	0	0	0
$001F	Highest Priority I Interrupt Register (HPRIO) See page 71.	Read:	1	1	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	0
		Write:								
		Reset:	1	1	1	1	0	0	1	0
$0020	Breakpoint Control Register 0 (BRKCT0) See page 301.	Read:	BKEN1	BKEN0	BKPM	0	BK1ALE	BK0ALE	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 2 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0021	Breakpoint Control Register 1 (BRKCT1) See page 302.	Read: Write:	0	BKDBE	BKMBH	BKMBL	BK1RWE	BK1RW	BK0RWE	BK0RW
		Reset:	0	0	0	0	0	0	0	0
$0022	Breakpoint Address Register High (BRKAH) See page 303.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0023	Breakpoint Address Register Low (BRKAL) See page 303.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0024	Breakpoint Data Register High (BRKDH) See page 304.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0025	Breakpoint Data Register Low (BRKDL) See page 304.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0026	Reserved		R	R	R	R	R	R	R	R
↓										
$003F	Reserved		R	R	R	R	R	R	R	R
$0040	PWM Clocks and Concatenate Register (PWCLK) See page 128.	Read: Write:	CON23	CON01	PCKA2	PCKA1	PCKA0	PCKB2	PCKB1	PCKB0
		Reset:	0	0	0	0	0	0	0	0
$0041	PWM Clock Select and Polarity Register (PWPOL) See page 129.	Read: Write:	PCLK3	PCLK2	PCLK1	PCLK0	PPOL3	PPOL2	PPOL1	PPOL0
		Reset:	0	0	0	0	0	0	0	0
$0042	PWM Enable Register (PWEN) See page 130.	Read:	0	0	0	0	PWEN3	PWEN2	PWEN1	PWEN0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0043	PWM Prescaler Counter Register (PWPRES) See page 131.	Read:	0	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0044	PWM Scale Register 0 (PWSCAL0) See page 131.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0045	PWM Scale Counter Register 0 (PWSCNT0) See page 131.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 3 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0046	PWM Scale Register 1 (PWSCAL1) See page 132.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0047	PWM Scale Counter Register 1 (PWSCNT1) See page 132.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	PWM Channel Counter Register 0 (PWCNT0) See page 133.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0049	PWM Channel Counter Register 1 (PWCNT1) See page 133.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$004A	PWM Channel Counter Register 2 (PWCNT2) See page 133.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$004B	PWM Channel Counter Register 3 (PWCNT3) See page 133.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$004C	PWM Channel Period Register 0 (PWPER0) See page 134.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$004D	PWM Channel Period Register 1 (PWPER1) See page 134.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$004E	PWM Channel Period Register 2 (PWPER2) See page 134.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$004F	PWM Channel Period Register 3 (PWPER3) See page 134.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0050	PWM Channel Duty Register 0 (PWDTY0) See page 135.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0051	PWM Channel Duty Register 1 (PWDTY1) See page 135.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 4 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0052	PWM Channel Duty Register 2 (PWDTY2) See page 135.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0053	PWM Channel Duty Register 3 (PWDTY3) See page 135.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0054	PWM Control Register (PWCTL) See page 136.	Read:	0	0	0	PSWAI	CENTR	RDPP	PUPP	PSBCK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0055	PWM Special Mode Register (PWTST) See page 137.	Read:	DISCR	DISCP	DISCAL	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0056	Port P Data Register (PORTP) See page 137.	Read:	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0057	Port P Data Direction Register (DDRP) See page 138.	Read:	DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0058	Reserved		R	R	R	R	R	R	R	R
↓										
$005F	Reserved		R	R	R	R	R	R	R	R
$0060	ATD Control Register 0 (ATDCTL0) See page 279.	Read:	0	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0061	ATD Control Register 1 (ATDCTL1) See page 279.	Read:	0	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0062	ATD Control Register 2 (ATDCTL2) See page 279.	Read:	ADPU	AFFC	AWAI	0	0	0	ASCIE	ASCIF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0063	ATD Control Register 3 (ATDCTL3) See page 280.	Read:	0	0	0	0	0	0	FRZ1	FRZ0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0064	ATD Control Register 4 (ATDCTL4) See page 281.	Read:	S10BM	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
		Write:								
		Reset:	0	0	0	0	0	0	0	1

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 5 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0065	ATD Control Register 5 (ATDCTL5) See page 282.	Read:		S8CM	SCAN	MULT	CD	CC	CB	CA
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0066	ATD Status Register (ATDSTAT) See page 284.	Read:	SCF	0	0	0	0	CC2	CC1	CC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0067	ATD Status Register (ATDSTAT) See page 284.	Read:	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0068	ATD Test Register High (ATDTSTH) See page 285.	Read:	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0069	ATD Test Register Low (ATDTSTL) See page 285.	Read:	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$006A	Reserved		R	R	R	R	R	R	R	R
↓										
$006E	Reserved		R	R	R	R	R	R	R	R
$006F	Port AD Data Input Register (PORTAD) See page 286.	Read:	PAD7	PAD6	PAD5	PAD4	PAD3	PAD2	PAD1	PAD0
		Write:								
		Reset:	After reset, reflect the state of the input pins							
$0070	ATD Result Register 0 (ADRx0H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$0071	ATD Result Register 0 (ADRx0L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$0072	ATD Result Register 1 (ADRx1H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$0073	ATD Result Register 1 (ADRx1L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$0074	ATD Result Register 2 (ADRx2H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 6 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0075	ATD Result Register 2 (ADRx2L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$0076	ATD Result Register 3 (ADRx3H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$0077	ATD Result Register 3 (ADRx3L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$0078	ATD Result Register 4 (ADRx4H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$0079	ATD Result Register 4 (ADRx4L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$007A	ATD Result Register 5 (ADRx5H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$007B	ATD Result Register 5 (ADRx5L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$007C	ATD Result Register 6 (ADRx6H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$007D	ATD Result Register 6 (ADRx6L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$007E	ATD Result Register 7 (ADRx7H) See page 286.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	Undefined							
$007F	ATD Result Register 7 (ADRx7L) See page 286.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Undefined							
$0080	Timer IC/OC Select Register (TIOS) See page 141.	Read:	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 7 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0081	Timer Compare Force Register (CFORC) See page 141.	Read:	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0082	Timer Output Compare 7 Mask Register (OC7M) See page 143.	Read:	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0083	Timer Output Compare 7 Data Register (OC7D) See page 143.	Read:	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0084	Timer Count Register High (TCNTH) See page 144.	Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0085	Timer Count Register Low (TCNTL) See page 144.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0086	Timer System Control Register (TSCR) See page 144.	Read:	TEN	TSWAI	TSBCK	TFFCA				
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0087	Reserved		R	R	R	R	R	R	R	R
$0088	Timer Control Register 1 (TCTL1) See page 145.	Read:	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0089	Timer Control Register 2 (TCTL2) See page 145.	Read:	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$008A	Timer Control Register 3 (TCTL3) See page 146.	Read:	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$008B	Timer Control Register 4 (TCTL4) See page 146.	Read:	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$008C	Timer Mask Register 1 (TMSK1) See page 146.	Read:	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$008D	Timer Mask Register 2 (TMSK2) See page 147.	Read:	TOI	0	PUPT	RDPT	TCRE	PR2	PR1	PR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 8 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$008E	Timer Interrupt Flag Register 1 (TFLG1) See page 148.	Read: Write:	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
		Reset:	0	0	0	0	0	0	0	0
$008F	Timer Interrupt Flag Register 2 (TFLG2) See page 148.	Read: Write:	TOF	0	0	0	0	0	0	0
		Reset:	0	0	0	0	0	0	0	0
$0090	Timer Input Capture/Output Compare 0 Register High (TC0H) See page 149.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0091	Timer Input Capture/Output Compare 0 Register Low (TC0L) See page 149.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0092	Timer Input Capture/Output Compare 1 Register High (TC1H) See page 149.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0093	Timer Input Capture/Output Compare 1 Register Low (TC1L) See page 149.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0094	Timer Input Capture/Output Compare 2 Register High (TC2H) See page 150.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0095	Timer Input Capture/Output Compare 2 Register Low (TC2L) See page 150.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0096	Timer Input Capture/Output Compare 3 Register High (TC3H) See page 150.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0097	Timer Input Capture/Output Compare 3 Register Low (TC3L) See page 150.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$0098	Timer Input Capture/Output Compare 4 Register High (TC4H) See page 150.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$0099	Timer Input Capture/Output Compare 4 Register Low (TC4L) See page 150.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 9 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$009A	Timer Input Capture/Output Compare 5 Register High (TC5H) See page 150.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$009B	Timer Input Capture/Output Compare 5 Register Low (TC5L) See page 150.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$009C	Timer Input Capture/Output Compare 6 Register High (TC6H) See page 151.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$009D	Timer Input Capture/Output Compare 6 Register Low (TC6L) See page 151.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$009E	Timer Input Capture/Output Compare 7 Register High (TC7H) See page 151.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$009F	Timer Input Capture/Output Compare 7 Register Low (TC7L) See page 151.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00A0	Pulse Accumulator Control Register (PACTL) See page 151.	Read:	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00A1	Pulse Accumulator Flag Register (PAFLG) See page 153.	Read: Write:	0	0	0	0	0	0	PAOVF	PAIF
		Reset:	0	0	0	0	0	0	0	0
$00A2	Pulse Accumulator Count Register 3 (PACN3) See page 178.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00A3	Pulse Accumulator Count Register 2 (PACN2) See page 178.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00A4	Pulse Accumulator Count Register 1 (PACN1) See page 178.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00A5	Pulse Accumulator Count Register 0 (PACN0) See page 178.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 10 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00A6	16-Bit Modulus Down-Counter Control Register (MCCTL) See page 179.	Read: Write:	MCZI	MODMC	RDMCL	ICLAT	FLMC	MCEN	MCPR1	MCPR0
		Reset:	0	0	0	0	0	0	0	0
$00A7	16-Bit Modulus Down-Counter Flag Register (MCFLG) See page 180.	Read: Write:	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
		Reset:	0	0	0	0	0	0	0	0
$00A8	Input Control Pulse Accumulators Control Register (ICPACR) See page 181.	Read: Write:	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
		Reset:	0	0	0	0	0	0	0	0
$00A9	Delay Counter Control Register (DLYCT) See page 181.	Read: Write:	0	0	0	0	0	0	DLY1	DLY0
		Reset:	0	0	0	0	0	0	0	0
$00AA	Input Control Overwrite Register (ICOVW) See page 182.	Read: Write:	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
		Reset:	0	0	0	0	0	0	0	0
$00AB	Input Control System Control Register (ICSYS) See page 182.	Read: Write:	SH37	SH26	SH15	HS04	TFMOD	PACMX	BUFEN	LATQ
		Reset:	0	0	0	0	0	0	0	0
$00AC	Reserved		R	R	R	R	R	R	R	R
$00AD	Timer Test Register (TIMTST) See page 183.	Read:	0	0	0	0	0	0	TCBYP	PCBYP(1)
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00AE	Timer Port Data Register (PORTT) See page 184.	Read: Write:	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0
		Reset:	0	0	0	0	0	0	0	0
$00AF	Data Direction Register for Timer Port (DDRT) See page 184.	Read: Write:	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0
		Reset:	0	0	0	0	0	0	0	0
$00B0	16-Bit Pulse Accumulator B Control Register (PBCTL) See page 185.	Read: Write:	0	PBEN	0	0	0	0	PBOV	0
		Reset:	0	0	0	0	0	0	0	0
$00B1	Pulse Accumulator B Flag Register (PBFLG) See page 185.	Read: Write:	0	0	0	0	0	0	PBOV	0
		Reset:	0	0	0	0	0	0	0	0
$00B2	8-Bit Pulse Accumulator Holding Register 3 (PA3H) See page 186.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 11 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00B3	8-Bit Pulse Accumulator Holding Register 2 (PA2H) See page 186.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00B4	8-Bit Pulse Accumulator Holding Register 1 (PA1H) See page 186.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00B5	8-Bit Pulse Accumulator Holding Register 0 (PA0H) See page 186.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00B6	Modulus Down-Counter Count Register (MCCNT) See page 187.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$00B7	Modulus Down-Counter Count Register (MCCNT) See page 187.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$00B8	Timer Input Capture Holding Register 0 (TC0H) See page 187.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	1	0	0	0	0	0
$00B9	Timer Input Capture Holding Register 0 (TC0H) See page 188.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00BA	Timer Input Capture Holding Register 1 (TC1H) See page 188.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$00BB	Timer Input Capture Holding Register 1 (TC1H) See page 188.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00BC	Timer Input Capture Holding Register 2 (TC2H) See page 188.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$00BD	Timer Input Capture Holding Register 2 (TC2H) See page 188.	Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$00BE	Timer Input Capture Holding Register 3 (TC3H) See page 188.	Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 12 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00BF	Timer Input Capture Holding Register 3 (TC3H) See page 188.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00C0	SCI 0 Baud Rate Control Register High (SC0BDH) See page 194.	Read:	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00C1	SCI 0 Baud Rate Control Register Low (SC0BDL) See page 194.	Read:	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
		Write:								
		Reset:	0	0	0	0	0	1	0	0
$00C2	SCI Control Register 1 (SC0CR1) See page 195.	Read:	LOOPS	WOMS	RSRC	M	WAKE	ILT	PE	PT
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00C3	SCI Control Register 2 (SC0CR2) See page 197.	Read:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00C4	SCI Status Register 1 (SC0SR1) See page 198.	Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$00C5	SCI Status Register 2 (SC0SR2) See page 199.	Read:	0	0	0	0	0	0	0	RAF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00C6	SCI Data Register High (SC0DRH) See page 200.	Read:	R8	T8	0	0	0	0	0	0
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$00C7	SCI Data Register Low (SC0DRL) See page 200.	Read:	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
		Write:								
		Reset:	Unaffected by reset							
$00C8 ↓ $00CF	Reserved		R	R	R	R	R	R	R	R
$00D0	SPI Control Register 1 (SP0CR1) See page 204.	Read:	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00D1	SPI Control Register 2 (SP0CR2) See page 205.	Read:	0	0	0	0	PUPS	RDS	0	SPC0
		Write:								
		Reset:	0	0	0	0	1	0	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 13 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00D2	SPI Baud Rate Register (SP0BR) See page 206.	Read:	0	0	0	0	0	SPR2	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00D3	SPI Status Register (SP0SR) See page 207.	Read:	SPIF	WCOL	0	MODF	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00D4	Reserved		R	R	R	R	R	R	R	R
$00D5	SPI Data Register (SP0DR) See page 207.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	Unaffected by reset							
$00D6	Port S Data Register (PORTS) See page 208.	Read:	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
		Write:								
		Reset:	After reset all bits configured as general-purpose inputs							
$00D7	Port S Data Direction Register (DDRS) See page 208.	Read:	DDS7	DDS6	DDS5	DDS4	DDS3	DDS2	DDS1	DDS0
		Write:								
		Reset:	After reset all bits configured as general-purpose inputs							
$00D8	Reserved		R	R	R	R	R	R	R	R
↓										
$00DA	Reserved		R	R	R	R	R	R	R	R
$00DB	Port S Pullup/Reduced Drive Register (PURDS) See page 209.	Read:	0	RDPS2	RDPS1	RDPS0	0	PUPS2	PUPS1	PUPS0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00DC	Reserved		R	R	R	R	R	R	R	R
↓										
$00DF	Reserved		R	R	R	R	R	R	R	R
$00E0	Slow Mode Divider Register (SLOW) See page 117.	Read:	0	0	0	0	0	SLDV2	SLDV1	SLDV0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00E1	Reserved		R	R	R	R	R	R	R	R
↓										
$00EF	Reserved		R	R	R	R	R	R	R	R
$00F0	EEPROM Configuration Register (EEMCR) See page 94.	Read:	1	1	1	1	1	EESWAI	PROTLCK	EERC
		Write:								
		Reset:	1	1	1	1	1	1	0	0

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 14 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00F1	EEPROM Block Protect Register (EEPROT) See page 95.	Read:	1	1	1	BRPROT4	BRPROT3	BRPROT2	BRPROT1	BRPROT0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$00F2	EEPROM Test Register (EETST) See page 95.	Read:	EEODD	EEVEN	MARG	EECPD	EECPRD	0	EECPM	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00F3	EEPROM Control Register (EEPROG) See page 96.	Read:	BULKP	0	0	BYTE	ROW	ERASE	EELAT	EEPGM
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$00F4	FLASH EEPROM Lock Control Register (FEELCK)(1) See page 100.	Read:	0	0	0	0	0	0	0	LOCK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00F5	FLASH EEPROM Configuration Register (FEEMCR)(1) See page 100.	Read:	0	0	0	0	0	0	0	BOOTP
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$00F6	FLASH EEPROM Test Register (FEETST)(1) See page 100.	Read:	FSTE	GADR	HVT	FENLV	FDISVFP	VTCK	STRE	MWPR
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00F7	FLASH EEPROM Control Register (FEECTL)(1) See page 102.	Read:	0	0	0	FEESWAI	SVFP	ERAS	LAT	ENPE
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00F8	BDLC Control Register 1 (BCR1)(2) See page 231.	Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
		Write:					R	R		
		Reset:	1	1	1	0	0	0	0	0
$00F9	BDLC State Vector Register (BSVR)(2) See page 237.	Read:	0	0	I3	I2	I1	I0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00FA	BDLC Control Register 2 (BCR2)(2) See page 233.	Read:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$00FB	BDLC Data Register (BDR)(2) See page 239.	Read:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
		Write:								
		Reset:	Indeterminate after reset							
$00FC	BDLC Analog Roundtrip Delay Register (BARD)(2) See page 240.	Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
		Write:								
		Reset:	1	1	0	0	0	1	1	1

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 15 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$00FD	Port DLC Control Register (DLCSCR)(2) See page 241.	Read:	0	0	0	0	0	BDLCEN	PUPDLC	RDPDLC
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00FE	Port DLC Data Register (PORTDLC)(2) See page 242.	Read:	0	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	U	U	U	U	U	U	U
$00FF	Port DLC Data Direction Register (DDRDLC)(2) See page 242.	Read:	0	DDDLC6	DDDLC5	DDDLC4	DDDLC3	DDDLC2	DDDLC1	DDDLC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0100	msCAN12 Module Control Register 0 (CMCR0)(3) See page 262.	Read:	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		Write:								
		Reset:	0	0	1	0	0	0	0	1
$0101	msCAN12 Module Control Register 1 (CMCR1)(3) See page 263.	Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0102	msCAN12 Bus Timing Register 0 (CBTR0)(3) See page 264.	Read:	SJW1	SJW0	BRP5	BRP4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0103	msCAN12 Bus Timing Register 1 (CBTR1)(3) See page 265.	Read:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0104	msCAN12 Receiver Flag Register (CRFLG)(3) See page 266.	Read:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0105	msCAN12 Receiver Interrupt Enable Register (CRIER)(3) See page 268.	Read:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0106	msCAN12 Transmitter Flag Register (CTFLG)(3) See page 269.	Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		Write:								
		Reset:	0	0	0	0	0	1	1	1
$0107	msCAN12 Transmitter Control Register (CTCR)(3) See page 270.	Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0108	msCAN12 Identifier Acceptance Control Register (CIDAC)(3) See page 270.	Read:	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0109	Reserved		R	R	R	R	R	R	R	R
↓										

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 16 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$010D	Reserved		R	R	R	R	R	R	R	R
$010E	msCAN12 Receive Error Counter (CRXERR)(3) See page 271.	Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$010F	msCAN12 Transmit Error Counter (CTXERR)(3) See page 272.	Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0110	msCAN12 Identifier Acceptance Register 0 (CIDAR0)(3) See page 272.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0111	msCAN12 Identifier Acceptance Register 1 (CIDAR1)(3) See page 272.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0112	msCAN12 Identifier Acceptance Register 2 (CIDAR2)(3) See page 272.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0113	msCAN12 Identifier Acceptance Register 3 (CIDAR3)(3) See page 272.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0114	msCAN12 Identifier Mask Register 0 (CIDMR0)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0115	msCAN12 Identifier Mask Register 1 (CIDMR1)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0116	msCAN12 Identifier Mask Register 2 (CIDMR2)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0117	msCAN12 Identifier Mask Register 3 (CIDMR3)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0118	msCAN12 Identifier Acceptance Register 4 (CIDAR4)(3) See page 273.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0119	msCAN12 Identifier Acceptance Register 5 (CIDAR5)(3) See page 273.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 17 of 19)

Downloaded from Arrow.com.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$011A	msCAN12 Identifier Acceptance Register 6 (CIDAR6)(3) See page 273.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$011B	msCAN12 Identifier Acceptance Register 7 (CIDAR7)(3) See page 273.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$011C	msCAN12 Identifier Mask Register 4 (CIDMR4)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$011D	msCAN12 Identifier Mask Register 5 (CIDMR5)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$011E	msCAN12 Identifier Mask Register 6 (CIDMR6)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$011F	msCAN12 Identifier Mask Register 7 (CIDMR7)(3) See page 274.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0120	Reserved		R	R	R	R	R	R	R	R
↓										
$013C	Reserved		R	R	R	R	R	R	R	R
$013D	msCAN12 Port CAN Control Register (PCTLCAN)(3) See page 275.	Read:	0	0	0	0	0	0	PUECAN	RDPCAN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$013E	msCAN12 Port CAN Data Register (PORTCAN)(3) See page 275.	Read:	PCAN7	PCAN6	PCAN5	PCAN4	PCAN2	PCAN2	TxCAN	RxCAN
		Write:								
		Reset:	Unaffected by reset							
$013F	msCAN12 Port CAN Data Direction Register (DDRCAN)(3) See page 276.	Read:	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0140 ↓ $014F	RECEIVE BUFFER (RxFG)(3) — SEE 16.3.2 Receive Structures									
$0150 ↓ $015F	TRANSMIT BUFFER 0 (Tx0)(3) — SEE 16.3.3 Transmit Structures									

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 18 of 19)

Downloaded from Arrow.com.

Addr.	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
$0160 ↓ $016F		TRANSMIT BUFFER 1 (TX1)(3) — SEE 16.3.3 Transmit Structures							
$0170 ↓ $017F		TRANSMIT BUFFER 2 (Tx2)(3) — SEE 16.3.3 Transmit Structures							

= Unimplemented R = Reserved U = Unaffected

Notes:
1. Available only on MC68HC912B32 and MC68HC912BC32 devices.
2. Available only on MC68HC912B32 and MC68HC12BE32 devices.
3. Available only on MC68HC(9)12BC32 devices.

Figure 2-1. Register Map (Sheet 19 of 19)

Downloaded from Arrow.com.

Chapter 3
Central Processor Unit (CPU)

3.1 Introduction

The CPU12 is a high-speed, 16-bit processor unit. It has full 16-bit data paths and wider internal registers (up to 20 bits) for high-speed extended math instructions. The instruction set is a proper superset of the M68HC11instruction set. The CPU12 allows instructions with odd byte counts, including many single-byte instructions. This provides efficient use of ROM space. An instruction queue buffers program information so the CPU always has immediate access to at least three bytes of machine code at the start of every instruction. The CPU12 also offers an extensive set of indexed addressing capabilities.

3.2 Programming Model

CPU12 registers are an integral part of the CPU and are not addressed as if they were memory locations. See Figure 3-1.

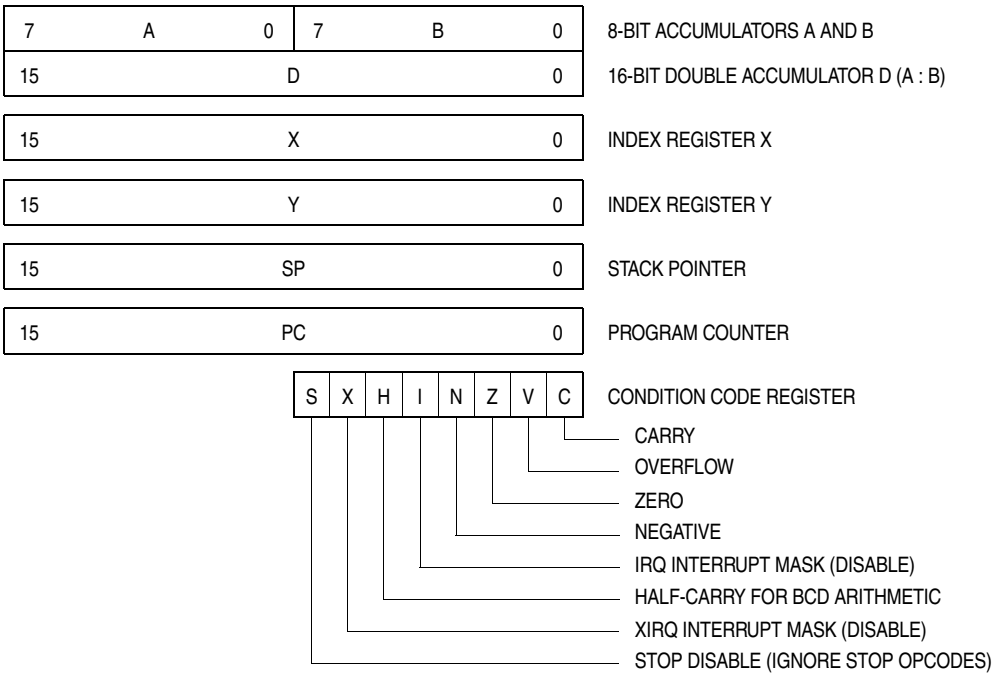


Figure 3-1. Programming Model

Downloaded from Arrow.com.

3.3 CPU Registers

This section describes the CPU registers.

3.3.1 Accumulators A and B

Accumulators A and B are general-purpose 8-bit accumulators that contain operands and results of arithmetic calculations or data manipulations.

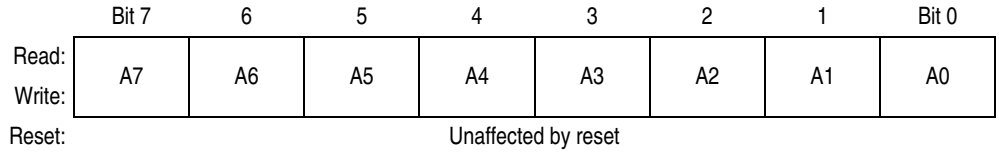


Figure 3-2. Accumulator A (A)

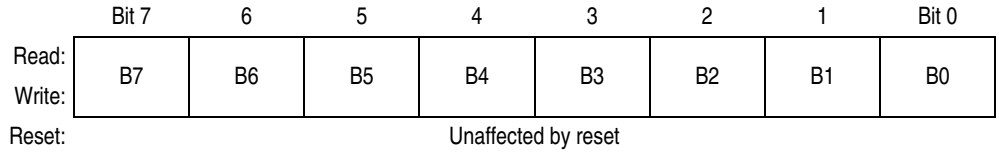


Figure 3-3. Accumulator B (B)

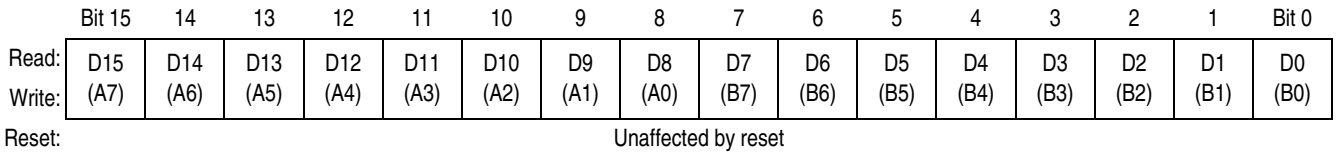
3.3.2 Accumulator D

Accumulator D is the concatenation of accumulators A and B. Some instructions treat the combination of these two 8-bit accumulators as a 16-bit double accumulator.

NOTE

The LDD and STD instructions can be used to manipulate data in and out of accumulator D.

Figure 3-4. Accumulator D (D)



Downloaded from Arrow.com.

3.3.3 Index Registers X and Y

Index registers X and Y are used for indexed addressing. Indexed addressing adds the value in an index register to a constant or to the value in an accumulator to form the effective address of the operand.

Index registers X and Y can also serve as temporary data storage locations.

NOTE

The LDX and STX instructions can be used to manipulate data in and out of index register X.

Figure 3-5. Index Register X (X)

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
Reset:	Unaffected by reset															

NOTE

The LDY and STY instructions can be used to manipulate data in and out of index register Y.

Figure 3-6. Index Register Y (Y)

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
Reset:	Unaffected by reset															

3.3.4 Stack Pointer

The stack pointer (SP) contains the last stack address used. The CPU12 supports an automatic program stack that is used to save system context during subroutine calls and interrupts.

The stack pointer can also serve as a temporary data storage location or as an index register for indexed addressing.

NOTE

The LDS and STS instructions can be used to manipulate data in and out of the stack pointer.

Figure 3-7. Stack Pointer (SP)

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Reset:	Unaffected by reset															

Downloaded from Arrow.com.

3.3.5 Program Counter

The program counter contains the address of the next instruction to be executed.

The program counter can also serve as an index register in all indexed addressing modes except autoincrement and autodecrement.

Figure 3-8. Program Counter (PC)

	Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
Read: Write:	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
Reset:	Unaffected by reset															

3.3.6 Condition Code Register

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	S	X	H	I	N	Z	V	C
Reset:	1	1	U	1	U	U	U	U

U = Unaffected

Figure 3-9. Condition Code Register (CCR)

S — Stop Disable Bit
Setting the S bit disables the STOP instruction.

X — XIRQ Interrupt Mask Bit
Setting the X bit masks interrupt requests from the $\overline{\text{XIRQ}}$ pin.

H — Half-Carry Flag
The H flag is used only for BCD arithmetic operations. It is set when an ABA, ADD, or ADC instruction produces a carry from bit 3 of accumulator A. The DAA instruction uses the H flag and the C flag to adjust the result to the correct BCD format.

I — Interrupt Mask Bit
Setting the I bit disables maskable interrupt sources.

N — Negative Flag
The N flag is set when the result of an operation is less than 0.

Z — Zero Flag
The Z flag is set when the result of an operation is all 0s.

V — Two's Complement Overflow Flag
The V flag is set when a two's complement overflow occurs.

C — Carry/Borrow Flag
The C flag is set when an addition or subtraction operation produces a carry or borrow.

Downloaded from Arrow.com.

3.4 Data Types

The CPU12 supports four data types:

1. Bit data
2. 8-bit and 16-bit signed and unsigned integers
3. 16-bit unsigned fractions
4. 16-bit addresses

A byte is eight bits wide and can be accessed at any byte location. A word is composed of two consecutive bytes with the most significant byte at the lower value address. There are no special requirements for alignment of instructions or operands.

3.5 Addressing Modes

Addressing modes determine how the CPU accesses memory locations to be operated upon. The CPU12 includes all of the addressing modes of the M68HC11 CPU as well as several new forms of indexed addressing. Table 3-1 is a summary of the available addressing modes.

Table 3-1. Addressing Mode Summary

Addressing Mode	Source Format	Abbreviation	Description
Inherent	INST	INH	Operands (if any) are in CPU registers.
Immediate	INST #*opr8i* or INST #*opr16i*	IMM	Operand is included in instruction stream 8- or 16-bit size implied by context.
Direct	INST *opr8a*	DIR	Operand is the lower 8 bits of an address in the range $0000–$00FF.
Extended	INST *opr16a*	EXT	Operand is a 16-bit address.
Relative	INST *rel8* or INST *rel16*	REL	An 8-bit or 16-bit relative offset from the current pc is supplied in the instruction.
Indexed 5-bit offset	INST *oprx5,xysp*	IDX	5-bit signed constant offset from x, y, sp, or pc
Indexed auto pre-decrement	INST *oprx3,–xys*	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexed auto pre-increment	INST *oprx3,+xys*	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexed auto post-decrement	INST *oprx3,xys–*	IDX	Auto post-decrement x, y, or sp by 1 ~ 8
Indexed auto post-increment	INST *oprx3,xys+*	IDX	Auto post-increment x, y, or sp by 1 ~ 8
Indexed accumulator offset	INST *abd,xysp*	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from x, y, sp, or pc
Indexed 9-bit offset	INST *oprx9,xysp*	IDX1	9-bit signed constant offset from x, y, sp, or pc (lower 8 bits of offset in one extension byte)
Indexed 16-bit offset	INST *oprx16,xysp*	IDX2	16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)

Table continued on next page

Downloaded from Arrow.com.

Table 3-1. Addressing Mode Summary (Continued)

Addressing Mode	Source Format	Abbreviation	Description
Indexed-Indirect 16-bit offset	INST [*oprx16,xysp*]	[IDX2]	Pointer to operand is found at 16-bit constant offset from x, y, sp, or pc (16-bit offset in two extension bytes)
Indexed-Indirect D accumulator offset	INST [D,*xysp*]	[D,IDX]	Pointer to operand is found at x, y, sp, or pc plus the value in D

3.6 Indexed Addressing Modes

The CPU12 indexed modes reduce execution time and eliminate code size penalties for using the Y index register. CPU12 indexed addressing uses a postbyte plus zero, one, or two extension bytes after the instruction opcode. The postbyte and extensions do these tasks:

- Specify which index register is used
- Determine whether a value in an accumulator is used as an offset
- Enable automatic pre- or post-increment or decrement
- Specify use of 5-bit, 9-bit, or 16-bit signed offsets

Table 3-2. Summary of Indexed Operations

Postbyte Code (xb)	Source Code Syntax	Comments

`rr0nnnnn`	`,r`
	`n,r`
	`-n,r`
		5-bit constant offset n = –16 to +15
		r can specify X, Y, SP, or PC

`111rr0zs`	`n,r` `-n,r`	**Constant offset** (9- or 16-bit signed) z:0 = 9-bit with sign in LSB of postbyte(s) 1 = 16-bit if z = s = 1, 16-bit offset indexed-indirect (see below) rr can specify X, Y, SP, or PC
`111rr011`	`[n,r]`	**16-bit offset indexed-indirect** rr can specify X, Y, SP, or PC
`rr1pnnnn`	`n,-r` `n,+r` `n,r-` `n,r+`	**Auto pre-decrement/increment** or **Auto post-decrement/increment**; p = pre-(0) or post-(1), n = –8 to –1, +1 to +8 rr can specify X, Y, or SP (PC not a valid choice)

`111rr1aa`	`A,r`
	`B,r`
	`D,r`
		Accumulator offset (unsigned 8-bit or 16-bit)
		aa:00 = A
		01 = B
		10 = D (16-bit)
		11 = see accumulator D offset indexed-indirect
		rr can specify X, Y, SP, or PC

`111rr111`	`[D,r]`	**Accumulator D offset indexed-indirect** rr can specify X, Y, SP, or PC

rr: 00 = X, 01 = Y, 10 = SP, 11 = PC

Downloaded from Arrow.com.

3.7 Opcodes and Operands

The CPU12 uses 8-bit opcodes. Each opcode identifies a particular instruction and associated addressing mode to the CPU. Several opcodes are required to provide each instruction with a range of addressing capabilities.

Only 256 opcodes would be available if the range of values were restricted to the number that can be represented by 8-bit binary numbers. To expand the number of opcodes, a second page is added to the opcode map. Opcodes on the second page are preceded by an additional byte with the value $18.

To provide additional addressing flexibility, opcodes can also be followed by a postbyte or extension bytes. Postbytes implement certain forms of indexed addressing, transfers, exchanges, and loop primitives. Extension bytes contain additional program information such as addresses, offsets, and immediate data.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 4
Resets and Interrupts

4.1 Introduction

Resets and interrupts are exceptions. Each exception has a 16-bit vector that points to the memory location of the associated exception-handling routine. Vectors are stored in the upper 128 bytes of the standard 64-Kbyte address map.

The six highest vector addresses are used for resets and non-maskable interrupt sources. The remainder of the vectors are used for maskable interrupts, and all must be initialized to point to the address of the appropriate service routine.

4.2 Exception Priority

A hardware priority hierarchy determines which reset or interrupt is serviced first when simultaneous requests are made. Six sources are not maskable. The remaining sources are maskable, and any one of them can be given priority over other maskable interrupts.

The priorities of the non-maskable sources are:

1. Power-on reset (POR) or $\overline{\text{RESET}}$ pin
2. Clock monitor reset
3. Computer operating properly (COP) watchdog reset
4. Unimplemented instruction trap
5. Software interrupt instruction (SWI)
6. $\overline{\text{XIRQ}}$ signal if X bit in CCR = 0

4.3 Maskable Interrupts

Maskable interrupt sources include on-chip peripheral systems and external interrupt service requests. Interrupts from these sources are recognized when the global interrupt mask bit (I) in the condition code register (CCR) is cleared. The default state of the I bit out of reset is 1, but it can be written at any time.

Interrupt sources are prioritized by default but any one maskable interrupt source may be assigned the highest priority by means of the HPRIO register. The relative priorities of the other sources remain the same.

An interrupt that is assigned highest priority is still subject to global masking by the I bit in the CCR or by any associated local bits. Interrupt vectors are not affected by priority assignment. HPRIO can be written only while the I bit is set (interrupts inhibited). Table 4-1 lists interrupt sources and vectors in default order of priority for all devices except the MC68HC(9)12BC32. Table 4-2 lists the interrupt sources and vectors for the MC68HC(9)12BC32.

Downloaded from Arrow.com.

Table 4-1. Interrupt Vector Map[1]

Vector Address	Interrupt Source	CCR Mask	Local Enable		HPRIO Value to Elevate to Highest I Bit
			Register	Bit(s)	
$FFFE, $FFFF	Reset	None	None	None	—
$FFFC, $FFFD	COP clock monitor fail reset	None	COPCTL	CME, FCME	—
$FFFA, $FFFB	COP failure reset	None	None	COP rate selected	—
$FFF8, $FFF9	Unimplemented instruction trap	None	None	None	—
$FFF6, $FFF7	SWI	None	None	None	—
$FFF4, $FFF5	XIRQ	X bit	None	None	—
$FFF2, $FFF3	IRQ	I bit	INTCR	IRQEN	$F2
$FFF0, $FFF1	Real-time interrupt	I bit	RTICTL	RTIE	$F0
$FFEE, $FFEF	Timer channel 0	I bit	TMSK1	C0I	$EE
$FFEC, $FFED	Timer channel 1	I bit	TMSK1	C1I	$EC
$FFEA, $FFEB	Timer channel 2	I bit	TMSK1	C2I	$EA
$FFE8, $FFE9	Timer channel 3	I bit	TMSK1	C3I	$E8
$FFE6, $FFE7	Timer channel 4	I bit	TMSK1	C4I	$E6
$FFE4, $FFE5	Timer channel 5	I bit	TMSK1	C5I	$E4
$FFE2, $FFE3	Timer channel 6	I bit	TMSK1	C6I	$E2
$FFE0, $FFE1	Timer channel 7	I bit	TMSK1	C7I	$E0
$FFDE, $FFDF	Timer overflow	I bit	TMSK2	TOI	$DE
$FFDC, $FFDD	Pulse accumulator overflow	I bit	PACTL	PAOVI	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I bit	PACTL	PAI	$DA
$FFD8, $FFD9	SPI serial transfer complete	I bit	SP0CR1	SPIE	$D8
$FFD6, $FFD7	SCI 0	I bit	SC0CR2	TIE, TCIE, RIE, ILIE	$D6
$FFD4, $FFD5	Reserved	I bit	—	—	$D4
$FFD2, $FFD3	ATD	I bit	ATDCTL2	ASCIE	$D2
$FFD0, $FFD1	BDLC	I bit	BCR1	IE	$D0
$FF80–$FFC1	Reserved (not implemented)	I bit	—	—	$80–$C0
$FFC2–$FFC9	Reserved (implemented)	I bit	—	—	$C2–$C8
$FFCA, $FFCB	Pulse accumulator B overflow	I bit	PBCTL	PBOVI	$CA
$FFCC, $FFCD	Modulus down counter underflow	I bit	MCCTL	MCZI	$CC
$FFCE, $FFCF	Reserved (implemented)	I bit	—	—	$CE

1. See Table 4-2 for the MC68HC(9)12BC32 interrupt vector map.

Downloaded from Arrow.com.

Table 4-2. MC68HC(9)12BC32 Interrupt Vector Map

Vector Address	Interrupt Source	CCR Mask	Local Enable		HPRIO Value to Elevate to Highest I Bit
			Register	Bit(s)	
$FFFE, $FFFF	Reset	None	None	None	—
$FFFC, $FFFD	COP clock monitor fail reset	None	COPCTL	CME, FCME	—
$FFFA, $FFFB	COP failure reset	None	None	COP rate selected	—
$FFF8, $FFF9	Unimplemented instruction trap	None	None	None	—
$FFF6, $FFF7	SWI	None	None	None	—
$FFF4, $FFF5	XIRQ	X bit	None	None	—
$FFF2, $FFF3	IRQ	I bit	INTCR	IRQEN	$F2
$FFF0, $FFF1	Real-time interrupt	I bit	RTICTL	RTIE	$F0
$FFEE, $FFEF	Timer channel 0	I bit	TMSK1	C0I	$EE
$FFEC, $FFED	Timer channel 1	I bit	TMSK1	C1I	$EC
$FFEA, $FFEB	Timer channel 2	I bit	TMSK1	C2I	$EA
$FFE8, $FFE9	Timer channel 3	I bit	TMSK1	C3I	$E8
$FFE6, $FFE7	Timer channel 4	I bit	TMSK1	C4I	$E6
$FFE4, $FFE5	Timer channel 5	I bit	TMSK1	C5I	$E4
$FFE2, $FFE3	Timer channel 6	I bit	TMSK1	C6I	$E2
$FFE0, $FFE1	Timer channel 7	I bit	TMSK1	C7I	$E0
$FFDE, $FFDF	Timer overflow	I bit	TMSK2	TOI	$DE
$FFDC, $FFDD	Pulse accumulator overflow	I bit	PACTL	PAOVI	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I bit	PACTL	PAI	$DA
$FFD8, $FFD9	SPI serial transfer complete	I bit	SP0CR1	SPIE	$D8
$FFD6, $FFD7	SCI 0	I bit	SC0CR2	TIE, TCIE, RIE, ILIE	$D6
$FFD4, $FFD5	Reserved	I bit	—	—	$D4
$FFD2, $FFD3	ATD	I bit	ATDCTL2	ASCIE	$D2
$FFD0, $FFD1	MSCAN wakeup	I bit	CRIER	WUPIE	$D0
$FFCA–$FFCF	Reserved (not implemented)	I bit	—	—	$CA–$CF
$FFC8–$FFC9	MSCAN errors	I bit	CRIER	RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE	$C8
$FFC6, $FFC7	MSCAN receive	I bit	CRIER	RXFIE	$C6
$FFC4, $FFC5	MSCAN transmit	I bit	CTCR	TXEIE[2:0]	$C4
$FF80, $FFC3	Reserved (implemented)	I bit	—	—	$80–$C3

Downloaded from Arrow.com.

4.4 Latching of Interrupts

$\overline{\text{XIRQ}}$ is always level triggered and $\overline{\text{IRQ}}$ can be selected as a level-triggered interrupt. These level-triggered interrupt pins should be released only during the appropriate interrupt service routine. Generally, the interrupt service routine will handshake with the interrupting logic to release the pin. In this way, the MCU will never start the interrupt service sequence only to determine that there is no longer an interrupt source. In the event that this does occur, the trap vector will be taken.

If $\overline{\text{IRQ}}$ is selected as an edge-triggered interrupt, the hold time of the level after the active edge is independent of when the interrupt is serviced. As long as the minimum hold time is met, the interrupt will be latched inside the MCU. In this case, the IRQ edge interrupt latch is cleared automatically when the interrupt is serviced.

All of the remaining interrupts are latched by the MCU with a flag bit. These interrupt flags should be cleared during an interrupt service routine or when the interrupts are masked by the I bit. By doing this, the MCU will never get an unknown interrupt source and take the trap vector.

4.5 Interrupt Control and Priority Registers

This section describes the interrupt control and priority registers.

4.5.1 Interrupt Control Register

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IRQE	IRQEN	DLY	0	0	0	0	0
Reset:	0	1	1	0	0	0	0	0

Figure 4-1. Interrupt Control Register (INTCR)

Read: Anytime

Write: Varies from bit to bit

IRQE — $\overline{\text{IRQ}}$ Edge-Sensitive Only Bit

IRQE can be written once in normal modes. In special modes, IRQE can be written anytime, but the first write is ignored.

1 = $\overline{\text{IRQ}}$ pin responds only to falling edges.
0 = $\overline{\text{IRQ}}$ pin responds to low levels.

IRQEN — External $\overline{\text{IRQ}}$ Enable Bit

IRQEN can be written anytime in all modes. The $\overline{\text{IRQ}}$ pin has an internal pullup.

1 = $\overline{\text{IRQ}}$ pin connected to interrupt logic
0 = $\overline{\text{IRQ}}$ pin disconnected from interrupt logic

DLY — Oscillator Startup Delay on Exit from Stop Mode Bit

DLY can be written once in normal modes. In special modes, DLY can be written anytime.
The delay time of about 4096 cycles is based on the E-clock rate.

1 = Stabilization delay on exit from stop mode
0 = No stabilization delay on exit from stop mode

Downloaded from Arrow.com.

4.5.2 Highest Priority I Interrupt Register

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	1	1	PSEL5	PSEL4	PSEL3	PSEL2	PSEL1	0
Write:								
Reset:	1	1	1	1	0	0	1	0

Figure 4-2. Highest Priority I Interrupt Register (HPRIO)

Read: Anytime

Write: Only if I bit in CCR = 1 (interrupts inhibited)

To give a maskable interrupt source highest priority, write the low byte of the vector address to the HPRIO register. For example, writing $F0 to HPRIO assigns highest maskable interrupt priority to the real-time interrupt timer ($FFF0). If an unimplemented vector address or a non-I-masked vector address (a value higher than $F2) is written, then $\overline{\text{IRQ}}$ is the default highest priority interrupt.

4.6 Resets

There are four possible sources of reset. POR and external reset on the $\overline{\text{RESET}}$ pin share the normal reset vector. COP reset and the clock monitor reset each has a vector. Entry into reset is asynchronous and does not require a clock, but the MCU cannot sequence out of reset without a system clock.

4.6.1 Power-On Reset (POR)

A positive transition on V_{DD} causes a POR. An external voltage level detector or other external reset circuits are the usual source of reset in a system. The POR circuit only initializes internal circuitry during cold starts and cannot be used to force a reset as system voltage drops.

4.6.2 External Reset

The CPU distinguishes between internal and external reset conditions by sensing whether the reset pin rises to a logic 1 in less than eight E-clock cycles after an internal device releases reset. When a reset condition is sensed, the $\overline{\text{RESET}}$ pin is driven low by an internal device for about 16 E-clock cycles, then released. Eight E-clock cycles later it is sampled. If the pin is still held low, the CPU assumes that an external reset has occurred. If the pin is high, it indicates that the reset was initiated internally by either the COP system or the clock monitor.

To prevent a COP or clock monitor reset from being detected during an external reset, hold the reset pin low for at least 32 cycles. An external resistor-capacitor (RC) power-up delay circuit on the reset pin is not recommended because circuit charge time can cause the MCU to misinterpret the type of reset that has occurred.

4.6.3 Computer Operating Properly (COP) Reset

The MCU includes a COP system to help protect against software failures. When COP is enabled, software must write $55 and $AA (in this order) to the COPRST register to keep a watchdog timer from timing out. Other instructions may be executed between these writes. A write of any value other than $55 or $AA or software failing to execute the sequence properly causes a COP reset to occur.

Downloaded from Arrow.com.

4.6.4 Clock Monitor Reset

If clock frequency falls below a predetermined limit when the clock monitor is enabled, a reset occurs.

4.7 Effects of Reset

When a reset occurs, MCU registers and control bits are changed to known startup states, as described here.

4.7.1 Operating Mode and Memory Map

The states of the BKGD, MODA, and MODB pins during reset determine the operating mode and default memory mapping. The SMODN, MODA, and MODB bits in the MODE register reflect the status of the mode-select inputs at the rising edge of reset. Operating mode and default maps can subsequently be changed according to strictly defined rules.

4.7.2 Clock and Watchdog Control Logic

Reset enables the COP watchdog with the CR2–CR0 bits set for the shortest timeout period. The clock monitor is disabled. The RTIF flag is cleared and automatic hardware interrupts are masked. The rate control bits are cleared and must be initialized before the return-from-interrupt (RTI) system is used. The DLY control bit is set to specify an oscillator startup delay upon recovery from stop mode.

4.7.3 Interrupts

Reset initializes the HPRIO register with the value $F2, causing the $\overline{\text{IRQ}}$ pin to have the highest I-bit interrupt priority. The $\overline{\text{IRQ}}$ pin is configured for level-sensitive operation (for wired-OR systems). However, the I and X bits in the CCR are set, masking $\overline{\text{IRQ}}$ and $\overline{\text{XIRQ}}$ interrupt requests.

4.7.4 Parallel Input/Output (I/O)

If the MCU comes out of reset in an expanded mode, port A and port B are the multiplexed address/data bus. Port E pins are normally used to control the external bus. The port E assignment register (PEAR) affects port E pin operation.

If the MCU comes out of reset in a single-chip mode, all ports are configured as general-purpose high-impedance inputs.

4.7.5 Central Processing Unit (CPU)

After reset, the CPU fetches a vector from the appropriate address and begins executing instructions. The stack pointer and other CPU registers are indeterminate immediately after reset. The condition code register (CCR) X and I interrupt mask bits are set to mask any interrupt requests. The S bit is also set to inhibit the STOP instruction.

4.7.6 Memory

After reset, the internal register block is located at $0000–$01FF, the register-following space is at $0200–$03FF, and RAM is at $0800–$0BFF. EEPROM is located at $0D00–$0FFF. FLASH EEPROM/ROM is located at $8000–$FFFF in single-chip modes and at $0000–$7FFF (but disabled) in expanded modes.

Downloaded from Arrow.com.

4.7.7 Other Resources

The timer, serial communications interface (SCI), serial peripheral interface (SPI), byte data link controller (BDLC), pulse-width modulator (PWM), analog-to-digital converter (ATD), and MSCAN are off after reset.

4.8 Interrupt Recognition

Once enabled, an interrupt request can be recognized at any time after the I bit in the CCR is cleared. When an interrupt request is recognized, the CPU responds at the completion of the instruction being executed. Interrupt latency varies according to the number of cycles required to complete the instruction. Some of the longer instructions can be interrupted and resume normally after servicing the interrupt.

When the CPU begins to service an interrupt request, it:

- Clears the instruction queue
- Calculates the return address
- Stacks the return address and the contents of the CPU registers as shown in Table 4-3

Table 4-3. Stacking Order on Entry to Interrupts

Memory Location	Stacked Values
SP – 2	RTN_H : RTN_L
SP – 4	Y_H : Y_L
SP – 6	X_H : X_L
SP – 8	B : A
SP – 9	CCR

After stacking the CCR, the CPU:

- Sets the I bit to prevent other interrupts from disrupting the interrupt service routine
- Sets the X bit if an $\overline{\text{XIRQ}}$ interrupt request is pending
- Fetches the interrupt vector for the highest-priority request that was pending at the beginning of the interrupt sequence
- Begins execution of the interrupt service routine at the location pointed to by the vector

If no other interrupt request is pending at the end of the interrupt service routine, a return-from-interrupt (RTI) instruction recovers the stacked values. Program execution resumes program at the return address.

If another interrupt request is pending at the end of an interrupt service routine, the RTI instruction recovers the stacked values. However, the CPU then:

- Adjusts the stack pointer to point again at the stacked CCR location, SP – 9
- Fetches the vector of the pending interrupt
- Begins execution of the interrupt service routine at the location pointed to by the vector

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 5
Operating Modes and Resource Mapping

5.1 Introduction

The MCU can operate in eight different modes. Each mode has a different default memory map and external bus configuration. After reset, most system resources can be mapped to other addresses by writing to the appropriate control registers.

5.2 Operating Modes

The states of the BKGD, MODB, and MODA pins during reset determine the operating mode after reset.

The SMODN, MODB, and MODA bits in the MODE register show current operating mode and provide limited mode switching during operation. The states of the BKGD, MODB, and MODA pins are latched into these bits on the rising edge of the reset signal. During reset an active pullup is connected to the BKGD pin (as input) and active pulldowns are connected to the MODB and MODA pins. If an open occurs on any of these pins, the device will operate in normal single-chip mode.

Table 5-1. Mode Selection

BKGD	MODB	MODA	Mode	Port A	Port B
0	0	0	Special single chip	General-purpose I/O	General-purpose I/O
0	0	1	Special expanded narrow	ADDR[15:8] DATA[7:0]	ADDR[7:0]
0	1	0	Special peripheral	ADDR DATA	ADDR DATA
0	1	1	Special expanded wide	ADDR DATA	ADDR DATA
1	0	0	Normal single chip	General-purpose I/O	General-purpose I/O
1	0	1	Normal expanded narrow	ADDR[15:8] DATA[7:0]	ADDR[7:0]
1	1	0	Reserved (forced to peripheral)	—	—
1	1	1	Normal expanded wide	ADDR DATA	ADDR DATA

The two basic types of operating modes are:

1. Normal modes — Some registers and bits are protected against accidental changes.
2. Special modes — Protected control registers and bits are allowed greater access for special purposes such as testing and emulation.

A system development and debug feature, background debug mode (BDM) is available in all modes. In special single-chip mode, BDM is active immediately after reset.

Downloaded from Arrow.com.

5.2.1 Normal Operating Modes

These modes provide three operating configurations. Background debugging is available in all three modes, but must first be enabled for some operations by means of a BDM command. BDM can then be made active by another BDM command.

5.2.1.1 Normal Expanded Wide Mode

The 16-bit external address and data buses use ports A and B. ADDR15–ADDR8 and DATA15–DATA8 are multiplexed on port A. ADDR7–ADDR0 and DATA7–DATA0 are multiplexed on port B.

5.2.1.2 Normal Expanded Narrow Mode

The 16-bit external address bus uses port A for the high byte and port B for the low byte. The 8-bit external data bus uses port A. ADDR15–ADDR8 and DATA7–DATA0 are multiplexed on port A.

5.2.1.3 Normal Single-Chip Mode

Normal single-chip mode has no external buses. Ports A, B, and E are configured for general-purpose input/output (I/O). Port E bits 1 and 0 are input only with internal pullups and the other 22 pins are bidirectional I/O pins that are initially configured as high-impedance inputs. Port E pullups are enabled on reset. Port A and B pullups are disabled on reset.

5.2.2 Special Operating Modes

Special operating modes are commonly used in factory testing and system development.

5.2.2.1 Special Expanded Wide Mode

This mode is for emulation of normal expanded wide mode and emulation of normal single-chip mode with a 16-bit bus. The bus-control pins of port E are all configured for their bus-control output functions rather than general-purpose I/O.

5.2.2.2 Special Expanded Narrow Mode

This mode is for emulation of normal expanded narrow mode. External 16-bit data is handled as two back-to-back bus cycles, one for the high byte followed by one for the low byte. Internal operations continue to use full 16-bit data paths.

5.2.2.3 Special Single-Chip Mode

This mode can be used to force the MCU to active BDM mode to allow system debug through the BKGD pin. The MCU does not fetch the reset vector and execute application code as it would in other modes. Instead, the active background mode is in control of CPU execution and BDM firmware waits for additional serial commands through the BKGD pin. There are no external address and data buses in this mode. The MCU operates as a stand-alone device and all program and data space are on-chip. External port pins can be used for general-purpose I/O.

Downloaded from Arrow.com.

5.2.2.4 Special Peripheral Mode

The CPU is not active in this mode. An external master can control on-chip peripherals for testing purposes. It is not possible to change to or from this mode without going through reset. Background debugging should not be used while the MCU is in special peripheral mode as internal bus conflicts between BDM and the external master can cause improper operation of both modes.

5.2.3 Background Debug Mode

Background debug mode (BDM) is an auxiliary operating mode that is used for system development. BDM is implemented in on-chip hardware and provides a full set of debug operations. Some BDM commands can be executed while the CPU is operating normally. Other BDM commands are firmware based and require the BDM firmware to be enabled and active for execution.

In special single-chip mode, BDM is enabled and active immediately out of reset. BDM is available in all other operating modes, but must be enabled before it can be activated. BDM should not be used in special peripheral mode because of potential bus conflicts.

Once enabled, background mode can be made active by a serial command sent via the BKGD pin or execution of a CPU12 BGND instruction. While background mode is active, the CPU can interpret special debugging commands, read and write CPU registers, peripheral registers, and locations in memory.

While BDM is active, the CPU executes code located in a small on-chip ROM mapped to addresses $FF00 to $FFFF; BDM control registers are accessible at addresses $FF00 to $FF06. The BDM ROM replaces the regular system vectors while BDM is active. While BDM is active, the user memory from $FF00 to $FFFF is not in the map except through serial BDM commands.

BDM allows read and write access to internal memory-mapped registers and RAM and read access to EEPROM, FLASH EEPROM, or ROM without interrupting the application code executing in the CPU. This non-intrusive mode uses dead bus cycles to access the memory and in most cases will remain cycle deterministic. Refer to 18.3 Background Debug Mode (BDM) for more details.

5.3 Internal Resource Mapping

The internal register block, RAM, FLASH EEPROM/ROM, and EEPROM have default locations within the 64-Kbyte standard address space but may be reassigned to other locations during program execution by setting bits in mapping registers INITRG, INITRM, and INITEE. During normal operating modes, these registers can be written once. It is advisable to explicitly establish these resource locations during the initialization phase of program execution, even if default values are chosen, to protect the registers from inadvertent modification later.

Writes to the mapping registers go into effect between the cycle that follows the write and the cycle after that. To assure that there are no unintended operations, a write to one of these registers should be followed with a no operation (NOP) instruction.

If conflicts occur when mapping resources, the register block will take precedence over the other resources; RAM, FLASH EEPROM/ROM, or EEPROM addresses occupied by the register block will not be available for storage. When active, BDM ROM takes precedence over other resources, although a conflict between BDM ROM and register space is not possible. Table 5-2 shows resource mapping precedence.

In expanded modes, all address space not utilized by internal resources is by default external memory.

Downloaded from Arrow.com.

Table 5-2. Mapping Precedence

Precedence	Resource
1	BDM ROM (if active)
2	Register space
3	RAM
4	EEPROM
5	FLASH EEPROM/ROM
6	External memory

5.4 Mode and Resource Mapping Registers

This section describes the mode and resource mapping registers.

5.4.1 Mode Register

The mode register (MODE) controls the MCU operating mode and various configuration options. This register is not in the map in peripheral mode.

Address: $000B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SMODN	MODB	MODA	ESTR	IVIS	EBSWAI	0	EME
Reset states:								
Normal expanded narrow:	1	0	1	1	0	0	0	0
Normal expanded wide:	1	1	1	1	0	0	0	0
Special expanded narrow:	0	0	1	1	1	0	0	1
Special expanded wide:	0	1	1	1	1	0	0	1
Peripheral:	0	1	0	1	1	0	0	1
Normal single-chip:	1	0	0	1	0	0	0	0
Special single-chip:	0	0	0	1	1	0	0	1

Figure 5-1. Mode Register (MODE)

Read: Anytime

Write: Varies from bit to bit

SMODN, MODB, MODA — Mode Select Special, B, and A Bits

These bits show the current operating mode and reflect the status of the BKGD, MODB, and MODA input pins at the rising edge of reset.

SMODN can be written only if SMODN = 0 (in special modes) but the first write is ignored; MODB, MODA may be written once if SMODN = 1; anytime if SMODN = 0, except that special peripheral and reserved modes cannot be selected.

Downloaded from Arrow.com.

ESTR — E Clock Stretch Enable Bit

ESTR determines if the E clock behaves as a simple free-running clock or as a bus control signal that is active only for external bus cycles. ESTR is always 1 in expanded modes since it is required for address demultiplexing and must follow stretched cycles.

1 = E stretches high during external access cycles and low during non-visible internal accesses
0 = E never stretches (always free running)

Normal modes: Write once
Special modes: Write anytime

IVIS — Internal Visibility Bit

IVIS determines whether internal ADDR/DATA, R/$\overline{W}$, and $\overline{LSTRB}$ signals can be seen on the bus during accesses to internal locations. In special expanded narrow mode, it is possible to configure the MCU to show internal accesses on an external 16-bit bus. The IVIS control bit must be set to 1. When the system is configured this way, visible internal accesses are shown as if the MCU was configured for expanded wide mode, but normal external accesses operate as if the bus in narrow mode. In normal expanded narrow mode, internal visibility is not allowed and IVIS is ignored.

1 = Internal bus operations visible on external bus
0 = No visibility of internal bus operations on external bus

Normal modes: Write once
Special modes: Write anytime except the first time

EBSWAI — External Bus Module Stop in Wait Bit

This bit controls access to the external bus interface when in wait mode. The module delays before shutting down in wait mode to allow for final bus activity to complete.

1 = External bus shut down during wait mode
0 = External bus and registers continue functioning in wait mode.

Normal modes: Write anytime
Special modes: Write never

EME — Emulate Port E Bit

Removing the registers from the map allows the user to emulate the function of these registers externally. In single-chip mode, port E data register (PORTE) and port E data direction register (DDRE) are always in the map regardless of the state of this bit.

1 = PORTE and DDRE removed from the memory map (expanded mode)
0 = PORTE and DDRE in the memory map

Normal modes: Write once
Special modes: Write anytime except the first time

Downloaded from Arrow.com.

5.4.2 Register Initialization Register

After reset, the 512-byte register block resides at location $0000 but can be reassigned to any 2-Kbyte boundary within the standard 64-Kbyte address space. Mapping of internal registers is controlled by five bits in the register initialization register (INITRG). The register block occupies the first 512 bytes of the 2-Kbyte block.

Address: $0011

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	REG15	REG14	REG13	REG12	REG11	0	0	MMSWAI
Reset:	0	0	0	0	0	0	0	0

Figure 5-2. Register Initialization Register (INITRG)

Read: Anytime

Write: Once in normal modes; anytime in special modes

REG15–REG11 — Register Position Bits

These bits specify the upper five bits of the 16-bit register address.

MMSWAI — Memory Mapping Interface Stop in Wait Control Bit

This bit controls access to the memory mapping interface when in wait mode.

0 = Memory mapping interface continues to function in wait mode.

1 = Memory mapping interface access shuts down in wait mode.

Normal modes: Write anytime

Special modes: Write never

5.4.3 RAM Initialization Register

After reset, addresses of the 1-Kbyte RAM array begin at location $0800 but can be assigned to any 2-Kbyte boundary within the standard 64-Kbyte address space. Mapping of internal RAM is controlled by five bits in the RAM initialization register (INITRM). The RAM array occupies the first 1 Kbyte of the 2-Kbyte block.

Address: $0010

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RAM15	RAM14	RAM13	RAM12	RAM11	0	0	0
Reset:	0	0	0	0	1	0	0	0

Figure 5-3. RAM Initialization Register (INITRM)

Read: Anytime

Write: Once in normal modes; anytime in special modes

RAM15–RAM11 — RAM Position Bits

These bits specify the upper five bits of the 16-bit RAM address.

Downloaded from Arrow.com.

5.4.4 EEPROM Initialization Register

The MCU has 768 bytes of EEPROM which are activated by the EEON bit in the EEPROM initialization register (INITEE).

Mapping of internal EEPROM is controlled by four bits in the INITEE register. After reset, EEPROM address space begins at location $0D00 but can be mapped to any 4-Kbyte boundary within the standard 64-Kbyte address space.

Address: $0012

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EE15	EE14	EE13	EE12	0	0	0	EEON
Reset:	0	0	0	0	0	0	0	1

Figure 5-4. EEPROM Initialization Register (INITEE)

Read: anytime

Write: varies from bit to bit

EE15–EE12 — Internal EEPROM Position Bits

These bits specify the upper four bits of the 16-bit EEPROM address. Write once in normal modes or anytime in special modes.

EEON — EEPROM On Bit

EEON allows read access to the EEPROM array. EEPROM control registers can be accessed and EEPROM locations can be programmed or erased regardless of the state of EEON.
EEON is forced to 1 in single-chip modes.
Write only in expanded and peripheral modes.

1 = EEPROM in memory map
0 = EEPROM removed from memory map

5.4.5 Miscellaneous Mapping Control Register

Additional mapping controls are available that can be used in conjunction with FLASH EEPROM/ROM and memory expansion.

The 32-Kbyte FLASH EEPROM/ROM can be mapped to either the upper or lower half of the 64-Kbyte address space. When mapping conflicts occur, registers, RAM, and EEPROM have priority over FLASH EEPROM.

NOTE

Only the MC68HC912B32 contains FLASH EEPROM. The MC68HC12BE32 contains ROM.

To use memory expansion, the part must be operated in one of the expanded modes.

Downloaded from Arrow.com.

Address: $0013

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	NDRF	RFSTR1	RFSTR0	EXSTR1	EXSTR0	MAPROM	ROMON
Reset states:								
Expanded modes:	0	0	0	0	1	1	0	0
Single-chip modes:	0	0	0	0	1	1	1	1

Figure 5-5. Miscellaneous Mapping Control Register (MISC)

Read: Anytime

Write: Once in normal modes; anytime in special modes

NDRF — Narrow Data Bus for Register-Following Map Bit

This bit enables a narrow bus feature for the 512-byte register-following map. In expanded narrow (8-bit) modes, single-chip modes, and peripheral mode, NDRF has no effect. The register-following map always begins at the byte following the 512-byte register map. If the registers are moved, this space moves also.

1 = Register-following map space acts the same as an 8-bit external data bus.
0 = Register-following map space acts as a full 16-bit external data bus.

RFSTR1 and RFSTR0 — Register-Following Stretch Bits

These bits determine the amount of clock stretch on accesses to the 512-byte register-following map. It is valid regardless of the state of the NDRF bit. In single-chip and peripheral modes, these bits have no meaning or effect. See Table 5-3.

Table 5-3. Register-Following Stretch Bit Function

RFSTR1 and RFSTR0	E Clocks Stretched
00	0
01	1
10	2
11	3

EXSTR1 and EXSTR0 — External Access Stretch Bit 1 and Bit 0

These bits determine the amount of clock stretch on accesses to the external address space. In single-chip and peripheral modes, these bits have no meaning or effect.

Table 5-4. Expanded Stretch Bit Function

EXSTR1 and EXSTR0	E Clocks Stretched
00	0
01	1
10	2
11	3

Downloaded from Arrow.com.

MAPROM — FLASH EEPROM/ROM Map Bit

This bit determines the location of the on-chip FLASH EEPROM/ROM. In expanded modes, it is reset to 0. In single-chip modes, it is reset to 1. If ROMON is 0, this bit has no meaning or effect.

1 = FLASH EEPROM/ROM is located from $8000 to $FFFF.

0 = FLASH EEPROM/ROM is located from $0000 to $7FFF.

ROMON — FLASH EEPROM/ROM Enable Bit

In expanded modes, ROMON is reset to 0. In single-chip modes, it is reset to 1. If the internal RAM, registers, EEPROM, or BDM ROM (if active) are mapped to the same space as the FLASH EEPROM/ROM, they will have priority over the FLASH EEPROM/ROM.

1 = Enables the FLASH EEPROM/ROM in the memory map

0 = Disables the FLASH EEPROM/ROM in the memory map

5.5 Memory Map

Figure 5-6 illustrates the memory map for each mode of operation immediately after reset.

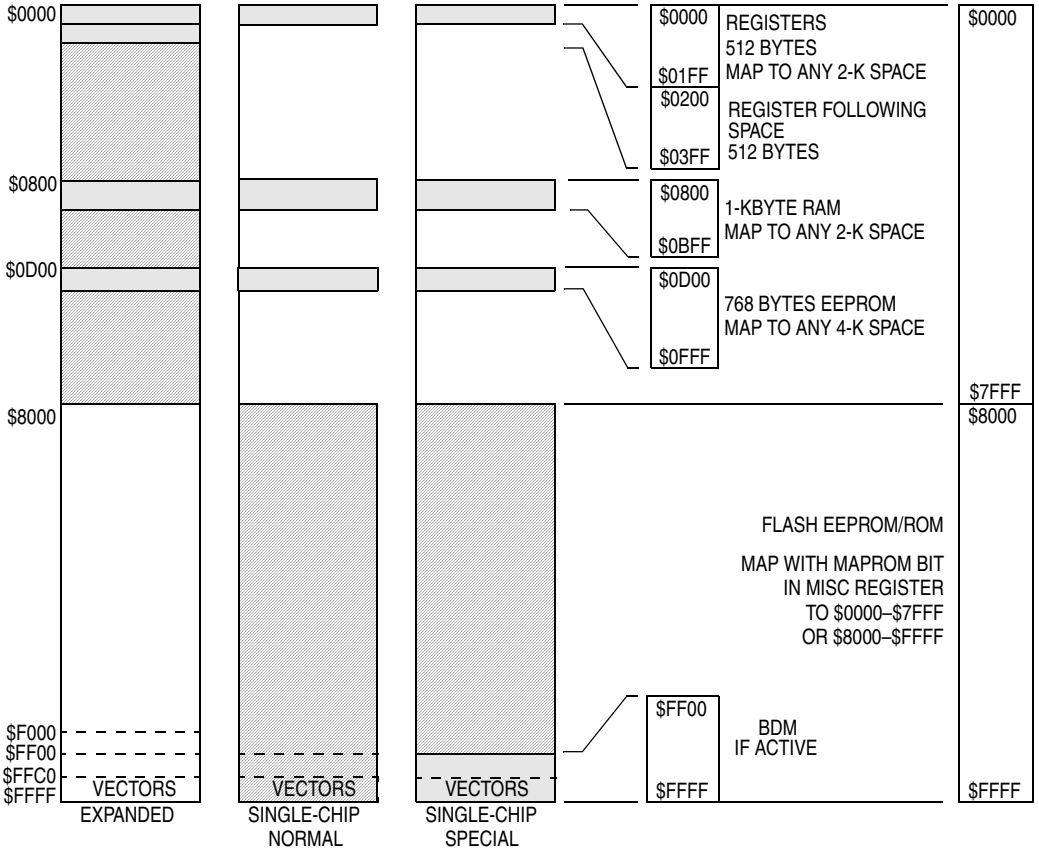


Figure 5-6. Memory Map

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 6
Bus Control and Input/Output (I/O)

6.1 Introduction

Internally, the MCU has full 16-bit data paths, but depending upon the operating mode and control registers, the external bus may be eight or 16 bits. There are cases where 8-bit and 16-bit accesses can appear on adjacent cycles using the $\overline{\text{LSTRB}}$ signal to indicate 8-bit or 16-bit data.

6.2 Detecting Access Type from External Signals

The external signals $\overline{\text{LSTRB}}$, R/$\overline{\text{W}}$, and A0 can be used to determine the type of bus access that is taking place. Accesses to the internal RAM module are the only accesses that produce $\overline{\text{LSTRB}}$ = A0 = 1, because the internal RAM is specifically designed to allow misaligned 16-bit accesses in a single cycle. In these cases, the data for the address that was accessed is on the low half of the data bus and the data for address +1 is on the high half of the data bus (data order is swapped).

Table 6-1. Detecting Access Type

$\overline{\text{LSTRB}}$	A0	R/$\overline{\text{W}}$	Type of Access
1	0	1	8-bit read of an even address
0	1	1	8-bit read of an odd address
1	0	0	8-bit write to an even address
0	1	0	8-bit write to an odd address
0	0	1	16-bit read of an even address
1	1	1	16-bit read of an odd address (low/high data swapped)
0	0	0	16-bit write to an even address
1	1	0	16-bit write to an odd address (low/high data swapped)

6.3 Registers

Under certain conditions, not all registers are visible in the memory map. In special peripheral mode, the first 16 registers associated with bus expansion are removed from the memory map.

In expanded modes, some or all of port A, port B, and port E are used for expansion buses and control signals. To allow emulation of the single-chip functions of these ports, some of these registers must be rebuilt in an external port replacement unit. In any expanded mode,
port A and port B are used for address and data lines so registers for these ports, as well as the data direction registers for these ports, are removed from the on-chip memory map and become external accesses.

In any expanded mode, port E pins may be needed for bus control (for example, ECLK and R/$\overline{\text{W}}$). To regain the single-chip functions of port E, the emulate port E (EME) control bit in the MODE register may

Downloaded from Arrow.com.

be set. In this special case of expanded mode and EME set, the port E data register (PORTE) and port E data direction register (DDRE) are removed from the on-chip memory map and become external accesses so port E may be rebuilt externally.

6.3.1 Port A Data Register

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Reset:	Unaffected by reset							
Expanded wide and peripheral:	ADDR15 DATA15	ADDR14 DATA14	ADDR13 DATA13	ADDR12 DATA12	ADDR11 DATA11	ADDR10 DATA10	ADDR9 DATA9	ADDR8 DATA8
Expanded narrow:	ADDR15 DATA15/7	ADDR14 DATA14/6	ADDR13 DATA13/5	ADDR12 DATA12/4	ADDR11 DATA11/3	ADDR10 DATA10/2	ADDR9 DATA9/1	ADDR8 DATA8/0

Figure 6-1. Port A Data Register (PORTA)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

Bits PA7–PA0 are associated with addresses ADDR15–ADDR8 and DATA15–DATA8. When this port is not used for external addresses and data, such as in single-chip mode, these pins can be used as general-purpose input/output (I/O). DDRA determines the primary direction of each pin. This register is not in the on-chip map in expanded and peripheral modes.

6.3.2 Port A Data Direction Register

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
Reset:	0	0	0	0	0	0	0	0

Figure 6-2. Port A Data Direction Register (DDRA)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

This register determines the primary direction for each port A pin when functioning as a general-purpose I/O port. DDRA is not in the on-chip map in expanded and peripheral modes.

1 = Associated pin is an output.
0 = Associated pin is a high-impedance input.

Downloaded from Arrow.com.

6.3.3 Port B Data Register

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Reset:	Unaffected by reset							
Alternate functions:								
Expanded wide and peripheral:	ADDR7 DATA7	ADDR6 DATA6	ADDR5 DATA5	ADDR4 DATA4	ADDR3 DATA3	ADDR2 DATA2	ADDR1 DATA1	ADDR0 DATA0
Expanded narrow:	ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0

Figure 6-3. Port B Data Register (PORTB)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

Bits PB7–PB0 are associated with addresses ADDR7–ADDR0 and DATA7–DATA0. When port B is not used for external addresses and data such as in single-chip mode, these pins can be used as general-purpose I/O. DDRB determines the primary direction of each pin. This register is not in the on-chip map in expanded and peripheral modes.

6.3.4 Port B Data Direction Register

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
Reset:	0	0	0	0	0	0	0	0

Figure 6-4. Port B Data Direction Register (DDRB)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

This register determines the primary direction for each port B pin when functioning as a general-purpose I/O port. DDRB is not in the on-chip map in expanded and peripheral modes.

1 = Associated pin is an output.

0 = Associated pin is a high-impedance input.

Downloaded from Arrow.com.

6.3.5 Port E Data Register

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
Reset:	0	0	0	0	0	0	0	0
Alternate function:	$\overline{DBE}$	MODB or IPIPE1	MODA or IPIPE0	ECLK	$\overline{LSTRB}$ or $\overline{TAGLO}$	R/$\overline{W}$	$\overline{IRQ}$	$\overline{XIRQ}$

Figure 6-5. Port E Data Register (PORTE)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

This register is associated with external bus control signals and interrupt inputs including:

- Data bus enable ($\overline{DBE}$)
- Mode select (MODB/IPIPE1 and MODA/IPIPE0)
- E clock
- Data size ($\overline{LSTRB}$/$\overline{TAGLO}$)
- Read/write (R/$\overline{W}$)
- $\overline{IRQ}$
- $\overline{XIRQ}$

When the associated pin is not used for one of these specific functions, the pin can be used as general-purpose I/O. The port E assignment register (PEAR) selects the function of each pin. DDRE determines the primary direction of each port E pin when configured to be general-purpose I/O.

Some of these pins have software selectable pullups ($\overline{DBE}$, $\overline{LSTRB}$, R/$\overline{W}$, and $\overline{XIRQ}$). A single control bit enables the pullups for all these pins which are configured as inputs. $\overline{IRQ}$ always has a pullup.

This register is not in the map in peripheral mode or expanded modes when the EME bit is set.

6.3.6 Port E Data Direction Register

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDE7	DDE6	DDE5	DDE4	DDE3	DDE2	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 6-6. Port E Data Direction Register (DDRE)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

This register determines the primary direction for each port E pin configured as general-purpose I/O.

1 = Associated pin is an output.

0 = Associated pin is a high-impedance input.

Downloaded from Arrow.com.

PE1 and PE0 are associated with $\overline{XIRQ}$ and $\overline{IRQ}$ and cannot be configured as outputs. These pins can be read regardless of whether the alternate interrupt functions are enabled.

This register is not in the map in peripheral mode and expanded modes while the EME control bit is set.

6.3.7 Port E Assignment Register

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	NDBE	CGMTE	PIPOE	NECLK	LSTRE	RDWE	0	0
Write:								
Reset states:								
Normal single-chip:	1	0	0	1	0	0	0	0
Special single-chip:	0	0	1	0	1	1	0	0
Normal expanded:	0	0	0	0	0	0	0	0
Special expanded:	0	0	1	0	1	1	0	0
Peripheral:	1	1	0	1	0	0	0	0

= Unimplemented

Figure 6-7. Port E Assignment Register (PEAR)

Read: Anytime, if register is in the map

Write: Varies from bit to bit, if register is in the map

The PEAR register is used to choose between the general-purpose I/O functions and the alternate bus control functions of port E. When an alternate control function is selected, the associated DDRE bits are overridden.

The reset condition of this register depends on the mode of operation because bus-control signals are needed immediately after reset in some modes.

In normal single-chip mode, no external bus control signals are needed, so all of port E is configured for general-purpose I/O.

In special single-chip mode, the E clock is enabled as a timing reference, and the other bits of port E are configured for general-purpose I/O.

In normal expanded modes, the reset vector is located in external memory. The E clock may be required for this access but R/$\overline{W}$ is only needed by the system when there are external writable resources. Therefore, in normal expanded modes, only the E clock is configured for its alternate bus control function and the other bits of port E are configured for general-purpose I/O. If the normal expanded system needs any other bus-control signals, PEAR would need to be written before any access that needed the additional signals.

In special expanded modes, IPIPE1, IPIPE0, E, R/$\overline{W}$, and $\overline{LSTRB}$ are configured as bus-control signals.

In peripheral mode, the PEAR register is not accessible for reads or writes.

NDBE — No Data Bus Enable Bit

Normal: Write once

Special: Write anytime except the first time

1 = PE7 used for general-purpose I/O

0 = PE7 used for external control of data enables on memories

Downloaded from Arrow.com.

CGMTE — CGM Test Output Enable

Normal: Write once
Special: Write anytime except the first time.
This bit is read at anytime.

1 = PE6 is a test signal output from the CGM module (no effect in single chip or normal expanded modes). PIPOE = 1 overrides this function and forces PE6 to be a pipe status output signal.
0 = PE6 is a general-purpose I/O or pipe output.

PIPOE — Pipe Signal Output Enable Bit

Normal: Write once
Special: Write anytime except the first time.
This bit has no effect in single chip modes.

1 = PE6–PE5 are outputs and indicate state of instruction queue.
0 = PE6–PE5 are general-purpose I/O.

NECLK — No External E Clock Bit

In expanded modes, writes to this bit have no effect. E clock is required for demultiplexing the external address; NECLK remains 0 in expanded modes. NECLK can be written once in normal single-chip mode and can be written anytime in special single-chip mode.

1 = PE4 is a general-purpose I/O pin.
0 = PE4 is the external E clock pin subject to this limitation: In single-chip modes, PE4 is general-purpose I/O unless NECLK = 0 and either IVIS = 1 or ESTR = 0. A 16-bit write to PEAR:MODE can configure all three bits in one operation.

LSTRE — Low Strobe ($\overline{\text{LSTRB}}$) Enable Bit

Normal: Write once
Special: Write anytime except the first time
This bit has no effect in single-chip modes or normal expanded narrow mode.

1 = PE3 is configured as the $\overline{\text{LSTRB}}$ bus-control output.
0 = PE3 is a general-purpose I/O pin.

$\overline{\text{LSTRB}}$ is used during external writes. After reset in normal expanded mode, $\overline{\text{LSTRB}}$ is disabled. If needed, it should be enabled before external writes. External reads do not normally need $\overline{\text{LSTRB}}$ because all 16 data bits can be driven even if the MCU only needs eight bits of data.
$\overline{\text{TAGLO}}$ is a shared function of the PE3/$\overline{\text{LSTRB}}$ pin. In special expanded modes with LSTRE set and the BDM instruction tagging on, a 0 at the falling edge of E tags the instruction word low byte being read into the instruction queue.

RDWE — Read/Write Enable Bit

Normal: Write once
Special: Write anytime except the first time
This bit has no effect in single-chip modes.

1 = PE2 configured as R/$\overline{\text{W}}$ pin
0 = PE2 configured as general-purpose I/O pin

R/$\overline{\text{W}}$ is used for external writes. After reset in normal expanded mode, it is disabled. If needed, it should be enabled before any external writes.

Downloaded from Arrow.com.

6.3.8 Pullup Control Register

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	PUPE	0	0	PUPB	PUPA
Write:								
Reset:	0	0	0	1	0	0	0	0

= Unimplemented

Figure 6-8. Pullup Control Register (PUCR)

Read: Anytime, if register is in the map

Write: Anytime, if register is in the map

These bits select pullup resistors for any pin in the corresponding port that is currently configured as an input. This register is not in the map in peripheral mode.

PUPE — Pullup Port E Enable Bit

Pin PE1 always has a pullup. Pins PE6, PE5, and PE4 never have pullups.

1 = Enable port E pullups on PE7, PE3, PE2, PE1, and PE0

0 = Disable port E pullups on PE7, PE3, PE2, PE1, and PE0

PUPB — Pullup Port B Enable Bit

1 = Enable pullups for all port B input pins

0 = Disable port B pullups

This bit has no effect if port B is used as part of the address/data bus (pullups are inactive).

PUPA — Pullup Port A Enable Bit

0 = Disable port A pullups

1 = Enable pullups for all port A input pins

This bit has no effect, if port A is used as part of the address/data bus (pullups are inactive).

Downloaded from Arrow.com.

6.3.9 Reduced Drive of I/O Lines

Address: $000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	RDPE	0	RDPB	RDPA
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 6-9. Reduced Drive of I/O Lines (RDRIV)

Read: Anytime, if register is in the map

Write: Once in normal modes; anytime, except the first time, in special modes

These bits select reduced drive for the associated port pins. This gives reduced power consumption and reduced radio frequency interference (RFI) with a slight increase in transition time (depending on loading). The reduced drive function is independent of which function is being used on a particular port. This register is not in the map in peripheral mode.

RDPE — Reduced Drive of Port E Bit

1 = Reduced drive for all port E output pins
0 = Full drive for all port E output pins

RDPB — Reduced Drive of Port B Bit

1 = Reduced drive for all port B output pins
0 = Full drive for all port B output pins

RDPA — Reduced Drive of Port A Bit

1 = Reduced drive for all port A output pins
0 = Full drive for all port A output pins

Downloaded from Arrow.com.

Chapter 7
EEPROM

7.1 Introduction

The MCU is electrically erasable, programmable read-only memory (EEPROM) serves as a 768-byte non-volatile memory which can be used for frequently accessed static data or as fast access program code.

The MCU's EEPROM is arranged in a 16-bit configuration. The EEPROM array may be read as either bytes, aligned words, or misaligned words. Access time is one bus cycle for byte and aligned word access and two bus cycles for misaligned word operations.

Programming is by byte or aligned word. Attempts to program or erase misaligned words will fail. Only the lower byte will be latched and programmed or erased. Programming and erasing of the user EEPROM can be done in all operating modes.

Each EEPROM byte or aligned word must be erased before programming. The EEPROM module supports byte, aligned word, row (32 bytes), or bulk erase, all using the internal charge pump. Bulk erasure of odd and even rows is also possible in test modes; the erased state is $FF. The EEPROM module has hardware interlocks which protect stored data from corruption by accidentally enabling the program/erase voltage. Programming voltage is derived from the internal V_{DD} supply with an internal charge pump. The EEPROM has a minimum program/erase life of 10,000 cycles over the complete operating temperature range.

7.2 EEPROM Programmer's Model

The EEPROM module consists of two separately addressable sections. The first is a 4-byte memory mapped control register block used for control, testing, and configuration of the EEPROM array. The second section is the EEPROM array itself.

At reset, the 4-byte register section starts at address $00F0 and the EEPROM array is located from addresses $0D00 to $0FFF (see Figure 7-1). For information on remapping the register block and EEPROM address space, refer to Chapter 5 Operating Modes and Resource Mapping.

Read access to the memory array section can be enabled or disabled by the EEON control bit in the EEPROM initialization register (INITEE). This feature allows the access of memory mapped resources that have lower priority than the EEPROM memory array. EEPROM control registers can be accessed and EEPROM locations may be programmed or erased regardless of the state of EEON.

Using the normal EEPROG control, it is possible to continue program/erase operations during wait. For lowest power consumption during wait, stop program/erase by turning off EEPGM.

If the stop mode is entered during programming or erasing, program/erase voltage will be turned off automatically and the resistor-capacitor (RC) clock (if enabled) is stopped. However, the EEPGM control bit will remain set. When stop mode is terminated, the program/erase voltage will be turned back on automatically if EEPGM is set.

Downloaded from Arrow.com.

At bus frequencies below 1 MHz, the RC clock must be turned on for program/erase.

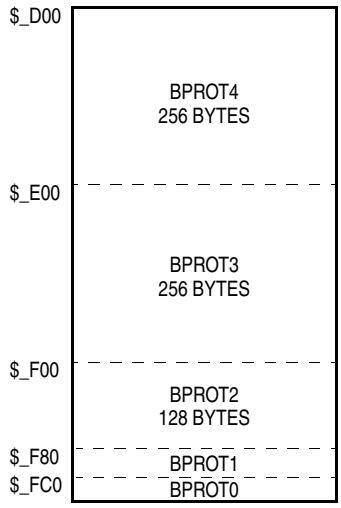


Figure 7-1. EEPROM Block Protect Mapping

7.3 EEPROM Control Registers

This section describes the EEPROM control registers.

7.3.1 EEPROM Module Configuration Register

Address: $00F0

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	1	1	1	1	1	EESWAI	PROTLCK	EERC
Reset:	1	1	1	1	1	1	0	0

Figure 7-2. EEPROM Module Configuration Register (EEMCR)

EESWAI — EEPROM Stops in Wait Mode Bit

0 = Module is not affected during wait mode.
1 = Module ceases to be clocked during wait mode.
This bit should be cleared if the wait mode vectors are mapped in the EEPROM array.

PROTLCK — Block Protect Write Lock Bit

0 = Block protect bits and bulk erase protection bit can be written.
1 = Block protect bits are locked.
Read anytime. Write once in normal modes (SMODN = 1); set and clear anytime in special modes (SMODN = 0).

EERC — EEPROM Charge Pump Clock Bit

0 = System clock is used as clock source for the internal charge pump. Internal RC oscillator is stopped.
1 = Internal RC oscillator drives the charge pump.
The RC oscillator is required when the system bus clock is lower than f_{PROG}.
Read and write anytime.

Downloaded from Arrow.com.

7.3.2 EEPROM Block Protect Register

Address: $00F1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	1	1	1	BRPROT4	BRPROT3	BRPROT2	BRPROT1	BRPROT0
Reset:	1	1	1	1	1	1	1	1

Figure 7-3. EEPROM Block Protect Register (EEPROT)

The EEPROM block protect register (EEPROT) prevents accidental writes to EEPROM.
Read anytime.
Write anytime if EEPGM = 0 and PROTLCK = 0.

BPROT4–BPROT0 — EEPROM Block Protection Bit

0 = Associated EEPROM block can be programmed and erased.
1 = Associated EEPROM block is protected from being programmed and erased.
Cannot be modified while programming is taking place (EEPGM = 1).

Table 7-1. 768-Byte EEPROM Block Protection

Bit Name	Block Protected	Block Size
BPROT4	$0D00 to $0DFF	256 bytes
BPROT3	$0E00 to $0EFF	256 bytes
BPROT2	$0F00 to $0F7F	128 bytes
BPROT1	$0F80 to $0FBF	64 bytes
BPROT0	$0FC0 to $0FFF	64 bytes

7.3.3 EEPROM Test Register

Address: $00F2

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EEODD	EEVEN	MARG	EECPD	EECPRD	0	EECPM	0
Reset:	0	0	0	0	0	0	0	0

Figure 7-4. EEPROM Test Register (EETST)

Read anytime. Write in special modes only (SMODN = 0). These bits are used for test purposes only. In normal modes, the bits are forced to 0.

EEODD — Odd Row Programming Bit

0 = Odd row bulk programming/erasing is disabled.
1 = Bulk program/erase all odd rows.
Refers to a physical location in the array rather than an odd byte address

EEVEN — Even Row Programming Bit

0 = Even row bulk programming/erasing is disabled.
1 = Bulk program/erase all even rows.
Refers to a physical location in the array rather than an even byte address.

Downloaded from Arrow.com.

MARG — Program and Erase Voltage Margin Test Enable Bit

0 = Normal operation
1 = Program and erase margin test
This bit is used to evaluate the program/erase voltage margin.

EECPD — Charge Pump Disable Bit

0 = Charge pump is turned on during program/erase.
1 = Disable charge pump.

EECPRD — Charge Pump Ramp Disable Bit

0 = Charge pump is turned on progressively during program/erase.
1 = Disable charge pump controlled ramp up.
Known to enhance write/erase endurance of EEPROM cells.

EECPM — Charge Pump Monitor Enable Bit

0 = Normal operation
1 = Output the charge pump voltage on the $\overline{IRQ}$/V_{PP} pin.

7.3.4 EEPROM Control Register

Address: $00F3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BULKP	0	0	BYTE	ROW	ERASE	EELAT	EEPGM
Reset:	1	0	0	0	0	0	0	0

Figure 7-5. EEPROM Control Register (EEPROG)

BULKP — Bulk Erase Protection Bit

0 = EEPROM can be bulk erased.
1 = EEPROM is protected from being bulk or row erased.
Read anytime. Write anytime if EEPGM = 0 and PROTLCK = 0.

BYTE — Byte and Aligned Word Erase Bit

0 = Bulk or row erase is enabled.
1 = One byte or one aligned word erase only
Read anytime. Write anytime if EEPGM = 0.

ROW — Row or Bulk Erase Bit (when BYTE = 0)

0 = Erase entire EEPROM array.
1 = Erase only one 32-byte row.
Read anytime. Write anytime if EEPGM = 0.
BYTE and ROW have no effect when ERASE = 0.

Table 7-2. Erase Selection

BYTE	ROW	Block Size
0	0	Bulk erase entire EEPROM array
0	1	Row erase 32 bytes
1	0	Byte or aligned word erase
1	1	Byte or aligned word erase

Downloaded from Arrow.com.

If BYTE = 1 and test mode is not enabled, only the location specified by the address written to the programming latches will be erased. The operation will be a byte or an aligned word erase depending on the size of written data.

ERASE — Erase Control Bit

0 = EEPROM configuration for programming or reading
1 = EEPROM configuration for erasure

Read anytime. Write anytime if EEPGM = 0.
Configures the EEPROM for erasure or programming.
When test mode is not enabled and unless BULKP is set, erasure is by byte, aligned word, row, or bulk.

EELAT — EEPROM Latch Control Bit

0 = EEPROM set up for normal reads
1 = EEPROM address and data bus latches set up for programming or erasing

Read anytime. Write anytime if EEPGM = 0.

NOTE

When EELAT is set, the entire EEPROM is unavailable for reads. Therefore, no program residing in the EEPROM can be executed while attempting to program unused EEPROM space. Care should be taken that no references to the EEPROM are used while programming. Interrupts should be turned off if the vectors are in the EEPROM. Timing and any serial communications must be done with polling during the programming process.

BYTE, ROW, ERASE, and EELAT bits can be written simultaneously or in any sequence.

EEPGM — Program and Erase Enable Bit

0 = Disables program/erase voltage to EEPROM
1 = Applies program/erase voltage to EEPROM

The EEPGM bit can be set only after EELAT has been set. When an attempt is made to set EELAT and EEPGM simultaneously, EEPGM remains clear but EELAT is set.

The BULKP, BYTE, ROW, ERASE, and EELAT bits cannot be changed when EEPGM is set. To complete a program or erase, a write to clear EEPGM and EELAT bits is required before reading the programmed data. A write to an EEPROM location has no effect when EEPGM is set. Latched address and data cannot be modified during program or erase.

A program or erase operation should follow this sequence:

1. Write BYTE, ROW, and ERASE to the desired value; write EELAT = 1.
2. Write a byte or an aligned word to an EEPROM address.
3. Write EEPGM = 1.
4. Wait for programming (t_{PROG}) or erase (t_{Erase}) delay time.
5. Write EEPGM = 0 and EELAT = 0.

To program/erase more bytes or words without intermediate EEPROM reads, only write EEPGM = 0 in step 5, leaving EELAT = 1, and jump to step 2.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 8
FLASH EEPROM

8.1 Introduction

The 32-Kbyte FLASH EEPROM module for the MC68HC912B32 and MC68HC912BC32 serves as electrically erasable and programmable, non-volatile ROM emulation memory. The module can be used for program code that must either execute at high speed or is frequently executed, such as operating system kernels and standard subroutines, or it can be used for static data which is read frequently. The FLASH EEPROM is ideal for program storage for single-chip applications allowing for field reprogramming.

NOTE

The MC68HC12BE32 and MC68HC12BC32 does not contain FLASH EEPROM.

The FLASH EEPROM array is arranged in a 16-bit configuration and may be read as either bytes, aligned words or misaligned words. Access time is one bus cycle for byte and aligned word access and two bus cycles for misaligned word operations.

The FLASH EEPROM module requires an external program/erase voltage (V_{FP}) to program or erase the FLASH EEPROM array. The external program/erase voltage is provided to the FLASH EEPROM module via an external V_{FP} pin. To prevent damage to the FLASH array, V_{FP} should always be greater than or equal to V_{DD} –0.35 V. Programming is by byte or aligned word. The FLASH EEPROM module supports bulk erase only.

The FLASH EEPROM module has hardware interlocks which protect stored data from accidental corruption. An erase- and program-protected 2-Kbyte block for boot routines is located at $7800–$7FFF or $F800–$FFFF, depending upon the mapped location of the FLASH EEPROM array. (The protected boot block on the initial mask sets, G86W and G75R, is 1-Kbyte and is located at $7C00–$7FFF or $FC00–$FFFF.)

8.2 FLASH EEPROM Array

After reset, the FLASH EEPROM array is located from addresses $8000 to $FFFF in single-chip mode. In expanded modes, the FLASH EEPROM array is located from address $0000 to $7FFF; however, it is disabled from the memory map. The FLASH EEPROM can be mapped to an alternate address range. See Chapter 5 Operating Modes and Resource Mapping.

8.3 FLASH EEPROM Registers

A 4-byte register block controls the FLASH EEPROM module operation. Configuration information is specified and programmed independently from the contents of the FLASH EEPROM array. At reset, the 4-byte register section starts at address $00F4.

Downloaded from Arrow.com.

8.3.1 FLASH EEPROM Lock Control Register

Address: $00F4

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	LOCK
Reset:	0	0	0	0	0	0	0	0

Figure 8-1. FLASH EEPROM Lock Control Register (FEELCK)

In normal modes, the LOCK bit can be written only once after reset.

LOCK — Lock Register Bit

0 = Enable write to FEEMCR register.
1 = Disable write to FEEMCR register.

8.3.2 FLASH EEPROM Module Configuration Register

Address: $00F5

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	BOOTP
Reset:	0	0	0	0	0	0	0	1

Figure 8-2. FLASH EEPROM Module Configuration Register (FEEMCR)

This register controls the operation of the FLASH EEPROM array. BOOTP cannot be changed when the LOCK control bit in the FEELCK register is set or if ENPE in the FEECTL register is set.

The boot block is located at $7800–$7FFF or $F800–$FFFF, depending upon the mapped location of the FLASH EEPROM array and mask set ($7C00–$7FFF or $FC00–$FFFF for 1-Kbyte block).

BOOTP — Boot Protect Bit

0 = Enable erase and program of 1-Kbyte or 2-Kbyte boot block.
1 = Disable erase and program of 1-Kbyte or 2-Kbyte boot block.

8.3.3 FLASH EEPROM Module Test Register

Address: $00F6

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	FSTE	GADR	HVT	FENLV	FDISVFP	VTCK	STRE	MWPR
Reset:	0	0	0	0	0	0	0	0

Figure 8-3. FLASH EEPROM Module Test Register (FEETST)

In normal mode, writes to FEETST control bits have no effect and always read 0. The FLASH EEPROM module cannot be placed in test mode inadvertently during normal operation.

FSTE — Stress Test Enable Bit

0 = Disables the gate/drain stress circuitry
1 = Enables the gate/drain stress circuitry

Downloaded from Arrow.com.

GADR — Gate/Drain Stress Test Select Bit

0 = Selects the drain stress circuitry
1 = Selects the gate stress circuitry

HVT — Stress Test High Voltage Status Bit

0 = High voltage not present during stress test
1 = High voltage present during stress test

FENLV — Enable Low Voltage Bit

0 = Disables low voltage transistor in current reference circuit
1 = Enables low voltage transistor in current reference circuit

FDISVFP — Disable Status V_{FP} Voltage Lock Bit

When the V_{FP} pin is below normal programming voltage, the FLASH module will not allow writing to the LAT bit; the user cannot erase or program the FLASH module. The FDISVFP control bit enables writing to the LAT bit regardless of the voltage on the V_{FP} pin.

0 = Enable the automatic lock mechanism if V_{FP} is low.
1 = Disable the automatic lock mechanism if V_{FP} is low.

VTCK — V_T Check Test Enable Bit

When VTCK is set, the FLASH EEPROM module uses the V_{FP} pin to control the control gate voltage; the sense amp timeout path is disabled. This allows for indirect measurements of the bit cells' program and erase threshold. If $V_{FP} < V_{ZBRK}$ (breakdown voltage), the control gate will equal the V_{FP} voltage. If $V_{FP} > V_{ZBRK}$, the control gate will be regulated by this equation:

Control gate voltage = $V_{ZBRK} + 0.44 \times (V_{FP} - V_{ZBRK})$

0 = V_T test disable
1 = V_T test enable

STRE — Spare Test Row Enable Bit

The spare test row consists of one FLASH EEPROM array row. The spare test row is reserved and contains production test information which must be maintained through several erase cycles. When STRE is set, the decoding for the spare test row overrides the address lines which normally select the other rows in the array.

0 = LIB accesses are to the FLASH EEPROM array.
1 = Spare test row in array enabled if SMOD is active

MWPR — Multiple Word Programming Bit

Used primarily for testing, if MPWR = 1, the two least significant address lines, ADDR1 and ADDR0, will be ignored when programming a FLASH EEPROM location. The word location addressed if ADDR1 and ADDR0 = 00, along with the word location addressed if ADDR1 and ADDR0 = 10, will both be programmed with the same word data from the programming latches. This bit should not be changed during programming.

0 = Multiple word programming disabled
1 = Program 32 bits of data

Downloaded from Arrow.com.

8.3.4 FLASH EEPROM Control Register

Address: $00F7

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	FEESWAI	SVFP	ERAS	LAT	ENPE
Reset:	0	0	0	0	0	0	0	0

Figure 8-4. FLASH EEPROM Control Register (FEECTL)

This register controls the programming and erasure of the FLASH EEPROM.

FEESWAI — FLASH EEPROM Stop in Wait Control Bit

0 = Do not halt FLASH EEPROM clock when in wait mode.
1 = Halt FLASH EEPROM clock when in wait mode.

NOTE

The FEESWAI bit cannot be asserted if the interrupt vector resides in the FLASH EEPROM array.

SVFP — Status V_{FP} Voltage Bit

SVFP is a read-only bit.

0 = Voltage of V_{FP} pin is below normal programming voltage levels.
1 = Voltage of V_{FP} pin is above normal programming voltage levels.

ERAS — Erase Control Bit

This bit can be read anytime or written when ENPE = 0. When set, all locations in the array will be erased at the same time. The boot block will be erased only if BOOTP = 0. This bit also affects the result of attempted array reads. See Table 8-1 for more information. Status of ERAS cannot change if ENPE is set.

0 = FLASH EEPROM configured for programming
1 = FLASH EEPROM configured for erasure

LAT — Latch Control Bit

This bit can be read anytime or written when ENPE = 0. When set, the FLASH EEPROM is configured for programming or erasure and, upon the next valid write to the array, the address and data will be latched for the programming sequence. See Table 8-1 for the effects of LAT on array reads. A high voltage detect circuit on the V_{FP} pin will prevent assertion of the LAT bit when the programming voltage is at normal levels.

0 = Programming latches disabled
1 = Programming latches enabled

ENPE — Enable Programming/Erase Bit

0 = Disables program/erase voltage to FLASH EEPROM
1 = Applies program/erase voltage to FLASH EEPROM

ENPE can be asserted only after LAT has been asserted and a write to the data and address latches has occurred. If an attempt is made to assert ENPE when LAT is negated, or if the latches have not been written to after LAT was asserted, ENPE will remain negated after the write cycle is complete. The LAT, ERAS, and BOOTP bits cannot be changed when ENPE is asserted. A write to FEECTL may affect only the state of ENPE. Attempts to read a FLASH EEPROM array location in the FLASH EEPROM module while ENPE is asserted will not return the data addressed. See Table 8-1 for more information.

Downloaded from Arrow.com.

FLASH EEPROM module control registers may be read or written while ENPE is asserted. If ENPE is asserted and LAT is negated on the same write access, no programming or erasure will be performed.

Table 8-1. Effects of ENPE, LAT, and ERAS on Array Reads

ENPE	LAT	ERAS	Result of Read
0	0	—	Normal read of location addressed
0	1	0	Read of location being programmed
0	1	1	Normal read of location addressed
1	—	—	Read cycle is ignored

8.4 Operation

The FLASH EEPROM can contain program and data. On reset, it can operate as a bootstrap memory to provide the CPU with internal initialization information during the reset sequence.

8.4.1 Bootstrap Operation Single-Chip Mode

After reset, the CPU controlling the system will begin booting up by fetching the first program address from address $FFFE.

8.4.2 Normal Operation

The FLASH EEPROM allows a byte or aligned word read/write in one bus cycle. Misaligned word read/write require an additional bus cycle. The FLASH EEPROM array responds to read operations only. Write operations are ignored.

8.4.3 Program/Erase Operation

An unprogrammed FLASH EEPROM bit has a logic state of 1. A bit must be programmed to change its state from 1 to 0. Erasing a bit returns it to a logic 1. The FLASH EEPROM has a minimum program/erase life of 100 cycles. Programming or erasing the FLASH EEPROM is accomplished by a series of control register writes and a write to a set of programming latches.

Programming is restricted to a single byte or aligned word at a time determined by internal signals SZ8 and ADDR0. The FLASH EEPROM must first be completely erased prior to programming final data values. It is possible to program a location in the FLASH EEPROM without erasing the entire array, if the new value does not require the changing of bit values from 0 to 1.

8.4.3.1 Read/Write Accesses During Program/Erase

During program or erase operations, read and write accesses may be different from those during normal operation and are affected by the state of the control bits in the FLASH EEPROM control register (FEECTL). The next write to any valid address to the array after LAT is set will cause the address and data to be latched into the programming latches. Once the address and data are latched, write accesses to the array will be ignored while LAT is set. Writes to the control registers will occur normally.

Downloaded from Arrow.com.

8.4.3.2 Program/Erase Verification

When programming or erasing the FLASH EEPROM array, a special verification method is required to ensure that the program/erase process is reliable and also to provide the longest possible life expectancy. This method requires stopping the program/erase sequence at periods of t_{PPULSE} (t_{EPULSE} for erasing) to determine if the FLASH EEPROM is programmed/erased. After the location reaches the proper value, it must continue to be programmed/erased with additional margin pulses to ensure that it will remain programmed/erased. Failure to provide the margin pulses could lead to corrupted or unreliable data.

8.4.3.3 Program/Erase Sequence

To begin a program or erase sequence, the external V_{FP} voltage must be applied and stabilized. The ERAS bit must be set or cleared, depending on whether a program sequence or an erase sequence is to occur. The LAT bit will be set to cause any subsequent data written to a valid address within the FLASH EEPROM to be latched into the programming address and data latches. The next FLASH array write cycle must be either to the location that is to be programmed if a programming sequence is being performed, or, if erasing, to any valid FLASH EEPROM array location. Writing the new address and data information to the FLASH EEPROM is followed by assertion of ENPE to turn on the program/erase voltage to program/erase the new location(s). The LAT bit must be asserted and the address and data latched to allow the setting of the ENPE control bit. If the data and address have not been latched, an attempt to assert ENPE will be ignored and ENPE will remain negated after the write cycle to FEECTL is completed. The LAT bit must remain asserted and the ERAS bit must remain in its current state as long as ENPE is asserted. A write to the LAT bit to clear it while ENPE is set will be ignored. That is, after the write cycle, LAT will remain asserted. Likewise, an attempt to change the state of ERAS will be ignored and the state of the ERAS bit will remain unchanged.

The programming software is responsible for all timing during a program sequence. This includes the total number of program pulses (n_{PP}), the length of the program pulse (t_{PPULSE}), the program margin pulses (p_m) and the delay between turning off the high voltage and verifying the operation (t_{VPROG}).

The erase software is responsible for all timing during an erase sequence. This includes the total number of erase pulses (e_m), the length of the erase pulse (t_{EPULSE}), the erase margin pulse or pulses, and the delay between turning off the high voltage and verifying the operation (t_{VERASE}).

Software also controls the supply of the proper program/erase voltage to the V_{FP} pin and should be at the proper level before ENPE is set during a program/erase sequence.

A program/erase cycle should not be in progress when starting another program/erase or while attempting to read from the array.

NOTE

Although clearing ENPE disables the program/erase voltage (V_{FP}) from the V_{FP} pin to the array, care must be taken to ensure that V_{FP} is at V_{DD} whenever programming/erasing is not in progress. Not doing so could damage the part. Ensuring that V_{FP} is always greater or equal to V_{DD} can be accomplished by controlling the V_{FP} power supply with the programming software via an output pin. Alternatively, all programming and erasing can be done prior to installing the device on an application circuit board which can always connect V_{FP} to V_{DD}. Programming can also be accomplished by plugging the board into a special programming fixture which provides program/erase voltage to the V_{FP} pin.

Downloaded from Arrow.com.

8.5 Programming the FLASH EEPROM

Programming the FLASH EEPROM is accomplished by this step-by-step procedure. The V_{FP} pin voltage must be at the proper level prior to executing step 4 the first time.

1. Apply program/erase voltage to the V_{FP} pin.
2. Clear ERAS and set the LAT bit in the FEECTL register to establish program mode and enable programming address and data latches.
3. Write data to a valid address. The address and data are latched. If BOOTP is asserted, an attempt to program an address in the boot block will be ignored.
4. Apply programming voltage by setting ENPE.
5. Delay for one programming pulse, t_{PPULSE}.
6. Remove programming voltage by clearing ENPE.
7. Delay while high voltage is turning off, t_{VPROG}.
8. Read the address location to verify that it has been programmed,
9. If the location is not programmed, repeat steps 4 through 7 until the location is programmed or until the specified maximum number of program pulses, n_{PP}, has been reached.
10. If the location is programmed, repeat the same number of pulses as required to program the location. This provides 100 percent program margin.
11. Read the address location to verify that it remains programmed.
12. Clear LAT.
13. If there are more locations to program, repeat steps 2 through 10.
14. Turn off V_{FP}. Reduce voltage on V_{FP} pin to V_{DD}.

The flowchart in Figure 8-5 demonstrates the recommended programming sequence.

Downloaded from Arrow.com.

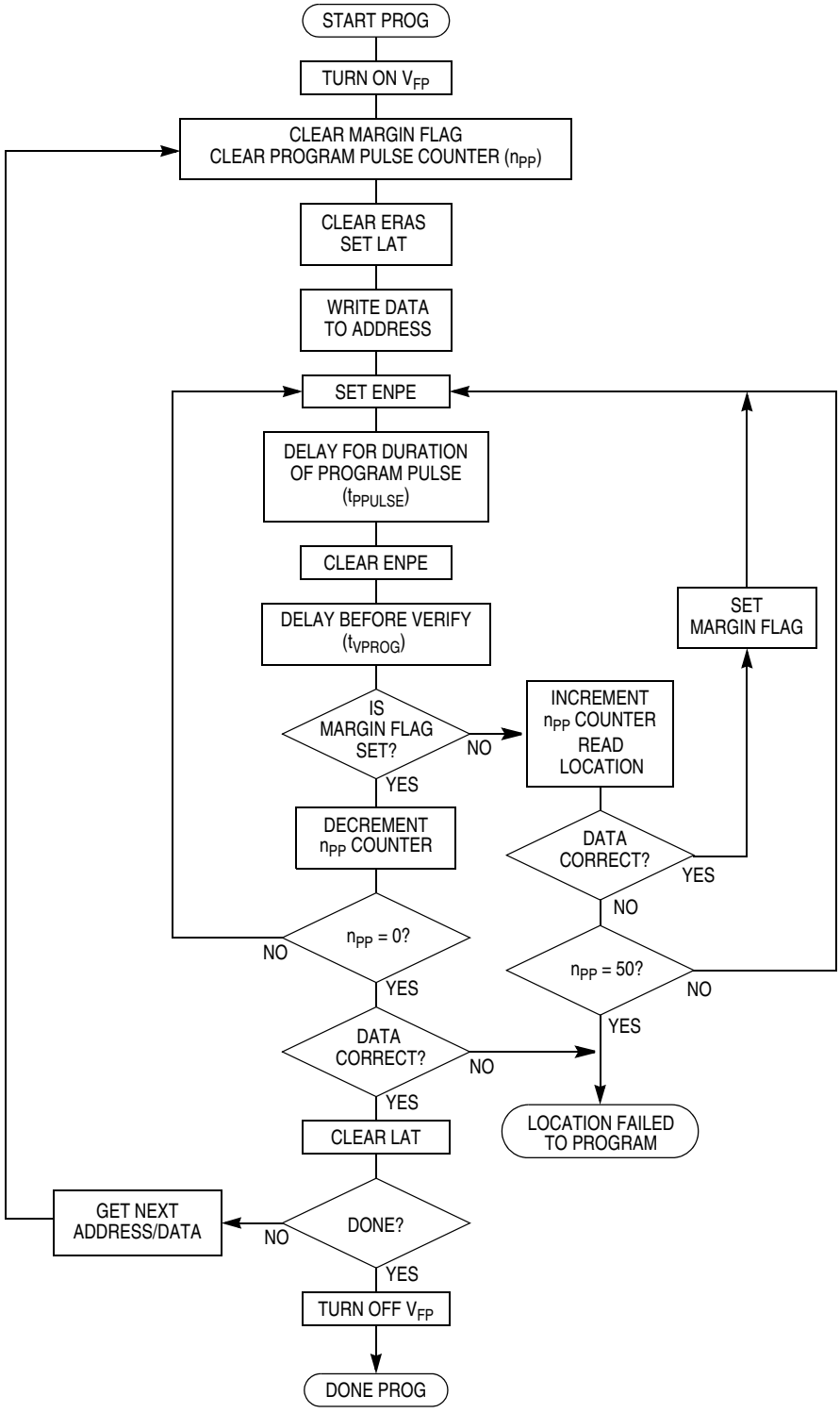


Figure 8-5. Program Sequence Flow

Downloaded from Arrow.com.

8.6 Erasing the FLASH EEPROM

This sequence demonstrates the recommended procedure for erasing the FLASH EEPROM. The V_{FP} pin voltage must be at the proper level prior to executing step 4 the first time.

1. Turn on V_{FP}. Apply program/erase voltage to the V_{FP} pin.
2. Set the LAT bit and ERAS bit to configure the FLASH EEPROM for erasing.
3. Write to any valid address in the FLASH array. This allows the erase voltage to be turned on; the data written and the address written are not important. The boot block will be erased only if the control bit BOOTP is negated.
4. Apply erase voltage by setting ENPE.
5. Delay for a single erase pulse, t_{EPULSE}.
6. Remove erase voltage by clearing ENPE.
7. Delay while high voltage is turning off, t_{VERASE}.
8. Read the entire array to ensure that the FLASH EEPROM is erased.
9. If all of the FLASH EEPROM locations are not erased, repeat steps 4 through 7 until either the remaining locations are erased or until the maximum erase pulses have been applied, n_{EP}.
10. If all of the FLASH EEPROM locations are erased, repeat the same number of pulses as required to erase the array. This provides 100 percent erase margin.
11. Read the entire array to ensure that the FLASH EEPROM is erased.
12. Clear LAT.
13. Turn off V_{FP}. Reduce voltage on V_{FP} pin to V_{DD}.

The flowchart in Figure 8-6 demonstrates the recommended erase sequence.

Downloaded from Arrow.com.

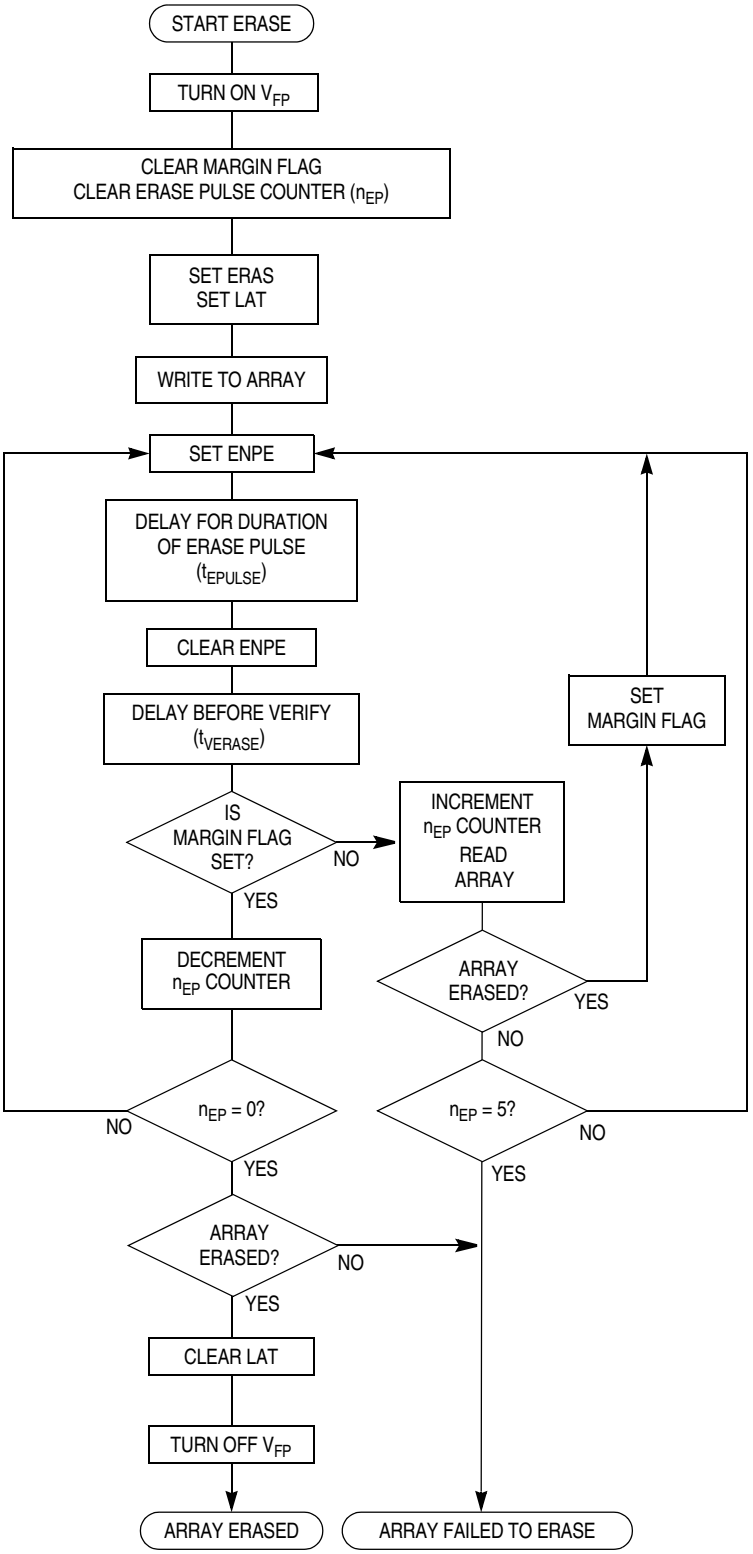


Figure 8-6. Erase Sequence Flow

Downloaded from Arrow.com.

8.7 Program/Erase Protection Interlocks

The FLASH EEPROM program and erase mechanisms provide maximum protection from accidental programming or erasure.

The voltage required to program/erase the FLASH EEPROM (V_{FP}) is supplied via an external pin. If V_{FP} is not present, no programming/erasing will occur. Furthermore, the program/erase voltage will not be applied to the FLASH EEPROM unless turned on by setting a control bit (ENPE). The ENPE bit may not be set unless the programming address and data latches have been written previously with a valid address. The latches may not be written unless enabled by setting a control bit (LAT). The LAT and ENPE control bits must be written on separate writes to the control register (FEECTL) and must be separated by a write to the programming latches. The ERAS and LAT bits are also protected when ENPE is set. This prevents inadvertent switching between erase/program mode and also prevents the latched data and address from being changed after a program cycle has been initiated.

8.8 Stop or Wait Mode

When STOP or WAIT commands are executed, the MCU puts the FLASH EEPROM in stop or wait mode. In these modes, the FLASH module will cease erasure or programming immediately.

NOTE

It is advised to not enter stop or wait modes when programming the FLASH array. The FLASH EEPROM module is not able to recover from stop mode without a 250-ns delay. If the operating bus frequency is greater than 4 MHz, the DLY bit must be set to1 to use the FLASH after recovering from stop mode. Other options are to map the EEPROM module over the FLASH module in the memory map with DLY = 0 and place the interrupt vectors in the EEPROM array or use reset to recover from a stop mode executed from FLASH EEPROM. Recovery from a STOP instruction executed from EEPROM and RAM operates normally.

8.9 Test Mode

The FLASH EEPROM has some special test functions which are only accessible when the device is in test mode. Test mode is indicated to the FLASH EEPROM module when the SMOD line on the LIB is asserted. When SMOD is asserted, the special test control bits may be accessed via the LIB to invoke the special test functions in the FLASH EEPROM module. When SMOD is not asserted, writes to the test control bits have no effect and all bits in the test register FEETST will be cleared. This ensures that FLASH EEPROM test mode cannot be invoked inadvertently during normal operation.

The FLASH EEPROM module will operate normally, even if SMOD is asserted, until a special test function is invoked. The test mode adds additional features over normal mode. These features allow the tests to be performed even after the device is installed in the final product.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 9
Read-Only Memory (ROM)

9.1 Introduction

The MC68HC12BE32 and MC68HC12BC32 contain 32 Kbytes of read-only memory (ROM). The ROM array is arranged in a 16-bit configuration and may be read as either bytes, aligned words or misaligned words. Access time is one bus cycle for byte and aligned word access and two bus cycles for misaligned word operations.

9.2 ROM Array

After reset, the ROM array is located from addresses $8000 to $FFFF in single-chip mode. In expanded modes, the ROM array is located from address $0000 to $7FFF; however, it is disabled from the memory map. The ROM can be mapped to an alternate address range. See Chapter 5 Operating Modes and Resource Mapping.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 10
Clock Generation Module (CGM)

10.1 Introduction

The clock generation module (CGM) generates the system clocks and generates and controls the timing of the reset and power-on reset (POR) logic. The CGM is composed of:

- Clock selection and generation circuitry
- Slow-mode clock divider
- Reset and stop generation timing and control

NOTE

Older device mask sets do not support the slow-mode clock divider feature. Register $00E0 is reserved in older devices and provides no function.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC912B32 include: G96P, G86W, and H91F.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC12BE32 include: H54T and J38M.

Mask sets that do not have the slow-mode clock divider feature on the MC68HC(9)12BC32 include: J15G.

10.2 Block Diagram

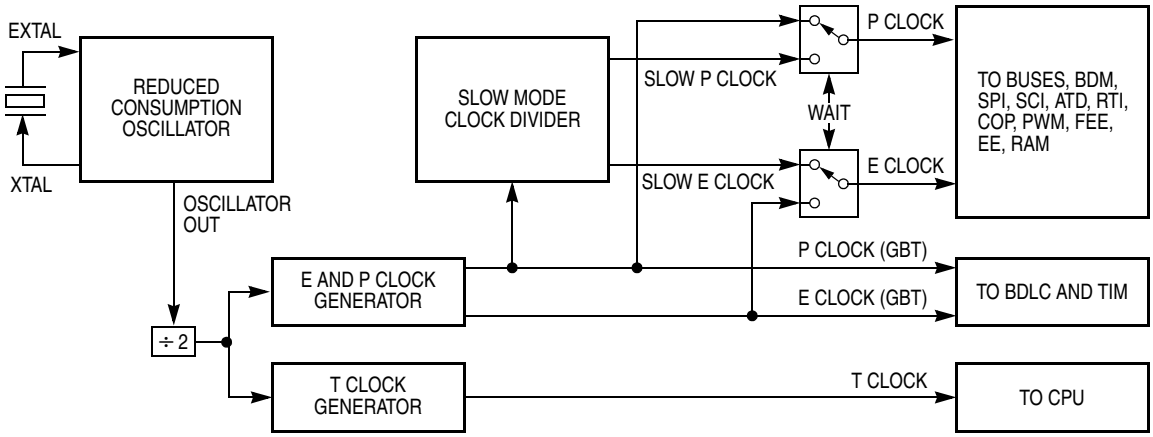


Figure 10-1. CGM Block Diagram

Downloaded from Arrow.com.

10.3 Register Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0014	Real-Time Interrupt Control Register (RTICTL) See page 118.	Read:	RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	Real-Time Interrupt Flag Register (RTIFLG) See page 119.	Read:	RTIF	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0016	COP Control Register (COPCTL) See page 119.	Read:	CME	FCME	FCM	FCOP	DISR	CR2	CR1	CR0
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$0017	Arm/Reset COP Timer Register (COPRST) See page 120.	Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$00E0	Slow Mode Divider Register (SLOW) See page 117.	Read:	0	0	0	0	0	SLDV2	SLDV1	SLDV0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-2. CGM Register Map

10.4 Clock Selection and Generation

The CGM generates the P clock, the E clock, and four T clocks. The P clock and E clock are used by all device modules except the CPU. The T clocks are used by the CPU.

Figure 10-3 shows clock timing relationships while in normal run modes.

There are two types of P clocks and E clocks while in wait mode:

- Global type (G), which is driven by the slow clock divider in wait mode and drives all on-chip peripherals except the BDLC and the TIMER
- Global type (GBT), which remains at the oscillator divide-by-2 rate in wait mode and drives the BDLC and the TIMER

Figure 10-4 shows clock timing relationships while in wait mode.

Downloaded from Arrow.com.

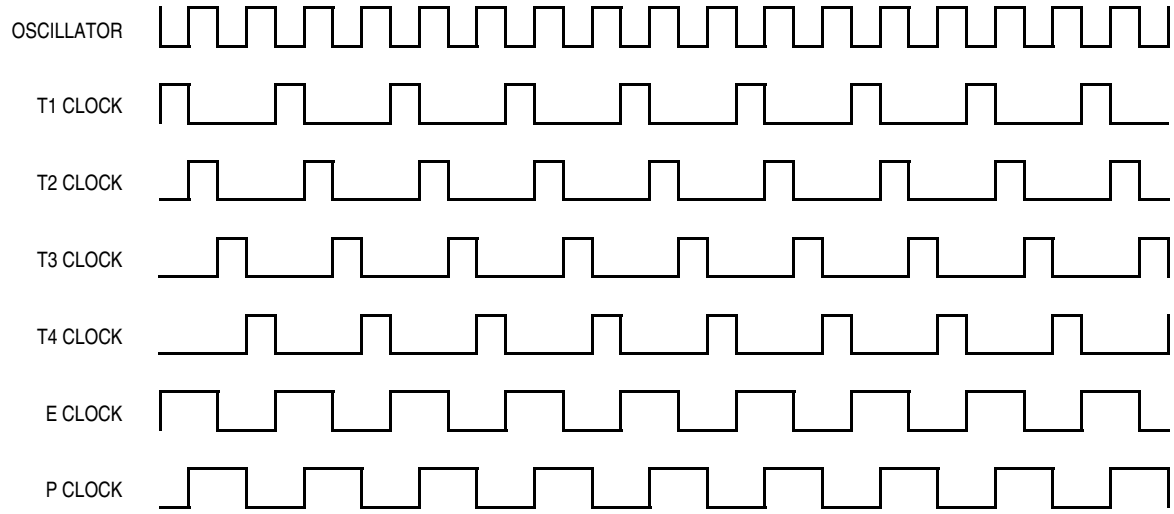
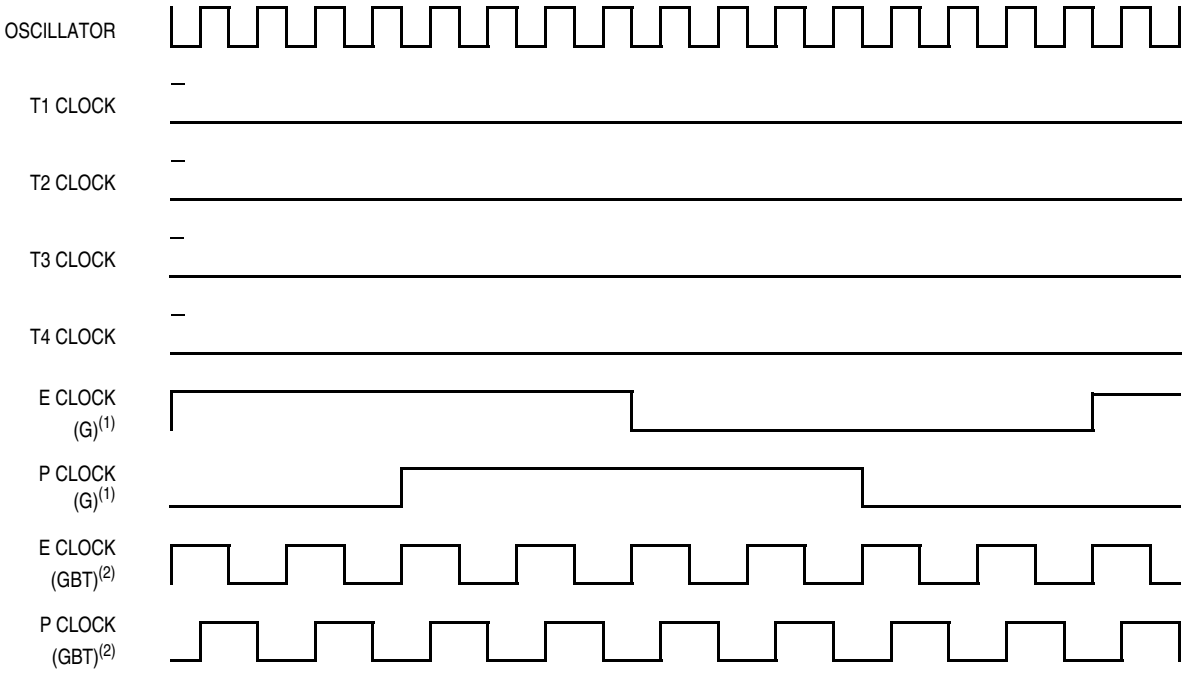


Figure 10-3. Internal Clock Relationships in Normal Run Modes



Notes:
1. Driven by slow clock divider in wait mode. Drives on-chip peripherals except BDLC and timer.
2. Remains at oscillator divided by 2 rate in wait mode. Drives BDLC and timer.

Figure 10-4. Internal Clock Relationships in Wait Mode

Downloaded from Arrow.com.

10.5 Slow Mode Divider

The slow mode divider is included to deliver a variable bus frequency to the MCU in wait mode. The bus clocks are derived from the constant P clock. The slow clock counter divides the P clock and E clock frequency in powers of 2, up to 128. When the slow control register is cleared or the part is not in wait mode, the slow mode divider is off and the bus clock's frequency is not changed.

NOTE

The clock monitor is clocked by the system clock (oscillator) reference; the slow mode divider allows operation of the MCU at clock periods longer than the clock monitor trigger time.

10.6 Clock Functions

The CGM generates and controls the timing of the reset and POR logic.

10.6.1 Computer Operating Properly (COP)

The computer operating properly (COP) or watchdog timer is an added check that a program is running and sequencing properly. When the COP is being used, software is responsible for keeping a free-running watchdog timer from timing out. If the watchdog timer times out, it is an indication that the software is no longer being executed in the intended sequence; thus, a system reset is initiated. Three control bits allow selection of seven COP timeout periods or COP disable. When COP is enabled, sometime during the selected period the program must write $55 and $AA (in this order) to the arm/reset COP register (COPRST). If the program fails to do this, the part resets. If any value other than $55 or $AA is written to COPRST, the part is reset.

10.6.2 Real-Time Interrupt

There is a real-time (periodic) interrupt available to the user. This interrupt occurs at one of seven selected rates. An interrupt flag and an interrupt enable bit are associated with this function. There are three bits for the rate select.

10.6.3 Clock Monitor

The clock monitor circuit is based on an internal resistor-capacitor (RC) time delay. If no MCU clock edges are detected within this RC time delay, the clock monitor can optionally generate a system reset. The clock monitor function is enabled/disabled by the CME control bit in the COP control register (COPCTL). This timeout is based on an RC delay so that the clock monitor can operate without any MCU clocks.

Clock monitor timeouts are shown in Table 10-1.

Table 10-1. Clock Monitor Timeout

Supply	Range
5 V ± 10%	2–20 µs

Downloaded from Arrow.com.

10.7 Clock Registers

This section describes the clock registers. All register addresses shown reflect the reset state. Registers may be mapped to any 2-Kbyte space.

10.7.1 Slow Mode Divider Register

Address: $00E0

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	SLDV2	SLDV1	SLDV0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-5. Slow Mode Divider Register (SLOW)

SLDV2–SLDV0 — Slow Mode Divisor Selector Bits

The value 2 raised to the power indicated by these three bits produces the slow mode frequency divider. The range of the divider is 2 to 128 by steps of power of 2. When the bits are clear, the divider is bypassed. Table 10-2 shows the divider for all bit conditions and the resulting bus rate for three example oscillator frequencies.

Table 10-2. Slow Mode Register Divider Rates

SLDV2	SLDV1	SLDV0	Divider (2^x)	Bus Rate (16-MHz Oscillator)	Bus Rate (8-MHz Oscillator)	Bus Rate (4-MHz Oscillator)
0	0	0	Off	8 MHz	4 MHz	2 MHz
0	0	1	2	4 MHz	2 MHz	1 MHz
0	1	0	4	2 MHz	1 MHz	500 kHz
0	1	1	8	1 MHz	500 kHz	250 kHz
1	0	0	16	500 kHz	250 kHz	125 kHz
1	0	1	32	250 kHz	125 kHz	62.5 kHz
1	1	0	64	125 kHz	62.5 kHz	31.2 kHz
1	1	1	128	62.5 kHz	31.2 kHz	15.6 kHz

Downloaded from Arrow.com.

10.7.2 Real-Time Interrupt Control Register

Address: $0014

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RTIE	RSWAI	RSBCK	0	RTBYP	RTR2	RTR1	RTR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-6. Real-Time Interrupt Control Register (RTICTL)

Read: Anytime

Write: Varies on a bit-by-bit basis

RTIE — Real-Time Interrupt Enable Bit

Write anytime.

0 = Interrupt requests from RTI are disabled.

1 = Interrupt is requested when RTI is set.

RSWAI — RTI and COP Stop While in Wait Bit

Write once in normal modes, anytime in special modes.

0 = Allows the RTI and COP to continue running in wait

1 = Disables both the RTI and COP when the part goes into wait

RSBCK — RTI and COP Stop While in Background Debug Mode Bit

Write once in normal modes, anytime in special modes.

0 = Allows the RTI and COP to continue running while in background mode

1 = Disables RTI and COP when the part is in background mode (useful for emulation)

RTBYP — Real-Time Interrupt Divider Chain Bypass Bit

Write is not allowed in normal modes, anytime in special modes.

0 = Divider chain functions normally.

1 = Divider chain is bypassed, allows faster testing. The divider chain is normally P divided by 2^{13}, when bypass becomes P divided by 4.

RTR2, RTR1, and RTR0 — Real-Time Interrupt Rate Select Bits

Write anytime.

Rate select for real-time interrupt. The E clock is used for this module.

Table 10-3. Real-Time Interrupt Rates

RTR2	RTR1	RTR0	Divide E By:	Timeout Period E = 4.0 MHz	Timeout Period E = 8.0 MHz
0	0	0	OFF	OFF	OFF
0	0	1	2^{13}	2.048 ms	1.024 ms
0	1	0	2^{14}	4.096 ms	2.048 ms
0	1	1	2^{15}	8.196 ms	4.096 ms
1	0	0	2^{16}	16.384 ms	8.196 ms
1	0	1	2^{17}	32.768 ms	16.384 ms
1	1	0	2^{18}	65.536 ms	32.768 ms
1	1	1	2^{19}	131.72 ms	65.536 ms

Downloaded from Arrow.com.

10.7.3 Real-Time Interrupt Flag Register

Address: $0015

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	RTIF	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 10-7. Real-Time Interrupt Flag Register (RTIFLG)

RTIF — Real-Time Interrupt Flag Bit

This bit is cleared automatically by a write to this register with this bit set.

0 = Timeout has not yet occurred.

1 = Set when the timeout period is met

10.7.4 COP Control Register

Address: $0016

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CME	FCME	FCM	FCOP	DISR	CR2	CR1	CR0
Normal Reset:	0	0	0	0	0	0	0	1
Special Reset:	0	0	0	0	1	0	0	1

Figure 10-8. COP Control Register (COPCTL)

Read: Anytime

Write: Varies on a bit by bit basis

CME — Clock Monitor Enable Bit

Write anytime.

If FCME is set, this bit has no meaning or effect.

0 = Clock monitor is disabled; slow clocks and STOP instruction may be used.

1 = Slow or stopped clocks (including the STOP instruction) cause a clock reset sequence.

FCME — Force Clock Monitor Enable Bit

Write once in normal modes, anytime in special modes.

In normal modes, when this bit is set, the clock monitor function cannot be disabled until a reset occurs.

0 = Clock monitor follows the state of the CME bit.

1 = Slow or stopped clocks cause a clock reset sequence.

To use both STOP and clock monitor, the CME bit should be cleared prior to executing a STOP instruction and set after recovery from STOP. Always keep FCME = 0, if STOP will be used.

FCM — Force Clock Monitor Reset Bit

Writes are not allowed in normal modes, anytime in special modes.

If DISR is set, this bit has no effect.

0 = Normal operation

1 = Force a clock monitor reset, if clock monitor is enabled.

Downloaded from Arrow.com.

FCOP — Force COP Watchdog Reset Bit

Writes are not allowed in normal modes; can be written anytime in special modes.
If DISR is set, this bit has no effect.

0 = Normal operation
1 = Force a COP reset, if COP is enabled.

DISR — Disable Resets from COP Watchdog and Clock Monitor Bit

Writes are not allowed in normal modes, anytime in special modes.

0 = Normal operation
1 = Regardless of other control bit states, COP and clock monitor do not generate a system reset.

CR2, CR1, and CR0 — COP Watchdog Timer Rate Select Bit

The COP system is driven by a constant frequency of $E/2^{13}$. (RTBYP in the RTICTL register allows all but two stages of this divider to be bypassed for testing in special modes only.) These bits specify an additional division factor to arrive at the COP timeout rate. The clock used for this module is the E clock.

Write once in normal modes, anytime in special modes.

Table 10-4. COP Watchdog Rates (RTBYP = 0)

CR2	CR1	CR0	Divide E By:	At E = 4.0-MHz Timeout 0 to +2.048 ms	At E = 8.0-MHz Timeout 0 to +1.024 ms
0	0	0	OFF	OFF	OFF
0	0	1	2^{13}	2.048 ms	1.024 ms
0	1	0	2^{15}	8.1920 ms	4.096 ms
0	1	1	2^{17}	32.768 ms	16.384 ms
1	0	0	2^{19}	131.072 ms	65.536 ms
1	0	1	2^{21}	524.288 ms	262.144 ms
1	1	0	2^{22}	1.048 s	524.288 ms
1	1	1	2^{23}	2.097 s	1.048576 s

10.7.5 Arm/Reset COP Timer Register

Address: $0017

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 10-9. Arm/Reset COP Timer Register (COPRST)

Always reads $00.

Writing $55 to this address is the first step of the COP watchdog sequence.

Writing $AA to this address is the second step of the COP watchdog sequence. Other instructions may be executed between these writes but both must be completed in the correct order prior to timeout to avoid a watchdog reset. Writing anything other than $55 or $AA causes a COP reset to occur.

Downloaded from Arrow.com.

10.8 Clock Divider Chains

Figure 10-10, Figure 10-11, Figure 10-12, and Figure 10-13 summarize the clock divider chains for these peripherals:

- SCI — Serial peripheral interface
- BDLC — Byte data link communications
- RTI — Real-time interrupt
- COP — Computer operating properly
- TIM — Standard timer module
- ECT — Enhanced capture timer
- SPI — Serial peripheral interface
- ATD — Analog-to-digital converter
- BDM — Background debug mode

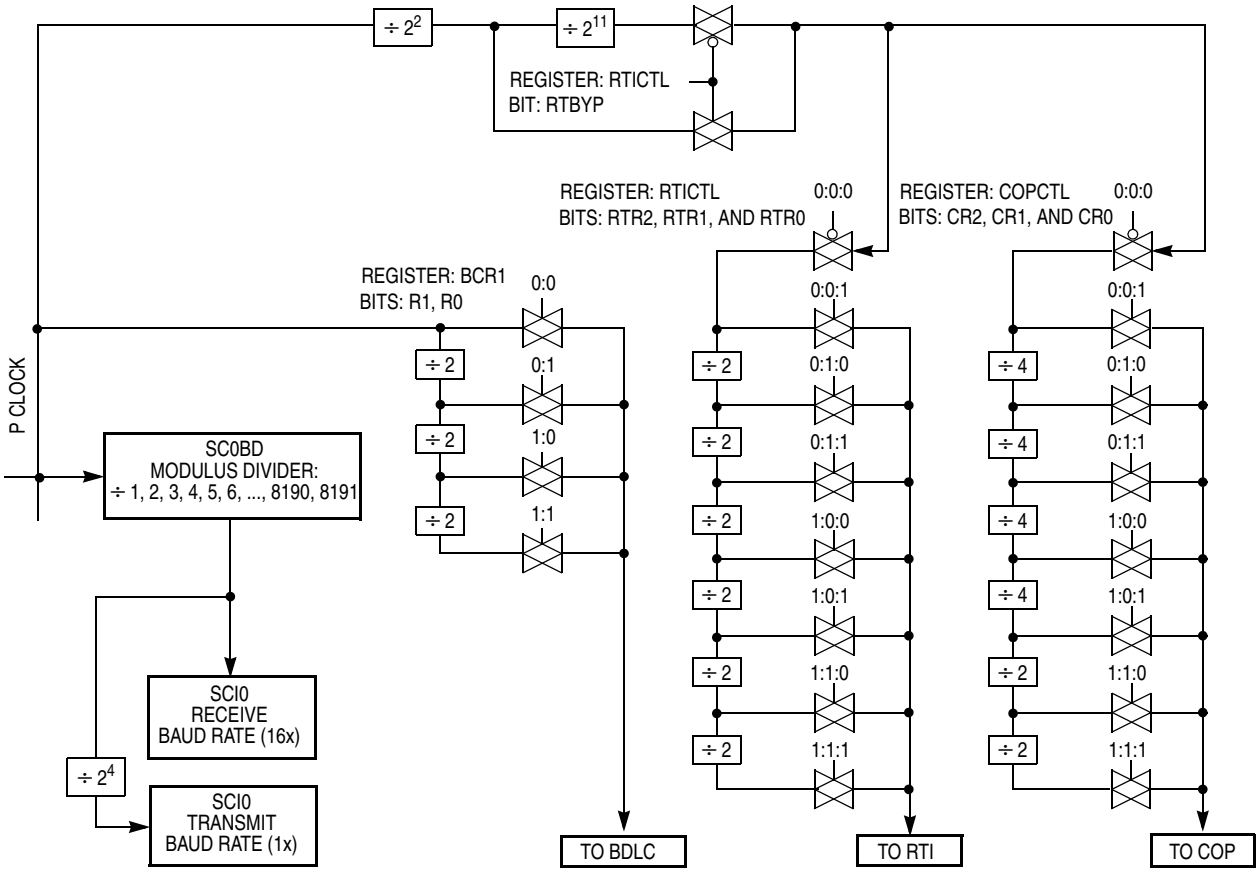


Figure 10-10. Clock Chain for SCI, BDLC, RTI, and COP

Downloaded from Arrow.com.

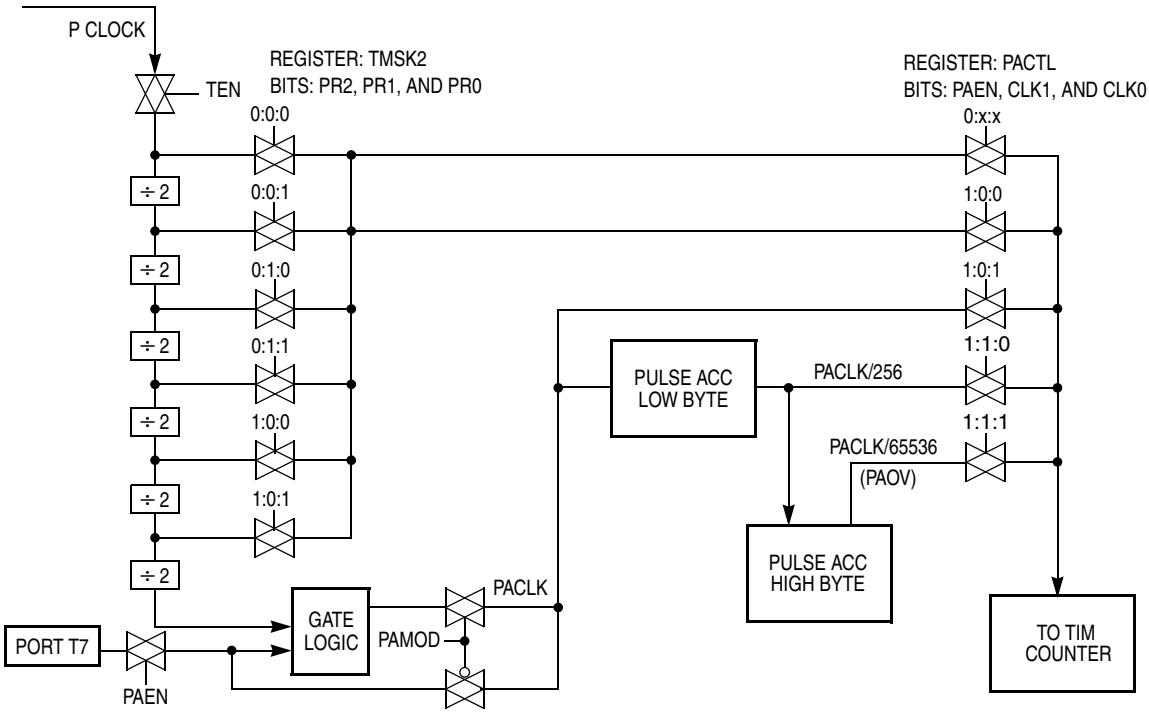


Figure 10-11. Clock Chain for TIM

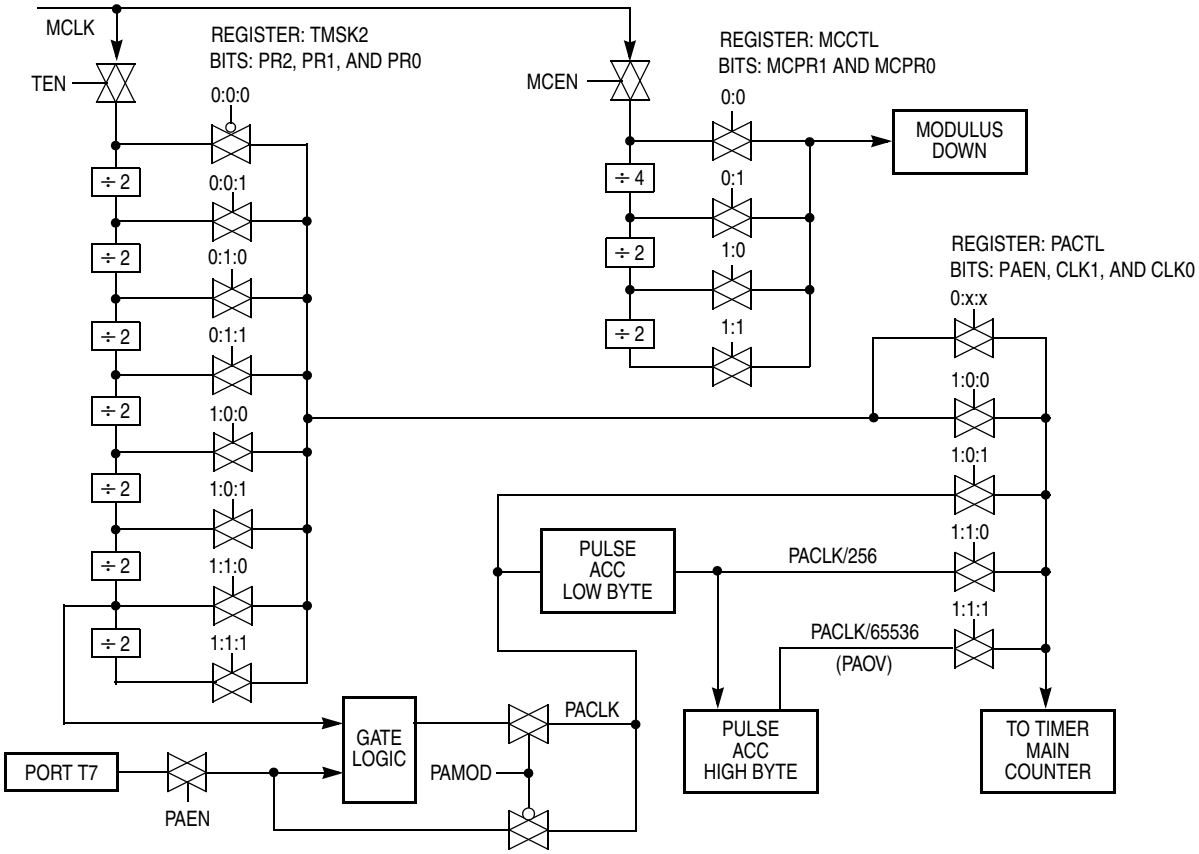


Figure 10-12. Clock Chain for ECT

Downloaded from Arrow.com.

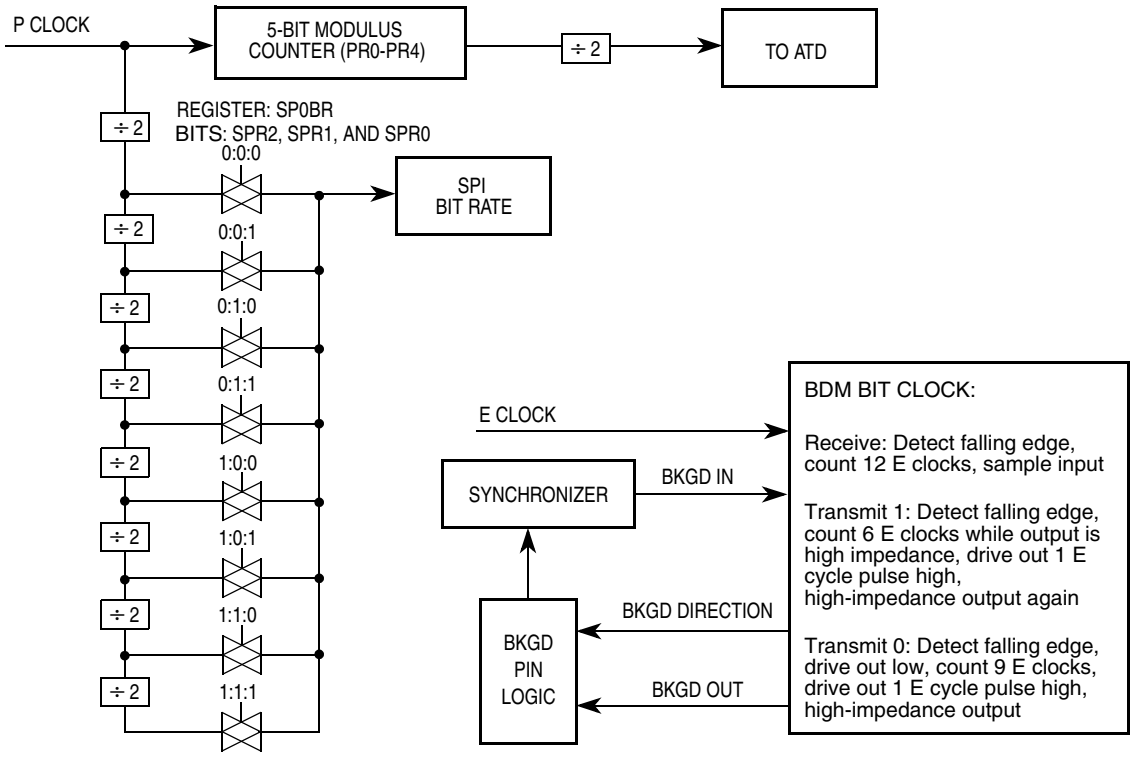


Figure 10-13. Clock Chain for SPI, ATD, and BDM

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 11
Pulse-Width Modulator (PWM)

11.1 Introduction

The pulse-width modulator (PWM) subsystem provides four independent 8-bit PWM waveforms or two 16-bit PWM waveforms or a combination of one 16-bit and two 8-bit PWM waveforms. Each waveform channel has a programmable period and a programmable duty cycle as well as a dedicated counter. A flexible clock select scheme allows four different clock sources to be used with the counters. Each of the modulators can create independent, continuous waveforms with software-selectable duty rates from 0 percent to 100 percent. The PWM outputs can be programmed as left-aligned outputs or center-aligned outputs. See Figure 11-1, Figure 11-2, and Figure 11-3.

The period and duty registers are double buffered so that if they change while the channel is enabled, the change does not take effect until the counter rolls over or the channel is disabled. If the channel is not enabled, then writes to the period and/or duty register go directly to the latches as well as the buffer, thus ensuring that the PWM output is always either the old waveform or the new waveform, not some variation in between.

A change in duty or period can be forced into immediate effect by writing the new value to the duty and/or period registers and then writing to the counter. This causes the counter to reset and the new duty and/or period values to be latched. In addition, since the counter is readable it is possible to know where the count is with respect to the duty value and software can be used to make adjustments by turning the enable bit off and on.

The four PWM channel outputs share general-purpose port P pins. Enabling PWM pins takes precedence over the general-purpose port. When PWM outputs are not in use, the port pins may be used for discrete input/output.

Downloaded from Arrow.com.

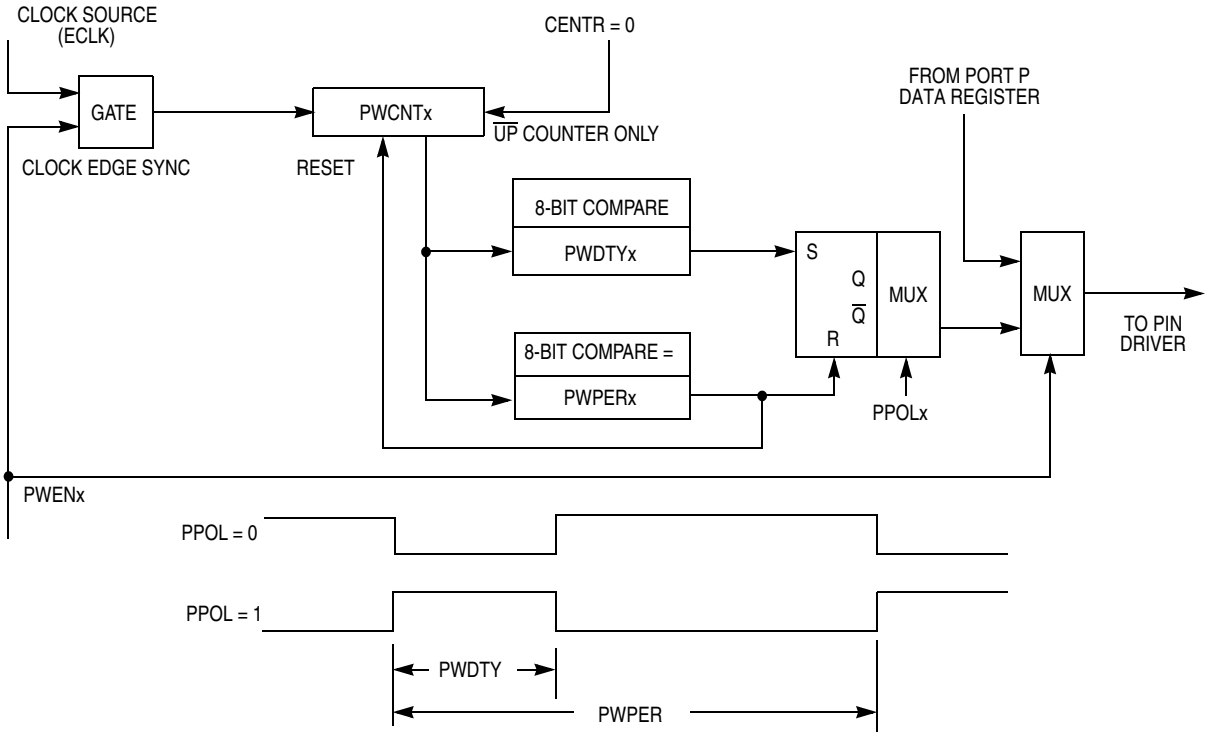


Figure 11-1. Block Diagram of PWM Left-Aligned Output Channel

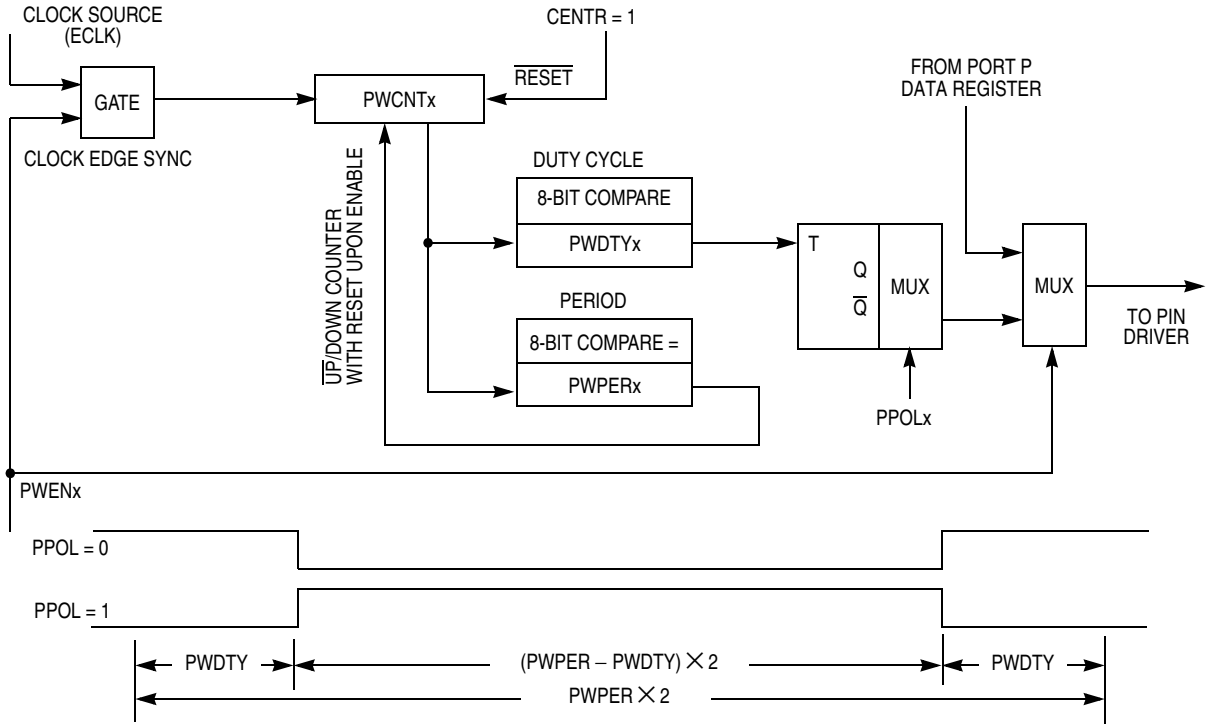


Figure 11-2. Block Diagram of PWM Center-Aligned Output Channel

Downloaded from Arrow.com.

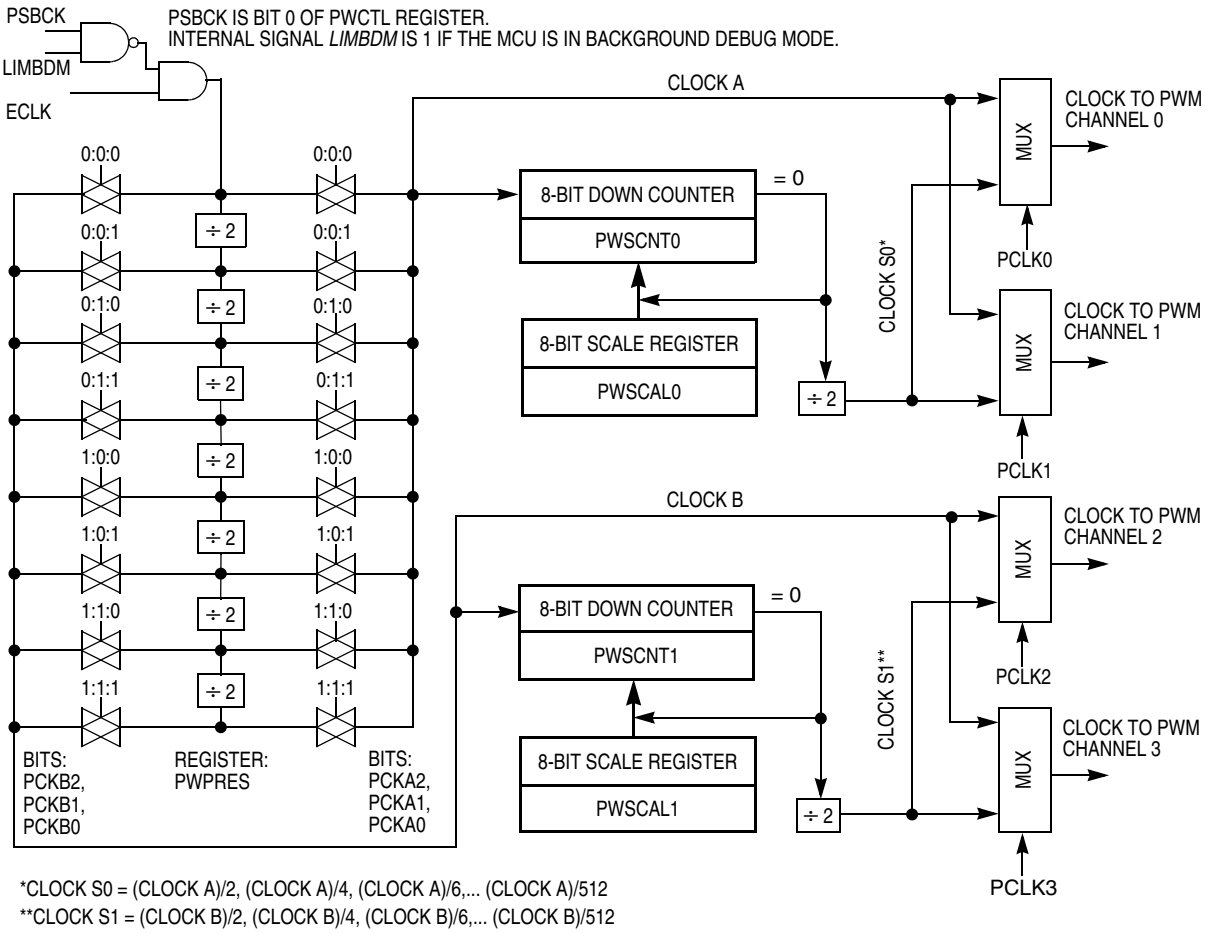


Figure 11-3. PWM Clock Sources

Downloaded from Arrow.com.

11.2 PWM Register Descriptions

This section provides descriptions of the PWM registers.

11.2.1 PWM Clocks and Concatenate Register

Address: $0040

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CON23	CON01	PCKA2	PCKA1	PCKA0	PCKB2	PCKB1	PCKB0
Reset:	0	0	0	0	0	0	0	0

Figure 11-4. PWM Clocks and Concatenate Register (PWCLK)

Read: Anytime

Write: Anytime

CON23 — Concatenate PWM Channels 2 and 3 Bit

When concatenated, channel 2 becomes the high-order byte and channel 3 becomes the low-order byte. Channel 2 output pin is used as the output for this 16-bit PWM (bit 2 of port P). Channel 3 clock-select control bits determines the clock source.

0 = Channels 2 and 3 are separate 8-bit PWMs.

1 = Channels 2 and 3 are concatenated to create one 16-bit PWM channel.

CON01 — Concatenate PWM Channels 0 and 1 Bit

When concatenated, channel 0 becomes the high-order byte and channel 1 becomes the low-order byte. Channel 0 output pin is used as the output for this 16-bit PWM (bit 0 of port P). Channel 1 clock-select control bits determine the clock source.

0 = Channels 0 and 1 are separate 8-bit PWMs.

1 = Channels 0 and 1 are concatenated to create one 16-bit PWM channel.

PCKA2–PCKA0 — Prescaler for Clock A Bits

Clock A is one of two clock sources which may be used for channels 0 and 1. These three bits determine the rate of clock A, as shown in Table 11-1.

PCKB2–PCKB0 — Prescaler for Clock B Bits

Clock B is one of two clock sources which may be used for channels 2 and 3. These three bits determine the rate of clock B, as shown in Table 11-1.

Table 11-1. Clock A and Clock B Prescaler

PCKA2 (PCKB2)	PCKA1 (PCKB1)	PCKA0 (PCKB0)	Value of Clock A (B)
0	0	0	E
0	0	1	E ÷ 2
0	1	0	E ÷ 4
0	1	1	E ÷ 8
1	0	0	E ÷ 16
1	0	1	E ÷ 32
1	1	0	E ÷ 64
1	1	1	E ÷ 128

Downloaded from Arrow.com.

11.2.2 PWM Clock Select and Polarity Register

Address: $0041

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PCLK3	PCLK2	PCLK1	PCLK0	PPOL3	PPOL2	PPOL1	PPOL0
Reset:	0	0	0	0	0	0	0	0

Figure 11-5. PWM Clock Select and Polarity Register (PWPOL)

Read: Anytime

Write: Anytime

PCLK3 — PWM Channel 3 Clock Select Bit

0 = Clock B is the clock source for channel 3.
1 = Clock S1 is the clock source for channel 3.

PCLK2 — PWM Channel 2 Clock Select Bit

0 = Clock B is the clock source for channel 2.
1 = Clock S1 is the clock source for channel 2.

PCLK1 — PWM Channel 1 Clock Select Bit

0 = Clock A is the clock source for channel 1.
1 = Clock S0 is the clock source for channel 1.

PCLK0 — PWM Channel 0 Clock Select Bit

0 = Clock A is the clock source for channel 0.
1 = Clock S0 is the clock source for channel 0.

If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse may occur during the transition.

PPOL3 — PWM Channel 3 Polarity Bit

0 = Channel 3 output is low at the beginning of the period, high when the duty count is reached.
1 = Channel 3 output is high at the beginning of the period, low when the duty count is reached.

PPOL2 — PWM Channel 2 Polarity Bit

0 = Channel 2 output is low at the beginning of the period, high when the duty count is reached.
1 = Channel 2 output is high at the beginning of the period, low when the duty count is reached.

PPOL1 — PWM Channel 1 Polarity Bit

0 = Channel 1 output is low at the beginning of the period, high when the duty count is reached.
1 = Channel 1 output is high at the beginning of the period, low when the duty count is reached.

PPOL0 — PWM Channel 0 Polarity Bit

0 = Channel 0 output is low at the beginning of the period, high when the duty count is reached.
1 = Channel 0 output is high at the beginning of the period, low when the duty count is reached.

Depending on the polarity bit, the duty registers may contain the count of either the high time or the low time. If the polarity bit is 0 and left alignment is selected, the duty registers contain a count of the low time. If the polarity bit is 1, the duty registers contain a count of the high time.

Downloaded from Arrow.com.

11.2.3 PWM Enable Register

Address: $0042

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	PWEN3	PWEN2	PWEN1	PWEN0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-6. PWM Enable Register (PWEN)

Read: Anytime

Write: Anytime

Setting any of the PWENx bits causes the associated port P line to become an output regardless of the state of the associated data direction register (DDRP) bit. This does not change the state of the data direction bit. When PWENx returns to 0, the data direction bit controls I/O direction. On the front end of the PWM channel, the scaler clock is enabled to the PWM circuit by the PWENx enable bit being high. When all four PWM channels are disabled, the prescaler counter shuts off to save power. There is an edge-synchronizing gate circuit to guarantee that the clock is only enabled or disabled at an edge.

PWEN3 — PWM Channel 3 Enable Bit

The pulse modulated signal will be available at port P bit 3 when its clock source begins its next cycle.

0 = Channel 3 disabled
1 = Channel 3 enabled

PWEN2 — PWM Channel 2 Enable Bit

The pulse modulated signal will be available at port P bit 2 when its clock source begins its next cycle.

0 = Channel 2 disabled
1 = Channel 2 enabled

PWEN1 — PWM Channel 1 Enable Bit

The pulse modulated signal will be available at port P bit 1 when its clock source begins its next cycle.

0 = Channel 1 disabled
1 = Channel 1 enabled

PWEN0 — PWM Channel 0 Enable Bit

The pulse modulated signal will be available at port P bit 0 when its clock source begins its next cycle.

0 = Channel 0 disabled
1 = Channel 0 enabled

Downloaded from Arrow.com.

11.2.4 PWM Prescale Counter

Address: $0043

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-7. PWM Prescale Counter (PWPRES)

Read: Anytime

Write: Only in special mode (SMOD = 1)

PWPRES is a free-running 7-bit counter.

11.2.5 PWM Scale Register 0

Address: $0044

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 11-8. PWM Scale Register 0 (PWSCAL0)

Read: Anytime

Write: Anytime

A write causes the scaler counter PWSCNT0 to load the PWSCAL0 value unless it is in special mode with DISCAL = 1 in the PWTST register.

PWM channels 0 and 1 can select clock S0 (scaled) as its input clock by setting the control bit PCLK0 and PCLK1 respectively. Clock S0 is generated by dividing clock A by the value in the PWSCAL0 register plus one and dividing again by two. When PWSCAL0 = $FF, clock A is divided by 256 then divided by two to generate clock S0.

11.2.6 PWM Scale Counter 0 Value

Address: $0045

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-9. PWM Scale Counter Register 0 (PWSCNT0)

Read: Anytime

PWSCNT0 is a down-counter that, upon reaching $00, loads the value of PWSCAL0.

Downloaded from Arrow.com.

11.2.7 PWM Scale Register 1

Address: $0046

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 11-10. PWM Scale Register 1 (PWSCAL1)

Read: Anytime

Write: Anytime

A write causes the scaler counter PWSCNT1 to load the PWSCAL1 value unless it is in special mode with DISCAL = 1 in the PWTST register.

PWM channels 2 and 3 can select clock S1 (scaled) as its input clock by setting the control bit PCLK2 and PCLK3 respectively. Clock S1 is generated by dividing clock B by the value in the PWSCAL1 register plus one and dividing again by two. When PWSCAL1 = $FF, clock B is divided by 256 then divided by two to generate clock S1.

11.2.8 PWM Scale Counter 1 Value

Address: $0047

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-11. PWM Scale Counter 1 Value (PWSCNT1)

Read: Anytime

PWSCNT1 is a down-counter that, upon reaching $00, loads the value of PWSCAL1.

Downloaded from Arrow.com.

11.2.9 PWM Channel Counters 0–3

Address: $0048

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 11-12. PWM Channel Counter 0 (PWCNT0)

Address: $0049

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 11-13. PWM Channel Counter 1 (PWCNT1)

Address: $004A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 11-14. PWM Channel Counter 2 (PWCNT2)

Address: $004B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 11-15. PWM Channel Counter 3 (PWCNT3)

Read: Anytime

Write: Anytime

A write causes the PWM counter to reset to $00.
In special mode, if DISCR = 1, a write does not reset the PWM counter.

Each counter may be read anytime without affecting the count or the operation of the corresponding PWM channel. Writes to a counter cause the counter to be reset to $00 and force an immediate load of both duty and period registers with new values. To avoid a truncated PWM period, write to a counter while the counter is disabled. In left-aligned output mode, resetting the counter and starting the waveform output is controlled by a match between the period register and the value in the counter. In center-aligned output mode, the counters operate as up/down counters, where a match in period changes the counter direction. The duty register changes the state of the output during the period to determine the duty.

When a channel is enabled, the associated PWM counter starts at the count in the PWCNTx register using the clock selected for that channel.

In special mode, when DISCP = 1 and is configured for left-aligned output, a match of period does not reset the associated PWM counter.

Downloaded from Arrow.com.

11.2.10 PWM Channel Period Registers 0–3

Address: $004C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-16. PWM Channel Period Register 0 (PWPER0)

Address: $004D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-17. PWM Channel Period Register 1 (PWPER1)

Address: $004E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-18. PWM Channel Period Register 2 (PWPER2)

Address: $004F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-19. PWM Channel Period Register 3 (PWPER3)

Read: Anytime

Write: Anytime

The value in the period register determines the period of the associated PWM channel. If written while the channel is enabled, the new value takes effect when the existing period terminates, forcing the counter to reset. The new period is then latched and is used until a new period value is written. Reading this register returns the most recent value written. To start a new period immediately, write the new period value and then write the counter, forcing a new period to start with the new period value.

Period = Channel-Clock-Period × (PWPER + 1)(CENTR = 0)

Period = Channel-Clock-Period × PWPER × 2(CENTR = 1)

Downloaded from Arrow.com.

11.2.11 PWM Channel Duty Registers 0–3

Address: $0050

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-20. PWM Channel Duty Register 0 (PWDTY0)

Address: $0051

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-21. PWM Channel Duty Register 1 (PWDTY1)

Address: $0052

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-22. PWM Channel Duty Register 2 (PWDTY2)

Address: $0053

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 11-23. PWM Channel Duty Register 3 (PWDTY3)

Read: Anytime

Write: Anytime

The value in each duty register determines the duty of the associated PWM channel. When the duty value is equal to the counter value, the output changes state. If the register is written while the channel is enabled, the new value is held in a buffer until the counter rolls over or the channel is disabled. Reading this register returns the most recent value written.

If the duty register is greater than or equal to the value in the period register, there is no duty change in state. If the duty register is set to $FF, the output is always in the state which would normally be the state opposite the PPOLx value.

Left-aligned output mode (CENTR = 0):

Duty cycle = [(PWDTYx + 1) / (PWPERx + 1)] × 100%(PPOLx = 1)

Duty cycle = [(PWPERx – PWDTYx) / (PWPERx + 1)] × 100%(PPOLx = 0)

Center-aligned output mode (CENTR = 1):

Duty cycle = [(PWPERx – PWDTYx) / PWPERx] × 100%(PPOLx = 0)

Duty cycle = (PWDTYx / PWPERx) × 100%(PPOLx = 1)

Downloaded from Arrow.com.

11.2.12 PWM Control Register

Address: $0054

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	PSWAI	CENTR	RDPP	PUPP	PSBCK
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-24. PWM Control Register (PWCTL)

Read: Anytime

Write: Anytime

PSWAI — PWM Halts While in Wait Mode Bit

0 = Continue PWM main clock generator while in wait mode.

1 = Halt PWM main clock generator when the part is in wait mode.

CENTR — Center-Aligned Output Mode Bit

To avoid irregularities in the PWM output mode, write the CENTR bit only when PWM channels are disabled.

0 = PWM channels operate in left-aligned output mode.

1 = PWM channels operate in center-aligned output mode.

RDPP — Reduced Drive of Port P Bit

0 = Full drive for all port P output pins

1 = Reduced drive for all port P output pins

PUPP — Pullup Port P Enable Bit

0 = Disable port P pullups

1 = Enable pullups for all port P input pins.

PSBCK — PWM Stops While in Background Mode Bit

0 = Allows PWM to continue while in background mode

1 = Disable PWM input clock while in background mode.

Downloaded from Arrow.com.

11.2.13 PWM Special Mode Register

Address: $0055

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DISCR	DISCP	DISCAL	0	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-25. PWM Special Mode Register (PWTST)

Read: Anytime

Write: Only in special mode (SMODN = 0)

These bits are available only in special mode and are reset in normal mode.

DISCR — Disable Channel Counter Reset Bit

This bit disables the normal operation of resetting the channel counter when the channel counter is written.

0 = Normal operation

1 = Write to PWM channel counter does not reset channel counter.

DISCP — Disable Compare Count Period Bit

0 = Normal operation

1 = In left-aligned output mode, match of the period does not reset the associated PWM counter register.

DISCAL — Disable Scale Counter Loading Bit

This bit disables the normal operation of loading scale counters on a write to the associated scale register.

0 = Normal operation

1 = Write to PWSCAL0 and PWSCAL1 does not load scale counters.

11.2.14 Port P Data Register

Address: $0056

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PP7	PP6	PP5	PP4	PP3	PP2	PP1	PP0
Write:								
PWM					PWM3	PWM2	PWM1	PWM0
Reset:	Unaffected by reset							

Figure 11-26. Port P Data Register (PORTP)

Read: Anytime

Write: Anytime

PWM functions share port P pins 3 to 0 and take precedence over the general-purpose port when enabled. When configured as input, a read returns the pin level. When configured as output, a read returns the latched output data.

A write drives associated pins only if configured for output and the corresponding PWM channel is not enabled. After reset, all pins are general-purpose, high-impedance inputs.

Downloaded from Arrow.com.

11.2.15 Port P Data Direction Register

Address: $0057

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDP7	DDP6	DDP5	DDP4	DDP3	DDP2	DDP1	DDP0
Reset:	0	0	0	0	0	0	0	0

Figure 11-27. Port P Data Direction Register (DDRP)

Read: Anytime

Write: Anytime

DDRP determines pin direction of port P when used for general-purpose I/O. When cleared, I/O pin is configured for input. When set, I/O pin is configured for output.

11.3 PWM Boundary Cases

The boundary conditions for the PWM channel duty registers and the PWM channel period registers cause the results shown in Table 11-2.

Table 11-2. PWM Boundary Conditions

PWDTYx	PWPERx	PPOLx	Output
$FF	>$00	1	Low
$FF	>$00	0	High
≥PWPERx	—	1	High
≥PWPERx	—	0	Low
—	$00	1	High
—	$00	0	Low

11.4 Using the Output Compare 7 Feature to Generate a PWM

This timer exercise is intended to utilize the output compare function along with the output compare 7 new feature to generate a PWM waveform. It must allow for the duty used to drive a DC motor on the UDLP1 board. The registers must be initialized accordingly TC7 = period and TC5 = high time (duty cycle). See Figure 11-28.

NOTE

To verify program is working the DC motor must turn, the frequency, high time, and duty cycle must displayed on the LCD.

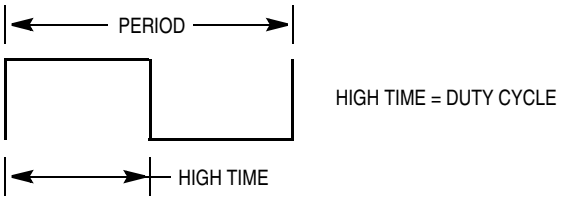


Figure 11-28. Example Waveform

Downloaded from Arrow.com.

11.4.1 PWM Period Calculation

These parameters were used to calculate the high-time values shown in Table 11-3:

- Period = $1000 (Hex) = 4096 (decimal)
- E clock = 8 MHz
- Prescaler = 4
- Frequency = (8 MHz) / (#clocks_count*prescaler) = (8 MHz) (4096*4) = 500 Hz
- If period ($4096 Clocks) => frequency = 500 Hz

Table 11-3. PWM Period Calculations

High-Time Values		Duty Cycle
HEX Count	Decimal Count	
$0020	32	0%
$0040	64	1.5%
$0080	128	3.1%
$0100	256	6.25%
$0200	512	12.50%
$0040	1024	25.00%
$0080	2048	50.00%
$0996	2454	60.00%
$0C00	3072	75.00%
$0D9A	3482	85.00%
$1000	4096	100.00%

11.4.2 Equipment

For this exercise, use the M68HC912B32EVB emulation board.

Downloaded from Arrow.com.

11.4.3 Code Listing

NOTE

A comment line is deliminted by a semi-colon. If there is no code before comment, an ";" must be placed in the first column to avoid assembly errors.

```
INCLUDE 'EQUATES.ASM'    ; Equates for all registers
; ---------------------------------------------------------------------
;           MAIN PROGRAM
; ---------------------------------------------------------------------
         ORG      $7000          ; 16K On-Board RAM, User code data area,
;                                ; start main program at $7000
MAIN:
         BSR      TIMERINIT      ; Subroutine used to initialize the timer:
;                                ; Out. comp. chan. using OC7 & OC5,
;                                ; no interrupts
;                                ; OC7 = PERIOD & OC5 = HIGH TIME of PWM
DONE:    BRA      DONE           ; Branch to itself, Convinient for Breakpoint
;* ------------------------------------------------------
;*  Subroutine TIMERINIT: Initialize Timer for PWM on OC5
;* ------------------------------------------------------
TIMERINIT:
         CLR      TMSK1          ; Disable All Interrupts

         MOVB     #$0A,TMSK2     ; Disable overflow interrupt, disable pull-up
;                                ; resistor function with normal drive capability
;                                ; and Cntr reset by a succesful OC7 compare,
;                                ; Prescaler = sys clock / 4.
         MOVB     #$88,TCTL1     ; Init. OC5 to Clear ouput line to zero on
;                                ; successful compare.
         MOVB     #$A0,TIOS      ; Select Channel 5 and 7 to act as output compare.
         MOVB     #$20,OC7M      ; Initialize OC7 compare to affect OC5 pin(OC7M)
         MOVB     #$20,OC7D      ; Enable OC7 to set output compare 5 pin high(OC7D).
         MOVW     #$0800,TC7H    ; Load TC7 with "PERIOD" of the PWM
         MOVW     #$0400,TC5H    ; Load TC5 with "HIGH TIME" of the PWM.
         MOVB     #$80,TSCR      ; Enable Timer, Timer runs during wait state,
;                                ; and while in Background Mode, also clear flags
;                                ; normally.
         RTS                     ; Return from Subroutine
```


Downloaded from Arrow.com.

Chapter 12
Standard Timer (TIM)

12.1 Introduction

The standard timer module (TIM) for the MC68HC912B32 and MC68HC(9)12BC32 consists of a 16-bit software-programmable counter driven by a prescaler. It contains eight complete 16-bit input capture/output compare channels and one 16-bit pulse accumulator. See Figure 12-2.

The MC68HC12BE32 contains an enhanced capture timer (ECT). The timer on the MC68HC12BE32 is backward compatible with code used on the MC68HC912B32. See Chapter 13 Enhanced Capture Timer (ECT) Module for technical information on this timer.

This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from less than a microsecond to many seconds. It can also generate pulse-width modulator (PWM) signals without CPU intervention.

12.2 Timer Registers

Input/output (I/O) pins default to general-purpose I/O lines until an internal function which uses that pin is specifically enabled. The timer overrides the state of the DDR to force the I/O state of each associated port line when an output compare using a port line is enabled. In these cases, the data direction bits will have no effect on these lines.

When a pin is assigned to output an on-chip peripheral function, writing to this PORTTn bit does not affect the pin, but the data is stored in an internal latch such that if the pin becomes available for general-purpose output the driven level will be the last value written to the PORTTn bit.

12.2.1 Timer Input Capture/Output Compare Select Register

Address: $0080

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset:	0	0	0	0	0	0	0	0

Figure 12-1. Timer Input Capture/Output Compare Select Register (TIOS)

Read: Anytime

Write: Anytime

IOS7–IOS0 —Input Capture or Output Compare Channel Designator Bits

0 = Corresponding channel acts as an input capture.
1 = Corresponding channel acts as an output compare.

Downloaded from Arrow.com.

12.3 Block Diagram

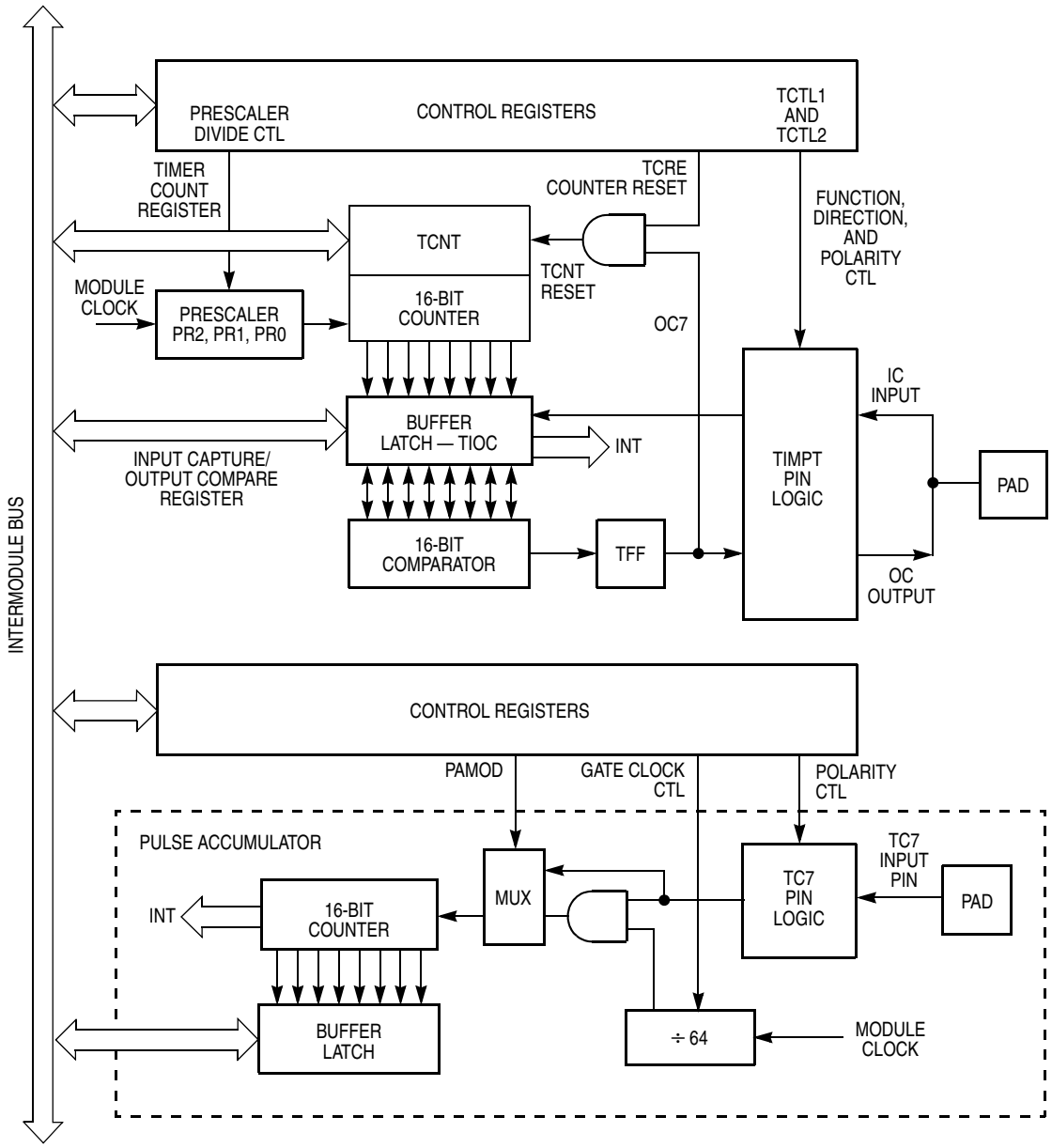


Figure 12-2. Timer Block Diagram

Downloaded from Arrow.com.

12.3.1 Timer Compare Force Register

Address: $0081

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset:	0	0	0	0	0	0	0	0

Figure 12-3. Timer Compare Force Register (CFORC)

Read: Anytime, always returns $00 (1 state is transient)

Write: Anytime

FOC7–FOC0 — Force Output Compare Action Bits for Channels 7–0

A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare n to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCn register except that the interrupt flag does not get set.

12.3.2 Output Compare 7 Mask Register

Address: $0082

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset:	0	0	0	0	0	0	0	0

Figure 12-4. Output Compare 7 Mask Register (OC7M)

Read: Anytime

Write: Anytime

The bits of OC7M correspond bit-for-bit with the bits of the timer port (PORTT). Setting the OC7Mn sets the corresponding port to be an output port regardless of the state of the DDRTn bit when the corresponding TIOSn bit is set to be an output compare. This does not change the state of the DDRT bits.

12.3.3 Output Compare 7 Data Register

Address: $0083

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
Reset:	0	0	0	0	0	0	0	0

Figure 12-5. Output Compare 7 Data Register (OC7D)

Read: Anytime

Write: Anytime

The bits of OC7D correspond bit-for-bit with the bits of the timer port (PORTT). When a successful OC7 compare occurs, for each bit that is set in OC7M, the corresponding data bit in OC7D is stored to the corresponding bit of the timer port.

When the OC7Mn bit is set, a successful OC7 action will override a successful OC6–OC0 compare action during the same cycle; therefore, the OCn action taken will depend on the corresponding OC7D bit.

Downloaded from Arrow.com.

12.3.4 Timer Count Register

Address: $0084

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

Address: $0085

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-6. Timer Count Register (TCNT)

Read: Anytime

Write: Has no meaning or effect in normal mode; only writable in special modes

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

12.3.5 Timer System Control Register

Address: $0086

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TEN	TSWAI	TSBCK	TFFCA	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-7. Timer System Control Register (TSCR)

Read: Anytime

Write: Anytime

TEN — Timer Enable Bit

If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator since the E ÷ 64 is generated by the timer prescaler.

0 = Disables timer, including the counter; can be used for reducing power consumption

1 = Allows timer to function normally

TSWAI — Timer Stops While in Wait Bit

Timer interrupts cannot be used to get the MCU out of wait.

0 = Allows timer to continue running during wait

1 = Disables timer when MCU is in wait mode

Downloaded from Arrow.com.

TSBCK — Timer Stops While in Background Mode Bit

0 = Allows timer to continue running while in background mode
1 = Disables timer when MCU is in background mode; useful for emulation

TFFCA — Timer Fast Flag Clear All Bit

0 = Allows timer flag clearing to function normally
1 = For TFLG1($8E), a read from an input capture or a write to the output compare channel ($90–$9F) causes the corresponding channel flag, CnF, to be cleared. For TFLG2 ($8F), any access to the TCNT register ($84, $85) clears the TOF flag. Any access to the PACNT register ($A2 and $A3) clears the PAOVF and PAIF flags in the PAFLG register ($A1).
This has the advantage of eliminating software overhead in a separate clear sequence.

NOTE

Extra care is required to avoid accidental flag clearing due to unintended accesses.

12.3.6 Timer Control Registers

Address: $0088

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset:	0	0	0	0	0	0	0	0

Figure 12-8. Timer Control Register 1 (TCTL1)

Address: $0089

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset:	0	0	0	0	0	0	0	0

Figure 12-9. Timer Control Register 2 (TCTL2)

Read: Anytime

Write: Anytime

OMn — Output mode
OLn — Output level

These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCn compare (see Table 12-1). When either OMn or OLn is 1, the pin associated with OCn becomes an output tied to OCn regardless of the state of the associated DDRT bit.

Table 12-1. Compare Result Output Action

OMn	OLn	Action
0	0	Timer disconnected from output pin logic
0	1	Toggle OCn output line
1	0	Clear OCn output line to 0
1	1	Set OCn output line to 1

Downloaded from Arrow.com.

Address: $008A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
Reset:	0	0	0	0	0	0	0	0

Figure 12-10. Timer Control Register 3 (TCTL3)

Address: $008B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset:	0	0	0	0	0	0	0	0

Figure 12-11. Timer Control Register 4 (TCTL4)

Read: Anytime

Write: Anytime

EDGnB and EDGnA — Input Capture Edge Control Bits

These 8 pairs of control bits configure the input capture edge detector circuits. See Table 12-2.

Table 12-2. Edge Detector Circuit Configuration

EDGnB	EDGnA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge (rising or falling)

12.3.7 Timer Interrupt Mask Registers

Address: $008C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
Reset:	0	0	0	0	0	0	0	0

Figure 12-12. Timer Interrupt Mask 1 Register (TMSK1)

Read: Anytime

Write: Anytime

The bits in TMSK1 correspond bit-for-bit with the bits in the TFLG1 status register. If cleared, the corresponding flag is disabled from causing a hardware interrupt. If set, the corresponding flag is enabled to cause a hardware interrupt.

Downloaded from Arrow.com.

C7I–C0I — Input Capture/Output Compare x Interrupt Enable Bits

Address: $008D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TOI	0	PUPT	RDPT	TCRE	PR2	PR1	PR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-13. Timer Interrupt Mask 2 Register (TMSK2)

Read: Anytime

Write: Anytime

TOI — Timer Overflow Interrupt Enable Bit

0 = Interrupt inhibited
1 = Hardware interrupt requested when TOF flag set

PUPT — Timer Pullup Resistor Enable Bit

This enable bit controls pullup resistors on the timer port pins when the pins are configured as inputs.
0 = Disable pullup resistor function
1 = Enable pullup resistor function

RDPT — Timer Drive Reduction Bit

This bit reduces the effective output driver size which can reduce power supply current and generated noise depending upon pin loading.
0 = Normal output drive capability
1 = Enable output drive reduction function

TCRE — Timer Counter Reset Enable Bit

This bit allows the timer counter to be reset by a successful output compare 7 event.
0 = Counter reset inhibited and counter free runs
1 = Counter reset by a successful output compare 7
If TC7 = $0000 and TCRE = 1, TCNT stays at $0000 continuously. If TC7 = $FFFF and TCRE = 1, TOF never gets set even though TCNT counts from $0000 through $FFFF.

PR2, PR1, and PR0 — Timer Prescaler Select Bits

These three bits specify the number of ÷2 stages that are to be inserted between the module clock and the timer counter. See Table 12-3.

Table 12-3. Prescaler Selection

PR2	PR1	PR0	Prescale Factor
0	0	0	1
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	Reserved
1	1	1	Reserved

The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal 0.

Downloaded from Arrow.com.

12.3.8 Timer Interrupt Flag Registers

Address: $008E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
Reset:	0	0	0	0	0	0	0	0

Figure 12-14. Timer Interrupt Flag 1 (TFLG1)

Read: Anytime

Write: Used in the clearing mechanism; set bits cause corresponding bits to be cleared

TFLG1 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write a 1 to the bit. Writing a logic 0 does not affect current status of the bit.

When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel ($90–$9F) causes the corresponding channel flag CnF to be cleared.

C7F–C0F — Input Capture/Output Compare Channel n Flag

Address: $008F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TOF	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 12-15. Timer Interrupt Flag 2 (TFLG2)

Read: Anytime

Write: Used in the clearing mechanism; set bits cause corresponding bits to be cleared

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, set the bit to 1.

Any access to TCNT clears TFLG2 register, if the TFFCA bit in TSCR register is set.

TOF — Timer Overflow Flag

Set when 16-bit free-running timer overflows from $FFFF to $0000. This bit is cleared automatically by a write to the TFLG2 register with bit 7 set. For additional information, see the TCRE control bit explanation found in 12.3.7 Timer Interrupt Mask Registers.

Downloaded from Arrow.com.

12.3.9 Timer Input Capture/Output Compare Registers

Address: $0090

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $0091

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-16. Timer Input Capture/Output Compare Register 0 (TC0)

Address: $0092

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $0093

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-17. Timer Input Capture/Output Compare Register 1 (TC1)

Address: $0094

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $0095

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-18. Timer Input Capture/Output Compare Register 2 (TC2)

Downloaded from Arrow.com.

Address: $0096

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $0097

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-19. Timer Input Capture/Output Compare Register 3 (TC3)

Address: $0098

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $0099

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-20. Timer Input Capture/Output Compare Register 4 (TC4)

Address: $009A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $009B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-21. Timer Input Capture/Output Compare Register 5 (TC5)

Downloaded from Arrow.com.

Address: $009C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $009D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-22. Timer Input Capture/Output Compare Register 6 (TC6)

Address: $009E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $009F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-23. Timer Input Capture/Output Compare Register 7 (TC7)

Read: Anytime

Write: Anytime for output compare function; has no meaning or effect during input capture

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

12.3.10 Pulse Accumulator Control Register

Address: $00A0

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-24. Pulse Accumulator Control Register (PACTL)

Read: Anytime

Write: Anytime

Downloaded from Arrow.com.

PAEN — Pulse Accumulator System Enable Bit

0 = Pulse accumulator system disabled
1 = Pulse accumulator system enabled

PAEN is independent from TEN.

PAMOD — Pulse Accumulator Mode Bit

0 = Event counter mode
1 = Gated time accumulation mode

PEDGE — Pulse Accumulator Edge Control Bit

For PAMOD = 0 (event counter mode)

0 = Falling edges on the pulse accumulator input pin (PT7/PAI) cause the count to be incremented.
1 = Rising edges on the pulse accumulator input pin cause the count to be incremented.

For PAMOD = 1 (gated time accumulation mode)

0 = Pulse accumulator input pin high enables E ÷ 64 clock to pulse accumulator and the trailing falling edge on the pulse accumulator input pin sets the PAIF flag.
1 = Pulse accumulator input pin low enables E ÷ 64 clock to pulse accumulator and the trailing rising edge on the pulse accumulator input pin sets the PAIF flag.

If the timer is not active (TEN = 0 in TSCR), there is no ÷64 clock since the E ÷ 64 clock is generated by the timer prescaler.

CLK1 and CLK0 — Clock Select Bits

Table 12-4. Clock Selection

CLK1	CLK0	Selected Clock
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65536 as timer counter clock frequency

If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

PAOVI — Pulse Accumulator Overflow Interrupt Enable Bit

0 = Interrupt inhibited
1 = Interrupt requested if PAOVF is set

PAI — Pulse Accumulator Input Interrupt Enable Bit

0 = Interrupt inhibited
1 = Interrupt requested if PAIF is set

Downloaded from Arrow.com.

12.3.11 Pulse Accumulator Flag Register

Address: $00A1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	PAOVF	PAIF
Reset:	0	0	0	0	0	0	0	0

Figure 12-25. Pulse Accumulator Flag Register (PAFLG)

Read: Anytime

Write: Anytime

When the TFFCA bit in the TSCR register is set, any access to the PACNT register clears all the flags in the PAFLG register.

PAOVF — Pulse Accumulator Overflow Flag

Set when the 16-bit pulse accumulator overflows from $FFFF to $0000. This bit is cleared automatically by a write to the PAFLG register with bit 1 set.

PAIF — Pulse Accumulator Input Edge Flag

Set when the selected edge is detected at the pulse accumulator input pin. In event mode, the event edge triggers PAIF. In gated time accumulation mode, the trailing edge of the gate signal at the pulse accumulator input pin triggers PAIF. This bit is cleared automatically by a write to the PAFLG register with bit 0 set.

12.3.12 16-Bit Pulse Accumulator Count Register

Address: $00A2

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $00A3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 12-26. 16-Bit Pulse Accumulator Count Register (PACNT)

Read: Anytime

Write: Anytime

Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

Downloaded from Arrow.com.

12.3.13 Timer Test Register

Address: $00AD

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	TCBYP	PCBYP
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 12-27. Timer Test Register (TIMTST)

Read: Anytime

Write: Only in special mode (SMODN = 0)

TCBYP — Timer Divider Chain Bypass Bit

0 = Normal operation

1 = 16-bit free-running timer counter is divided into two 8-bit halves and the prescaler is bypassed. The clock drives both halves directly.

PCBYP — Pulse Accumulator Divider Chain Bypass Bit

0 = Normal operation

1 = 16-bit pulse accumulator counter is divided into two 8-bit halves and the prescaler is bypassed. The clock drives both halves directly.

12.3.14 Timer Port Data Register

Address: $00AE

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0
Reset:	0	0	0	0	0	0	0	0
Timer:	IOC7	IOC6	IOC5	IOC4	IOC3	IOC2	IOC1	IOC0
PA:	PAI							

Figure 12-28. Timer Port Data Register (PORTT)

Read: Anytime; inputs return pin level; outputs return pin driver input level

Write: Data stored in an internal latch; drives pins only if configured for output

NOTE

Writes do not change pin state when the pin is configured for timer output. The minimum pulse width for pulse accumulator input should always be greater than two module clocks due to input synchronizer circuitry. The minimum pulse width for the input capture should always be greater than the width of two module clocks due to input synchronizer circuitry.

Downloaded from Arrow.com.

12.3.15 Data Direction Register for Timer Port

Address: $00AF

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0
Reset:	0	0	0	0	0	0	0	0

Figure 12-29. Data Direction Register for Timer Port (DDRT)

Read: Anytime

Write: Anytime

0 = Configures the corresponding I/O pin for input only
1 = Configures the corresponding I/O pin for output

The timer forces the I/O state to be an output for each timer port pin associated with an enabled output compare. In these cases the data direction bits will not be changed, but they have no affect on the direction of these pins. The DDRT will revert to controlling the I/O direction of a pin when the associated timer output compare is disabled. Input captures do not override the DDRT settings.

12.4 Timer Operation in Modes

Stop — Timer is off since both PCLK and ECLK are stopped.

BDM— Timer keeps running, unless TSBCK = 1.

Wait — Counters keep running, unless TSWAI = 1.

Normal — Timer keeps running, unless TEN = 0.

TEN = 0 —All timer operations are stopped, registers may be accessed.
Gated pulse accumulator ÷64 clock is also disabled.

PAEN = 0 —All pulse accumulator operations are stopped.
Registers may be accessed.

Downloaded from Arrow.com.

12.5 Using the Output Compare Function to Generate a Square Wave

This timer exercise is intended to utilize the output compare function to generate a square wave of predetermined duty cycle and frequency.

Square wave frequency 1000 Hz, duty cycle 50%

The program generates a square wave, 50 percent duty cycle, on output compare 2 (OC2). The signal will be measured by the HC11 on the UDLP1 board. It assumes a 8.0 MHz operating frequency for the E clock. The control registers are initialized to disable interrupts, configure for proper pin control action and also the TC2H register for desired compare value. The appropiate count must be calculated to achieve the desired frequency and duty cycle. For example: for a 50 percent duty, 1 KHz signal each period must consist of 2048 counts or 1024 counts high and 1024 counts low. In essence a $0400 is added to generate a frequency of 1 kHz.

12.5.1 Sample Calculation to Obtain Period Counts

The sample calculation to obtain period counts is:

- For 1000 Hz frequency:
  - E-clock = 8 MHz
  - IC/OC resolution factor = 1/(E-clock/Prescaler)
  - If the Prescaler = 4, then output compare resolution is 0.5 µs
- For a 1 KHz, 50 percent duty cycle:
  - 1/F = T = 1/1000 = 1 ms
  - F for output compare = prescaler/E clock = 2 MHz

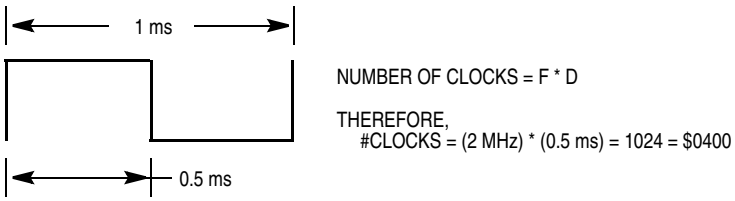


Figure 12-30. Example Waveform

12.5.2 Equipment

For this exercise, use the M68HC912B32EVB emulation board.

12.5.3 Code Listing

NOTE

A comment line is deliminted by a semi-colon. If there is no code before comment, an ";" must be placed in the first column to avoid assembly errors.

Downloaded from Arrow.com.

```
------------------------------------------------------------------------
;            MAIN PROGRAM
; ---------------------------------------------------------------------
          ORG      $7000          ; 16K On-Board RAM, User code data area,
;                                 ; start main program at $7000
MAIN:
          BSR      TIMERINIT      ; Subroutine used to initialize the timer:
;                                 ; Output compare channel, no interrupts
          BSR      SQWAVE         ; Subroutine to generate square wave
DONE:     BRA      DONE           ; Branch to itself, Convinient for Breakpoint

;* ---------------------------------------------------------------------
;*  Subroutine TIMERINIT: Initialize Timer for Output Compare on OC2
;* ---------------------------------------------------------------------
TIMERINIT:
          CLR      TMSK1          ; Disable All Interrupts

          MOVB     #$02,TMSK2     ; Disable overflow interrupt, disable pull-up
;                                 ; resistor function with normal drive capability
;                                 ;and free running counter, Prescaler = sys clock/4.

          MOVB     #$10,TCTL2     ; Initialize OC2 to toggle on successful compare.

          MOVB     #$04,TIOS      ; Select Channel 2 to act as output compare.

          MOVW     #$0400,TC2H    ; Load TC2 Reg with initial compare value.

          MOVB     #$80,TSCR      ; Enable Timer, Timer runs during wait state, and
;                                 ; while in Background Mode, also clear flags
;                                 ; normally.

          RTS                     ; Return from Subroutine

;* ------------------------------
;*      SUBROUTINE:  SQWAVE
;* ------------------------------
SQWAVE:

;* -------
CLEARFLG:
;* -------
;*  To clear the C2F flag: 1) read TFLG1 when
;*  C2F is set and then 2) write a logic "one" to C2F.

          LDAA     TFLG1             ; To clear OC2 Flag, first it must be read,
          ORAA     #$04              ; then a "1" must be written to it
          STAA     TFLG1

WTFLG:    BRCLR    TFLG1,#$04,WTFLG  ; Wait (Polling) for C2F Flag

          LDD      TC2H              ; Loads value of compare from TC2 Reg.
          ADDD     #$0400            ; Add hex value of 500us High Time
          STD      TC2H              ; Set-up next transition time in 500 us

          BRA      CLEARFLG          ; Continuously add 500 us, branch to CLEARFLAG

          RTS                        ; return from Subroutine

          END                        ; End of program
```


Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 13
Enhanced Capture Timer (ECT) Module

13.1 Introduction

The M68HC12 enhanced capture timer (ECT) module has the features of the M68HC12 standard timer (TIM) module enhanced by additional features in order to enlarge the field of applications.

These additional features are:

- 16-bit buffer register for four input capture (IC) channels
- Four 8-bit pulse accumulators:
  - 8-bit buffer registers associated with the four buffered IC channels
  - Configurable as two 16-bit pulse accumulators
- 16-bit modulus down-counter with 4-bit prescaler
- Four user selectable delay counters for input noise immunity increase
- Main timer prescaler extended to 7-bit

This section describes the standard timer found on the MC68HC912B32 as well as the additional features found on the MC68HC12BE32.

13.2 Basic Timer Overview

The basic timer consists of a 16-bit, software-programmable counter driven by a prescaler. This timer can be used for many purposes, including input waveform measurements while simultaneously generating an output waveform. Pulse widths can vary from microseconds to many seconds.

A full access for the counter registers or the input capture/output compare registers should take place in one clock cycle. Accessing high byte and low byte separately for all of these registers may not yield the same result as accessing them in one word.

13.3 Enhanced Capture Timer Modes of Operation

The enhanced capture timer has eight input capture, output compare (IC/OC) channels same as on the M68HC12 standard timer (timer channels TC0–TC7). When channels are selected as input capture by selecting the IOSx bit in the timer input capture/output compare select register (TIOS), they are called input capture (IC) channels.

Four IC channels are the same as on the standard timer with one capture register which memorizes the timer value captured by an action on the associated input pin. Four other IC channels, in addition to the capture register, also have one buffer called holding register. This permits the register to memorize two different timer values without generation of any interrupt.

Four 8-bit pulse accumulators are associated with the four buffered IC channels. Each pulse accumulator has a holding register to memorize their value by an action on its external input. Each pair of pulse accumulators can be used as a 16-bit pulse accumulator.

Downloaded from Arrow.com.

The 16-bit modulus down-counter can control the transfer of the IC register's contents and the pulse accumulators to the respective holding registers for a given period, every time the count reaches 0. The modulus down-counter can also be used as a stand-alone timebase with periodic interrupt capability.

13.3.1 IC Channels

The IC channels are composed of four standard IC registers and four buffered IC channels.

- An **IC register** is **empty** when it has been read or latched into the holding register.
- A **holding register** is **empty** when it has been read.

13.3.1.1 Non-Buffered IC Channels

The main timer value is memorized in the IC register by a valid input pin transition.

- If the corresponding NOVWx bit of the input control overwrite register (ICOVW) is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value.
- If the corresponding NOVWx bit of the ICOVW register is set, the capture register cannot be written unless it is empty.

This will prevent the captured value to be overwritten until it is read.

13.3.1.2 Buffered IC Channels

There are two modes of operations for the buffered IC channels.

IC Latch Mode (see Figure 13-1):

When enabled (LATQ = 1), the main timer value is memorized in the IC register by a valid input pin transition. The value of the buffered IC register is latched to its holding register by the modulus counter for a given period when the count reaches 0, by a write $0000 to the modulus counter or by a write to ICLAT in the 16-bit modulus down-counter control register (MCCTL).

- If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the contents of IC register are overwritten by the new value. In case of latching, the contents of its holding register are overwritten.
- If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see 13.3.1 IC Channels). This will prevent the captured value to be overwritten until it is read or latched in the holding register.

IC Queue Mode (see Figure 13-2):

When enabled (LATQ = 0), the main timer value is memorized in the IC register by a valid input pin transition.

- If the corresponding NOVWx bit of the ICOVW register is cleared, with a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.
- If the corresponding NOVWx bit of the ICOVW register is set, the capture register or its holding register cannot be written by an event unless they are empty (see 13.3.1 IC Channels).

In queue mode, reads of the holding register will latch the corresponding pulse accumulator value to its holding register.

Downloaded from Arrow.com.

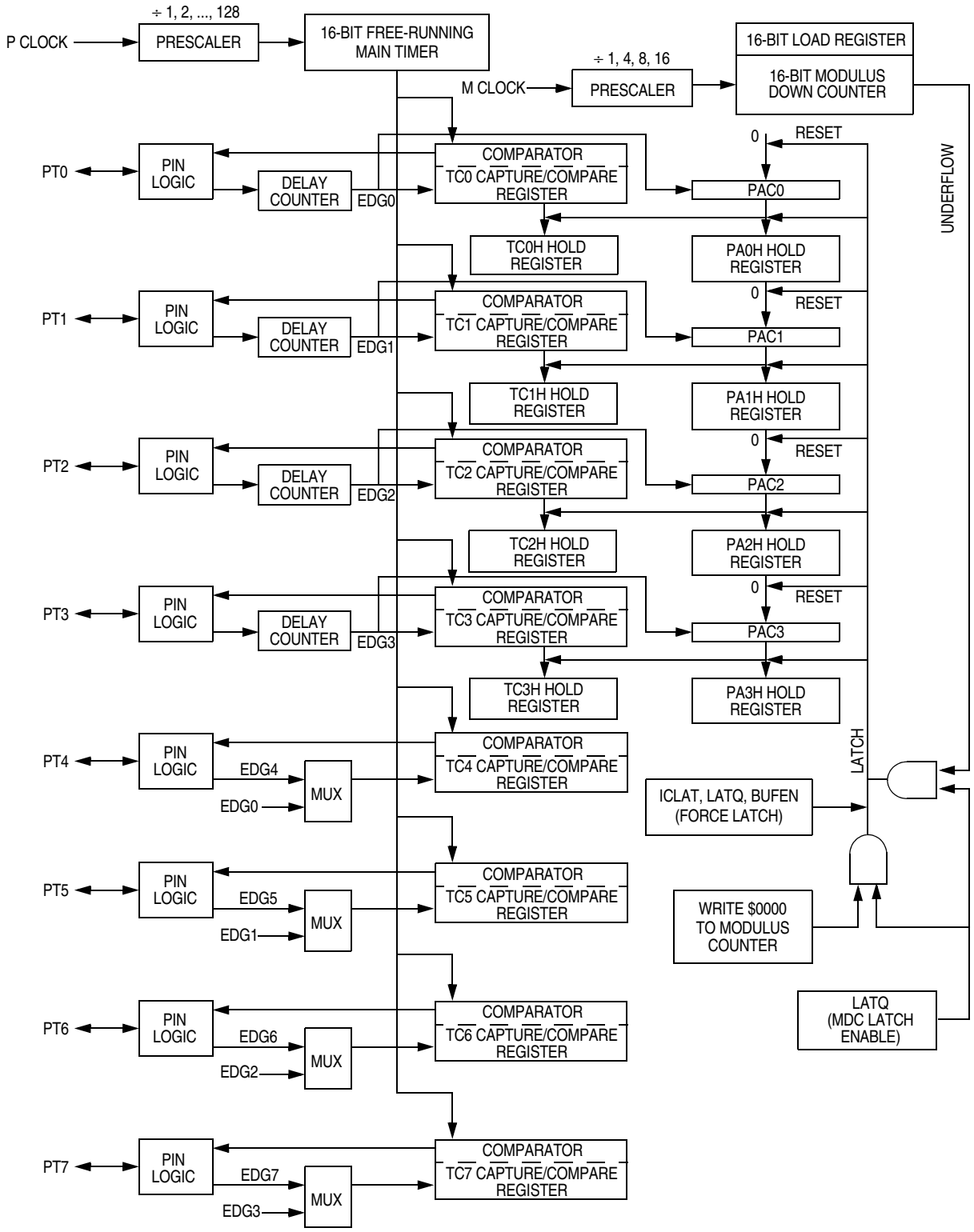


Figure 13-1. Timer Block Diagram in Latch Mode

Downloaded from Arrow.com.

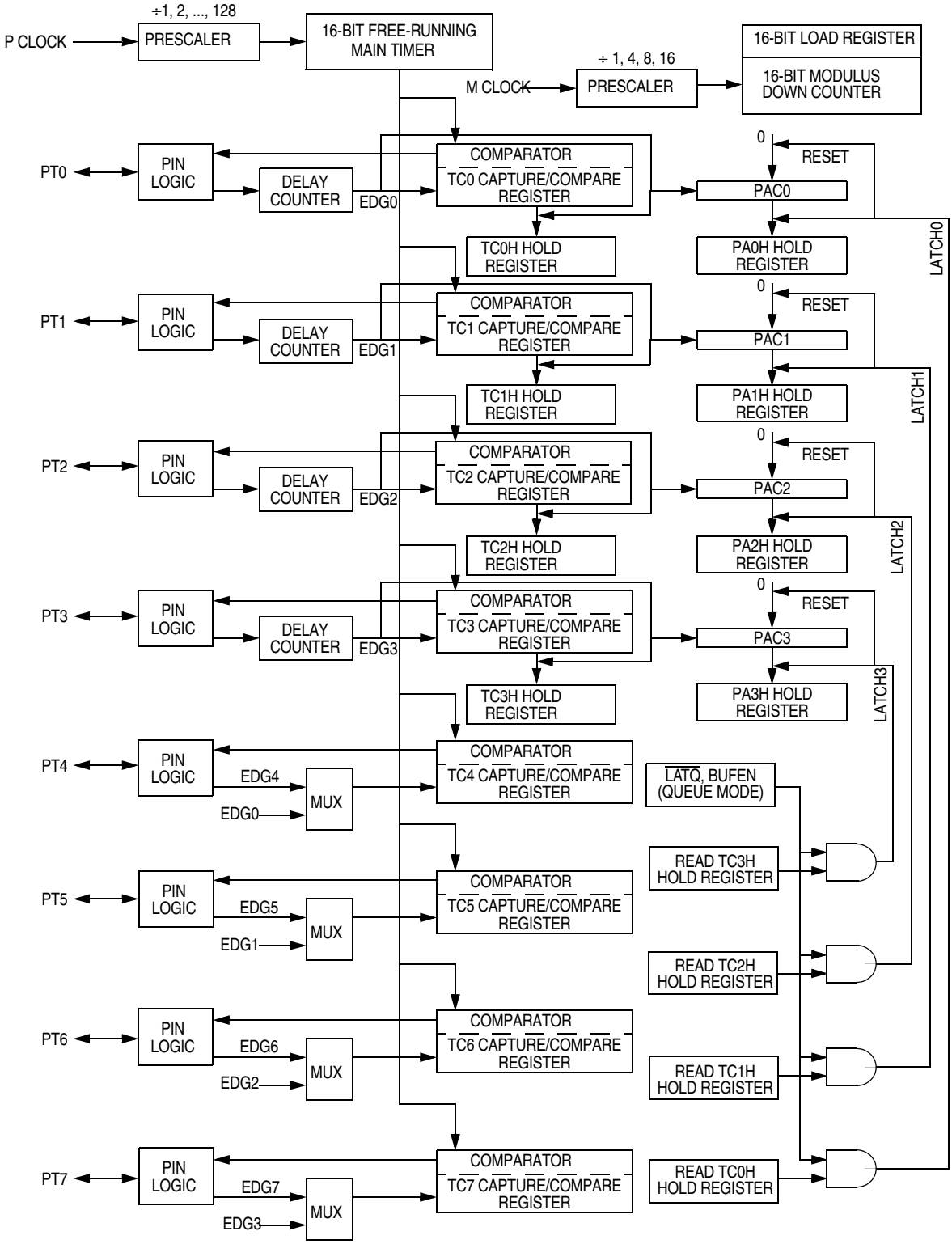


Figure 13-2. Timer Block Diagram in Queue Mode

Downloaded from Arrow.com.

13.3.2 Pulse Accumulators

Four 8-bit pulse accumulators with four 8-bit holding registers are associated with the four IC buffered channels. See Figure 13-3. A pulse accumulator counts the number of active edges at the input of its channel.

The user can prevent 8-bit pulse accumulators from counting further than $FF by PACMX control bit in input control system control register (ICSYS). In this case, a value of $FF means that 255 counts or more have occurred.

Each pair of pulse accumulators can be used as a 16-bit pulse accumulator. See Figure 13-4.

For more information on the two modes of operation for the pulse accumulators, see 13.3.2.1 Pulse Accumulator Latch Mode and 13.3.2.2 Pulse Accumulator Queue Mode.

13.3.2.1 Pulse Accumulator Latch Mode

The value of the pulse accumulator is transferred to its holding register when the modulus down-counter reaches zero, a write $0000 to the modulus counter, or when the force latch control bit ICLAT is written. At the same time, the pulse accumulator is cleared.

13.3.2.2 Pulse Accumulator Queue Mode

When queue mode is enabled, reads of an input capture holding register will transfer the contents of the associated pulse accumulator to its holding register. At the same time, the pulse accumulator is cleared.

13.3.3 Modulus Down-Counter

The modulus down-counter can be used as a timebase to generate a periodic interrupt. It can also be used to latch the values of the IC registers and the pulse accumulators to their holding registers. The action of latching can be programmed to be periodic or only once.

13.4 Timer Registers

Input/output pins default to general-purpose input/output (I/O) lines until an internal function which uses that pin is specifically enabled. The timer overrides the state of the DDR to force the I/O state of each associated port line when an output compare using a port line is enabled. In these cases, the data direction bits will have no effect on these lines.

When a pin is assigned to output an on-chip peripheral function, writing to this PORTT bit does not affect the pin. The data is stored in an internal latch such that if the pin becomes available for general-purpose output, the driven level will be the last value written to the PORTT bit.

Downloaded from Arrow.com.

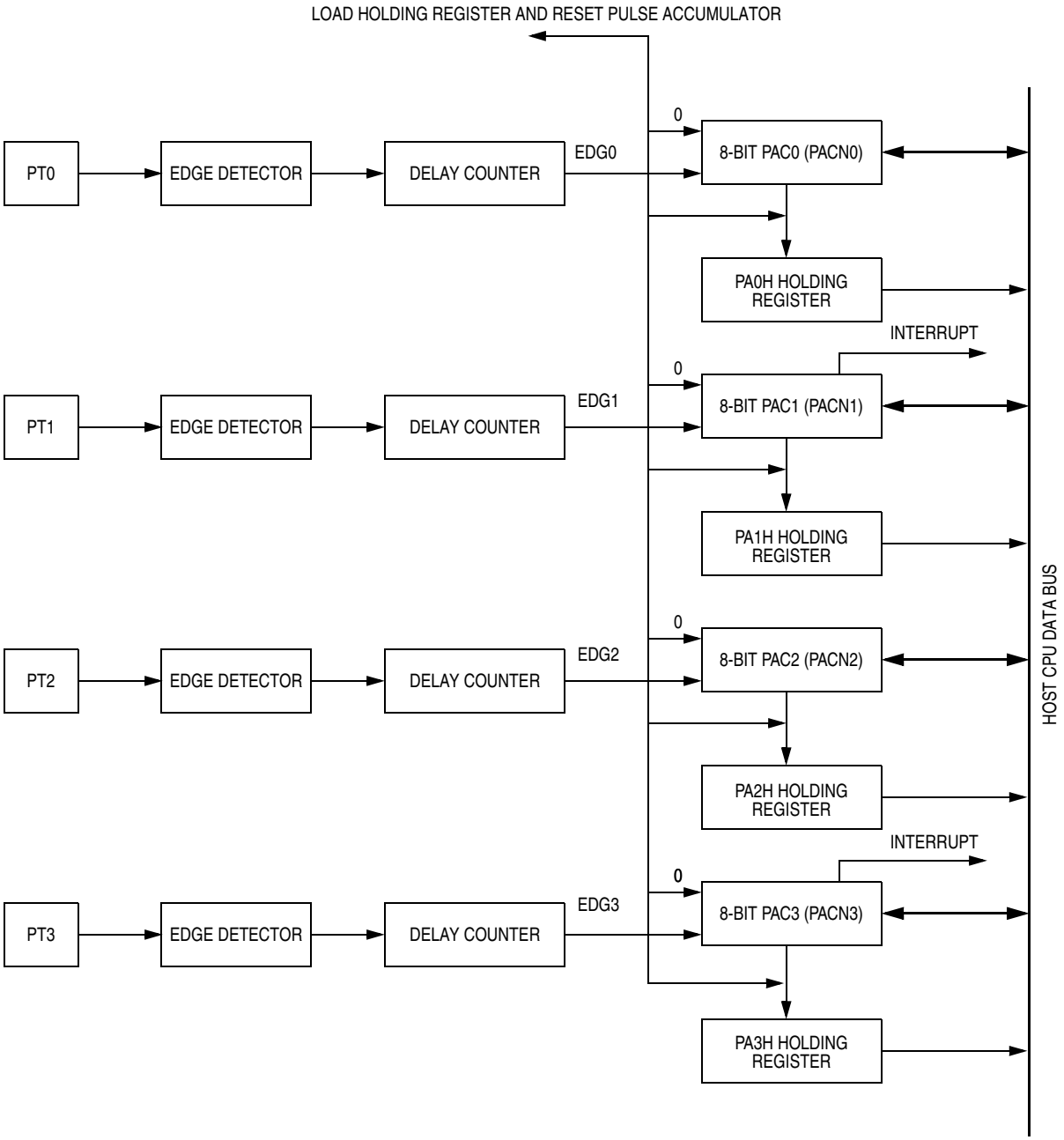


Figure 13-3. 8-Bit Pulse Accumulators Block Diagram

Downloaded from Arrow.com.

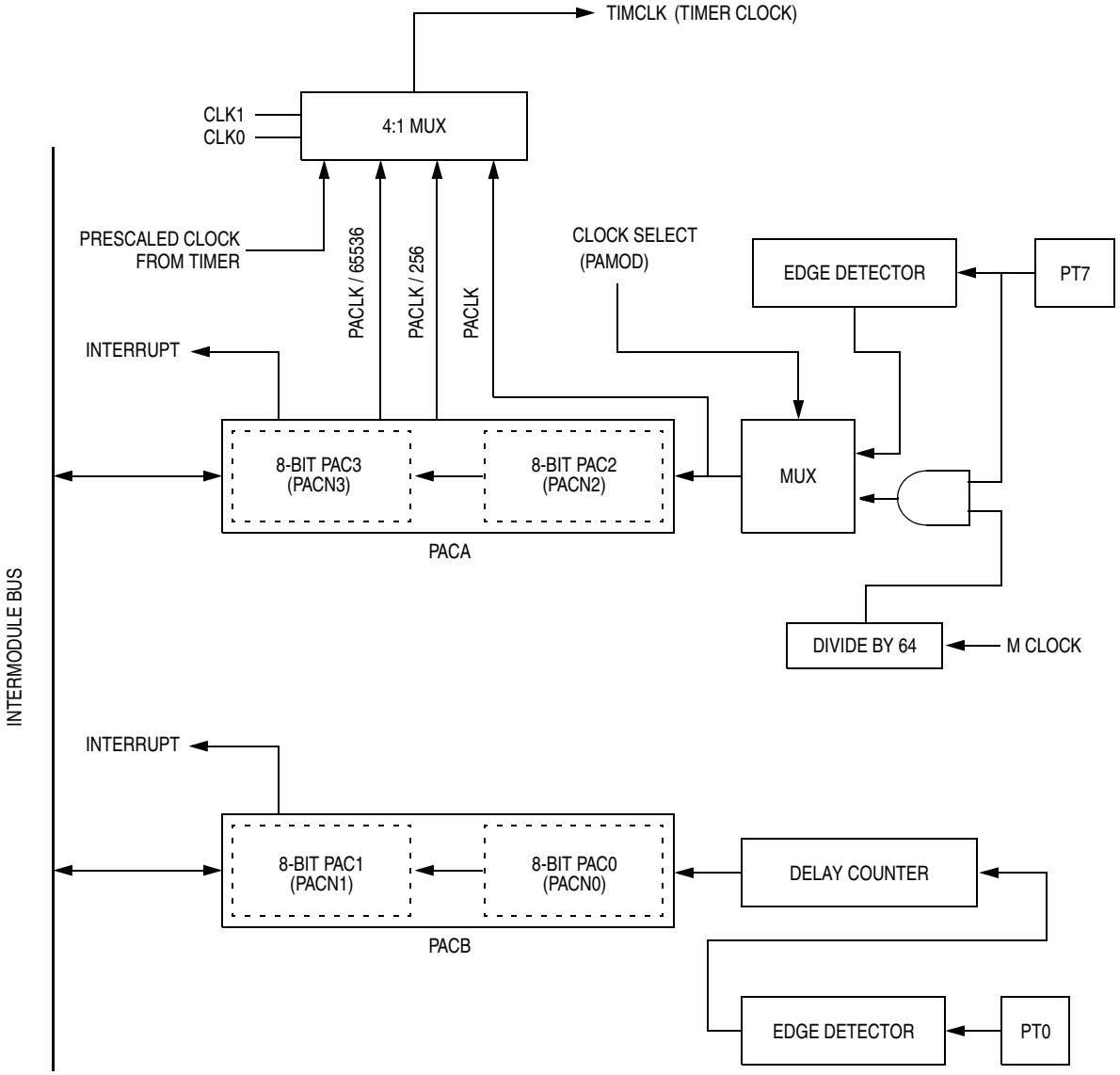


Figure 13-4. 16-Bit Pulse Accumulators Block Diagram

Downloaded from Arrow.com.

13.4.1 Timer Input Capture/Output Compare Select Register

Address: $0080

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset:	0	0	0	0	0	0	0	0

Figure 13-5. Timer Input Capture/Output Compare Select Register (TIOS)

Read: Anytime

Write: Anytime

IOS[7:0] — Input Capture or Output Compare Channel Configuration Bits

0 = The corresponding channel acts as an input capture.

1 = The corresponding channel acts as an output compare.

13.4.2 Timer Compare Force Register

Address: $0081

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset:	0	0	0	0	0	0	0	0

Figure 13-6. Timer Compare Force Register (CFORC)

Read: Anytime but, will always return $00 (1 state is transient).

Write: Anytime

FOC[7:0] — Force Output Compare Action Bits for Channel 7–0

A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “n” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCn register except the interrupt flag does not get set.

13.4.3 Output Compare 7 Mask Register

Address: $0082

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset:	0	0	0	0	0	0	0	0

Figure 13-7. Output Compare 7 Mask Register (OC7M)

Read: Anytime

Write: Anytime

Downloaded from Arrow.com.

OC7M[7:0] Bits

The bits of OC7M correspond bit-for-bit with the timer port (PORTT) bits. Setting the OC7Mn will set the corresponding port to be an output port regardless of the state of the DDRTn bit, when the corresponding TIOSn bit is set to be an output compare. This does not change the state of the DDRT bits. At successful OC7, for each bit that is set in OC7M, the corresponding data bit OC7D is stored to the corresponding bit of the timer port. See Figure 13-8.

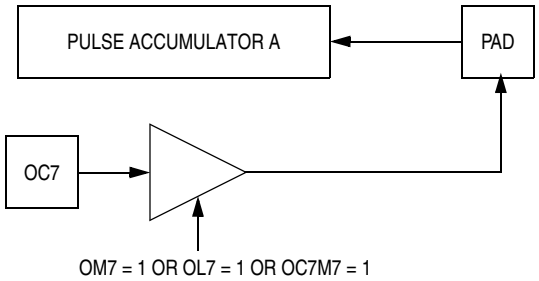


Figure 13-8. Block Diagram for Port 7 with Output Compare/Pulse Accumulator A

NOTE

OC7M has priority over output action on the timer port enabled by OMn and OLn bits in TCTL1 and TCTL2. If an OC7M bit is set, it prevents the action of corresponding OM and OL bits on the selected timer port.

13.4.4 Output Compare 7 Data Register

Address: $0083

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OC7D7	OC7D6	OC7D5	OC7D4	OC7D3	OC7D2	OC7D1	OC7D0
Reset:	0	0	0	0	0	0	0	0

Figure 13-9. Output Compare 7 Data Register (OC7D)

Read: Anytime

Write: Anytime

OC7D[7:0] Bits

The bits of OC7D correspond bit-for-bit with the bits of the timer port (PORTT). When a successful OC7 compare occurs, for each bit that is set in OC7M, the corresponding data bit in OC7D is stored to the corresponding bit of the timer port.

When the OC7Mn bit is set, a successful OC7 action will override a successful OC[6:0] compare action during the same cycle; therefore, the OCn action taken will depend on the corresponding OC7D bit.

Downloaded from Arrow.com.

13.4.5 Timer Count Registers

Address: $0084

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	0	0	0	0	0	0	0	0

Address: $0085

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-10. Timer Count Registers (TCNT)

Read: Anytime

Write: Has no meaning or effect in the normal mode; only writable in special modes (SMODN = 0)

The 16-bit main timer is an up-counter.

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

The period of the first count after a write to the TCNT registers may be a different size because the write is not synchronized with the prescaler clock.

13.4.6 Timer System Control Register

Address: $0086

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TEN	TSWAI	TSBCK	TFFCA				
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-11. Timer System Control Register (TSCR)

Read: Anytime

Write: Anytime

TEN — Timer Enable Bit

If for any reason the timer is not active, there is no ÷64 clock for the pulse accumulator since the E ÷ 64 is generated by the timer prescaler.

0 = Disables the main timer, including the counter; can be used for reducing power consumption

1 = Allows the timer to function normally

Downloaded from Arrow.com.

TSWAI — Timer Module Stops While in Wait Bit

TSWAI also affects pulse accumulators and modulus down counters.

0 = Allows the timer module to continue running during wait

1 = Disables the timer module when the MCU is in wait mode. Timer interrupts cannot be used to get the MCU out of wait.

TSBCK — Timer and Modulus Counter Stop While in Background Mode Bit

TBSCK does not stop the pulse accumulator.

0 = Allows the timer and modulus counter to continue running while in background mode

1 = Disables the timer and modulus counter whenever the MCU is in background mode. This is useful for emulation.

TFFCA — Timer Fast Flag Clear All Bit

0 = Allows the timer flag clearing to function normally

1 = For TFLG1($8E), a read from an input capture or a write to the output compare channel ($90–$9F) causes the corresponding channel flag, CnF, to be cleared. For TFLG2 ($8F), any access to the TCNT register ($84, $85) clears the TOF flag. Any access to the PACN3 and PACN2 registers ($A2, $A3) clears the PAOVF and PAIF flags in the PAFLG register ($A1). Any access to the PACN1 and PACN0 registers ($A4, $A5) clears the PBOVF flag in the PBFLG register ($B1). This has the advantage of eliminating software overhead in a separate clear sequence. Extra care is required to avoid accidental flag clearing due to unintended accesses.

13.4.7 Timer Control Registers

Address: $0088

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset:	0	0	0	0	0	0	0	0

Figure 13-12. Timer Control Register 1 (TCTL1)

Address: $0089

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset:	0	0	0	0	0	0	0	0

Figure 13-13. Timer Control Register 2 (TCTL2)

Read: Anytime

Write: Anytime

OMn Bits — Output Mode
OLn Bits — Output Level

These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCn compare (see Table 13-1). When either OMn or OLn is 1, the pin associated with OCn becomes an output tied to OCn regardless of the state of the associated DDRT bit.

NOTE

To enable output action by OMn and OLn bits on the timer port, the corresponding bit in OC7M should be cleared.

Downloaded from Arrow.com.

Table 13-1. Compare Result Output Action

OMn	OLn	Action
0	0	Timer disconnected from output pin logic
0	1	Toggle OCn output line
1	0	Clear OCn output line to 0
1	1	Set OCn output line to 1

To operate the 16-bit pulse accumulators A and B (PACA and PACB) independently of input capture or output compare 7 and 0, respectively, the user must set the corresponding bits IOSn = 1, OMn = 0, and OLn = 0. OC7M7 or OC7M0 in the OC7M register must also be cleared.

Address: $008A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
Reset:	0	0	0	0	0	0	0	0

Figure 13-14. Timer Control Register 3 (TCTL3)

Address: $008B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset:	0	0	0	0	0	0	0	0

Figure 13-15. Timer Control Register 4 (TCTL4)

Read: Anytime

Write: Anytime

EDGnB and EDGnA — Input Capture Edge Control Bits

These eight pairs of control bits configure the input capture edge detector circuits. See Table 13-2.

Table 13-2. Edge Detector Circuit Configuration

EDGnB	EDGnA	Configuration
0	0	Capture disabled
0	1	Capture on rising edges only
1	0	Capture on falling edges only
1	1	Capture on any edge (rising or falling)

Downloaded from Arrow.com.

13.4.8 Timer Interrupt Mask Registers

Address: $008C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I
Reset:	0	0	0	0	0	0	0	0

Figure 13-16. Timer Interrupt Mask 1 Register (TMSK1)

Read: Anytime

Write: Anytime

C7I–C0I — Input Capture/Output Compare x Interrupt Enable Bits

The bits in TMSK1 correspond bit-for-bit with the bits in the TFLG1 status register. If cleared, the corresponding flag is disabled from causing a hardware interrupt. If set, the corresponding flag is enabled to cause a hardware interrupt.

Address: $008D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TOI	0	PUPT	RDPT	TCRE	PR2	PR1	PR0
Reset:	0	0	0	0	0	0	0	0

Figure 13-17. Timer Interrupt Mask 2 Register (TMSK2)

Read: Anytime

Write: Anytime

TOI — Timer Overflow Interrupt Enable Bit

0 = Interrupt inhibited
1 = Hardware interrupt requested when TOF flag set

PUPT — Timer Port Pullup Resistor Enable Bit

This enable bit controls pullup resistors on the timer port pins when the pins are configured as inputs.

0 = Disable pullup resistor function
1 = Enable pullup resistor function

RDPT — Timer Port Drive Reduction Bit

This bit reduces the effective output driver size which can reduce power supply current and generated noise depending upon pin loading.

0 = Normal output drive capability
1 = Enable output drive reduction function

TCRE — Timer Counter Reset Enable Bit

This bit allows the timer counter to be reset by a successful output compare 7 event. This mode of operation is similar to an up-counting modulus counter.

0 = Counter reset inhibited and counter runs free
1 = Counter reset by a successful output compare 7

NOTE

If TC7 = $0000 and TCRE = 1, TCNT will stay at $0000 continuously.
If TC7 = $FFFF and TCRE = 1, TOF will never be set when TCNT is reset from $FFFF to $0000.

Downloaded from Arrow.com.

PR2, PR1, and PR0 — Timer Prescaler Select Bits

These three bits specify the number of ÷2 stages that are to be inserted between the module clock and the main timer counter. See Table 13-3. The newly selected prescale factor will not take effect until the next synchronized edge where all prescale counter stages equal 0.

Table 13-3. Prescaler Selection

Value	PR2	PR1	PR0	Prescale Factor
0	0	0	0	1
1	0	0	1	2
2	0	1	0	4
3	0	1	1	8
4	1	0	0	16
5	1	0	1	32
6	1	1	0	64
7	1	1	1	128

13.4.9 Main Timer Interrupt Flag Registers

Address: $008E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F
Reset:	0	0	0	0	0	0	0	0

Figure 13-18. Main Timer Interrupt Flag 1 (TFLG1)

Read: Anytime

Write: Used in the clearing mechanism (set bits cause corresponding bits to be cleared). Writing a 0 will not affect current bit status.

C7F–C0F — Input Capture/Output Compare Channel n Flag

TFLG1 indicates when interrupt conditions have occurred. To clear a bit in the flag register, write a 1 to the bit.

Use of the TFMOD bit in the input control system control register (ICSYS) register ($AB) in conjunction with the use of the ICOVW register ($AA) allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture.
When TFFCA bit in TSCR register is set, a read from an input capture or a write into an output compare channel ($90–$9F) will cause the corresponding channel flag CnF to be cleared. See Figure 13-19.

Downloaded from Arrow.com.

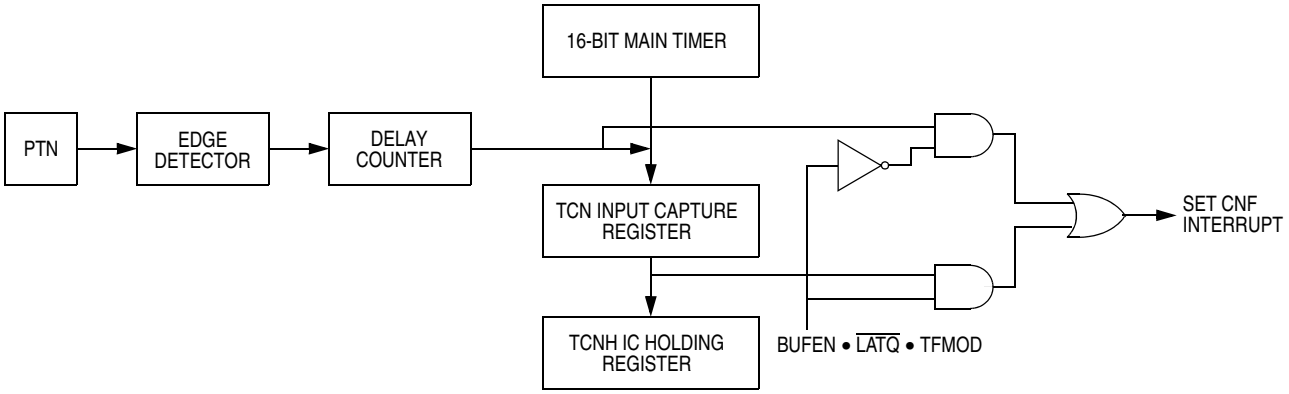


Figure 13-19. C3F–C0F Interrupt Flag Setting

Address: $008F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TOF	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 13-20. Main Timer Interrupt Flag 2 (TFLG2)

Read: Anytime

Write: Used in clearing mechanism (set bits cause corresponding bits to be cleared). Any access to TCNT will clear the TFLG2 register, if the TFFCA bit in the TSCR register is set.

TFLG2 indicates when interrupt conditions have occurred. To clear a bit in the flag register, set the bit to 1.

TOF — Timer Overflow Flag

TOF is set when the 16-bit free-running timer overflows from $FFFF to $0000. This bit is cleared automatically by a write to the TFLG2 register with bit 7 set. See the explanation of the TCRE control bit in 13.4.8 Timer Interrupt Mask Registers.)

13.4.10 Timer Input Capture/Output Compare Registers

Address: $0090–$0091

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-21. Timer Input Capture/Output Compare Register 0 (TC0)

Downloaded from Arrow.com.

Address: $0092–$0093

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-22. Timer Input Capture/Output Compare Register 1 (TC1)

Address: $0094–$0095

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-23. Timer Input Capture/Output Compare Register 2 (TC2)

Address: $0096–$0097

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-24. Timer Input Capture/Output Compare Register 3 (TC3)

Address: $0098–$0099

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-25. Timer Input Capture/Output Compare Register 4 (TC4)

Downloaded from Arrow.com.

Address: $009A–$009B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-26. Timer Input Capture/Output Compare Register 5 (TC5)

Address: $009C–$009D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-27. Timer Input Capture/Output Compare Register 6 (TC6)

Address: $009E–$009F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-28. Timer Input Capture/Output Compare Register 7 (TC7)

Read: Anytime

Write: Anytime for output compare function

Depending on the TIOS bit for the corresponding channel, these registers are used to latch the value of the free-running counter when a defined transition is sensed by the corresponding input capture edge detector or to trigger an output action for output compare.

Writes to these registers have no meaning or effect during input capture. All timer input capture/output compare registers are reset to $0000.

Downloaded from Arrow.com.

13.4.11 16-Bit Pulse Accumulator A Control Register

Address: $00A0

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-29. 16-Bit Pulse Accumulator A Control Register (PACTL)

Read: Anytime

Write: Anytime

Sixteen-bit pulse accumulator A (PACA) is formed by cascading the 8-bit pulse accumulators PAC3 and PAC2. When PAEN is set, the PACA is enabled. The PACA shares the input pin with IC7.

PAEN — Pulse Accumulator A System Enable Bit

PAEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless the pulse accumulator is disabled.

0 = 16-bit pulse accumulator A system disabled. Eight-bit PAC3 and PAC2 can be enabled when their related enable bits in ICPACR ($A8) are set. Pulse accumulator input edge flag (PAIF) function is disabled.

1 = Pulse accumulator A system enabled. The two 8-bit pulse accumulators, PAC3 and PAC2, are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled, the PACN3 and PACN2 registers' contents are, respectively, the high and low byte of the PACA. PA3EN and PA2EN control bits in ICPACR ($A8) have no effect. Pulse accumulator input edge flag (PAIF) function is enabled.

PAMOD — Pulse Accumulator Mode Bit

0 = Event counter mode

1 = Gated time accumulation mode

PEDGE — Pulse Accumulator Edge Control Bit

For PAMOD bit = 0, event counter mode

0 = Falling edges on PT7 pin cause the count to be incremented.

1 = Rising edges on PT7 pin cause the count to be incremented.

For PAMOD bit = 1, gated time accumulation mode

0 = PT7 input pin high enables M divided by 64 clock to pulse accumulator and the trailing falling edge on PT7 sets the PAIF flag.

1 = PT7 input pin low enables M divided by 64 clock to pulse accumulator and the trailing rising edge on PT7 sets the PAIF flag.

PAMOD	PEDGE	Pin Action
0	0	Falling edge
0	1	Rising edge
1	0	Divide by 64 clock enabled with pin high level
1	1	Divide by 64 clock enabled with pin low level

NOTE

If the timer is not active (TEN = 0 in TSCR), there is no divide-by-64 since the E ÷ 64 clock is generated by the timer prescaler.

Downloaded from Arrow.com.

CLK1 and CLK0 — Clock Select Bits

CLK1	CLK0	Clock Source
0	0	Use timer prescaler clock as timer counter clock
0	1	Use PACLK as input to timer counter clock
1	0	Use PACLK/256 as timer counter clock frequency
1	1	Use PACLK/65,536 as timer counter clock frequency

If the pulse accumulator is disabled (PAEN = 0), the prescaler clock from the timer is always used as an input clock to the timer counter. The change from one selected clock to the other happens immediately after these bits are written.

PAOVI — Pulse Accumulator A Overflow Interrupt Enable Bit

0 = Interrupt inhibited
1 = Interrupt requested if PAOVF is set

PAI — Pulse Accumulator Input Interrupt Enable Bit

0 = Interrupt inhibited
1 = Interrupt requested if PAIF is set

13.4.12 Pulse Accumulator A Flag Register

Address: $00A1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	PAOVF	PAIF
Reset:	0	0	0	0	0	0	0	0

Figure 13-30. Pulse Accumulator A Flag Register (PAFLG)

Read: Anytime

Write: Anytime

When the TFFCA bit in the TSCR register is set, any access to the PACNT register will clear all the flags in the PAFLG register.

PAOVF — Pulse Accumulator A Overflow Flag

Set when the 16-bit pulse accumulator A overflows from $FFFF to $0000 or when 8-bit pulse accumulator 3 (PAC3) overflows from $FF to $00. This bit is cleared automatically by a write to the PAFLG register with bit 1 set.

PAIF — Pulse Accumulator Input Edge Flag

Set when the selected edge is detected at the PT7 input pin. In event mode, the event edge triggers PAIF and, in gated time accumulation mode, the trailing edge of the gate signal at the PT7 input pin triggers PAIF. This bit is cleared by a write to the PAFLG register with bit 0 set. Any access to the PACN3 and PACN2 registers will clear all the flags in this register when TFFCA bit in register TSCR ($86) is set.

Downloaded from Arrow.com.

13.4.13 Pulse Accumulators Count Registers

Address: $00A2

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-31. Pulse Accumulator Count Register 3 (PACN3)

Address: $00A3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-32. Pulse Accumulator Count Register 2 (PACN2)

Read: Anytime

Write: Anytime

The two 8-bit pulse accumulators, PAC3 and PAC2, are cascaded to form the PACA 16-bit pulse accumulator. When PACA in enabled (PAEN = 1 in PACTL, $A0) the PACN3 and PACN2 registers' contents are, respectively, the high and low bytes of the PACA.

When PACN3 overflows from $FF to $00, the interrupt flag PAOVF in PAFLG ($A1) is set. Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

Address: $00A4

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-33. Pulse Accumulator Count Register 1 (PACN1)

Address: $00A5

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-34. Pulse Accumulator Count Register 0 (PACN0)

Read: Anytime

Write: Anytime

Downloaded from Arrow.com.

The two 8-bit pulse accumulators, PAC1 and PAC0, are cascaded to form the PACB 16-bit pulse accumulator. When PACB in enabled, (PBEN = 1 in PBCTL, $B0) the PACN1 and PACN0 register contents are, respectively, the high and low bytes of the PACB.

When PACN1 overflows from $FF to $00, the interrupt flag PBOVF in PBFLG ($B1) is set. Full count register access should take place in one clock cycle. A separate read/write for high byte and low byte will give a different result than accessing them as a word.

13.4.14 16-Bit Modulus Down-Counter Control Register

Address: $00A6

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	MCZI	MODMC	RDMCL	ICLAT	FLMC	MCEN	MCPR1	MCPR0
Reset:	0	0	0	0	0	0	0	0

Figure 13-35. 16-Bit Modulus Down-Counter Control Register (MCCTL)

Read: Anytime

Write: Anytime

MCZI — Modulus Counter Underflow Interrupt Enable Bit

0 = Modulus counter interrupt is disabled.
1 = Modulus counter interrupt is enabled.

MODMC — Modulus Mode Enable Bit

0 = The counter counts once from the value written to it and will stop at $0000.
1 = Modulus mode is enabled. When the counter reaches $0000, the counter is loaded with the latest value written to the modulus count register.

NOTE

For proper operation, the MCEN bit should be cleared before modifying the MODMC bit to reset the modulus counter to $FF.

RDMCL — Read Modulus Down-Counter Load Bit

0 = Reads of the modulus count register will return the present value of the count register.
1 = Reads of the modulus count register will return the contents of the load register.

ICLAT — Input Capture Force Latch Action Bit

When input capture latch mode is enabled (LATQ and BUFEN bit in ICSYS ($AB) are set), writing 1 to this bit immediately forces the contents of the input capture registers TC0 to TC3 and their corresponding 8-bit pulse accumulators to be latched into the associated holding registers. The pulse accumulators will be automatically cleared when the latch action occurs.

Writing 0 to this bit has no effect. Read of this bit aways will return 0.

FLMC — Force Load Register into the Modulus Counter Count Register Bit

This bit is active only when the modulus down-counter is enabled (MCEN = 1). Writing a 1 into this bit loads the load register into the modulus counter count register. This also resets the modulus counter prescaler. Writing 0 to this bit has no effect.

When MODMC = 0, the counter starts counting and stops at $0000. Reads of this bit will return always 0.

Downloaded from Arrow.com.

MCEN — Modulus Down-Counter Enable Bit

When MCEN = 0, the counter is preset to $FFFF. This will prevent an early interrupt flag when the modulus down-counter is enabled.

0 = Modulus counter disabled.

1 = Modulus counter is enabled.

MCPR1 and MCPR0 — Modulus Counter Prescaler Select Bits

These two bits specify the division rate of the modulus counter prescaler. The newly selected prescaler division rate will not be effective until a load of the load register into the modulus counter count register occurs.

MCPR1	MCPR0	Prescalar Division Rate
0	0	1
0	1	4
1	0	8
1	1	16

13.4.15 16-Bit Modulus Down-Counter Flag Register

Address: $00A7

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	MCZF	0	0	0	POLF3	POLF2	POLF1	POLF0
Reset:	0	0	0	0	0	0	0	0

Figure 13-36. 16-Bit Modulus Down-Counter Flag Register (MCFLG)

Read: Anytime

Write: Only for clearing bit 7

MCZF — Modulus Counter Underflow Interrupt Flag

The flag is set when the modulus down-counter reaches $0000. Writing 1 to this bit clears the flag. Writing 0 has no effect. Any access to the MCCNT register will clear the MCZF flag in this register when TFFCA bit in register TSCR ($86) is set.

POLF3–POLF0 — First Input Capture Polarity Status Bits

This are read-only bits. Writing to these bits has no effect. Each status bit gives the polarity of the first edge which has caused an input capture to occur after capture latch has been read. Each POLFx corresponds to a timer PORTx input.

0 = The first input capture has been caused by a falling edge.

1 = The first input capture has been caused by a rising edge.

Downloaded from Arrow.com.

13.4.16 Input Control Pulse Accumulators Control Register

Address: $00A8

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	PA3EN	PA2EN	PA1EN	PA0EN
Reset:	0	0	0	0	0	0	0	0

Figure 13-37. Input Control Pulse Accumulators Control Register (ICPACR)

Read: Anytime

Write: Anytime

The 8-bit pulse accumulators, PAC3 and PAC2, can be enabled only if PAEN in PATCL ($A0) is cleared. If PAEN is set, PA3EN and PA2EN have no effect. The 8-bit pulse accumulators, PAC1 and PAC0, can be enabled only if PBEN in PBTCL ($B0) is cleared. If PBEN is set, PA1EN and PA0EN have no effect.

PAxEN — 8-Bit Pulse Accumulator x Enable Bits

0 = 8-bit pulse accumulator disabled
1 = 8-bit pulse accumulator enabled

13.4.17 Delay Counter Control Register

Address: $00A9

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	DLY1	DLY0
Reset:	0	0	0	0	0	0	0	0

Figure 13-38. Delay Counter Control Register (DLYCT)

Read: Anytime

Write: Anytime

If enabled, after detection of a valid edge on input capture pin, the delay counter counts the pre-selected number of P clock (module clock) cycles, then it will generate a pulse on its output. The pulse is generated only if the level of input signal, after the preset delay, is the opposite of the level before the transition.This will avoid reaction to narrow input pulses.

After counting, the counter will be cleared automatically. Delay between two active edges of the input signal period should be longer than the selected counter delay.

DLYx — Delay Counter Select Bits

DLY1	DLY0	Delay
0	0	Disabled (bypassed)
0	1	256 P clock cycles
1	0	512 P clock cycles
1	1	1024 P clock cycles

Downloaded from Arrow.com.

13.4.18 Input Control Overwrite Register

Address: $00AA

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	NOVW7	NOVW6	NOVW5	NOVW4	NOVW3	NOVW2	NOVW1	NOVW0
Reset:	0	0	0	0	0	0	0	0

Figure 13-39. Input Control Overwrite Register (ICOVW)

Read: Anytime

Write: Anytime

An IC register is empty when it has been read or latched into the holding register. A holding register is empty when it has been read.

NOVWx — No Input Capture Overwrite Bits

0 = The contents of the related capture register or holding register can be overwritten when a new input capture or latch occurs.

1 = The related capture register or holding register cannot be written by an event unless they are empty (see 13.3.1 IC Channels). This will prevent the captured value to be overwritten until it is read or latched in the holding register.

13.4.19 Input Control System Control Register

Address: $00AB

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SH37	SH26	SH15	SH04	TFMOD	PACMX	BUFEN	LATQ
Reset:	0	0	0	0	0	0	0	0

Figure 13-40. Input Control System Control Register (ICSYS)

Read: Anytime

Write: May be written once (SMODN = 1). Writes are always permitted when SMODN = 0.

SHxy — Share Input Action of Input Capture Channels x and y Bits

0 = Normal operation

1 = The channel input x causes the same action on the channel y. The port pin x and the corresponding edge detector is used to be active on the channel y.

TFMOD — Timer Flag-Setting Mode Bit

Use of the TFMOD bit in the ICSYS register ($AB) in conjunction with the use of the ICOVW register ($AA) allows a timer interrupt to be generated after capturing two values in the capture and holding registers instead of generating an interrupt for every capture.

Downloaded from Arrow.com.

By setting TFMOD in queue mode, when NOVW bit is set and the corresponding capture and holding registers are emptied, an input capture event will first update the related input capture register with the main timer contents. At the next event, the TCn data is transferred to the TCnH register, the TCn is updated, and the CnF interrupt flag is set. See Figure 13-19. In all other input capture cases, the interrupt flag is set by a valid external event on PTn.

0 = The timer flags C3F–C0F in TFLG1 ($8E) are set when a valid input capture transition on the corresponding port pin occurs.

1 = If in queue mode (BUFEN = 1 and LATQ = 0), the timer flags C3F–C0F in TFLG1 ($8E) are set only when a latch on the corresponding holding register occurs. If the queue mode is not engaged, the timer flags C3F–C0F are set the same way as for TFMOD = 0.

PACMX — 8-Bit Pulse Accumulators Maximum Count Bit

0 = Normal operation. When the 8-bit pulse accumulator has reached the value $FF, with the next active edge, it will be incremented to $00.

1 = When the 8-bit pulse accumulator has reached the value $FF, it will not be incremented further. The value $FF indicates a count of 255 or more.

BUFEN — IC Buffer Enable Bit

0 = Input capture and pulse accumulator holding registers are disabled.

1 = Input capture and pulse accumulator holding registers are enabled. The latching mode is defined by LATQ control bit. Writing a 1 into ICLAT bit in MCCTL ($A6) when LATQ is set, will produce latching of input capture and pulse accumulator registers into their holding registers.

LATQ — Input Control Latch or Queue Mode Enable Bit

The BUFEN control bit should be set to enable the IC and pulse accumulators' holding registers. Otherwise, LATQ latching modes are disabled.

Writing one into ICLAT bit in MCCTL ($A6), when LATQ and BUFEN are set will produce latching of input capture and pulse accumulators registers into their holding registers.

0 = Queue mode of input capture is enabled. The main timer value is memorized in the IC register by a valid input pin transition. With a new occurrence of a capture, the value of the IC register will be transferred to its holding register and the IC register memorizes the new timer value.

1 = Latch mode is enabled. Latching function occurs when modulus down-counter reaches 0 or a 0 is written into the count register MCCNT (see 13.3.1.2 Buffered IC Channels). With a latching event the contents of IC registers and 8-bit pulse accumulators are transferred to their holding registers. The 8-bit pulse accumulators are cleared.

13.4.20 Timer Test Register

Address: $00AD

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	TCBYP	PCBYP(1)
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

1. Available only on MC68HC912B32 devices.

Figure 13-41. Timer Test Register (TIMTST)

Read: Anytime

Write: Only in special mode (SMOD = 1)

Downloaded from Arrow.com.

TCBYP — Main Timer Divider Chain Bypass Bit

0 = Normal operation

1 = For testing only. The 16-bit free-running timer counter is divided into two 8-bit halves and the prescaler is bypassed. The clock drives both halves directly. When the high byte of timer counter TCNT ($84) overflows from $FF to $00, the TOF flag in TFLG2 ($8F) will be set.

13.4.21 Timer Port Data Register

Address: $00AE

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0
Timer:	I/0C7	I/OC6	I/OC5	I/OC4	I/OC3	I/OC2	I/OC1	I/OC0
Reset:	0	0	0	0	0	0	0	0

Figure 13-42. Timer Port Data Register (PORTT)

Read: Anytime (input return pin level; outputs return data register contents)

Write: Data stored in an internal latch (drives pins only if configured for output)

Since the output compare 7 register (OC7) shares pins with the pulse accumulator input, the only way for the pulse accumulator to receive an independent input from OC7 is by setting both OM7 and OL7 to be 0, and also OC7M7 in OC7M register to be 0. OC7 can still reset the counter if enabled while PT7 is used as an input to the pulse accumulator.

PORTT can be read anytime. When configured as an input, a read will return the pin level. When configured as an output, a read will return the latched output data.

NOTE

Writes do not change pin state when the pin is configured for timer output. The minimum pulse width for pulse accumulator input should always be greater than the width of two module clocks due to input synchronizer circuitry. The minimum pulse width for the input capture should always be greater than the width of two module clocks due to input synchronizer circuitry.

13.4.22 Data Direction Register for Timer Port

Address: $00AF

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDT7	DDT6	DDT5	DDT4	DDT3	DDT2	DDT1	DDT0
Reset:	0	0	0	0	0	0	0	0

Figure 13-43. Data Direction Register for Timer Port (DDRT)

Read: Anytime

Write: Anytime

Downloaded from Arrow.com.

DDT[7:0] — Data Direction Bits for Timer Port

The timer forces the I/O state to be an output for each timer port line associated with an enabled output compare. In these cases the data direction bits will not be changed, but have no effect on the direction of these pins. The DDRT will revert to controlling the I/O direction of a pin when the associated timer output compare is disabled. Input captures do not override the DDRT settings.

0 = Configures the corresponding I/O pin for input only
1 = Configures the corresponding I/O pin for output

13.4.23 16-Bit Pulse Accumulator B Control Register

Address: $00B0

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	PBEN	0	0	0	0	PBOV	0
Reset:	0	0	0	0	0	0	0	0

Figure 13-44. 16-Bit Pulse Accumulator B Control Register (PBCTL)

Read: Anytime

Write: Anytime

Sixteen-bit pulse accumulator B (PACB) is formed by cascading the 8-bit pulse accumulators PAC1 and PAC0. When PBEN is set, the PACB is enabled. The PACB shares the input pin with IC0.

PBEN — Pulse Accumulator B System Enable Bit

PBEN is independent from TEN. With timer disabled, the pulse accumulator can still function unless pulse accumulator is disabled.

0 = 16-bit pulse accumulator system disabled. Eight-bit PAC1 and PAC0 can be enabled when their related enable bits in ICPACR ($A8) are set.
1 = Pulse accumulator B system enabled. The two 8-bit pulse accumulators PAC1 and PAC0 are cascaded to form the PACB 16-bit pulse accumulator. When PACB is enabled, the PACN1 and PACN0 register contents are, respectively, the high and low byte of the PACB. PA1EN and PA0EN control bits in ICPACR ($A8) have no effect.

PBOVI — Pulse Accumulator B Overflow Interrupt Enable Bit

0 = Interrupt inhibited
1 = Interrupt requested if PBOVF is set

13.4.24 Pulse Accumulator B Flag Register

Address: $00B1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	PBOVF	0
Reset:	0	0	0	0	0	0	0	0

Figure 13-45. Pulse Accumulator B Flag Register (PBFLG)

- Read:Anytime
- Write:Anytime

Downloaded from Arrow.com.

PBOVF — Pulse Accumulator B Overflow Flag

This bit is set when the 16-bit pulse accumulator B overflows from $FFFF to $0000 or when 8-bit pulse accumulator 1 (PAC1) overflows from $FF to $00. This bit is cleared by a write to the PBFLG register with bit 1 set. Any access to the PACN1 and PACN0 registers will clear the PBOVF flag in this register when TFFCA bit in register TSCR ($86) is set.

13.4.25 8-Bit Pulse Accumulators Holding Registers

Address: $00B2

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-46. 8-Bit Pulse Accumulator Holding Register 3 (PA3H)

Address: $00B3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-47. 8-Bit Pulse Accumulator Holding Register 2 (PA2H)

Address: $00B4

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-48. 8-Bit Pulse Accumulator Holding Register 1 (PA1H)

Address: $00B5

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-49. 8-Bit Pulse Accumulator Holding Register 0 (PA0H)

Read: Anytime

Write: Anytime

These registers are used to latch the value of the corresponding pulse accumulator when the related bits in register ICPACR ($A8) are enabled (see 13.3.2 Pulse Accumulators).

Downloaded from Arrow.com.

13.4.26 Modulus Down-Counter Count Registers

Address: $00B6

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	1	1	1	1	1	1	1	1

Address: $00B7

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	1	1	1	1	1	1	1	1

Figure 13-50. Modulus Down-Counter Count Registers (MCCNT)

Read: Anytime

Write: Anytime

A full access for the counter register should take place in one clock cycle. A separate read/write for high byte and low byte will give different results than accessing them as a word. If the RDMCL bit in MCCTL register is cleared, reads of the MCCNT register will return the present value of the count register. If the RDMCL bit is set, reads of the MCCNT will return the contents of the load register.

If a $0000 is written into MCCNT and modulus counter while LATQ and BUFEN in ICSYS ($AB) register are set, the input capture and pulse accumulator registers will be latched. With a $0000 write to the MCCNT, the modulus counter will stay at 0 and does not set the MCZF flag in MCFLG register.

If modulus mode is enabled (MODMC = 1), a write to this address will update the load register with the value written to it. The count register will not be updated with the new value until the next counter underflow. The FLMC bit in MCCTL ($A6) can be used to immediately update the count register with the new value if an immediate load is desired.

If modulus mode is not enabled (MODMC = 0), a write to this address will clear the prescaler and will immediately update the counter register with the value written to it and down-counts once to $0000.

13.4.27 Timer Input Capture Holding Registers

Address: $00B8

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $00B9

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-51. Timer Input Capture Holding Register 0 (TC0H)

Downloaded from Arrow.com.

Address: $00BA

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $00BB

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-52. Timer Input Capture Holding Register 1 (TC1H)

Address: $00BC

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $00BD

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-53. Timer Input Capture Holding Register 2 (TC2H)

Address: $00BE

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset:	0	0	0	0	0	0	0	0

Address: $00BF

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	0	0	0	0	0	0	0	0

Figure 13-54. Timer Input Capture Holding Register 3 (TC3H)

Read: Anytime

Write: Has no effect

These registers are used to latch the value of the input capture registers TC0–TC3. The corresponding IOSx bits in TIOS ($80) should be cleared (see 13.3.1 IC Channels).

Downloaded from Arrow.com.

13.5 Timer and Modulus Counter Operation in Different Modes

STOP

Timer and modulus counter are off since clocks are stopped.

BGDM

Timer and modulus counter keep on running unless bit 5, TSBCK, of TSCR is set to 1. See 13.4.6 Timer System Control Register.

WAIT

Counters keep on running, unless the TSWAI bit in TSCR is set to 1. See 13.4.6 Timer System Control Register.

NORMAL

Timer and modulus counter keep on running, unless the TEN bit in TSCR and MCEN in MCCTL, respectively, are cleared. See 13.4.6 Timer System Control Register and 13.4.14 16-Bit Modulus Down-Counter Control Register.

TEN = 0

All 16-bit timer operations are stopped; can only access the registers

MCEN = 0

Modulus counter is stopped.

PAEN = 1

Sixteen-bit pulse accumulator A is active.

PAEN = 0

Eight-bit pulse accumulators 3 and 2 can be enabled. See 13.4.16 Input Control Pulse Accumulators Control Register.

PBEN = 1

Sixteen-bit pulse accumulator B is active.

PBEN = 0

Eight-bit pulse accumulators 1 and 0 can be enabled. See 13.4.16 Input Control Pulse Accumulators Control Register.

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 14
Serial Interface

14.1 Introduction

- The serial interface of the MCU consists of two independent serial input/output (I/O) subsystems:
- Serial communication interface (SCI)
- Serial peripheral interface (SPI)

Each serial pin shares function with the general-purpose port pins of port S. The SCI is an NRZ (non-return to zero) type system that is compatible with standard RS-232 systems. The SCI system has a single-wire operation mode which allows the unused pin to be available as general-purpose I/O. The SPI subsystem, which is compatible with the M68HC11 SPI, includes new features such as $\overline{SS}$ output and bidirectional mode.

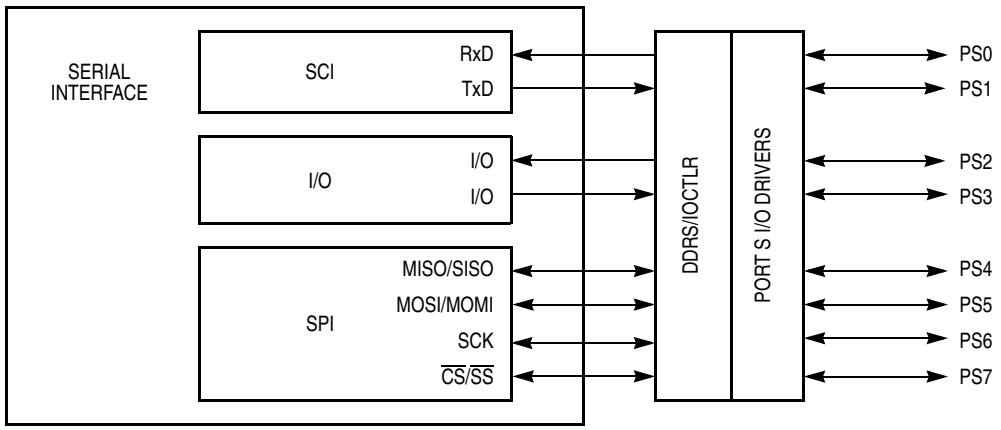


Figure 14-1. Serial Interface Block Diagram

Downloaded from Arrow.com.

14.2 Serial Communication Interface (SCI)

The SCI on the MCU is an NRZ format (one start, eight or nine data, and one stop bit) asynchronous communication system with independent internal baud rate generation circuitry and an SCI transmitter and receiver. It can be configured for eight or nine data bits (one of which may be designated as a parity bit, odd or even). If enabled, parity is generated in hardware for transmitted and received data. Receiver parity errors are flagged in hardware. The baud rate generator is based on a modulus counter, allowing flexibility in choosing baud rates. There is a receiver wakeup feature, an idle line detect feature, a loop-back mode, and various error detection features. Two port pins provide the external interface for the transmitted data (TXD) and the received data (RXD). See Figure 14-2.

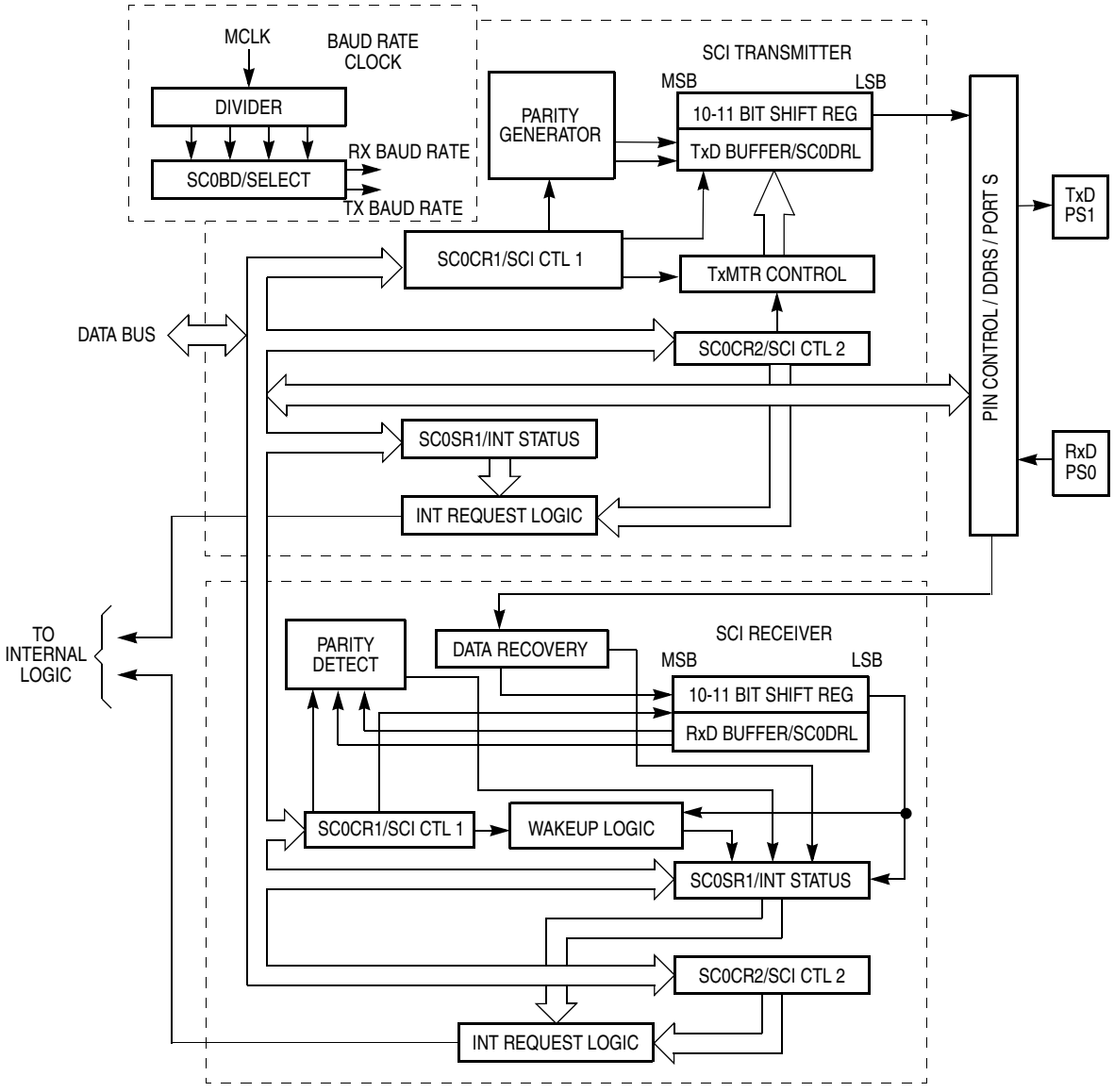


Figure 14-2. Serial Communications Interface Block Diagram

Downloaded from Arrow.com.

14.2.1 Data Format

The serial data format requires these conditions:

- An idle-line in the high state before transmission or reception of a message
- A start bit (logic 0), transmitted or received, that indicates the start of each character
- Data that is transmitted or received least significant bit (LSB) first
- A stop bit (logic 1) used to indicate the end of a frame

A frame consists of:

- A start bit
- A character of eight or nine data bits
- A stop bit

A BREAK is defined as the transmission or reception of a logic 0 for one frame or more.

This SCI supports hardware parity for transmit and receive.

14.2.2 SCI Baud Rate Generation

The basis of the SCI baud rate generator is a 13-bit modulus counter. This counter gives the generator the flexibility necessary to achieve a reasonable level of independence from the CPU operating frequency and still be able to produce standard baud rates with a minimal amount of error. The clock source for the generator comes from the P clock.

Table 14-1. Baud Rate Generation

Desired SCI Baud Rate	BR Divisor for P = 4.0 MHz	BR Divisor for P = 8.0 MHz
110	2273	4545
300	833	1667
600	417	833
1200	208	417
2400	104	208
4800	52	104
9600	26	52
14,400	17	35
19,200	13	26
38,400	—	13

Downloaded from Arrow.com.

14.2.3 SCI Register Descriptions

Control and data registers for the SCI subsystem are described here. The memory address indicated for each register is the default address that is in use after reset. The entire 512-byte register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space.

14.2.3.1 SCI Baud Rate Control Register

Address: $00C0

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BTST	BSPL	BRLD	SBR12	SBR11	SBR10	SBR9	SBR8
Reset:	0	0	0	0	0	0	0	0

Figure 14-3. SCI Baud Rate Control Register (SC0BDH)

Address: $00C1

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
Reset:	0	0	0	0	0	1	0	0

Figure 14-4. SCI Baud Rate Control Register (SC0BDL)

Read: Anytime

Write: SBR12–SBR0 anytime; low-order byte must be written for change to take effect — SBR15–SBR13 only in special modes

SC0BDH and SC0BDL are considered together as a 16-bit baud rate control register.

The value in SBR12–SBR0 determines the baud rate of the SCI. The desired baud rate is determined by the following formula:

$$\text{SCI baud rate} = \frac{\text{MCLK}}{16 \times \text{BR}}$$

Which is equivalent to:

$$\text{BR} = \frac{\text{MCLK}}{16 \times \text{SCI baud rate}}$$

BR is the value written to bits SBR12–SBR0 to establish the baud rate.

NOTE

The baud rate generator is disabled until the TE or RE bit in SC0CR2 register is set for the first time after reset and/or the baud rate generator is disabled when SBR12–SBR0 = 0.

BTST — Reserved for test function

BSPL — Reserved for test function

BRLD — Reserved for test function

Downloaded from Arrow.com.

14.2.3.2 SCI Control Register 1

Address: $00C2

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LOOPS	WOMS	RSRC	M	WAKE	ILT	PE	PT
Reset:	0	0	0	0	0	0	0	0

Figure 14-5. SCI Control Register 1 (SC0CR1)

Read: Anytime

Write: Anytime

LOOPS — SCI LOOP Mode/Single-Wire Mode Enable Bit

0 = SCI transmit and receive sections operate normally.

1 = SCI receive section is disconnected from the RXD pin and the RXD pin is available as general-purpose I/O.

The receiver input is determined by the RSRC bit. The transmitter output is controlled by the associated DDRS bit. Both the transmitter and the receiver must be enabled to use the LOOP or the single-wire mode.

If the DDRS bit associated with the TXD pin is set during the LOOPS = 1, the TXD pin outputs the SCI waveform. If the DDRS bit associated with the TXD pin is clear during the LOOPS = 1, the TXD pin becomes high (IDLE line state) for RSRC = 0 and high impedance for RSRC = 1. Refer to Table 14-2.

Table 14-2. Loop Mode Functions

LOOPS	RSRC	DDS1	WOMS	Function of Port S Bit 1/3
0	x	x	x	Normal operations
1	0	0	0/1	LOOP mode without TXD output (TXD = high impedance)
1	0	1	0	LOOP mode with TXD output (CMOS)
1	0	1	1	LOOP mode with TXD output (open-drain)
1	1	0	x	Single-wire mode without TXD output (pin is used as receiver input only, TXD = high Impedance)
1	1	1	0	Single-wire mode with TXD output (The output is also fed back to receiver input, CMOS.)
1	1	1	1	Single wire mode for the receiving and transmitting (open-drain)

WOMS — Wired-OR Mode for Serial Pins

This bit controls the two pins (TXD and RXD) associated with the SCIx section.

0 = Pins operate in a normal mode with both high and low drive capability. To affect the RXD bit, that bit would have to be configured as an output (via DDS0/2) which is the single-wire case when using the SCI. WOMS bit still affects general-purpose output on TXD and RXD pins when SCIx is not using these pins.

1 = Each pin operates in an open drain fashion if that pin is declared as an output.

Downloaded from Arrow.com.

RSRC — Receiver Source Bit

When LOOPS = 1, the RSRC bit determines the internal feedback path for the receiver.

0 = Receiver input connected to the transmitter internally
(not TXD pin)
1 = Receiver input connected to the TXD pin

M — Mode Bit (select character format)

0 = One start, eight data, one stop bit
1 = One start, eight data, ninth data, one stop bit

WAKE — Wakeup by Address Mark/Idle Bit

0 = Wakeup by IDLE line recognition
1 = Wakeup by address mark (last data bit set)

ILT — Idle Line Type Bit

This bit determines which of two types of idle line detection is used by the SCI receiver.

0 = Short idle line mode enabled
1 = Long idle line mode detected

In short mode, the SCI circuitry begins counting 1s in the search for the idle line condition immediately after the start bit. This means that the stop bit and any bits that were 1s before the stop bit could be counted in that string of 1s, resulting in earlier recognition of an idle line.

In long mode, the SCI circuitry does not begin counting 1s in the search for the idle line condition until a stop bit is received. Therefore, the last byte's stop bit and preceding 1 bits do not affect how quickly an idle line condition can be detected.

PE — Parity Enable Bit

0 = Parity disabled
1 = Parity enabled

PT — Parity Type Bit

If parity is enabled, this bit determines even or odd parity for both the receiver and the transmitter. An even number of 1s in the data character causes the parity bit to be 0 and an odd number of 1s causes the parity bit to be 1.

0 = Even parity selected
1 = Odd parity selected

Downloaded from Arrow.com.

14.2.3.3 SCI Control Register 2

Address: $00C3

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
Reset:	0	0	0	0	0	0	0	0

Figure 14-6. SCI Control Register 2 (SC0CR2)

Read: Anytime

Write: Anytime

TIE — Transmit Interrupt Enable Bit

0 = TDRE interrupts disabled
1 = SCI interrupt requested when TDRE status flag is set

TCIE — Transmit Complete Interrupt Enable Bit

0 = TC interrupts disabled
1 = SCI interrupt requested when TC status flag is set

RIE — Receiver Interrupt Enable Bit

0 = RDRF and OR interrupts disabled
1 = SCI interrupt requested when RDRF status flag or OR status flag is set

ILIE — Idle Line Interrupt Enable Bit

0 = IDLE interrupts disabled
1 = SCI interrupt requested when IDLE status flag is set

TE — Transmitter Enable Bit

0 = Transmitter disabled
1 = SCI transmit logic is enabled and the TXD pin (port S bit 1) is dedicated to the transmitter. The TE bit can be used to queue an idle preamble.

RE — Receiver Enable Bit

0 = Receiver disabled
1 = Enables the SCI receive circuitry

RWU — Receiver Wakeup Control Bit

0 = Normal SCI receiver
1 = Enables the wakeup function and inhibits further receiver interrupts. Normally, hardware wakes the receiver by automatically clearing this bit.

SBK — Send Break Bit

0 = Break generator off
1 = Generate a break code, at least 10 or 11 contiguous 0s.

As long as SBK remains set, the transmitter sends 0s. When SBK is changed to 0, the current frame of all 0s is finished before the TxD line goes to the idle state. If SBK is toggled on and off, the transmitter sends only 10 (or 11) 0s and then reverts to mark idle or sending data.

Downloaded from Arrow.com.

14.2.3.4 SCI Status Register 1

Address: $00C4

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
Write:								
Reset:	1	1	0	0	0	0	0	0

= Unimplemented

Figure 14-7. SCI Status Register 1 (SC0SR1)

Read: Anytime; used in auto clearing mechanism

Write: Has no meaning or effect

The bits in these registers are set by various conditions in the SCI hardware and are cleared automatically by special acknowledge sequences. The receive related flag bits in SC0SR1 (RDRF, IDLE, OR, NF, FE, and PF) are all cleared by a read of the SC0SR1 register followed by a read of the transmit/receive data register low byte. However, only those bits which were set when SC0SR1 was read will be cleared by the subsequent read of the transmit/receive data register low byte. The transmit related bits in SC0SR1 (TDRE and TC) are cleared by a read of the SC0SR1 register followed by a write to the transmit/receive data register low byte.

TDRE — Transmit Data Register Empty Flag

New data is not transmitted unless SC0SR1 is read before writing to the transmit data register. Reset sets this bit.

0 = SC0DR busy

1 = Any byte in the transmit data register is transferred to the serial shift register so new data may now be written to the transmit data register.

TC — Transmit Complete Flag

Flag is set when the transmitter is idle (no data, preamble, or break transmission in progress). Clear by reading SC0SR1 with TC set and then writing to SC0DR.

0 = Transmitter busy

1 = Transmitter idle

RDRF — Receive Data Register Full Flag

Once cleared, IDLE is not set again until the RxD line has been active and becomes idle again. RDRF is set if a received character is ready to be read from SC0DR. Clear the RDRF flag by reading SC0SR1 with RDRF set and then reading SC0DR.

0 = SC0DR empty

1 = SC0DR full

IDLE — Idle Line Detected Flag

Receiver idle line is detected (the receipt of a minimum of 10 or 11 consecutive 1s). This bit is not set by the idle line condition when the RWU bit is set. Once cleared, IDLE is not set again until after RDRF has been set (after the line has been active and becomes idle again).

0 = RxD line active

1 = RxD line idle

Downloaded from Arrow.com.

OR — Overrun Error Flag

New byte is ready to be transferred from the receive shift register to the receive data register and the receive data register is already full (RDRF bit is set). Data transfer is inhibited until this bit is cleared.

0 = No overrun
1 = Overrun detected

NF — Noise Error Flag

Set during the same cycle as the RDRF bit but not set in the case of an overrun (OR).

0 = Unanimous decision
1 = Noise on a valid start bit, any of the data bits, or on the stop bit

FE — Framing Error Flag

Set when a 0 is detected where a stop bit was expected. Clear the FE flag by reading SC0SR1 with FE set and then reading SC0DR.

0 = Stop bit detected
1 = Zero detected rather than a stop bit

PF — Parity Error Flag

Indicates if received data's parity matches parity bit. This feature is active only when parity is enabled. The type of parity tested for is determined by the PT (parity type) bit in SC0CR1.

0 = Parity correct
1 = Incorrect parity detected

14.2.3.5 SCI Status Register 2

Address: $00C5

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	RAF
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-8. SCI Status Register 2 (SC0SR2)

Read: Anytime

Write: Has no meaning or effect

RAF — Receiver Active Flag

This bit is controlled by the receiver front end. It is set during the RT1 time period of the start bit search. It is cleared when an idle state is detected or when the receiver circuitry detects a false start bit (generally due to noise or baud rate mismatch).

0 = Character is not being received.
1 = Character is being received.

Downloaded from Arrow.com.

14.2.3.6 SCI Data Register

Address: $00C6

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	0	0	0	0	0	0
Write:								
Reset:	U	U	0	0	0	0	0	0

= Unimplemented U = Unaffected

Figure 14-9. SCI Data Register High (SC0DRH)

Address: $00C7

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
Write:								
Reset:	Unaffected by reset							

Figure 14-10. SCI Data Register Low (SC0DRL)

Read: Anytime

Write: Varies on a bit by bit basis

R8 — Receive Bit 8

Write has no meaning or effect.

This bit is the ninth serial data bit received when the SCI system is configured for 9-data-bit operation.

T8 — Transmit Bit 8

Write anytime.

This bit is the ninth serial data bit transmitted when the SCI system is configured for 9-data-bit operation. When using 9-bit data format, this bit does not have to be written for each data word. The same value is transmitted as the ninth bit until this bit is rewritten.

R7T7–R0T0 — Receive/Transmit Data Bits 7 to 0

Reads access the eight bits of the read-only SCI receive data register (RDR). Writes access the eight bits of the write-only SCI transmit data register (TDR). SC0DRL and SC0DRH form the 9-bit data word for the SCI. If the SCI is being used with a 7- or 8-bit data word, only SC0DRL needs to be accessed. If a 9-bit format is used, the upper register should be written first to ensure that it is transferred to the transmitter shift register with the lower register.

Downloaded from Arrow.com.

14.3 Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) allows the MCU to communicate synchronously with peripheral devices and other microprocessors. The SPI system in the MCU can operate as a master or as a slave. The SPI is also capable of interprocessor communications in a multiple master system.

When the SPI is enabled, all pins that are defined by the configuration as inputs will be inputs regardless of the state of the DDRS bits for those pins. All pins that are defined as SPI outputs will be outputs only if the DDRS bits for those pins are set. Any SPI output whose corresponding DDRS bit is cleared can be used as a general-purpose input.

A bidirectional serial pin is possible using the DDRS as the direction control.

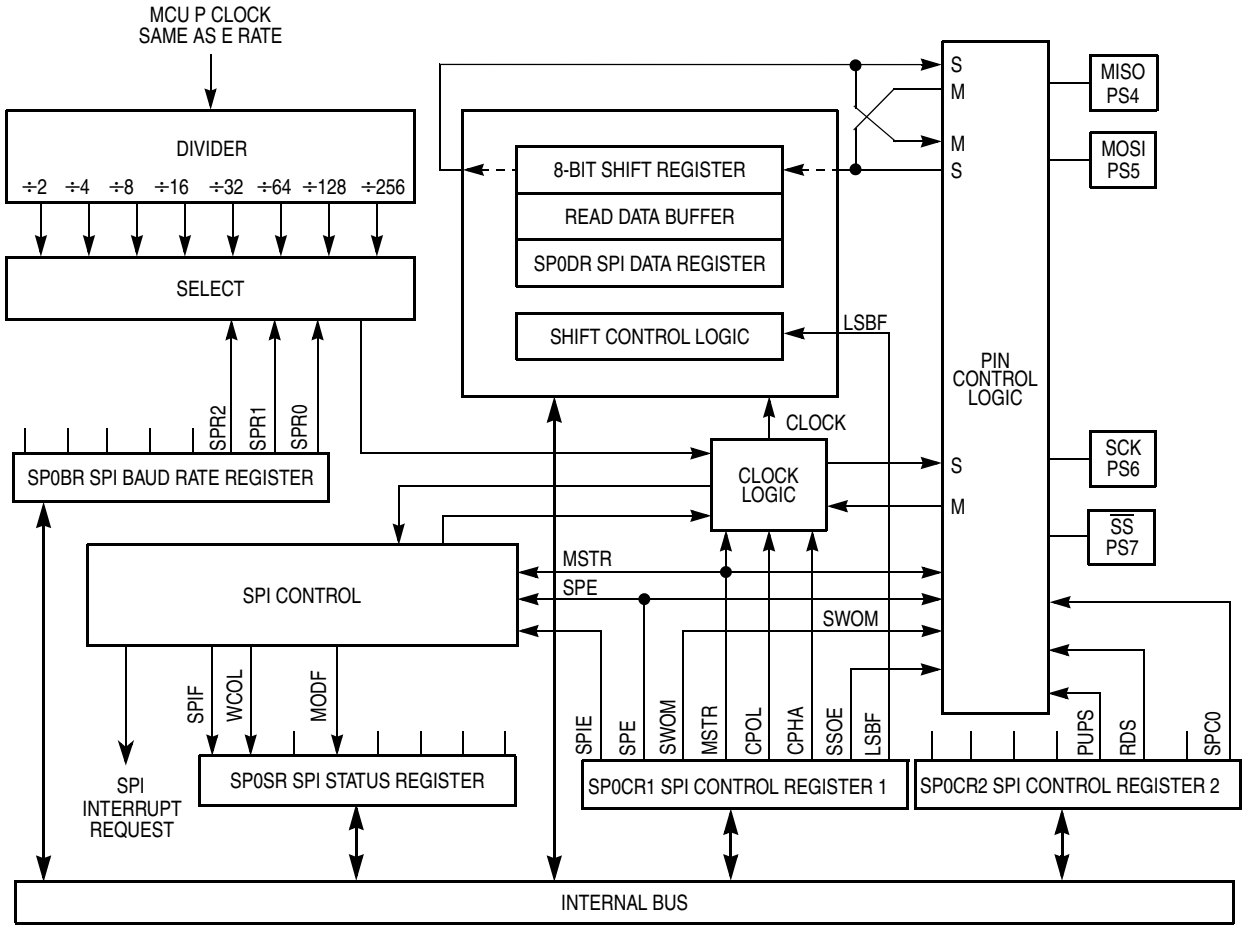


Figure 14-11. Serial Peripheral Interface Block Diagram

Downloaded from Arrow.com.

14.3.1 SPI Baud Rate Generation

The P clock is input to a divider series and the resulting SPI clock rate may be selected to be P divided by 2, 4, 8, 16, 32, 64, 128, or 256. Three bits in the SP0BR register control the SPI clock rate. This baud rate generator is activated only when SPI is in the master mode and serial transfer is taking place. Otherwise, this divider is disabled to save power.

14.3.2 SPI Operation

In the SPI system, the 8-bit data register in the master and the 8-bit data register in the slave are linked to form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master so the data is effectively exchanged between the master and the slave. Data written to the SP0DR register of the master becomes the output data for the slave and data read from the SP0DR register of the master after a transfer operation is the input data from the slave.

A clock phase control bit (CPHA) and a clock polarity control bit (CPOL) in the SPI control register 1 (SP0CR1) select one of four possible clock formats to be used by the SPI system. The CPOL bit simply selects non-inverted or inverted clock. The CPHA bit is used to accommodate two fundamentally different protocols by shifting the clock by one half cycle or no phase shift.

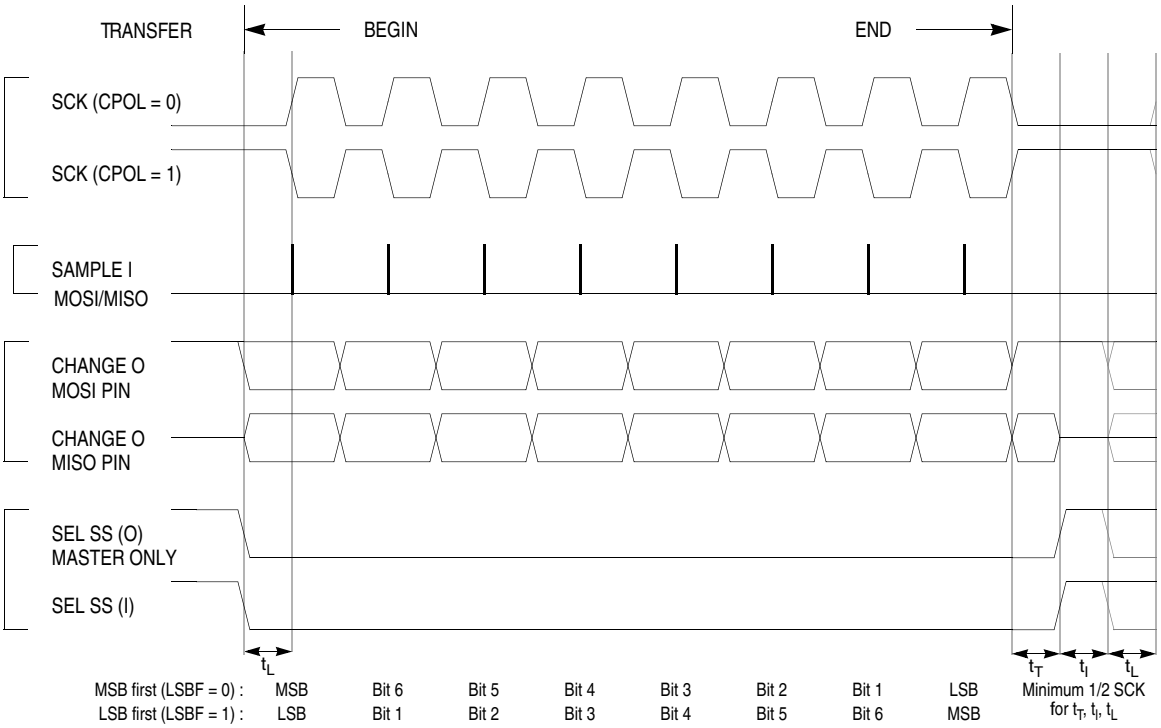


Figure 14-12. SPI Clock Format 0 (CPHA = 0)

Downloaded from Arrow.com.

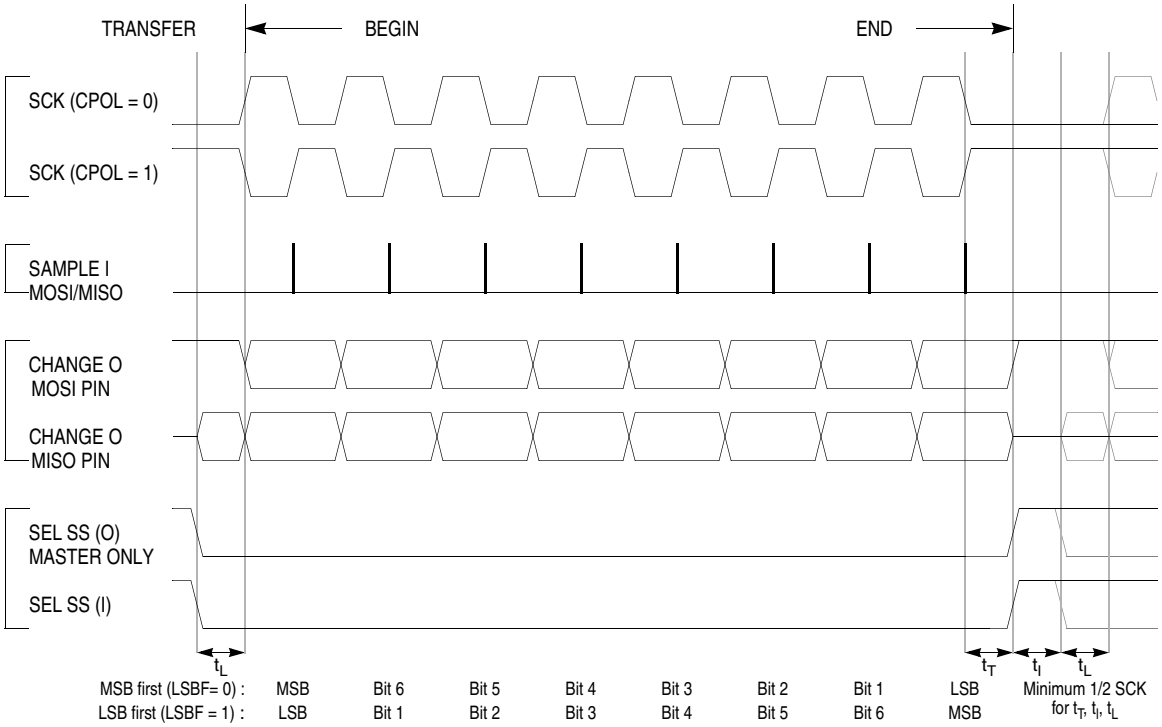


Figure 14-13. SPI Clock Format 1 (CPHA = 1)

14.3.3 $\overline{SS}$ Output

Available in master mode only, $\overline{SS}$ output is enabled with the SSOE bit in the SP0CR1 register if the corresponding DDRS bit is set. The $\overline{SS}$ output pin is connected to the $\overline{SS}$ input pin of the external slave device. The $\overline{SS}$ output automatically goes low for each transmission to select the external device and it goes high during each idling state to deselect external devices.

Table 14-3. $\overline{SS}$ Output Selection

DDS7	SSOE	Master Mode	Slave Mode
0	0	$\overline{SS}$ input with MODF feature	$\overline{SS}$ input
0	1	Reserved	$\overline{SS}$ input
1	0	General-purpose output	$\overline{SS}$ input
1	1	$\overline{SS}$ output	$\overline{SS}$ input

Downloaded from Arrow.com.

14.3.4 Bidirectional Mode (MOMI or SISO)

In bidirectional mode, the SPI uses only one serial data pin for external device interface. The MSTR bit decides which pin to be used. The MOSI pin becomes a serial data I/O (MOMI) pin for the master mode, and the MISO pin becomes a serial data I/O (SISO) pin for the slave mode. The direction of each serial I/O pin depends on the corresponding DDRS bit.

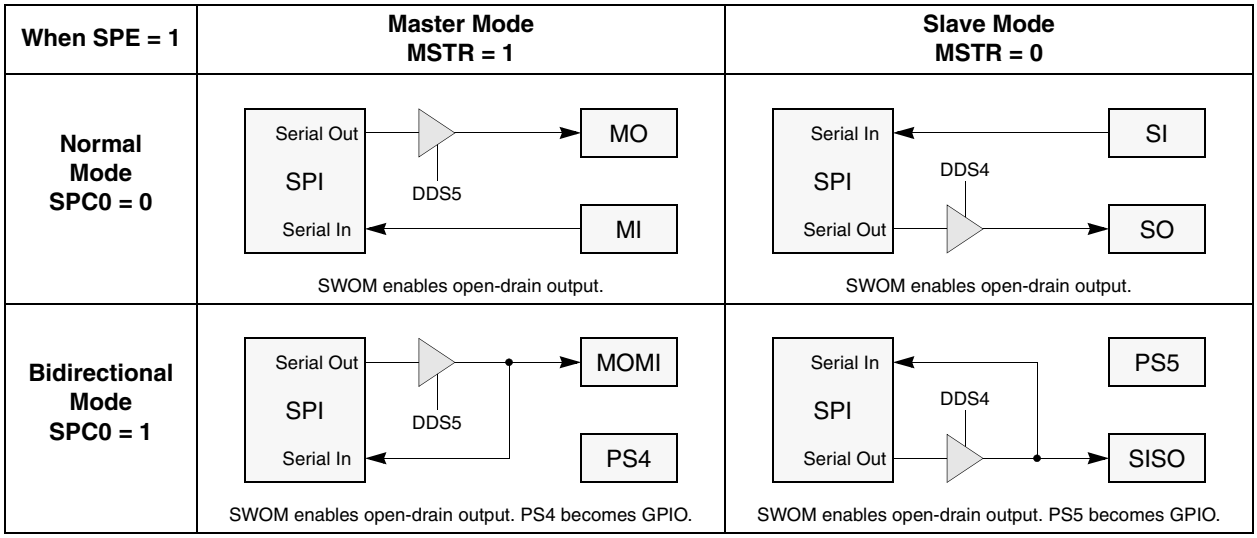


Figure 14-14. Normal Mode and Bidirectional Mode

14.3.5 SPI Register Descriptions

Control and data registers for the SPI subsystem are described in this section. The memory address indicated for each register is the default address that is in use after reset. The entire 512-byte register block can be mapped to any 2-Kbyte boundary within the standard 64-Kbyte address space. For more information, refer to Chapter 5 Operating Modes and Resource Mapping.

14.3.5.1 SPI Control Register 1

Address: $00D0

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SPIE	SPE	SWOM	MSTR	CPOL	CPHA	SSOE	LSBF
Reset:	0	0	0	0	0	1	0	0

Figure 14-15. SPI Control Register 1 (SP0CR1)

Read: Anytime

Write: Anytime

SPIE — SPI Interrupt Enable Bit

0 = SPI interrupts are inhibited.

1 = Hardware interrupt sequence is requested each time the SPIF or MODF status flag is set.

Downloaded from Arrow.com.

SPE — SPI System Enable Bit

0 = SPI internal hardware is initialized and SPI system is in a low-power disabled state.
1 = PS4–PS7 are dedicated to the SPI function.

When MODF is set, SPE always reads 0. SP0CR1 must be written as part of a mode fault recovery sequence.

SWOM — Port S Wired-OR Mode Bit

Controls not only SPI output pins but also the general-purpose output pins (PS4–PS7) which are not used by SPI.

0 = SPI and/or PS4–PS7 output buffers operate normally.
1 = SPI and/or PS4–PS7 output buffers behave as open-drain outputs.

MSTR — SPI Master/Slave Mode Select Bit

0 = Slave mode
1 = Master mode

CPOL and CPHA — SPI Clock Polarity, Clock Phase Bits

These two bits are used to specify the clock format to be used in SPI operations. When the clock polarity bit is cleared and data is not being transferred, the SCK pin of the master device is low. When CPOL is set, SCK idles high. See Figure 14-12 and Figure 14-13.

SSOE — Slave Select Output Enable Bit

The $\overline{SS}$ output feature is enabled only in master mode by asserting the SSOE and DDS7.

LSBF — SPI LSB First Enable Bit

0 = Data is transferred most-significant bit (MSB) first.
1 = Data is transferred least-significant bit (LSB) first.

Normally, data is transferred MSB first. This bit does not affect the position of the MSB and LSB in the data register. Reads and writes of the data register always have MSB in bit 7.

14.3.5.2 SPI Control Register 2

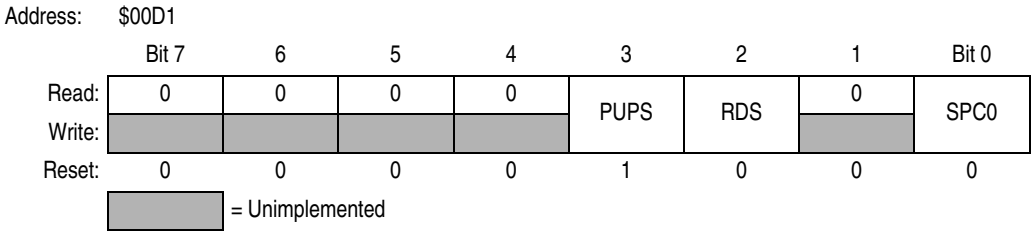


Figure 14-16. SPI Control Register 2 (SP0CR2)

Read: Anytime

Write: Anytime

PUPS — Pullup Port S Enable Bit

0 = No internal pullups on port S
1 = All port S input pins have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

RDS — Reduce Drive of Port S Bit

0 = Port S output drivers operate normally.
1 = All port S output pins have reduced drive capability for lower power and less noise.

Downloaded from Arrow.com.

SPC0 — Serial Pin Control 0 Bit

This bit decides serial pin configurations with MSTR control bit.

Table 14-4. Serial Pin Control

Pin Mode		SPC0[1]	MSTR	MISO[2]	MOSI[3]	SCK[4]	$\overline{SS}$[5]
#1	Normal	0	0	Slave out	Slave in	SCK in	$\overline{SS}$ in
#2			1	Master in	Master out	SCK out	$\overline{SS}$ I/O
#3	Bidirectional	1	0	Slave I/O	General-purposeI/O	SCK in	$\overline{SS}$ in
#4			1	General-purposeI/O	Master I/O	SCK out	$\overline{SS}$ I/O

1. The serial pin control 0 bit enables bidirectional configurations.
2. Slave output is enabled if DDS4 = 1, SS = 0, and MSTR = 0. (#1, #3)
3. Master output is enabled if DDS5 = 1 and MSTR = 1. (#2, #4)
4. SCK output is enabled if DDS6 = 1 and MSTR = 1. (#2, #4)
5. $\overline{SS}$ output is enabled if DDS7 = 1, SSOE = 1, and MSTR = 1. (#2, #4)

14.3.5.3 SPI Baud Rate Register

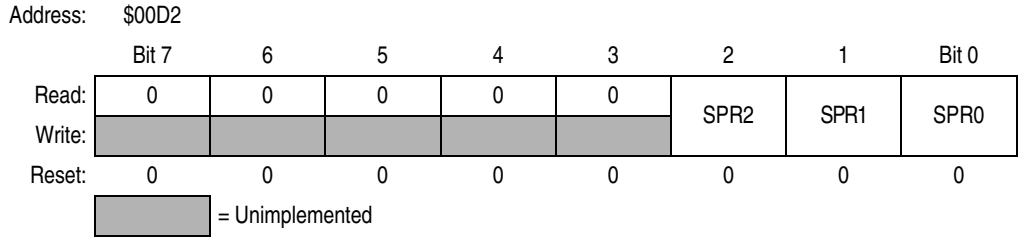


Figure 14-17. SPI Baud Rate Register (SP0BR)

Read: Anytime

Write: Anytime

At reset, E clock divided by 2 is selected.

SPR2–SPR0 — SPI Clock (SCK) Rate Select Bits

These bits are used to specify the SPI clock rate.

Table 14-5. SPI Clock Rate Selection

SPR2	SPR1	SPR0	E Clock Divisor	Frequency at E Clock = 4 MHz	Frequency at E Clock = 8 MHz
0	0	0	2	2.0 MHz	4.0 MHz
0	0	1	4	1.0 MHz	2.0 MHz
0	1	0	8	500 kHz	1.0 MHz
0	1	1	16	250 kHz	500 kHz
1	0	0	32	125 kHz	250 kHz
1	0	1	64	62.5 kHz	125 kHz
1	1	0	128	31.3 kHz	62.5 kHz
1	1	1	256	15.6 kHz	31.3 kHz

Downloaded from Arrow.com.

14.3.5.4 SPI Status Register

Address:	$00D3							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SPIF	WCOL	0	MODF	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-18. SPI Status Register (SP0SR)

Read: Anytime

Write: Has no meaning or effect

SPIF — SPI Interrupt Request Bit

SPIF is set after the eighth SCK cycle in a data transfer, and it is cleared by reading the SP0SR register (with SPIF set) followed by an access (read or write) to the SPI data register.

WCOL — Write Collision Status Flag

The MCU write is disabled to avoid writing over the data being transferred. No interrupt is generated because the error status flag can be read upon completion of the transfer that was in progress at the time of the error. This bit is cleared automatically by a read of the SP0SR (with WCOL set) followed by an access (read or write) to the SP0DR register.

0 = No write collision

1 = Indicates that a serial transfer was in progress when the MCU tried to write new data into the SP0DR data register

MODF — SPI Mode Error Interrupt Status Flag

This bit is set automatically by SPI hardware, if the MSTR control bit is set and the slave select input pin becomes 0. This condition is not permitted in normal operation. In the case where DDRS bit 7 is set, the PS7 pin is a general-purpose output pin or $\overline{SS}$ output pin rather than being dedicated as the $\overline{SS}$ input for the SPI system. In this special case, the mode fault function is inhibited and MODF remains cleared. This flag is cleared automatically by a read of the SP0SR (with MODF set) followed by a write to the SP0CR1 register.

14.3.5.5 SPI Data Register

Address:	$00D5							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset:	Unaffected by reset							

Figure 14-19. SPI Data Register (SP0DR)

Read: Anytime; normally, only after SPIF flag set

Write: Anytime; see WCOL write collision flag in 14.3.5.4 SPI Status Register

This 8-bit register is both the input and output register for SPI data. Reads of this register are double buffered but writes cause data to bewritten directly into the serial shifter. In the SPI system, the 8-bit data register in the master and the 8-bit data register in the slave are linked by the MOSI and MISO wires to

Downloaded from Arrow.com.

form a distributed 16-bit register. When a data transfer operation is performed, this 16-bit register is serially shifted eight bit positions by the SCK clock from the master so the data is exchanged effectively between the master and the slave.

NOTE

Some slave devices are simple and either accept data from the master without returning data to the master or pass data to the master without requiring data from the master.

14.4 Port S

In all modes, port S bits PS7–PS0 can be used for either general-purpose I/O or with the SCI and SPI subsystems. During reset, port S pins are configured as high-impedance inputs (DDRS is cleared).

14.4.1 Port S Data Register

Address: $00D6

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
Pin Function	$\overline{SS}$ $\overline{CS}$	SCK	MOSI MOMI	MISO SISO	—	—	TXD0	RXD0
Reset:	After reset all bits configured as general-purpose inputs							

Figure 14-20. Port S Data Register (PORTS)

Read: Anytime; inputs return pin level; outputs return pin driver input level

Write: Data stored in internal latch; drives pins only if configured for output; does not change pin state when pin configured for SPI or SCI output

Port S shares function with the on-chip serial systems, SPI0 and SCI0.

14.4.2 Port S Data Direction Register

Address: $00D7

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDS7	DDS6	DDS5	DDS4	DDS3	DDS2	DDS1	DDS0
Reset:	0	0	0	0	0	0	0	0

Figure 14-21. Port S Data Direction Register (DDRS)

Read: Anytime

Write: Anytime

After reset, all general-purpose I/O are configured for input only.

0 = Configure the corresponding I/O pin for input only.

1 = Configure the corresponding I/O pin for output.

Downloaded from Arrow.com.

DDS0 — Data Direction for Port S Bit 0

If the SCI receiver is configured for 2-wire SCI operation, corresponding port S pins are input regardless of the state of these bits.

DDS1 — Data Direction for Port S Bit 1

If the SCI transmitter is configured for 2-wire SCI operation, corresponding port S pins are output regardless of the state of these bits.

DDS2 and DDS3 — Data Direction for Port S Bit 2 and Bit 3

These bits are for general-purpose only.

DDS6–DDS4 — Data Direction for Port S Bits 6–4

If the SPI is enabled and expects the corresponding port S pin to be an input, it will be an input regardless of the state of the DDRS bit. If the SPI is enabled and expects the bit to be an output, it will be an output only if the DDRS bit is set.

DDS7 — Data Direction for Port S Bit 7

In SPI slave mode, DDS7 has no meaning or effect; the PS7 pin is dedicated as the $\overline{SS}$ input. In SPI master mode, DDS7 determines whether PS7 is an error detect input to the SPI or a general-purpose or slave select output line.

14.4.3 Pullup and Reduced Drive Register for Port S

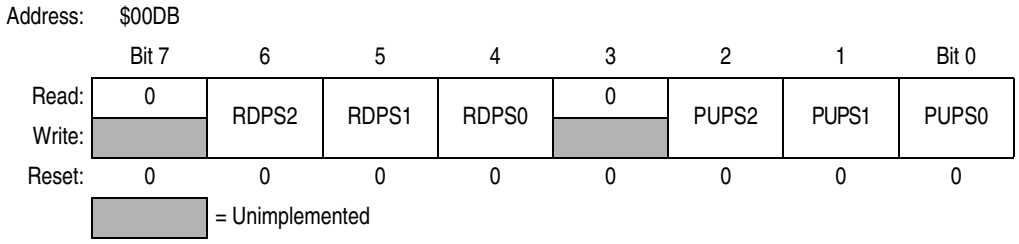


Figure 14-22. Pullup and Reduced Drive Register for Port S (PURDS)

Read: Anytime

Write: Anytime

RDPS2 — Reduce Drive of PS7–PS4

0 = Port S output drivers for bits 7–4 operate normally.
1 = Port S output pins for bits 7–4 have reduced drive capability for lower power and less noise.

RDPS1 — Reduce Drive of PS3 and PS2

0 = Port S output drivers for bits 3 and 2 operate normally.
1 = Port S output pins for bits 3 and 2 have reduced drive capability for lower power and less noise.

RDPS0 — Reduce Drive of PS1 and PS0

0 = Port S output drivers for bits 1 and 0 operate normally.
1 = Port S output pins for bits 1 and 0 have reduced drive capability for lower power and less noise.

PUPS2 — Pullup Port S Enable PS7–PS4

0 = No internal pullups on port S bits 7–4.
1 = Port S input pins for bits 7–4 have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

Downloaded from Arrow.com.

PUPS1 — Pullup Port S Enable PS3 and PS2 Bit

0 = No internal pullups on port S bits 3 and 2
1 = Port S input pins for bits 3 and 2 have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

PUPS0 — Pullup Port S Enable PS1 and PS0 Bit

0 = No internal pullups on port S bits 1 and 0
1 = Port S input pins for bits 1 and 0 have an active pullup device. If a pin is programmed as output, the pullup device becomes inactive.

14.5 Serial Character Transmission using the SCI

Code is intended to use SCI1 to serially transmit characters using polling to the LCD display on the UDLP1 board: when the transmission data register is empty a flag will get set, which is telling us that SC1DR is ready so we can write another byte. The transmission is performed at a baud rate of 9600. Since the SCI1 is only being used for transmit data, the data register will not be used bidirectionally for received data.

14.5.1 Equipment

For this exercise, use the M68HC912B32EVB emulation board.

14.5.2 Code Listing

NOTE

A comment line is deliminted by a semi-colon. If there is no code before comment, an ";" must be placed in the first column to avoid assembly errors.

```
INCLUDE 'EQUATES.ASM'       ; Equates for registers

; User Variables

; Bit Equates

; ---------------------------------------------------------------------
;           MAIN PROGRAM
; ---------------------------------------------------------------------
        ORG     $7000             ; 16K On-Board RAM, User code data area,
;                                 ; start main program at $4000
MAIN:
        BSR     INIT              ; Subroutine to Initialize SCI0 registers
        BSR     TRANS             ; Subroutine to start transmission
DONE:   BRA     DONE              ; Always branch to DONE, convenient for breakpoint

; ---------------------------------------------------------------------
;         SUBROUTINE INIT:
; ---------------------------------------------------------------------

INIT:   TPA                       ; Transfer CCR to A accumulator
        ORAA    #$10              ; ORed A with #$10 to Set I bit
        TAP                       ; Transfer A to CCR

        MOVB    #$34,SC1BDL       ; Set BAUD =9600, in SCI1 Baud Rate Reg.

        MOVB    #$00,SC1CR1       ; Initialize for 8-bit Data format,
;                                 ; Loop Mode  and parity disabled,(SC1CR1)
```


Downloaded from Arrow.com.

```
        MOVB    #$08,SC1CR2       ; Set for No Ints, and Transmitter enabled(SC1CR2)

        LDAA    SC1SR1            ; 1st step to clear TDRE flag: Read SC1SR1
        STD     SC1DRH            ; 2nd step to clear TDRE flag: Write SC1DR register

        LDX     #DATA             ; Use X as a pointer to DATA.

        RTS                       ; Return from subroutine
; ---------------------------------------------------------------------
;       TRANSMIT SUBROUTINE
; ---------------------------------------------------------------------
TRANS:  BRCLR   SC1SR1,#$80, TRANS ; Wait for TDRE flag
        MOVB    1,X+,SC1DRL       ; Transmit character, increment X pointer
        CPX     #EOT              ; Detect if last character has been transmitted
        BNE     TRANS             ; If last char. not equal to "eot", Branch to TRANS
        RTS                       ; else Transmission complete, Return from Subroutine

; ---------------------------------------------------------------------
;  TABLE : DATA TO BE TRANSMITTED
; ---------------------------------------------------------------------
DATA:   DC.B    'Freescale HC12 Banner - June, 1999'
        DC.B    $0D,$0A           ; Return (cr) ,Line Feed (LF)
        DC.B    'Scottsdale, Arizona'
        DC.B    $0D,$0A           ; Return (cr) ,Line Feed (LF)
EOT:    DC.B    $04               ; Byte used to test end of data = EOT

        END                       ; End of program
```

14.6 Synchronous Character Transmission using the SPI

This program is intended to communicate with the HC11 on the UDLP1 board. It utilizes the SPI to transmit synchronously characters in a string to be displayed on the LCD display. The program must configure the SPI as a master, and non-interrupt driven. The slave peripheral is chip-selected with the $\overline{SS}$ line at low voltage level. Between 8 bit transfers the $\overline{SS}$ line is held high. Also the clock idles low and takes data on the rising clock edges. The serial clock is set not to exceed 100 kHz baud rate.

14.6.1 Equipment

For this exercise, use the M68HC912B32EVB emulation board.

14.6.2 Code Listing

NOTE

A comment line is deliminted by a semi-colon. If there is no code before comment, an ";" must be placed in the first column to avoid assembly errors.

Downloaded from Arrow.com.

```
INCLUDE 'EQUATES.ASM'      ;Equates for all registers

; User Variables

; Bit Equates

; ---------------------------------------------------------------------
;           MAIN PROGRAM
; ---------------------------------------------------------------------
        ORG     $7000             ; 16K On-Board RAM, User code data area,
;                                 ; start main program at $7000
MAIN:
        BSR     INIT              ; Subroutine to initialize SPI registers
        BSR     TRANSMIT          ; Subroutine to start transmission
FINISH:
        BRA     FINIS             ; Finished transmitting all DATA
; ---------------------------------------------------------------------
;*         SUBROUTINE INIT:
; ---------------------------------------------------------------------
INIT:
        BSET    PORTS,#$80        ; SET SS Line High to prevent glitch

        MOVB    #$E0,DDRS         ; Configure PORT S input/ouput levels
;                                 ; MOSI, SCK, SS* = ouput, MISO=Input

        MOVB    #$07,SP0BR        ; Select serial clock baud rate < 100 KHz

        MOVB    #$12,SP0CR1       ; Configure SPI(SP0CR1): No SPI interrupts,
;                                 ; MSTR=1, CPOL=0, CPHA=0

        MOVB    #$08,SP0CR2       ; Config. PORTS output drivers to operate normally,
;                                 ; and with active pull-up devices.

        LDX     #DATA             ; Use X register as pointer to first character

        LDAA    SP0SR             ; 1st step to clear SPIF Flag, Read SP0SR
        LDAA    SP0DR             ; 2nd step to clear SPIF Flag, Access SP0DR

        BSET    SP0CR1,#$40       ; Enable the SPI (SPE=1)

        RTS                       ; Return from subroutine

; ---------------------------------------------------------------------
;*        TRANSMIT SUBROUTINE
; ---------------------------------------------------------------------
TRANSMIT:
        LDAA    1,X+              ; Load Acc. with "NEW" character to send, Inc X
        BEQ     DONE              ; Detect if last character(0) has been transmitted
;                                 ; If last char. branch to DONE, else
        BCLR    PORTS,#$80        ; Assert SS Line to start X-misssion.
        STAA    SP0DR             ; Load Data into Data Reg.,X-mit.
;                                 ; it is also the 2nd step to clear SPIF flag.
FLAG:   BRCLR   SP0SR,#$80,FLAG   ;Wait for flag.
        BSET    PORTS,#$80        ; Disassert SS Line.
        BRA     TRANSMIT          ; Continue sending characters, Branch to TRANSMIT.

DONE:   RTS                       ; Return from subroutine

; ---------------------------------------------------------------------
;  TABLE OF DATA TO BE TRANSMITTED
; ---------------------------------------------------------------------
DATA:   DC.B    'Freescale'
        DC.B    $0D,$0A           ; Return (cr) ,Line Feed (LF)
EOT:    DC.B    $00               ; Byte used to test end of data = EOT

        END                       ; End of program
```


Downloaded from Arrow.com.

Chapter 15
Byte Data Link Communications (BDLC)

15.1 Introduction

The byte data link communications module (BDLC) provides access to an external serial communication multiplex bus, operating according to the SAE J1850 protocol.

15.2 Features

Features of the BDLC module include:

- *SAE J1850 Class B Data Communications Network Interface* compatible and ISO compatible for low-speed (≤125 Kbps) serial data communications in automotive applications
- 10.4 Kbps variable pulse width (VPW) bit format
- Digital noise filter
- Collision detection
- Hardware cyclical redundancy check generation and checking
- Two power-saving modes with automatic wakeup on network activity
- Polling or CPU interrupts
- Block mode receive and transmit
- 4X receive mode, 41.6 Kbps
- Digital loopback mode
- Analog loopback mode
- In-frame response (IFR) types 0, 1, 2, and 3

NOTE

Familiarity with the SAE Standard J1850 Class B Data Communication Network Interface specification is recommended before proceeding. First-time users of the BDLC should obtain the Byte Data Link Controller Reference Manual, Freescale document order number BDLCRM/AD.

15.3 Functional Description

Figure 15-1 shows the organization of the BDLC module. The CPU interface contains the software addressable registers and provides the link between the CPU and the buffers. The buffers provide storage for data received and data to be transmitted onto the J1850 bus. The protocol handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX interface provides the link between the BDLC digital section and the analog physical interface. The wave shaping, driving, and digitizing of data is performed by the physical interface.

Use of the BDLC module in message networking fully implements the *SAE Standard J1850 Class B Data Communication Network Interface* specification.

Downloaded from Arrow.com.

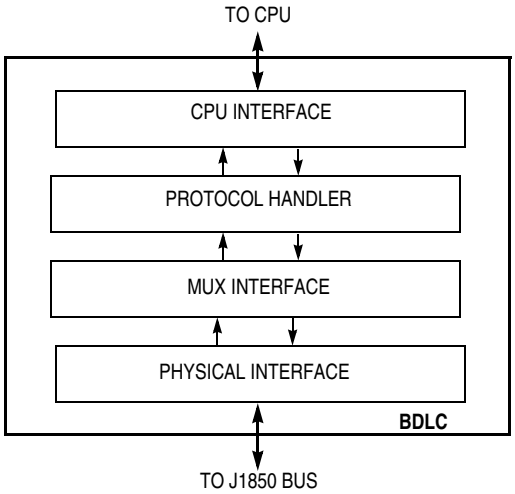


Figure 15-1. BDLC Block Diagram

15.4 BDLC Operating Modes

The BDLC has five main modes of operation which interact with the power supplies, pins, and rest of the MCU as shown in Figure 15-2.

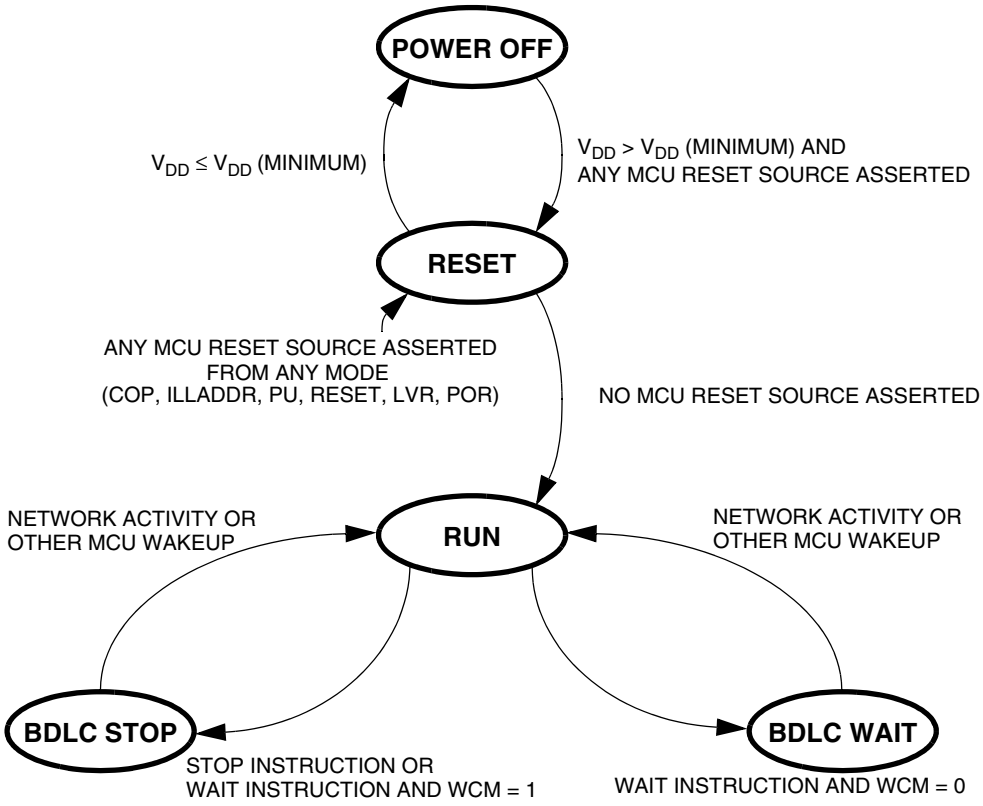


Figure 15-2. BDLC Operating Modes State Diagram

Downloaded from Arrow.com.

15.4.1 Power Off Mode

For guaranteed BDLC operation, this mode is entered from reset mode when the BDLC supply voltage, V_{DD}, drops below its minimum specified value. The BDLC is placed in reset mode by low-voltage reset (LVR) before being powered down. In power off mode, the pin input and output specifications are not guaranteed.

15.4.2 Reset Mode

This mode is entered from power off mode when the BDLC supply voltage, V_{DD}, rises above its minimum specified value (V_{DD} –10%) and an MCU reset source is asserted. The internal MCU reset must be asserted while powering up the BDLC or an unknown state is entered and correct operation cannot be guaranteed. Reset mode is also entered from any other mode when any reset source is asserted.

In reset mode, the internal BDLC voltage references are operative, V_{DD} is supplied to the internal circuits which are held in their reset state, and the internal BDLC system clock is running. Registers assume their reset condition. Because outputs are held in their programmed reset state, inputs and network activity are ignored.

15.4.3 Run Mode

This mode is entered from reset mode after all MCU reset sources are no longer asserted. Run mode is entered from the BDLC wait mode when activity is sensed on the J1850 bus.

Run mode is entered from the BDLC stop mode when network activity is sensed, although messages are not received properly until the clocks have stabilized and the CPU is also in run mode.

In this mode, normal network operation takes place. Ensure that all BDLC transmissions have ceased before exiting this mode.

15.5 Power-Conserving Modes

The BDLC has three power-conserving modes:

1. BDLC wait and CPU wait mode
2. BDLC stop and CPU wait mode
3. BDLC stop and CPU stop mode

Depending upon the logic level of the WCM bit in BDLC control register 1 (BCR1), the BDLC enters a power-conserving mode when the CPU executes the STOP or WAIT instruction.

When a power-conserving mode is entered, any activity on the J1850 network causes the BDLC to exit low-power mode. When exiting from BDLC stop mode, the BDLC generates an unmaskable interrupt of the CPU. This wakeup interrupt state is reflected in the BDLC state vector register (BSVR) and encoded as the highest priority interrupt.

Wait mode or stop mode does not reset the BDLC registers upon BDLC wakeup.

Downloaded from Arrow.com.

To disengage a BDLC node from receiving J1850 traffic:

- Verify all BSVR flags are clear.
- Do not load the BDR.
- Set the ALOOP bit (after placing the analog transceiver into loopback mode) or DLOOP bit in BCR2.

The BDLC can then be put into wait mode or stop mode and does not wake up with J1850 traffic.

Depending upon which low-power mode instruction the CPU executes and which mode the BDLC enters, the message which wakes up the BDLC (and the CPU) may not be received correctly. Three possibilities are described here. These descriptions apply regardless of whether the BDLC is in normal or 4X mode when the STOP or WAIT instruction is executed.

15.5.1 BDLC Wait and CPU Wait Mode

This power-saving mode is entered automatically from run mode when the WCM bit in BCR1 register is cleared followed by a CPU WAIT instruction. In BDLC wait mode, the BDLC cannot drive data. A subsequent J1850 network rising edge wakes up the BDLC.

In this mode, the BDLC internal clocks continue to run as do the MCU clocks. The first passive-to-active transition on the J1850 network generates a CPU interrupt request by the BDLC which wakes up the BDLC and CPU. The BDLC correctly receives the entire message which generated the CPU interrupt request.

NOTE

Ensure that all transmissions are complete or aborted prior to putting the BDLC into wait mode (WCM = 0 in BCR1).

15.5.2 BDLC Stop and CPU Wait Mode

This power-conserving mode is entered automatically from run mode when the WCM bit in the BCR1 register is set followed by a CPU WAIT instruction. This is the lowest-power mode that the BDLC can enter.

In this mode:

- The BDLC internal clocks are stopped.
- The CPU internal clocks continue to run.
- The BDLC awaits J1850 network activity.

The first passive-to-active transition on the J1850 network generates a non-maskable ($20) CPU interrupt request by the BDLC, allowing the CPU to restart the BDLC internal clocks.

To correctly receive future J1850 wakeup traffic, users must read an EOF (end of frame) in the BSVR prior to placing the BDLC into stop mode (WCM = 1). Then, the new message which wakes up the BDLC from the BDLC stop mode and the CPU from the CPU wait mode, is received correctly.

NOTE

Ensure that all transmissions are complete or aborted prior to putting the BDLC into stop mode (WCM = 1 in BCR1).

Downloaded from Arrow.com.

15.5.3 BDLC Stop and CPU Stop Mode

This power-conserving mode is entered automatically from run mode when the WCM bit in the BCR1 register is set followed by a CPU STOP instruction. This is the lowest-power mode that the BDLC can enter.

In this mode:

- The BDLC internal clocks are stopped.
- The CPU internal clocks are stopped.
- The BDLC awaits J1850 network activity.

The first passive-to-active transition on the J1850 network generates a non-maskable ($20) CPU interrupt request by the BDLC, allowing the CPU clocks to restart and the BDLC internal clocks to restart. Therefore, the new message which wakes up the BDLC from the BDLC stop mode and the CPU from the CPU wait mode are not received correctly. This is due primarily to the time required for the MCU's oscillator to stabilize before the clocks can be applied internally to the other MCU modules, including the BDLC.

NOTE

Ensure that all transmissions are complete or aborted prior to putting the BDLC into stop mode (WCM = 1 in BCR1).

15.6 Loopback Modes

Two loopback modes are used to determine the source of bus faults.

15.6.0.1 Digital Loopback Mode

When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node's internal circuits or elsewhere in the network, including the node's analog physical interface. In this mode, the transmit digital output pin (BDTxD) and the receive digital input pin (BDRxD) of the digital interface are disconnected from the analog physical interface and tied together to allow the digital portion of the BDLC to transmit and receive its own messages without driving the J1850 bus.

15.6.0.2 Analog Loopback Mode

Analog loopback mode is used to determine if a bus fault has been caused by a failure in the node's off-chip analog transceiver or elsewhere in the network. The BDLC analog loopback mode does not modify the digital transmit or receive functions of the BDLC. It does, however, ensure that once analog loopback mode is exited, the BDLC waits for an idle bus condition before participation in network communication resumes. If the off-chip analog transceiver has a loopback mode, it usually causes the input to the output drive stage to be looped back into the receiver, allowing the node to receive messages it has transmitted without driving the J1850 bus. In this mode, the output to the J1850 bus typically is high impedance. This allows the communication path through the analog transceiver to be tested without interfering with network activity. Using the BDLC analog loopback mode in conjunction with the analog transceiver's loopback mode ensures that, once the off-chip analog transceiver has exited loopback mode, the BDLC does not begin communicating before a known condition exists on the J1850 bus.

Downloaded from Arrow.com.

15.7 BDLC MUX Interface

The MUX (multiplex) interface is responsible for bit encoding/decoding and digital noise filtering between the protocol handler and the physical interface.

15.7.1 Rx Digital Filter

The receiver section of the BDLC includes a digital low-pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 15-3.

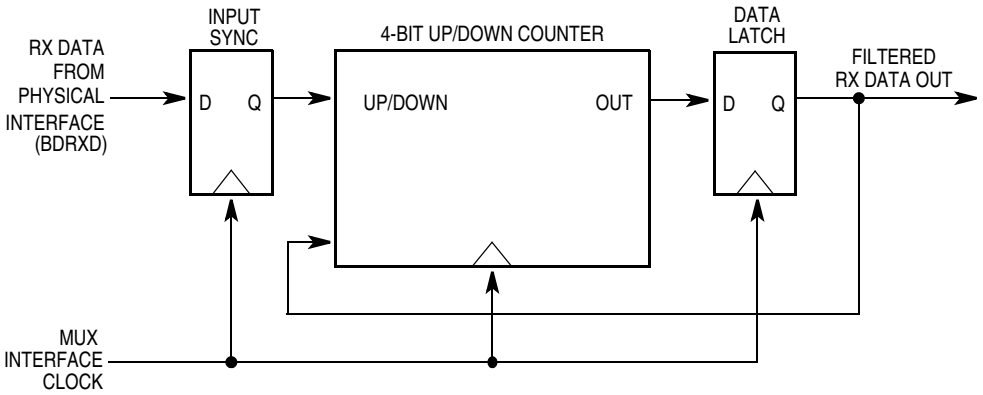


Figure 15-3. BDLC Rx Digital Filter Block Diagram

15.7.1.1 Operation

The clock for the digital filter is provided by the MUX interface clock (see f_{BDLC} parameter in Table 15-2). At each positive edge of the clock signal, the current state of the receiver physical interface (BDRxD) signal is sampled. The BDRxD signal state is used to determine whether the counter should increment or decrement at the next negative edge of the clock signal.

The counter increments if the input data sample is high but decrements if the input sample is low. Therefore, the counter progresses either up toward 15 if, on average, the BDRxD signal remains high or progresses down toward 0 if, on average, the BDRxD signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can be decremented only from this state.

Alternatively, should the counter eventually reach the value 0, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can be incremented only from this state.

The data latch retains its value until the counter next reaches the opposite end point, signifying a definite transition of the signal.

Downloaded from Arrow.com.

15.7.1.2 Performance

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the BDRxD signal transitions, there is a delay before that transition appears at the filtered Rx data output signal. This delay is between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This filter delay must be taken into account when performing message arbitration.

For example, if the frequency of the MUX interface clock (f_{BDLC}) is 1.0486 MHz, then the period (t_{BDLC}) is 954 ns and the maximum filter delay in the absence of noise is 15.259 µs.

The effect of random noise on the BDRxD signal depends on the characteristics of the noise itself. Narrow noise pulses on the BDRxD signal is ignored completely if they are shorter than the filter delay. This provides a degree of low pass filtering.

If noise occurs during a symbol transition, the detection of that transition can be delayed by an amount equal to the length of the noise burst. This is a reflection of the uncertainty of where the transition is actually occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length, are detected by the next stage of the BDLC's receiver as an invalid symbol.

Noise pulses that are longer than the shortest allowable symbol length are detected normally as an invalid symbol or as invalid data when the frame's CRC is checked.

15.7.2 J1850 Frame Format

All messages transmitted on the J1850 bus are structured using the format shown in Figure 15-4.

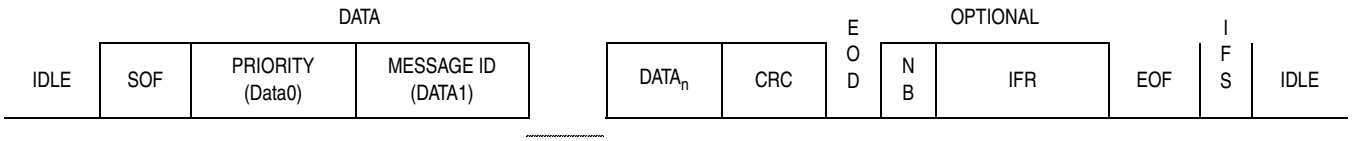


Figure 15-4. J1850 Bus Message Format (VPW)

J1850 states that each message has a maximum length of 101 PWM bit times or 12 VPW bytes, excluding SOF, EOD, NB, and EOF, with each byte transmitted most significant bit (MSB) first.

All VPW symbol lengths in the following descriptions are typical values at a 10.4-Kbps bit rate.

15.7.2.1 SOF — Start-of-Frame Symbol

All messages transmitted onto the J1850 bus must begin with a long-active 200 µs period SOF symbol. This indicates the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

Downloaded from Arrow.com.

15.7.2.2 Data — In-Message Data Bytes

The data bytes contained in the message include the message priority/type, message ID byte (typically, the physical address of the responder), and any actual data being transmitted to the receiving node. The message format used by the BDLC is similar to the 3-byte consolidated header message format outlined by the SAE J1850 document. See *SAE J1850 – Class B Data Communications Network Interface* specification for more information about 1- and 3-byte headers.

Messages transmitted by the BDLC onto the J1850 bus must contain at least one data byte, and, therefore, can be as short as one data byte and one CRC byte. Each data byte in the message is eight bits in length and is transmitted MSB (most significant bit) to LSB (least significant bit).

15.7.2.3 CRC — Cyclical Redundancy Check Byte

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus. It also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial $X^8 + X^4 + X^3 + X^2 + 1$. The remainder polynomial initially is set to all 1s. Each byte in the message after the start-of-frame (SOF) symbol is processed serially through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes, in MSB-to-LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and end-of-data symbols (EOD) but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial equals $X^7 + X^6 + X^2 = \$C4$, regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC recognizes this as a CRC error and sets the CRC error flag in the BSVR.

15.7.2.4 EOD — End-of-Data Symbol

The EOD symbol is a long 200-µs passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

15.7.2.5 IFR — In-Frame Response Bytes

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the *SAE J1850 – Class B Data Communications Network Interface* specification.

15.7.2.6 EOF — End-of-Frame Symbol

This symbol is a long 280-µs passive period on the J1850 bus and is longer than an end-of-data (EOD) symbol, which signifies the end of a message. Since an EOF symbol is longer than a 200-µs EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

15.7.2.7 IFS — Interframe Separation Symbol

The IFS symbol is a 20-µs passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node after the completion of the end-of-frame (EOF) period and, therefore, is seen as a 300-µs passive period.

Downloaded from Arrow.com.

When the last byte of a message has been transmitted onto the J1850 bus and the EOF symbol time has expired, all nodes then must wait for the IFS symbol time to expire before transmitting a start-of-frame (SOF) symbol, marking the beginning of another message.

However, if the BDLC is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it synchronizes internally to that edge.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. To allow for individual clock tolerances, receivers must synchronize to any SOF occurring during an IFS period.

15.7.2.8 BREAK — Break

The BDLC cannot transmit a BREAK symbol.

If the BDLC is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred and halts transmission.

If the BDLC detects a BREAK symbol while receiving a message, it treats the BREAK as a reception error and sets the invalid symbol flag in the BSVR, also ignoring the frame it was receiving. If while receiving a message in 4X mode, the BDLC detects a BREAK symbol, it treats the BREAK as a reception error, sets the invalid symbol flag, and exits 4X mode (for example, the RX4XE bit in BCR2 is cleared automatically). If bus control is required after the BREAK symbol is received and the IFS time has elapsed, the programmer must resend the transmission byte using highest priority.

15.7.2.9 IDLE — Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period (for example, > 300 μs). Any node sensing an idle bus condition can begin transmission immediately.

15.7.3 J1850 VPW Symbols

Huntsinger's variable pulse-width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions and by the level of the bus between transitions (for instance, active or passive). Active and passive bits are used alternately. This encoding technique is used to reduce the number of bus transitions for a given bit rate.

Each logic 1 or logic 0 contains a single transition and can be at either the active or passive level and one of two lengths, either 64 μs or 128 μs (t_{NOM} at 10.4 Kbps baud rate), depending upon the encoding of the previous bit. The start-of-frame (SOF), end-of-data (EOD), end-of-frame (EOF), and inter-frame separation (IFS) symbols are always encoded at an assigned level and length. See Figure 15-5.

Each message begins with an SOF symbol, an active symbol, and, therefore, each data byte (including the CRC byte) begins with a passive bit, regardless of whether it is a logic 1 or a logic 0.

All VPW bit lengths stated in the descriptions here are typical values at a 10.4-Kbps bit rate. EOF, EOD, IFS, and IDLE, however, are not driven J1850 bus states. They are passive bus periods observed by each node's CPU.

Downloaded from Arrow.com.

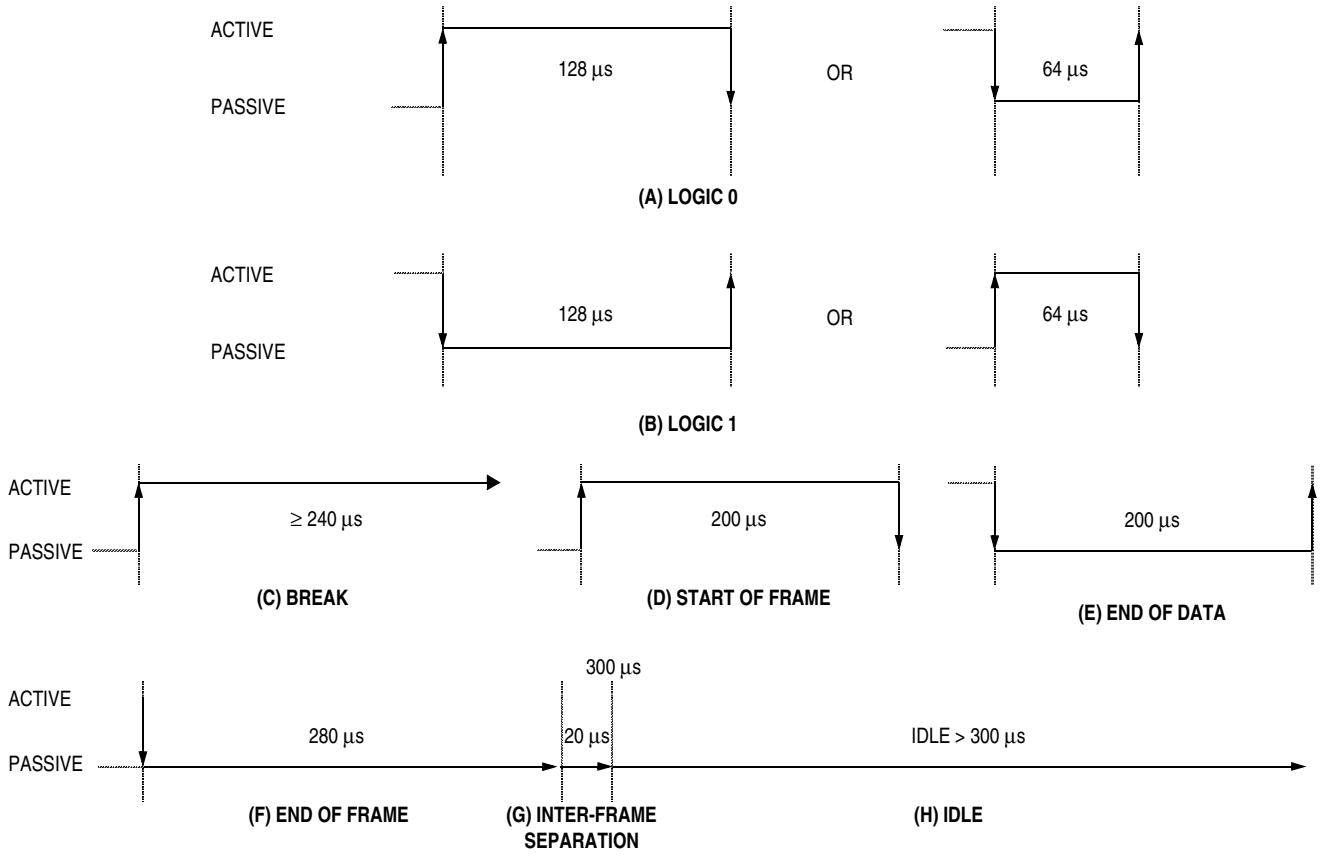


Figure 15-5. J1850 VPW Symbols with Nominal Symbol Times

15.7.3.1 Logic 0

A logic 0 is defined as:

- An active-to-passive transition followed by a passive period 64 µs in length, or
- A passive-to-active transition followed by an active period 128 µs in length

See Figure 15-5(A).

15.7.3.2 Logic 1

A logic 1 is defined as:

- An active-to-passive transition followed by a passive period 128 µs in length, or
- A passive-to-active transition followed by an active period 64 µs in length

See Figure 15-5(B).

15.7.3.3 Normalization Bit (NB)

The NB symbol has the same property as a logic 1 or a logic 0. It is used only in IFR message responses.

15.7.3.4 Break Signal (BREAK)

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 µs (see Figure 15-5(C)).

Downloaded from Arrow.com.

15.7.3.5 Start-of-Frame Symbol (SOF)

The SOF symbol is defined as passive-to-active transition followed by an active period 200 µs in length (see Figure 15-5(D)). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

15.7.3.6 End-of-Data Symbol (EOD)

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 µs in length (see Figure 15-5(E)).

15.7.3.7 End-of-Frame Symbol (EOF)

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280 µs in length (see Figure 15-5(F)). If no IFR byte is transmitted after an EOD symbol is transmitted, another 80 µs the EOD becomes an EOF, indicating completion of the message.

15.7.3.8 Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 300 µs in length. The 20-µs IFS symbol contains no transition, since when it is used it always appends to a 280-µs EOF symbol (see Figure 15-5(G)).

15.7.3.9 Idle

An idle is defined as a passive period greater than 300 µs in length.

15.7.4 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for receiving data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases, the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC for determining what symbol is being received is equal to a single period of the MUX interface clock (t_{BDLC}), an apparent separation in these maximum time/minimum time concurrences equals one cycle of t_{BDLC}.

This one clock resolution allows the BDLC to differentiate properly between the different bits and symbols. This is done without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus, which has varying oscillator frequencies.

In Huntsinger's variable pulse width (VPW) modulation bit encoding, the tolerances for both the passive and active data bits received and the symbols received are defined with no gaps between definitions. For example, the maximum length of a passive logic 0 is equal to the minimum length of a passive logic 1, and the maximum length of an active logic 0 is equal to the minimum length of a valid SOF symbol.

See Figure 15-6, Figure 15-7, and Figure 15-8.

Downloaded from Arrow.com.

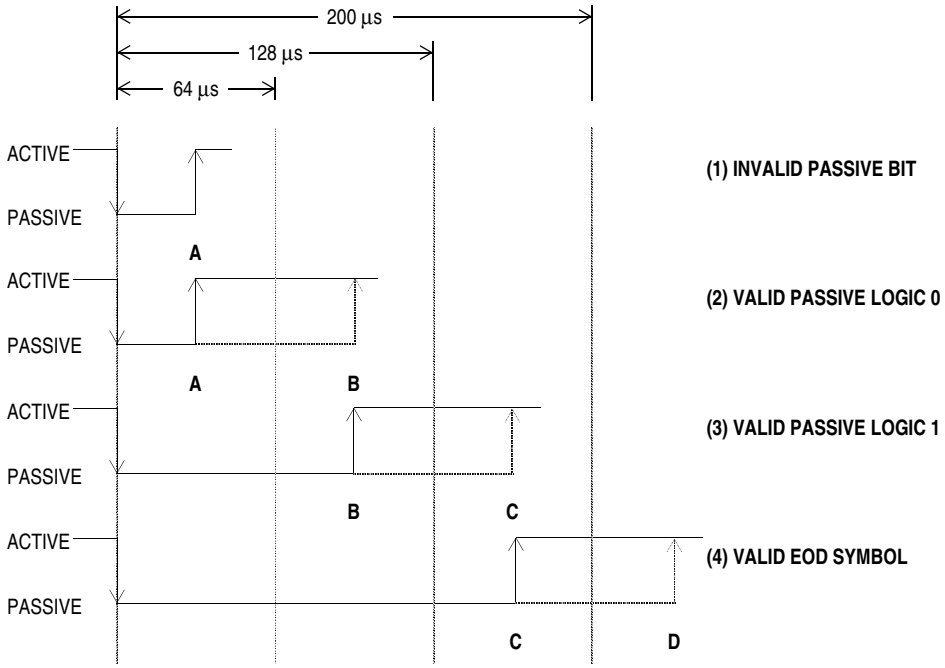


Figure 15-6. J1850 VPW Received Passive Symbol Times

15.7.4.1 Invalid Passive Bit

See Figure 15-6(1). If the passive-to-active received transition beginning the next data bit or symbol occurs between the active-to-passive transition beginning the current data bit (or symbol) and **A**, the current bit would be invalid.

15.7.4.2 Valid Passive Logic 0

See Figure 15-6(2). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **A** and **B**, the current bit would be considered a logic 0.

15.7.4.3 Valid Passive Logic 1

See Figure 15-6(3). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **B** and **C**, the current bit would be considered a logic 1.

15.7.4.4 Valid EOD Symbol

See Figure 15-6(4). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **C** and **D**, the current symbol would be considered a valid end-of-data symbol (EOD).

Downloaded from Arrow.com.

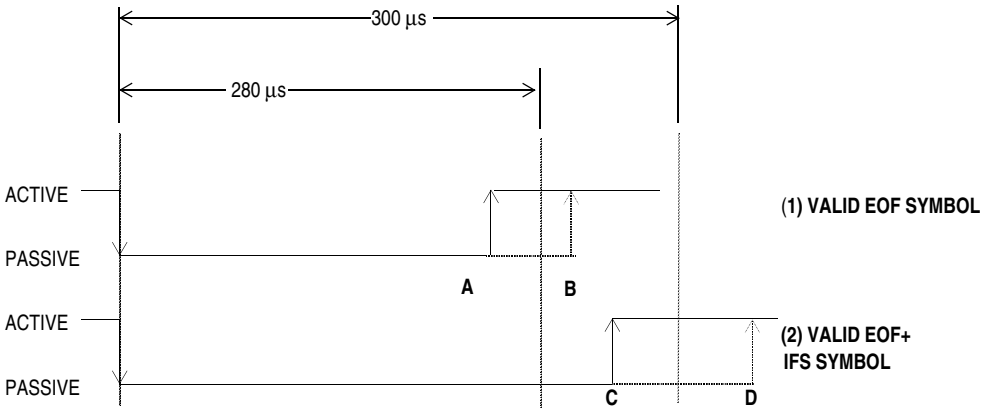


Figure 15-7. VPW Received Passive EOF and IFS Symbol Times

15.7.4.5 Valid EOF and IFS Symbols

In Figure 15-7(1), if the passive-to-active received transition beginning the SOF symbol of the next message occurs between **A** and **B**, the current symbol is considered a valid end-of-frame (EOF) symbol.

See Figure 15-7(2). If the passive-to-active received transition beginning the SOF symbol of the next message occurs between **C** and **D**, the current symbol is considered a valid EOF symbol followed by a valid inter-frame separation symbol (IFS). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others and immediately begin transmitting. Therefore, any time a node waiting to transmit detects a passive-to-active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

15.7.4.6 Idle Bus

In Figure 15-7(2), if the passive-to-active received transition beginning the start-of-frame (SOF) symbol of the next message does not occur before **D,** the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

Downloaded from Arrow.com.

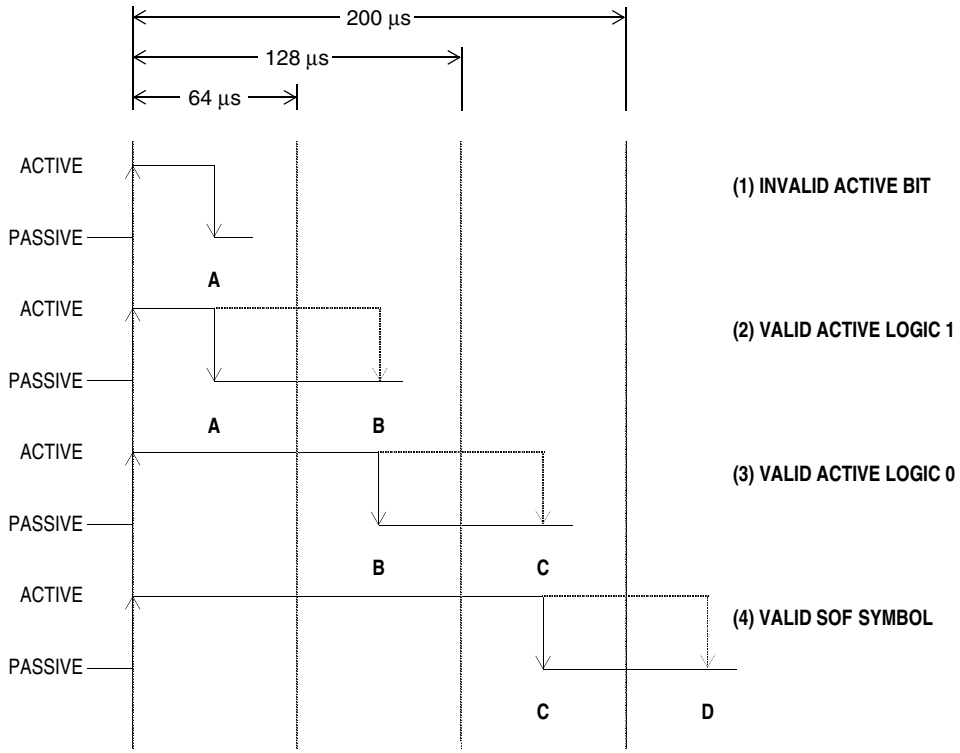


Figure 15-8. J1850 VPW Received Active Symbol Times

15.7.4.7 Invalid Active Bit

In Figure 15-8(1), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between the passive-to-active transition beginning the current data bit (or symbol) and **A**, the current bit would be invalid.

15.7.4.8 Valid Active Logic 1

In Figure 15-8(2), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **A** and **B**, the current bit would be considered a logic 1.

15.7.4.9 Valid Active Logic 0

In Figure 15-8(3), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **B** and **C**, the current bit would be considered a logic 0.

15.7.4.10 Valid SOF Symbol

In Figure 15-8(4), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **C** and **D**, the current symbol would be considered a valid SOF symbol.

Downloaded from Arrow.com.

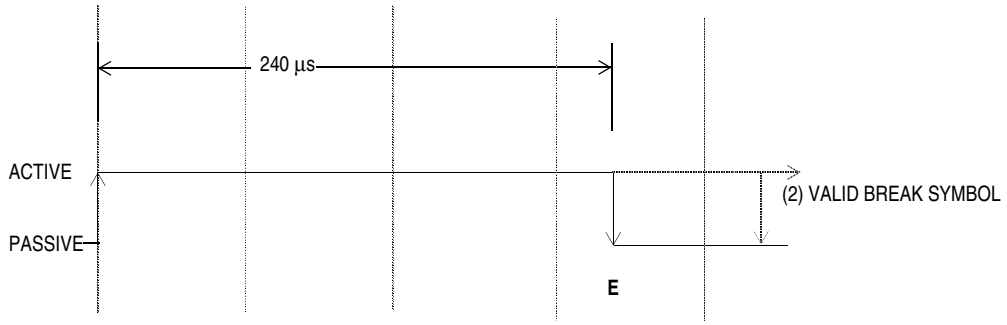


Figure 15-9. J1850 VPW Received BREAK Symbol Times

15.7.4.11 Valid BREAK Symbol

In Figure 15-9, if the next active-to-passive received transition does not occur until after **E**, the current symbol is considered a valid BREAK symbol. A BREAK symbol should be followed by a start-of-frame (SOF) symbol beginning the next message to be transmitted onto the J1850 bus. See 15.7.2 J1850 Frame Format for BDLC response to BREAK symbols.

15.7.5 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC wants to transmit onto the J1850 bus, but detects that another message is in progress, it waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration occurs beginning with the first bit after the SOF symbol and continues with each bit thereafter. If a write to the BDR (for instance, to initiate transmission) occurred on or before $104 \bullet t_{BDLC}$ from the received rising edge, then the BDLC transmits and arbitrates for the bus. If a CPU write to the BDR occurred after $104 \bullet t_{BDLC}$ from the detection of the rising edge, then the BDLC does not transmit, but waits for the next IFS period to expire before attempting to transmit the byte.

The variable pulse-width modulation (VPW) symbols and J1850 bus electrical characteristics are chosen carefully so that a logic 0 (active or passive type) always dominates over a logic 1 (active or passive type) simultaneously transmitted. Hence, logic 0s are said to be dominant and logic 1s are said to be recessive.

When a node detects a dominant bit on BDRxD when it transmitted a recessive bit, it loses arbitration and immediately stops transmitting. This is known as bitwise arbitration (see Figure 15-10).

Since a logic 0 dominates a logic 1, the message with the lowest value has the highest priority and always wins arbitration. For instance, a message with priority 000 wins arbitration over a message with priority 011.

This method of arbitration works no matter how many bits of priority encoding are contained in the message.

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is stopped immediately unless it occurs on the eighth bit of a byte. In this case, the BDLC automatically appends up to two extra logic 1 bits and then stops transmitting. These two extra bits are arbitrated normally and thus do not interfere with another message. The second logic 1 bit is not sent if

Downloaded from Arrow.com.

the first loses arbitration. If the BDLC has lost arbitration to another valid message, then the two extra logic 1s do not corrupt the current message. However, if the BDLC has lost arbitration due to noise on the bus, then the two extra logic 1s ensure that the current message is detected and ignored as a noise-corrupted message.

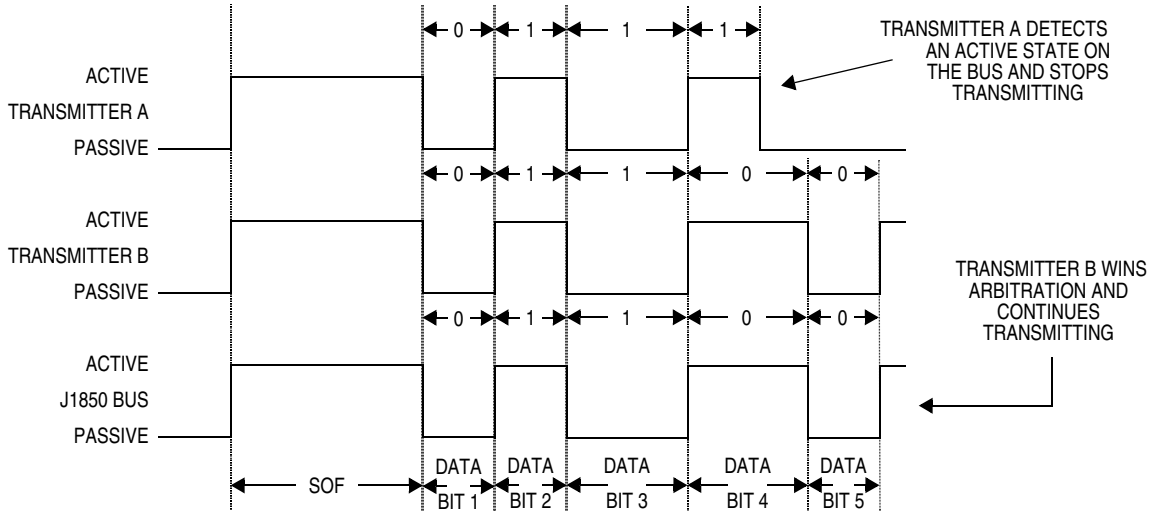


Figure 15-10. J1850 VPW Bitwise Arbitrations

15.8 BDLC Protocol Handler

The protocol handler is responsible for framing, arbitration, CRC generation/checking, and error detection. The protocol handler conforms to SAE J1850 – Class B Data Communications Network Interface.

15.8.1 Protocol Architecture

The protocol handler contains the state machine, Rx shadow register, Tx shadow register, Rx shift register, Tx shift register, and loopback multiplexer as shown in Figure 15-11.

15.8.2 Rx and Tx Shift Registers

The Rx shift register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx shadow register. The Tx shift register takes data, in parallel form, from the Tx shadow register and presents it serially to the state machine so that it can be transmitted onto the J1850 bus.

15.8.3 Rx and Tx Shadow Registers

Immediately after the Rx shift register has completed shifting in a byte of data, this data is transferred to the Rx shadow register and RDRF or RXIFR is set (see 15.9.3 BDLC State Vector Register). An interrupt is generated if the interrupt enable bit (IE) in BCR1 is set. After the transfer takes place, this new data byte in the Rx shadow register is available to the CPU interface, and the Rx shift register is ready to shift in the next byte of data. Data in the Rx shadow register must be retrieved by the CPU before it is overwritten by new data from the Rx shift register.

Downloaded from Arrow.com.

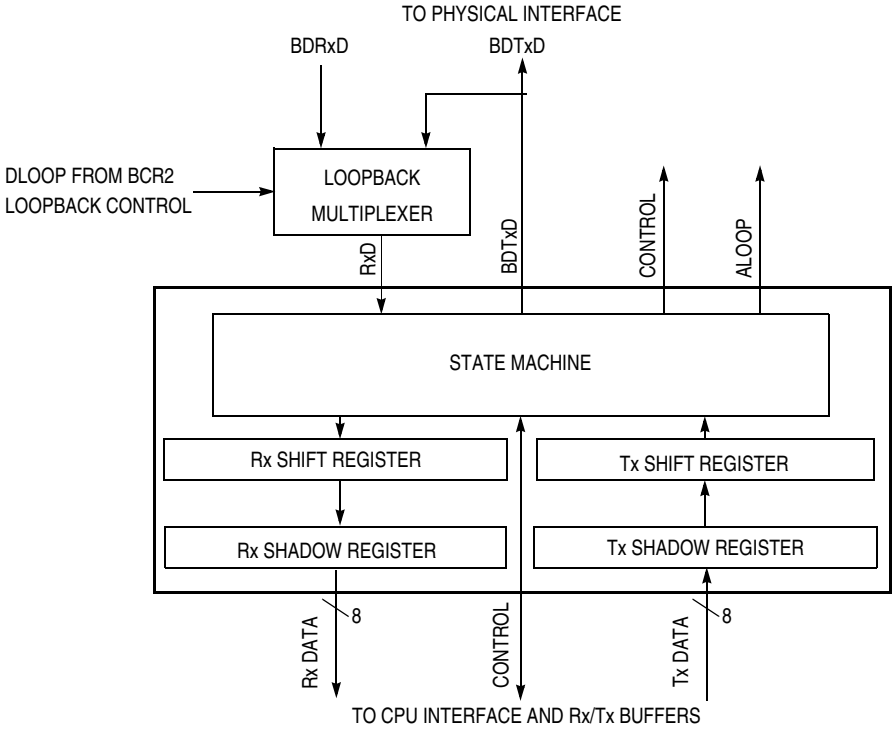


Figure 15-11. BDLC Protocol Handler Outline

Once the Tx shift register has completed its shifting operation for the current byte, the data byte in the Tx shadow register is loaded into the Tx shift register. After this transfer takes place, the Tx shadow register is ready to accept new data from the CPU when the TDRE flag in the BSVR is set.

15.8.4 Digital Loopback Multiplexer

The digital loopback multiplexer connects RxD to either BDTxD or BDRxD, depending on the state of the DLOOP bit in the BCR2. (See 15.9.2 BDLC Control Register 2.)

15.8.5 State Machine

All functions associated with performing the protocol are executed or controlled by the state machine. The state machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. These sections describe the BDLC's actions in a variety of situations.

15.8.5.1 4X Mode

The BDLC can exist on the same J1850 bus as modules which use a special 4X (41.6 Kbps) mode of J1850 variable pulse width modulation (VPW) operation. The BDLC cannot transmit in 4X mode, but it can receive messages in 4X mode, if the RX4X bit is set in BCR2. If the RX4X bit is not set in the BCR2, any 4X message on the J1850 bus is treated as noise by the BDLC and is ignored.

15.8.5.2 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the BDLC does allow for a special block mode of operation of the receiver. As far as the BDLC is concerned, a block mode message is simply a long J1850 frame

Downloaded from Arrow.com.

that contains an indefinite number of data bytes. All other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

15.8.5.3 Transmitting a Message in Block Mode

A block mode message is transmitted inherently by simply loading the bytes one by one into the BDR until the message is complete. The programmer should wait until the TDRE flag (see 15.9.3 BDLC State Vector Register) is set prior to writing a new byte of data into the BDR. The BDLC does not contain any predefined maximum J1850 message length requirement.

15.8.5.4 J1850 Bus Errors

The BDLC detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

Transmission error — If the message transmitted by the BDLC contains invalid bits or framing symbols on non-byte boundaries, this constitutes a transmission error. When a transmission error is detected, the BDLC immediately ceases transmitting. The error condition is reflected in the BSVR (see Table 15-1). If the interrupt enable bit (IE in BCR1) is set, a CPU interrupt request from the BDLC is generated.

CRC error — A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed and the CRC calculation result is not equal. The CRC code detects any single and 2-bit errors, as well as all 8-bit burst errors and almost all other types of errors. The CRC error flag (in BSVR) is set when a CRC error is detected. (See 15.9.3 BDLC State Vector Register.)

Symbol error — A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. The invalid symbol is set when a symbol error is detected. (See 15.9.3 BDLC State Vector Register.)

Framing error — A framing error is detected if an EOD or EOF symbol is detected on a non-byte boundary from the J1850 bus. A framing error also is detected if the BDLC is transmitting the EOD and instead receives an active symbol. The symbol invalid, or the out-of-range flag, is set when a framing error is detected. (See 15.9.3 BDLC State Vector Register.)

Bus fault — If a bus fault occurs, the response of the BDLC depends upon the type of bus fault.

If the bus is shorted to battery, the BDLC waits for the bus to fall to a passive state before it attempts to transmit a message. As long as the short remains, the BDLC never attempts to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC sees an idle bus, begins to transmit the message, and then detects a transmission error (in BSVR), since the short to ground does not allow the bus to be driven to the active (dominant) SOF state. The BDLC aborts that transmission and waits for the next CPU command to transmit.

In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC resumes normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus. (See 15.9.3 BDLC State Vector Register.)

BREAK — If a BREAK symbol is received while the BDLC is transmitting or receiving, an invalid symbol (in BSVR) interrupt is generated. Reading the BSVR (see 15.9.3 BDLC State Vector Register) clears this interrupt condition. The BDLC waits for the bus to idle, then waits for a start-of-frame (SOF) symbol.

Downloaded from Arrow.com.

The BDLC cannot transmit a BREAK symbol. It can receive a BREAK symbol only from the J1850 bus.

15.8.5.5 Summary

Table 15-1 provides a bus error summary.

Table 15-1. BDLC J1850 Bus Error Summary

Error Condition	BDLC Function
Transmission error	For invalid bits or framing symbols on non-byte boundaries, invalid symbol interrupt is generated. BDLC stops transmission.
Cyclical redundancy check (CRC) error	CRC error interrupt is generated. BDLC waits for EOF.
Invalid symbol: BDLC transmits, but receives invalid bits (noise)	The BDLC aborts transmission immediately. Invalid symbol interrupt is generated.
Framing error	Invalid symbol interrupt is generated. BDLC waits for end of frame (EOF).
Bus short to V_{DD}	The BDLC does not transmit until the bus is idle. Invalid symbol interrupt is generated. EOF interrupt also must be seen before another transmission attempt. Depending on length of the short, LOA flag also may be set.
Bus short to GND	Thermal overload shuts down physical interface. Fault condition is seen as invalid symbol flag. EOF interrupt must also be seen before another transmission attempt.
BDLC receives BREAK symbol	Invalid symbol interrupt is generated. BDLC waits for the next valid SOF.

15.9 BDLC Registers

Eight registers are available for controlling operation of the BDLC and for communicating data and status information. A full description of each register is given here.

15.9.1 BDLC Control Register 1

Address: $00F8

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
Write:					R	R		
Reset:	1	1	1	0	0	0	0	0

R = Reserved

Figure 15-12. BDLC Control Register 1 (BCR1)

IMSG — Ignore Message Bit

This bit disables the receiver until a new start-of-frame (SOF) is detected. The bit is cleared automatically by the reception of an SOF symbol or a BREAK symbol. It then generates interrupt requests and allows changes of the status register to occur. However, these interrupts may still be masked by the interrupt enable (IE) bit. When set, all BDLC interrupt requests are masked (except $20 in BSVR) and the status bits are held in their reset state. If this bit is set while the BDLC is receiving a message, the rest of the incoming message is ignored.

1 = Disable receiver
0 = Enable receiver

Downloaded from Arrow.com.

CLKS — Clock Select Bit

For J1850 bus communications to take place, the nominal BDLC operating frequency (f_{BDLC}) must always be 1.048576 MHz or 1 MHz. The CLKS register bit allows the user to select the frequency (1.048576 MHz or 1 MHz) used to automatically adjust symbol timing.

1 = Binary frequency, 1.048576 MHz
0 = Integer frequency, 1 MHz

R1 and R0 — Rate Select Bits

These bits determine the amount by which the frequency of the MCU CGMXCLK signal is divided to form the MUX interface clock (f_{BDLC}) which defines the basic timing resolution of the MUX interface. They may be written only once after reset, after which they become read-only bits.

The nominal frequency of f_{BDLC} must always be 1.048576 MHz or 1.0 MHz for J1850 bus communications to take place. Hence, the value programmed into these bits is dependent on the chosen MCU system clock frequency per Table 15-2.

Table 15-2. BDLC Rate Selection

f_{XCLK} Frequency	R1	R0	Division	f_{BDLC}
1.049 MHz	0	0	1	1.049 MHz
2.097 MHz	0	1	2	1.049 MHz
4.194 MHz	1	0	4	1.049 MHz
8.389 MHz	1	1	8	1.049 MHz
1.000 MHz	0	0	1	1.00 MHz
2.000 MHz	0	1	2	1.00 MHz
4.000 MHz	1	0	4	1.00 MHz
8.000 MHz	1	1	8	1.00 MHz

IE — Interrupt Enable Bit

This bit determines whether the BDLC generates CPU interrupt requests in run mode. It does not clear BSVR interrupts when exiting the BDLC stop or BDLC wait modes. Interrupt requests are maintained until all of the interrupt request sources are cleared by performing the specified actions upon the BDLC's registers (or an MCU reset sets BSVR bits to $00). Interrupts that were pending at the time that this bit is cleared may be lost.

If the programmer does not want to use the interrupt capability of the BDLC, the BDLC state vector register (BSVR) can be polled periodically to determine BDLC states.

1 = Enable interrupt requests from BDLC
0 = Disable interrupt requests from BDLC

WCM — Wait Clock Mode Bit

This bit determines the operation of the BDLC during CPU wait mode.

0 = Run BDLC internal clocks during CPU wait mode.
1 = Stop BDLC internal clocks during CPU wait mode.

Downloaded from Arrow.com.

15.9.2 BDLC Control Register 2

Address: $00FA

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
Reset:	1	1	0	0	0	0	0	0

Figure 15-13. BDLC Control Register 2 (BCR2)

This register controls transmitter operations of the BDLC.

ALOOP — Analog Loopback Mode Bit

This bit determines if the J1850 bus is driven by the analog physical interface's final drive stage. The programmer places the transceiver into loopback mode first, then sets ALOOP which resets the BDLC state machine to a known state. When the user clears ALOOP, to indicate the transceiver has been taken out of loopback mode, the BDLC waits for an EOF symbol before attempting to transmit. Most transceivers have the ALOOP feature available.

1 = Input to the analog physical interface's final drive stage is looped back to the BDLC receiver. The J1850 bus is not driven.

0 = BDLC digital circuitry drives an output for the J1850 bus. After the bit is cleared, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol time (t_{TRV4}) before message reception or a minimum of inter-frame symbol time (t_{TRV6}) before message transmission.

DLOOP — Digital Loopback Mode Bit

This bit determines the source to which the BDLC internal digital receive input is connected and can be used to isolate bus fault conditions. If a fault condition has been detected on the bus, this control bit allows the programmer to connect the digital transmit output to the digital receive input. In this configuration, data sent from the transmit buffer is reflected back into the receive buffer. If no faults exist in the BDLC, the fault is in the physical interface block or elsewhere on the J1850 bus. When the DLOOP bit is set, the BDLC is disengaged from the J1850 bus. Therefore, the BDLC does not receive an edge from the J1850 bus which would normally cause a BSVR non-maskable wakeup interrupt.

1 = BDRxD is connected to BDTxD. The BDLC is in digital loopback mode.

0 = BDTxD is not connected to BDRxD. The BDLC is taken out of digital loopback mode and can now drive or receive the J1850 bus normally (given ALOOP is not set). After clearing DLOOP, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol (t_{tv4}) time before allowing reception of a message. The BDLC requires the bus to be idle for a minimum of inter-frame separator symbol (t_{tv6}) time before allowing a message to be transmitted.

NOTE

The DLOOP bit is a fault condition aid and should never be altered after the BDR is loaded for transmission. Changing DLOOP during a transmission may cause corrupted data to be transmitted onto the J1850 network.

Before going into digital loopback mode, the RXPOL bit in the BARD register must be set so that the receive polarity is not expected to be inverted.

Downloaded from Arrow.com.

RX4XE — Receive 4X Enable Bit

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 Kbps) or receive only at 41.6 Kbps. This feature is useful for fast downloading data into a J1850 node for diagnostic or factory programming.

1 = BDLC is put in 4X receive-only operation.

0 = BDLC transmits and receives at 10.4 Kbps. Reception of a BREAK symbol automatically clears this bit and sets BDLC state vector register (BSVR) to $001C.

NBFS — Normalization Bit Format Select Bit

This bit controls the format of the normalization bit (NB). (See **Figure 15-14**.) SAE J1850 encourages using an active long (logic 0) for in-frame responses containing cyclical redundancy check (CRC) and an active short (logic 1) for in-frame responses without CRC.

0 = NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) does not end with a CRC byte.

1 = NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) does not end with a CRC byte.

TEOD — Transmit End of Data Bit

This bit is set by the programmer to indicate the end of a message is being sent by the BDLC. It appends an 8-bit CRC after completing transmission of the current byte. This bit also is used to end an in-frame response (IFR). If the transmit shadow register is full when TEOD is set, the CRC byte is transmitted after the current byte in the Tx shift register and the byte in the Tx shadow register have been transmitted. (See 15.8.3 Rx and Tx Shadow Registers for a description of the transmit shadow register.) Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC state vector register (BSVR) is cleared to allow lower priority interrupts to occur. (See 15.9.3 BDLC State Vector Register.)

1 = Transmit end-of-data (EOD) symbol

0 = TEOD bit is cleared automatically at the rising edge of the first CRC bit that is sent or if an error is detected. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol or an error condition occurs.

TSIFR, TMIFR1, TMIFR0 — Transmit In-Frame Response Bits

These bits control the type of in-frame response being sent. Only one of these bits should be set at a time. If more than one are set, the priority encoding logic forces the bits to a known value as shown in Table 15-3. For example, if 011 is written to TSIFR, TMIFR1, and TMIFR0, then internally they are encoded as 010. However, when these bits are read back, they read 011.

Table 15-3. Transmit In-Frame Response Bit Encoding

Write/Read[1]			Internal Interpretation		
TSIFR	TMIFR1	TMIFR0	TSIFR	TMIFR1	TMIFR0
0	0	0	0	0	0
1			1	0	0
0	1		0	1	0
0	0	1	0	0	1

1. Shaded cells indicate bits which do not affect internal interpretation. These bits read back as written.

Downloaded from Arrow.com.

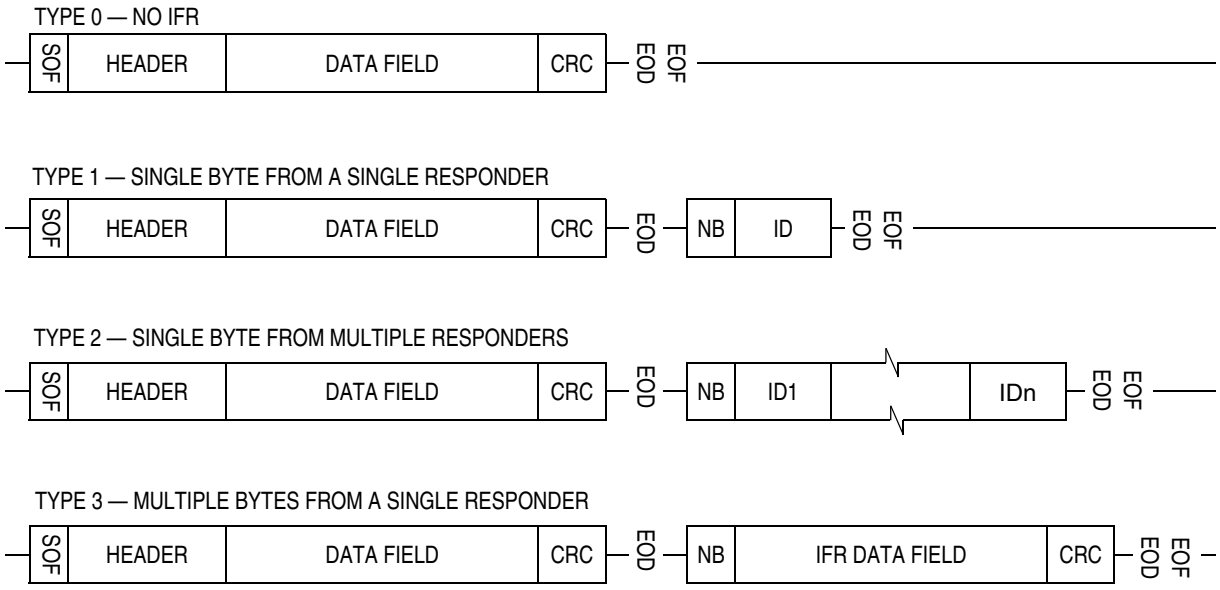


Figure 15-14. Types of In-Frame Response

The BDLC supports the in-frame response (IFR) features of J1850. The four types of J1850 IFR are shown in Figure 15-14.

The purpose of the in-frame response modes is to allow multiple nodes to acknowledge receipt of the data by responding with their personal ID or physical address in a concatenated manner after they have seen the EOD symbol. If transmission arbitration is lost by a node while sending its response, it continues to transmit its ID/address until observing its unique byte in the response stream. For VPW modulation, the first bit of the IFR is always passive; therefore, an active normalization bit must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync their responses.

TSIFR — Transmit Single Byte IFR with No CRC Bit (Type 1 and Type 2)

The TSIFR bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node. See Figure 15-14.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC attempts to transmit the appropriate normalization bit followed by the byte in the BDR.

0 = TSIFR bit is cleared automatically, once the BDLC has successfully transmitted the byte in the BDR onto the bus, or TEOD is set, or an error is detected on the bus.

If the programmer attempts to set the TSIFR bit immediately after the EOD symbol has been received from the bus, the TSIFR bit remains in the reset state and no attempt is made to transmit the IFR byte. If a loss of arbitration occurs when the BDLC attempts to transmit and after the IFR byte winning arbitration completes transmission, the BDLC again attempts to transmit the BDR (with no normalization bit). The BDLC continues transmission attempts until an error is detected on the bus, or TEOD is set, or the BDLC transmission is successful.

If loss of arbitration occurs in the last bit of the IFR byte, two additional 1 bits are not sent out because the BDLC attempts to retransmit the byte in the transmit shift register after the IRF byte winning arbitration completes transmission.

Downloaded from Arrow.com.

TMIFR0 — Transmit Multiple Byte IFR without CRC (Type 3)

The TMIFR0 bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums (see 15.7.2 J1850 Frame Format and Figure 15-14).

1 = If set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR. After TEOD has been set, the last IFR byte to be transmitted is the last byte which was written into the BDR.

0 = Bit is cleared automatically once the BDLC has successfully transmitted the EOD symbol, by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR0 bit is set, the BDLC attempts to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see 15.9.3 BDLC State Vector Register) occurs similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BCR2. This instructs the BDLC to transmit an EOD symbol once the byte in the BDR is transmitted, indicating the end of the IFR portion of the message frame. The BDLC does not append a CRC when the TMIFR0 is set.

If the programmer attempts to set the TMIFR0 bit after the EOD symbol has been received from the bus, the TMIFR0 bit remains in the reset state, and no attempt is made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting, the TMIFR0 bit is cleared, and no attempt is made to retransmit the byte in the BDR. If loss of arbitration occurs in the last bit of the IFR byte, two additional 1 bits are sent out.

NOTE

The extra logic 1 bits are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise onto the J1850 bus from a corrupted message.

TMIFR1 — Transmit Multiple Byte IFR with CRC Bit (Type 3)

The TMIFR1 bit requests the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are still subject to J1850 message length maximums (see 15.7.2 J1850 Frame Format and Figure 15-14).

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received, the BDLC attempts to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

0 = The bit is cleared automatically, once the BDLC has successfully transmitted the CRC byte and EOD symbol, by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

Downloaded from Arrow.com.

If the TMIFR1 bit is set, the BDLC attempts to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see 15.9.3 BDLC State Vector Register) occurs similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BDLC control register 2 (BCR2). This instructs the BDLC to transmit a CRC byte once the byte in the BDR is transmitted, and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, to transmit a single byte followed by a CRC byte, the programmer should load the byte into the BDR before the EOD symbol has been received, and then set the TMIFR1 bit. Once the TDRE interrupt occurs, the programmer sets the TEOD bit in the BCR2. This results in the byte in the BDR being the only byte transmitted before the IFR CRC byte, and no TDRE interrupt is generated.

If the programmer attempts to set the TMIFR1 bit immediately after the EOD symbol has been received from the bus, the TMIFR1 bit remains in the reset state, and no attempt is made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting any byte of a multiple byte IFR, the BDLC goes to the loss of arbitration state, sets the appropriate flag, and ceases transmission.

If the BDLC loses arbitration during the IFR, the TMIFR1 bit is cleared and no attempt is made to retransmit the byte in the BDR. If loss of arbitration occurs in the last bit of the IFR byte, two additional 1 bits are sent out.

NOTE

The extra logic 1 bits are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.

15.9.3 BDLC State Vector Register

Address: $00F9

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	I3	I2	I1	I0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 15-15. BDLC State Vector Register (BSVR)

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a multiplex protocol. It provides an index offset that is directly related to the BDLC's current state, which can be used with a user-supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

I0, I1, I2, I3 — Interrupt Source Bits

These bits indicate the source of the pending interrupt request. Bits are encoded according to Table 15-4.

Downloaded from Arrow.com.

Table 15-4. Interrupt Sources

BSVR	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No interrupts pending	0 (lowest)
$04	0	0	0	1	Received EOF	1
$08	0	0	1	0	Received IFR byte (RXIFR)	2
$0C	0	0	1	1	BDLC Rx data register full (RDRF)	3
$10	0	1	0	0	BDLC Tx data register empty (TDRE)	4
$14	0	1	0	1	Loss of arbitration	5
$18	0	1	1	0	Cyclical redundancy check (CRC) error	6
$1C	0	1	1	1	Symbol invalid or out of range	7
$20	1	0	0	0	Wakeup	8 (highest)

Bits I0, I1, I2, and I3 are cleared by a read of the BSVR register except when the BDLC data register needs servicing (RDRF, RXIFR, or TDRE conditions). RXIFR and RDRF can only be cleared by a read of the BSVR register followed by a read of BDR. TDRE can either be cleared by a read of the BSVR register followed by a write to the BDLC BDR register, or by setting the TEOD bit in BCR2. Clearing an invalid symbol flag requires an EOF flag to be received before the BDLC can receive or transmit. If ALOOP or DLOOP in BCR2 is set, the BDLC node is disengaged from the J1850 bus. Therefore, the BDLC does not receive any data from the J1850 bus which normally generates BSVR flags.
Upon receiving a BDLC interrupt, the user may read the value within the BSVR, transferring it to the CPU's index register. The value can be used to index a jump table to access a service routine. For example:

```
SERVICE LDX     BSVR  Fetch State Vector Number
        JMP     JMPTAB,XEnter service routine,
*                     (must end in an RTI)
*
JMPTAB  JMP     SERVE0Service condition #0
        NOP
        JMP     SERVE1Service condition #1
        NOP
        JMP     SERVE2Service condition #2
        NOP
        .
        .
        .
        JMP     SERVE8Service condition #8
        END
```

NOTE

NOP instructions are used only to align the JMP instructions onto 4-byte boundaries so that the value in the BSVR can be used intact. Each of the service routines must end with an RTI instruction to guarantee correct continued operation of the device. The first entry can be omitted since it does not correspond to an interrupt.

Downloaded from Arrow.com.

The service routines should clear all of the sources that are causing the pending interrupts. Clearing a high priority interrupt may still leave a lower priority interrupt pending, in which case bits I0, I1, and I2 of the BSVR reflect the source of the remaining interrupt request.

If fewer states are used or if a different software approach is taken, the jump table can be made smaller or omitted altogether.

15.9.4 BDLC Data Register

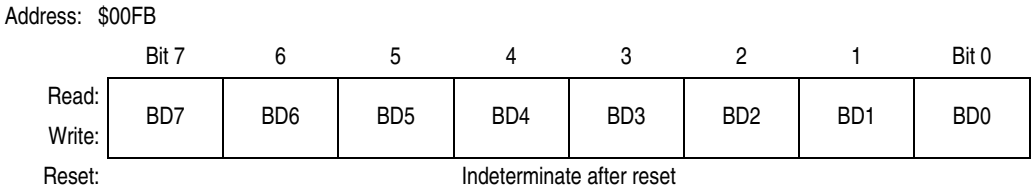


Figure 15-16. BDLC Data Register (BDR)

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC. It is also used to pass data received from the J1850 bus to the CPU. Each data byte (after the first one) should be written only after a Tx data register empty (TDRE) state is indicated in the BSVR.

Data read from this register is the last data byte received from the J1850 bus. This received data should only be read after an Rx data register full (RDRF) interrupt has occurred. (See 15.9.3 BDLC State Vector Register.)

The BDR is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next data byte. The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag is set to indicate that a new byte of data has been received. The programmer has one BDLC byte reception time to read the shadow register and clear the RDRF flag before the shadow register is overwritten by the newly received byte.

To abort an in-progress transmission, the programmer should stop loading data into the BDR. This causes a transmitter underrun error and the BDLC automatically disables the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte plus two extra logic 1 bits have been transmitted. The receiver picks this up as an error and relays it in the state vector register as an invalid symbol error.

NOTE

The extra logic 1 bits are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise on the J1850 bus from corrupting a message.

Downloaded from Arrow.com.

15.9.5 BDLC Analog Roundtrip Delay Register

Address: $00FC

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
Write:								
Reset:	1	1	0	0	0	1	1	1

= Unimplemented

Figure 15-17. BDLC Analog Roundtrip Delay Register (BARD)

Read: Anytime

Write: Once in normal modes or anytime in special mode

BARD programs the BDLC to compensate for various delays of external transceivers.

ATE — Analog Transceiver Enable Bit

The ATE bit is used to select either the on-board or an off-chip analog transceiver.

0 = Select off-chip analog transceiver.

1 = Select on-board analog transceiver.

NOTE

This device does not contain an on-board transceiver; ATE should be cleared for proper operation.

RXPOL — Receive Pin Polarity Bit

This bit selects the polarity of an incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding it back to the digital receive pin.

0 = Select inverted polarity, where external transceiver inverts the receive signal.

1 = Select normal/true polarity; true non-inverted signal from J1850 bus, for example, the external transceiver does not invert the receive signal.

BO3–BO0 — BARD Offset Bits

These bits are used to compensate for the analog transceiver roundtrip delay. Table 15-5 shows the expected transceiver delay with respect to BARD offset values.

Downloaded from Arrow.com.

Table 15-5. Offset Bit Values and Transceiver Delay

BO3–BO0	Expected Delay (μs)
0000	9
0001	10
0010	11
0011	12
0100	13
0101	14
0110	15
0111	16
1000	17
1001	18
1010	19
1011	20
1100	21
1101	22
1110	23
1111	24

15.9.6 Port DLC Control Register

Address: $00FD

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	BDLCEN	PUPDLC	RDPDLC
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 15-18. Port DLC Control Register (DLCSCR)

Read: Anytime

Write: Anytime

The BDLC port DLC functions as a general-purpose I/O port. BDLC functions takes precedence over the general-purpose port when enabled.

BDLCEN — BDLC Enable Bit

1 = Configure I/O pins for BDLC function. BDLC is active.
0 = Configure BDLC I/O pins as general-purpose I/O. BDLC is off.

PUPDLC — BDLC Pullup Enable Bit

1 = Connects internal pullups to PORTDLC I/O pins
0 = Disconnects internal pullups from PORTDLC I/O pins

RDPDLC — BDLC Reduced Drive Bit

1 = Configure PORTDLC I/O pins for reduced drive strength.
0 = Configure PORTDLC I/O pins for normal drive strength.

Downloaded from Arrow.com.

15.9.7 Port DLC Data Register

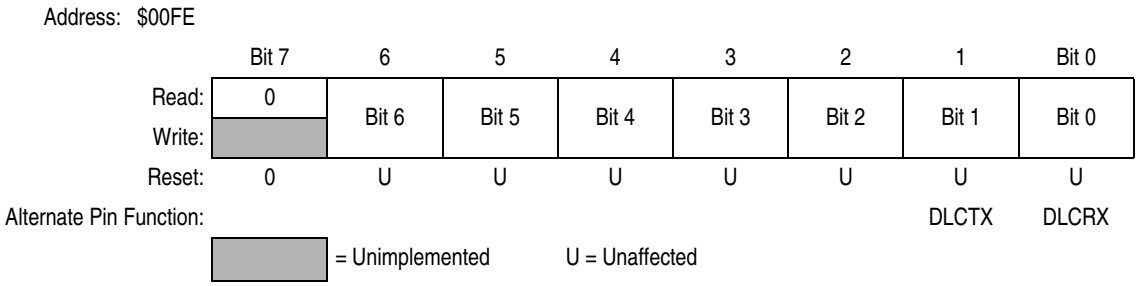


Figure 15-19. Port DLC Data Register (PORTDLC)

Read: Anytime

Write: Anytime

This register holds data to be driven out on port DLC pins or data received from port DLC pins.

When configured as an input, a read returns the pin level. When configured as output, a read returns the latched output data.

Writes do not change pin state when the pins are configured for BDLC output. Upon reset, pins are configured for general-purpose high impedance inputs.

15.9.8 Port DLC Data Direction Register

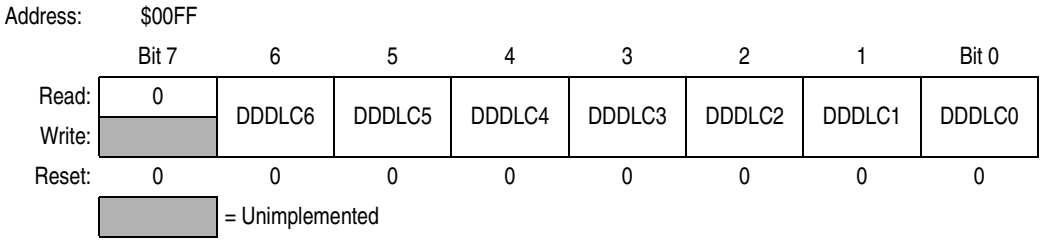


Figure 15-20. Port DLC Data Direction Register (DDRDLC)

Read: Anytime

Write: Anytime

DDDLC6–DDDLC0 — Data Direction Port DLC Bits

1 = Configure I/O pin for output
0 = Configure I/O pin for input only

Downloaded from Arrow.com.

Chapter 16
msCAN12 Controller

16.1 Introduction

The msCAN12 is the specific implementation of the msCAN concept targeted for the Freescale M68HC12 Family of microcontrollers (MCU).

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the specification from Robert Bosch GmbH dated September 1991.

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field such as:

- Real-time processing
- Reliable operation in the electromagnetic interference (EMI) environment of a vehicle
- Cost effectiveness
- Required bandwidth

The msCAN12 simplifies the application software by utilizing an advanced buffer arrangement resulting in a predictable real-time behavior.

16.2 External Pins

The msCAN12 uses two external pins, one input (RxCAN), and one output (TxCAN). The TxCAN output pin represents the logic level on the CAN: 0 is for a dominant state, and 1 is for a recessive state.

RxCAN is on bit 0 of port CAN, and TxCAN is on bit 1. The remaining six pins of port CAN are controlled by registers in the msCAN12 address space. See 16.12.14 msCAN12 Port CAN Control Register and 16.12.16 msCAN12 Port CAN Data Direction Register.

A typical CAN system with msCAN12 is shown in Figure 16-1.

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defected CAN or defected stations.

16.3 Message Storage

msCAN12 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

Downloaded from Arrow.com.

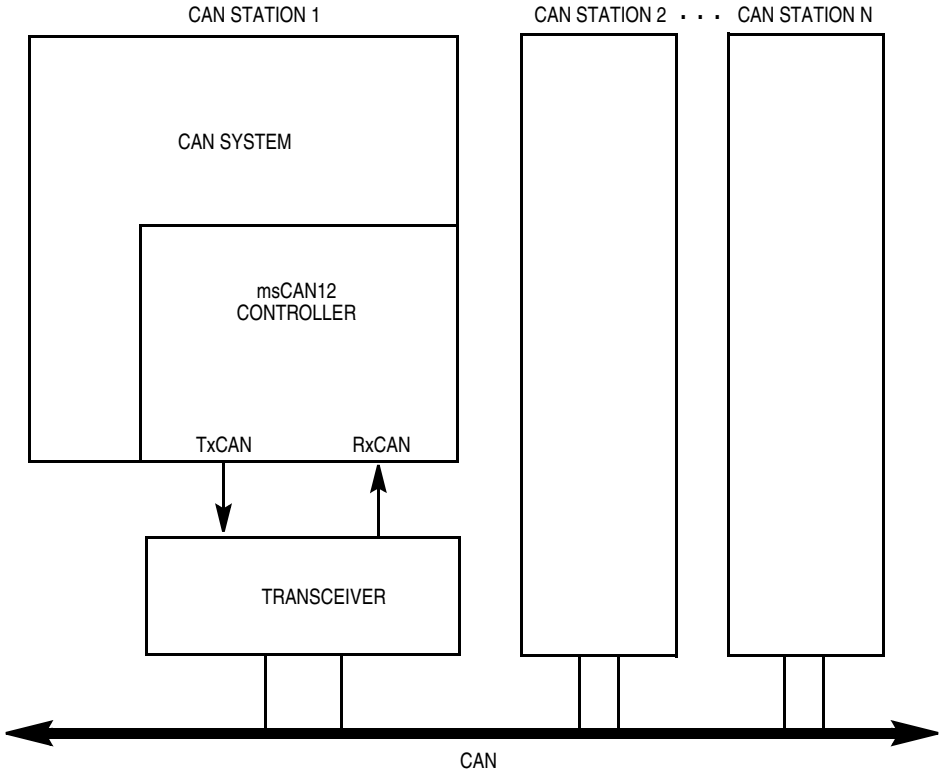


Figure 16-1. Typical CAN System with msCAN12

16.3.1 Background

Modern application layer software is built on two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized so that the highest priority message will be sent out first if more than one message is ready to be sent.

This behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the central processor unit (CPU) reacts with short latencies to the transmit interrupt.

A double buffer scheme would decouple the reloading of the transmit buffers from the actual message being sent and as such reduces the reaction requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU reloads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first requirement under all circumstances. The msCAN12 has three transmit buffers.

The second requirement calls for some sort of internal priorization which the msCAN12 implements with the “local priority” concept described here.

Downloaded from Arrow.com.

16.3.2 Receive Structures

The received messages are stored in a 2-stage first-in/first-out (FIFO) input. The two message buffers are mapped into a single memory area (see Figure 16-2). While the background receive buffer (RxBG) is exclusively associated to the msCAN12, the foreground receive buffer (RxFG) is addressable by the CPU12. This scheme simplifies the handler software since only one address area is applicable for the receive process.

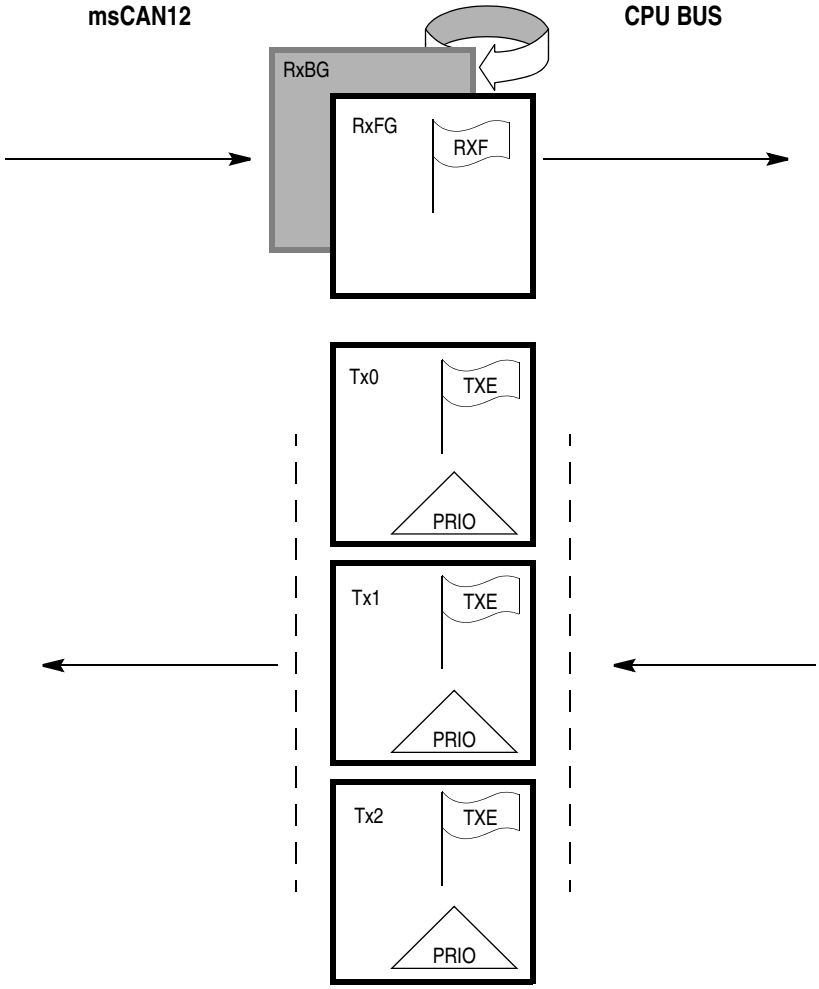


Figure 16-2. User Model for Message Buffer Organization

Both buffers have 13 bytes for storing the CAN control bits, the identifier (standard or extended), and the data contents. For details, see 16.11 Programmer's Model of Message Storage.

The receiver full flag (RXF) in the msCAN12 receiver flag register (CRFLG) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set. See 16.12.5 msCAN12 Receiver Flag Register.

On reception, each message is checked to see if it passes the filter (for details see 16.4 Identifier Acceptance Filter) and in parallel is written into RxBG. The msCAN12 copies the content of RxBG into RxFG[(1)], sets the RXF flag, and emits a receive interrupt to the CPU[(2)]. The user's receive handler has to read the received message from RxFG and then reset the RXF flag to acknowledge the interrupt and

Downloaded from Arrow.com.

to release the foreground buffer. A new message, which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The over-writing of the background buffer is independent of the identifier filter function.

When the msCAN12 module is transmitting, the msCAN12 receives its own messages into the background receive buffer, RxBG, but does not overwrite RxFG, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see 16.12.2 msCAN12 Module Control Register 1) where the msCAN12 treats its own messages exactly like all other incoming messages. The msCAN12 receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the msCAN12 must be prepared to become the receiver.

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled. The msCAN12 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

16.3.3 Transmit Structures

The msCAN12 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 16-2.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see 16.11 Programmer's Model of Message Storage). An additional transmit buffer priority register (TBPR) contains an 8-bit local priority field (PRIO). See 16.11.5 Transmit Buffer Priority Register.

To transmit a message, the CPU12 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the msCAN12 transmitter flag register (CTFLG). See 16.12.7 msCAN12 Transmitter Flag Register.

The CPU12 then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The msCAN12 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt will be emitted(1) when TXE is set, and this can be used to drive the application software to reload the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the msCAN12 uses the local priority setting of the three buffers for prioritizing. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority.

The internal scheduling process takes place whenever the msCAN12 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

1. Only if the RXF flag is not set
2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.
1. The transmit interrupt is generated only if not masked. A polling scheme can be applied on TXE also.

Downloaded from Arrow.com.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. Because messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The msCAN12 grants the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent in the meantime (ABTAK = 0).

16.4 Identifier Acceptance Filter

The identifier acceptance registers (CIDAR0–CIADAR7) define the acceptable patterns of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked don't care in the identifier mask registers (CIDMR0–CIDMR7).

A filter hit is indicated to the application software by:

- A set RXF (receive buffer full flag)
  See **16.12.5 msCAN12 Receiver Flag Register**) and
- Three bits in the identifier acceptance control register
  See **16.12.9 msCAN12 Identifier Acceptance Control Register**).

These identifier hit flags (IDHIT2–IDHIT0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. When more than one hit occurs (two or more filters match), the lower hit has priority.

A flexible, programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

1. Two identifier acceptance filters, each to be applied to the full 29 bits of the identifier and to three bits of the CAN frame: RTR, IDE, and SRR. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure 16-3 shows how the first 32-bit filter bank (CIDAR0–CIDAR3, CIDMR0–CIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CIDAR4–CUIDAR7, CIDMR4–CIDMR7) produces a filter 1 hit.

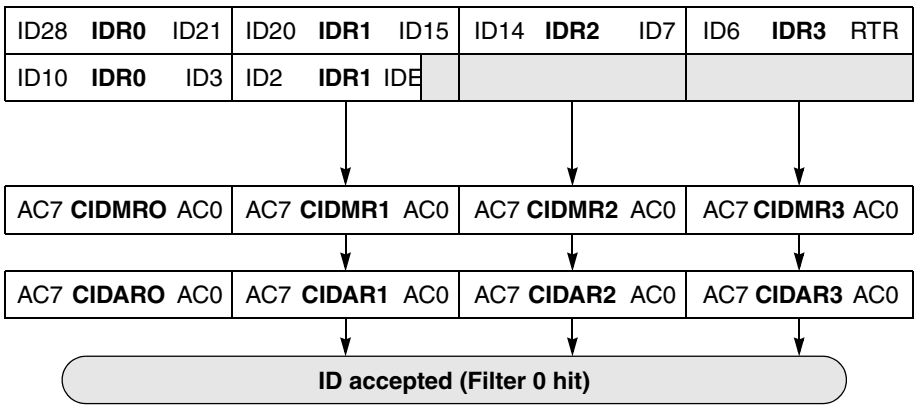


Figure 16-3. 32-Bit Maskable Identifier Acceptance Filter

Downloaded from Arrow.com.

2. Four identifier acceptance filters, each to be applied to:
   a. 11 bits of the identifier and the RTR bit of CAN 2.0A messages, or
   b. 14 most significant bits of the identifier of CAN 2.0B messages

   Figure 16-4 shows how the first 32-bit filter bank (CIDAR0–CIDAR3, CIDMR0–CIDMR3) produces filter 0 and 1 hit. Similarly, the second filter bank (CIDAR4–CUIDAR7, CIDMR4–CIDMR7) produces filter 2 and three hits.

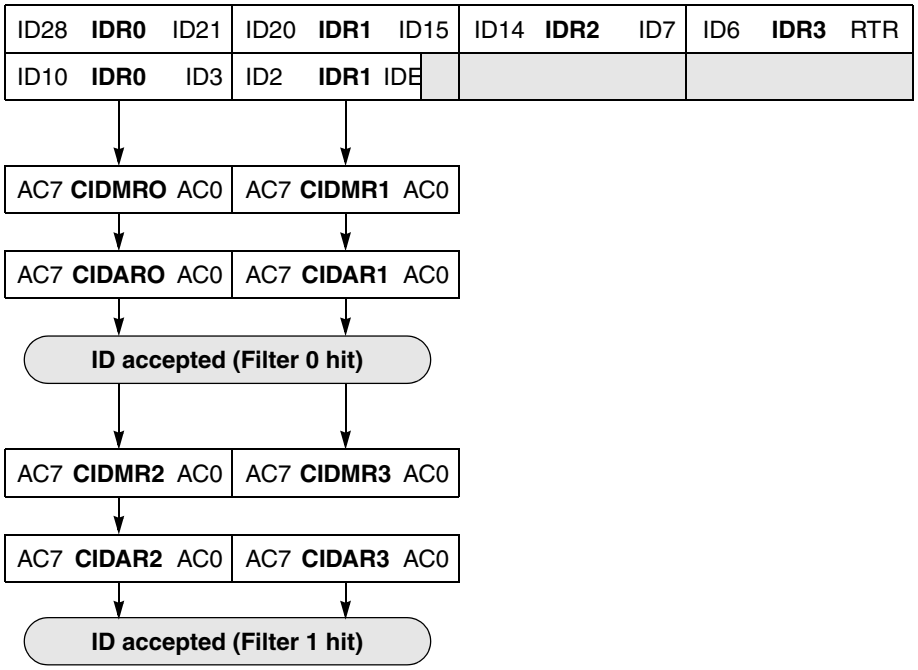


Figure 16-4. 16-Bit Maskable Acceptance Filters

3. Eight identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements eight independent filters for the first eight bits of a CAN 2.0A compliant standard identifier or of a CAN 2.0B compliant extended identifier. Figure 16-5 shows how the first 32-bit filter bank (CIDAR0–CIDAR3, CIDMR0–CIDMR3) produces filter 0 to three hits. Similarly, the second filter bank (CIDAR4–CUIDAR7, CIDMR4–CIDMR7) produces filter 4 to seven hits.
4. Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

Downloaded from Arrow.com.

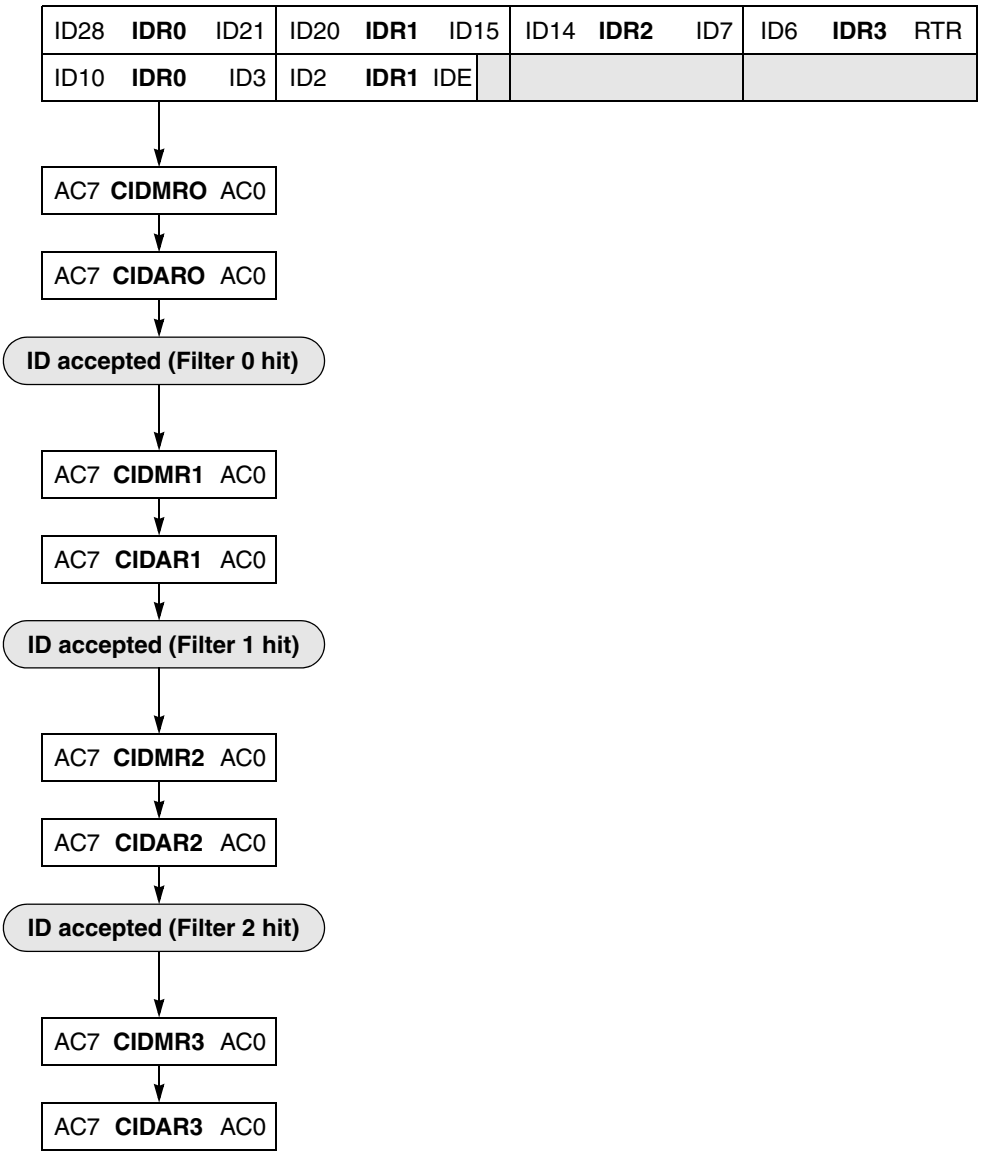


Figure 16-5. 8-Bit Maskable Acceptance Filters

Downloaded from Arrow.com.

16.5 Interrupts

The msCAN12 supports four interrupt vectors mapped onto 11 different interrupt sources, any of which can be individually masked. For details, see 16.12.5 msCAN12 Receiver Flag Register to 16.12.8 msCAN12 Transmitter Control Register.

1. *Transmit interrupt:* At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The empty message buffers TXE flags are set.
2. *Receive interrupt:* A message has been successfully received and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the end of frame (EOF) symbol. The RXF flag is set.
3. *Wakeup interrupt:* An activity on the CAN bus occurred during msCAN12 internal sleep mode.
4. *Error interrupt:* An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of these conditions:
   - *Overrun:* An overrun condition as described in **16.3.2 Receive Structures** has occurred.
   - *Receiver warning:* The receive error counter has reached the CPU warning limit of 96.
   - *Transmitter warning:* The transmit error counter has reached the CPU warning limit of 96.
   - *Receiver error passive:* The receive error counter has exceeded the error passive limit of 127 and msCAN12 has gone to error passive state.
   - *Transmitter error passive:* The transmit error counter has exceeded the error passive limit of 127 and msCAN12 has gone to error passive state.
   - *Bus off:* The transmit error counter has exceeded 255 and msCAN12 has gone to bus-off state.

16.5.1 Interrupt Acknowledge

Interrupts are associated directly with one or more status flags in either the msCAN12 receiver flag register (CRFLG) or the msCAN12 transmitter control register (CTCR). Interrupts are pending as long as one of the corresponding flags is set. The flags in the aforementioned registers must be reset within the interrupt handler to handshake the interrupt. The flags are reset through writing a 1 to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

NOTE

Bit manipulation instructions (BSET) must not be used to clear interrupt flags.

16.5.2 Interrupt Vectors

The msCAN12 supports four interrupt vectors as shown in Table 16-1. The vector addresses are dependent on the chip integration and to be defined. The relative interrupt priority is also integration dependent and to be defined.

Downloaded from Arrow.com.

Table 16-1. msCAN12 Interrupt Vectors

Function	Source	Local Mask	Global Mask
Wakeup	WUPIF	WUPIE	I bit
Error interrupts	RWRNIF	RWRNIE	
	TWRNIF	TWRNIE	
	RERRIF	RERRIE	
	TERRIF	TERRIE	
	BOFFIF	BOFFIE	
	OVRIF	OVRIE	
Receive	RXF	RXFIE	
Transmit	TXE0	TXEIE0	
	TXE1	TXEIE1	
	TXE2	TXEIE2	

16.6 Protocol Violation Protection

The msCAN12 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements these features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the msCAN12 cannot be modified while the msCAN12 is online. The SFTRES bit in CMCR0 (see **16.12.1 msCAN12 Module Control Register 0**) serves as a lock to protect these registers:
  - msCAN12 module control register 1 (CMCR1)
  - msCAN12 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - msCAN12 identifier acceptance control register (CIDAC)
  - msCAN12 identifier acceptance registers (CIDAR0–CIDAR7)
  - msCAN12 identifier mask registers (CIDMR0–CIDMR7)
- The TxCAN pin is forced to recessive if the CPU goes into stop mode.

16.7 Low-Power Modes

In addition to normal mode, the msCAN12 has three modes with reduced power consumption compared to normal mode. In sleep and soft-reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In power-down mode, all clocks are stopped and no power is consumed.

The wait-for-interrupt (WAI) and STOP instructions put the MCU in low power-consumption standby modes. Table 16-2 summarizes the combinations of msCAN12 and CPU modes. A particular combination of modes is entered for the given settings of the bits CSWAI, SLPAK, and SFTRES. For all modes, an msCAN wakeup interrupt can occur only if SLPAK = WUPIE = 1. While the CPU is in wait mode, the msCAN12 can be operated in normal mode and emit interrupts. (Registers can be accessed via background debug mode.)

Downloaded from Arrow.com.

Table 16-2. msCAN12 versus CPU Operating Modes

msCAN Mode	CPU Mode		
	Stop	Wait	Run
Power-down	CSWAI = X[1] SLPAK = X SFTRES = X	CSWAI = 1 SLPAK = X SFTRES = X	
Sleep		CSWAI = 0 SLPAK = 1 SFTRES = 0	CSWAI = X SLPAK = 1 SFTRES = 0
Soft reset		CSWAI = 0 SLPAK = 0 SFTRES = 1	CSWAI = X SLPAK = 0 SFTRES = 1
Normal		CSWAI = 0 SLPAK = 0 SFTRES = 0	CSWAI = X SLPAK = 0 SFTRES = 0

1. X means don't care.

16.7.1 msCAN12 Sleep Mode

The CPU can request the msCAN12 to enter this low-power mode by asserting the SLPRQ bit in the module configuration register. See Figure 16-6.

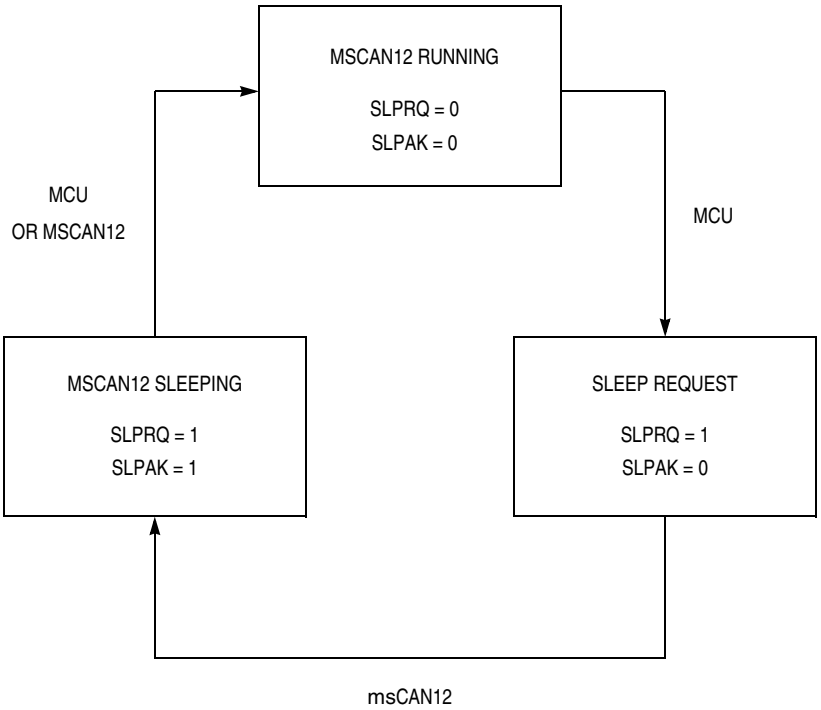


Figure 16-6. Sleep Request/Acknowledge Cycle

Downloaded from Arrow.com.

The time when the msCAN12 enters sleep mode depends on its activity. For instance,

- If it is transmitting, it continues to transmit until there are no more messages to be transmitted and then goes into sleep mode.
- If it is receiving, it waits for the end of this message and then goes into sleep mode.
- If it is neither transmitting nor receiving, it immediately goes into sleep mode.

The application software must avoid setting up a transmission (by clearing one or more TXE flag(s)) and immediately request sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations whether the msCAN12 starts transmitting or goes into sleep mode directly.

During sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into sleep mode. When in sleep mode, the msCAN12 stops its internal clocks. However, clocks to allow register accesses still run. If the msCAN12 is in bus-off state, it stops counting the 128 x 11 consecutive recessive bits due to the stopped clocks. The TxCAN pin stays in recessive state. If RXF = 1, the message can be read and RXF can be cleared. Copying RxBG into RxFG doesn't take place while in sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in sleep mode.

NOTE

The msCAN12 leaves sleep mode (wakeup) when one of these occurs:

- Bus activity occurs.
- MCU clears the SLPRQ bit.
- MCU sets SFTRES.

NOTE

The MCU cannot clear the SLPRQ bit before the msCAN12 is in sleep mode (SLPAK = 1).

After wakeup, the msCAN12 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the msCAN12 is wakened by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions are executed upon wakeup: copying of RxBG into RxFG, message aborts, and message transmissions. If the msCAN12 is still in bus-off state after sleep mode was left, it continues counting the 128 x 11 consecutive recessive bits.

16.7.2 msCAN12 Soft-Reset Mode

In soft-reset mode, the msCAN12 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing, and the CAN message filter. See **16.12.1 msCAN12 Module Control Register 0** for a complete description of the soft-reset mode.

When setting the SFTRES bit, the msCAN12 immediately stops all ongoing transmissions and receptions, potentially causing the CAN protocol violations.

NOTE

The user is responsible for ensuring that the msCAN12 is not active when soft-reset mode is entered. The recommended procedure is to put the msCAN12 into sleep mode before the SFTRES bit is set.

Downloaded from Arrow.com.

16.7.3 msCAN12 Power-Down Mode

The msCAN12 is in power-down mode when either of these occurs:

- CPU is in stop mode.
- CPU is in wait mode and the CSWAI bit is set. See **16.12.1 msCAN12 Module Control Register 0** and **16.12.2 msCAN12 Module Control Register 1**.

When entering power-down mode, the msCAN12 immediately stops all on-going transmissions and receptions, potentially causing CAN protocol violations.

NOTE

The user should be careful that the msCAN12 is not active when power-down mode is entered. The recommended procedure is to put the msCAN12 into sleep mode before the STOP instruction — or the WAI instruction, if CSWAI is set — is executed.

To protect the CAN bus system from fatal consequences of violations to this rule, the msCAN12 will drive the TxCAN pin into recessive state.

In power-down mode, no registers can be accessed.

16.7.4 Programmable Wakeup Function

The msCAN12 can be programmed to apply a low-pass filter function to the RxCAN input line while in sleep mode. See control bit WUPM in the module control register, **16.12.2 msCAN12 Module Control Register 1**. This feature can be used to protect the msCAN12 from wakeup due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

16.8 Timer Link

The msCAN12 generates a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal is generated right after the EOF. A pulse of one bit time is generated. As the msCAN12 receiver engine also receives the frames being sent by itself, a timer signal also is generated after a successful transmission.

The previously described timer signal can be routed into the on-chip timer interface module (TIM). This signal is connected to the timer n channel m input[1] under the control of the timer link enable (TLNKEN) bit in the CMCR0.

After timer n has been programmed to capture rising edge events, it can be used under software control to generate 16 bit-time stamps which can be stored with the received message.

16.9 Clock System

Figure 16-7 shows the structure of the msCAN12 clock generation circuitry. With this flexible clocking scheme the msCAN12 is able to handle CAN bus rates ranging from 10 kbps to 1 Mbps.

1. The timer channel being used for the timer link is integration dependent.

Downloaded from Arrow.com.

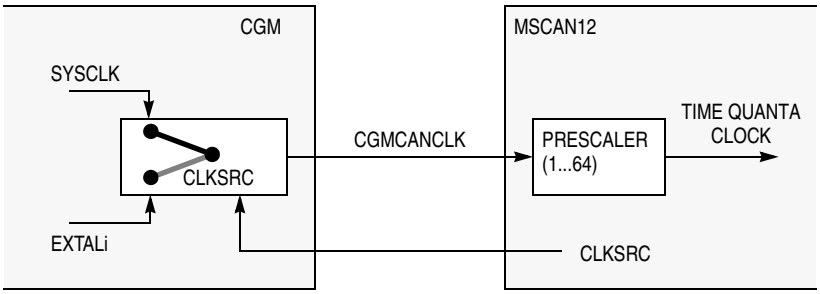


Figure 16-7. Clocking Scheme

The clock source bit (CLKSRC) in the msCAN12 module control register (CMCR1) (see 16.12.3 msCAN12 Bus Timing Register 0) defines whether the msCAN12 is connected to the output of the crystal oscillator (EXTALi) or to a clock twice as fast as the system clock (ECLK).

The clock source has to be chosen so that the tight oscillator tolerance requirements (up to 0.4 percent) of the CAN protocol are met. Additionally, for high CAN bus rates (1 Mbps), a 50 percent duty cycle of the clock is required.

For microcontrollers without the CGM module, CGMCANCLK is driven from the crystal oscillator (EXTALi).

A programmable prescaler is used to generate out of msCANCLK the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the msCAN12.

$$f_{Tq} = \frac{f_{CGMCANCLK}}{Presc\ \ value}$$

A bit time is subdivided into three segments[1]:

- SYNC_SEG — This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1 — This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.
- Time segment 2 — This segment represents the PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta.

$$BitRate = \frac{f_{Tq}}{number\ \ of\ \ TimeQuanta}$$

The synchronization jump width can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter. These parameters can be set by programming the bus timing registers (CBTR0 and CBTR1). See **16.12.3 msCAN12 Bus Timing Register 0** and 16.12.4 msCAN12 Bus Timing Register 1.

NOTE

It is the user's responsibility to make sure that the bit time settings are in compliance with the CAN standard. Table 16-3 gives an overview on the CAN-conforming segment settings and the related parameter values.

1. For further explanation of the underlying concepts, refer to ISO/DIS 11519-1, Section 10.3.

Downloaded from Arrow.com.

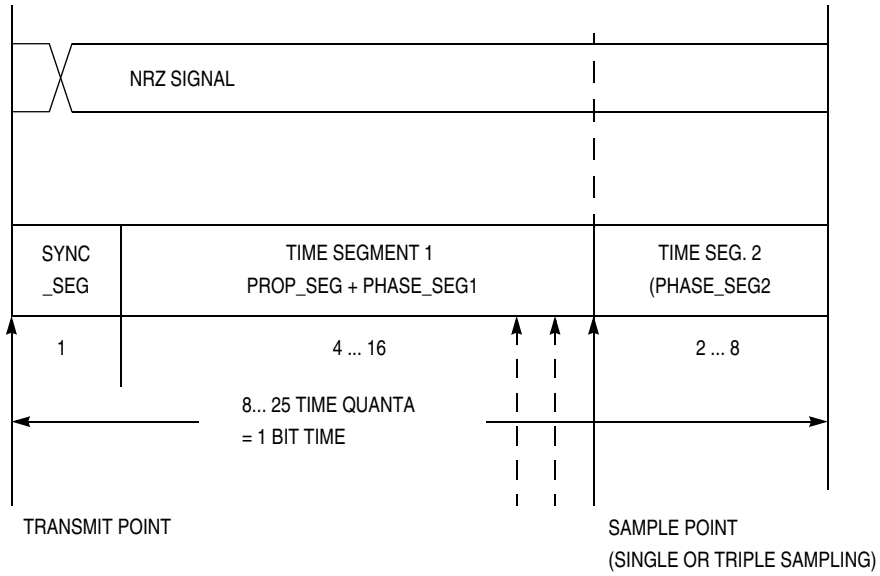


Figure 16-8. Segments within the Bit Time

Table 16-3. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	TSEG1	Time Segment 2	TSEG2	Synchronize Jump Width	SJW
5.. 10	4 .. 9	2	1	1 .. 2	0 .. 1
4 .. 11	3 .. 10	3	2	1 .. 3	0 .. 2
5 .. 12	4 .. 11	4	3	1 .. 4	0 .. 3
6 .. 13	5 .. 12	5	4	1 .. 4	0 .. 3
7 .. 14	6 .. 13	6	5	1 .. 4	0 .. 3
8 .. 15	7 .. 14	7	6	1 .. 4	0 .. 3
9 .. 16	8 .. 15	8	7	1 .. 4	0 .. 3

Downloaded from Arrow.com.

16.10 Memory Map

The msCAN12 occupies 128 bytes in the CPU12 memory space. The background receive buffer can be read only in test mode.

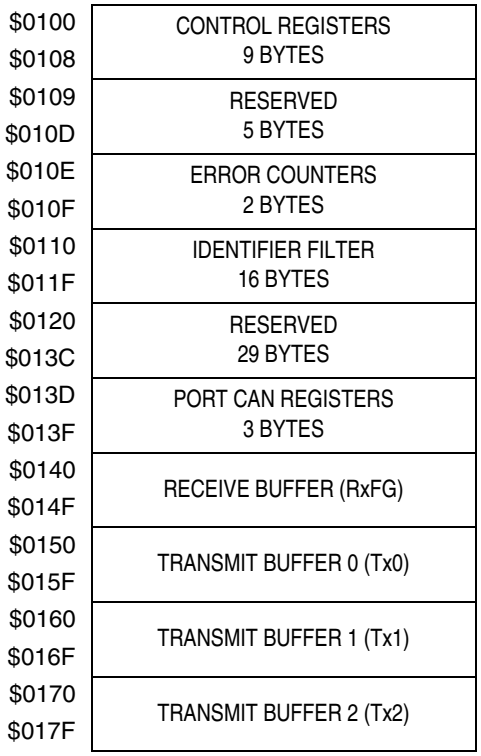


Figure 16-9. msCAN12 Memory Map

16.11 Programmer's Model of Message Storage

This subsection details the organization of the receive and transmit message buffers and the associated control registers.

16.11.1 Message Buffer Organization

Figure 16-10 shows the organization of a single message buffer. For reasons of programmer interface simplification, the receive and transmit message buffers have the same register organization. Each message buffer allocates 16 bytes in the memory map containing:

- 13-byte data structure which includes an identifier section (IDRn), a data section (DSRn), and the data length register (DLR)
- Transmit buffer priority register (TBPR) which is only applicable for transmit buffers. See 16.11.5 Transmit Buffer Priority Register
- Two unused bytes

All bits of the 13-byte data structure are undefined out of reset.

NOTE

The receive buffer can be read anytime but cannot be written. The transmit buffers can be read or written anytime.

Downloaded from Arrow.com.

Address[1]	Register Name
01x0	IDENTIFIER REGISTER 0
01x1	IDENTIFIER REGISTER 1
01x2	IDENTIFIER REGISTER 2
01x3	IDENTIFIER REGISTER 3
01x4	DATA SEGMENT REGISTER 0
01x5	DATA SEGMENT REGISTER 1
01x6	DATA SEGMENT REGISTER 2
01x7	DATA SEGMENT REGISTER 3
01x8	DATA SEGMENT REGISTER 4
01x9	DATA SEGMENT REGISTER 5
01xA	DATA SEGMENT REGISTER 6
01xB	DATA SEGMENT REGISTER 7
01xC	DATA LENGTH REGISTER
01xD	TRANSMIT BUFFER PRIORITY REGISTER[2]
01xE	UNUSED
01xF	UNUSED

1. x is 4, 5, 6, or 7 depending on which buffer, RxFG, Tx0, Tx1, or Tx2, respectively.
2. Not applicable for receive buffers.

Figure 16-10. Message Buffer Organization

16.11.2 Identifier Registers

The Bosch CAN 2.0A and 2.0B protocol specifications allow for two different sizes of message identifiers. The Bosch CAN 2.0A specification requires an 11-bit identifier in the message buffer and the Bosch CAN 2.0B specification requires a 29-bit identifier in the message buffer. The resulting message buffers are referred to as standard and extended formats, respectively.

The identifier registers (IDRn) in the memory map can be configured to create either the 11-bit (IDR10–IDR0) identifier necessary for the standard format or the 29-bit (IDR28–IDR0) identifier necessary for the extended format. Figure 16-11 details the identifier structure used in the standard format while Figure 16-12 details the identifier structure used in the extended format. ID10/ID28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be the highest for the smallest binary number.

Downloaded from Arrow.com.

Addr.(1)	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$01x0	Identifier Register 0 (IDR0)	Read: Write:	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
		Reset:	Undefined out of reset							
$01x1	Identifier Register 1 (IDR1)	Read: Write:	ID2	ID1	ID0	RTR	IDE			
		Reset:	Undefined out of reset							
$01x2	Identifier Register 2 (IDR2)	Read: Write:								
		Reset:	Undefined out of reset							
$01x3	Identifier Register 3 (IDR3)	Read: Write:								
		Reset:	Undefined out of reset							

Note 1. x is 4, 5, 6, or 7 depending on which buffer, RxFG, Tx0, Tx1, or Tx3, respectively. = Unimplemented

Figure 16-11. Identifier Mapping in the Standard Format

Addr.(1)	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$01x0	Identifier Register 0 (IDR0)	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		Reset:	Undefined out of reset							
$01x1	Identifier Register 1 (IDR1)	Read: Write:	ID20	ID19	ID18	SRR	IDE	ID17	ID16	ID15
		Reset:	Undefined out of reset							
$01x2	Identifier Register 2 (IDR2)	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		Reset:	Undefined out of reset							
$01x3	Identifier Register 3 (IDR3)	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		Reset:	Undefined out of reset							

Note 1. x is 4, 5, 6, or 7 depending on which buffer, RxFG, Tx0, Tx1, or Tx3, respectively.

Figure 16-12. Identifier Mapping in the Extended Format

SRR — Substitute Remote Request Bit

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

IDE — ID Extended Flag

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer the flag indicates to the msCAN12 what type of identifier to send.

0 = Standard format (11 bit)
1 = Extended format (29 bit)

Downloaded from Arrow.com.

RTR — Remote Transmission Request Flag

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

0 = Data frame
1 = Remote frame

16.11.3 Data Length Register

The data length register (DLR) keeps the data length field of the CAN frame.

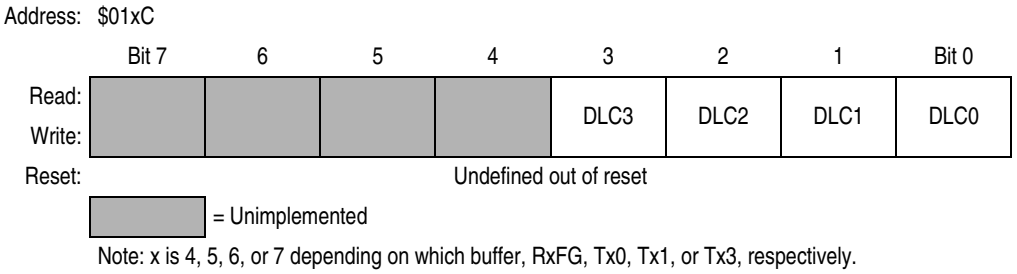


Figure 16-13. Data Length Register (DLR)

DLC3–DLC0 — Data Length Code Bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 16-4 shows the effect of setting the DLC bits.

Table 16-4. Data Length Codes

Data Length Code				Data Byte Count
DLC3	DLC2	DLC1	DLC0	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

Downloaded from Arrow.com.

16.11.4 Data Segment Registers

The eight data segment registers (DSR0–DSR7) contain the data to be transmitted or being received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR. Data is transmitted starting from the data segment register 0 (DSR0), beginning with the most significant bit (DB7), and continuing until the number of bytes specified in the data length register (DLR) is complete.

Addr.(1)	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$01x4	Data Segment Register 0 (DSR0)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01x5	Data Segment Register 1 (DSR1)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01x6	Data Segment Register 2 (DSR2)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01x7	Data Segment Register 3 (DSR3)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01x8	Data Segment Register 4 (DSR4)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01x9	Data Segment Register 5 (DSR5)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01xA	Data Segment Register 6 (DSR6)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							
$01xB	Data Segment Register 7 (DSR7)	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Undefined out of reset							

Note 1.: x is 4, 5, 6, or 7 depending on which buffer, RxFG, Tx0, Tx1, or Tx3, respectively.

Figure 16-14. Data Segment Registers (DSRn)

16.11.5 Transmit Buffer Priority Register

Address: $01xD

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PRIO7	PRIO6	PRIO5	PRIO4	PRIO3	PRIO2	PRIO1	PRIO0
Reset:	Unaffected by reset							

Note 1. x is 5, 6, or 7 depending on which buffer Tx0, Tx1, or Tx2, respectively.

Figure 16-15. Transmit Buffer Priority Register (TBPR)

Downloaded from Arrow.com.

PRIO7–PRIO0— Local Priority

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritization process of the msCAN12 and is defined to be highest for the smallest binary number. The msCAN12 implements this internal prioritization mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritization right before the start of frame (SOF) is sent.
- The transmission buffer with the lowest local priority field wins the prioritization.
- In case of more than one buffer having the same lowest priority, the message buffer with the lowest index number wins.

16.12 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency.

16.12.1 msCAN12 Module Control Register 0

Address: $0100

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	CSWAI	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
Write:								
Reset:	0	0	1	0	0	0	0	1

= Unimplemented

Figure 16-16. msCAN12 Module Control Register 0 (CMCR0)

CSWAI — CAN Stops in Wait Mode Bit

0 = The module is not affected during wait mode.
1 = The module ceases to be clocked during wait mode.

SYNCH — Synchronized Status Bit

This bit indicates whether the msCAN12 is synchronized to the CAN bus and as such can participate in the communication process.

0 = msCAN12 is not synchronized to the CAN bus.
1 = msCAN12 is synchronized to the CAN bus.

TLNKEN — Timer Enable Flag

This flag is used to establish a link between the msCAN12 and the on-chip timer. See **16.8 Timer Link**.

0 = Port is connected to the timer input.
1 = msCAN12 timer signal output is connected to the timer input.

SLPAK — Sleep Mode Acknowledge Flag

This flag indicates whether the msCAN12 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request. See 16.7.1 msCAN12 Sleep Mode.

0 = Wakeup – The msCAN12 is not in sleep mode.
1 = Sleep – The msCAN12 is in sleep mode.

SLPRQ — Sleep Request, Go To Sleep Mode Flag

This flag requests the msCAN12 to go into an internal power-saving mode (see **16.7.1 msCAN12 Sleep Mode**).

0 = Wakeup – The msCAN12 will function normally.
1 = Sleep request – The msCAN12 will go into sleep mode.

Downloaded from Arrow.com.

SFTRES — Soft-Reset Bit

When this bit is set by the CPU, the msCAN12 immediately enters the soft-reset state. Any on-going transmission or reception is aborted and synchronization to the bus is lost.

These registers will go into and stay in the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

Registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–CIDAR7, and CIDMR0–CIDMR7 can be written only by the CPU when the msCAN12 is in soft-reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the msCAN12 will try to synchronize to the CAN bus. For example, if the msCAN12 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the msCAN12 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

0 = Normal operation
1 = msCAN12 in soft-reset state

16.12.2 msCAN12 Module Control Register 1

Address: $0101

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-17. msCAN12 Module Control Register 1 (CMCR1)

LOOPB — Loop Back Self-Test Mode Bit

When this bit is set, the msCAN12 performs an internal loop back which can be used for self-test operation. The bit stream output of the transmitter is fed back to the receiver. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (1). In this state the msCAN12 ignores the bit sent during the ACK slot of the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

0 = Normal operation
1 = Activate loop back self-test mode

NOTE

The ACK bit is added to the CAN frame by the protocol. For more information on the CAN frame and the ACK bit, refer to the Bosch CAN 2.0 specification.

WUPM — Wakeup Mode Flag

This flag defines whether the integrated low-pass filter is applied to protect the msCAN12 from spurious wakeups. See **16.7.4 Programmable Wakeup Function**.

0 = msCAN12 will wake up the CPU after any recessive-to- dominant edge on the CAN bus.
1 = msCAN12 will wake up the CPU only in the case of a dominant pulse on the bus which has a length of approximately t_{WUP}.

Downloaded from Arrow.com.

CLKSRC — msCAN12 Clock Source Flag

This flag defines which clock source the msCAN12 module is driven from (only for system with CGM module. See **16.9 Clock System** and Figure 16-7.

0 = msCAN12 clock source is EXTALi.

1 = msCAN12 clock source is twice the frequency of ECLK.

NOTE

The CMCR1 register can be written only if the SFTRES bit in CMCR0 is set.

16.12.3 msCAN12 Bus Timing Register 0

Address: $0102

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SJW1	SJW0	BRP5	BRP4	BPR3	BPR2	BPR1	BPR0
Reset:	0	0	0	0	0	0	0	0

Figure 16-18. msCAN12 Bus Timing Register 0 (CBTR0)

SJW1 and SJW0 — Synchronization Jump Width Bits

The synchronization jump width defines the maximum number of time quanta (Tq) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 16-5).

Table 16-5. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 Tq clock cycle
0	1	2 Tq clock cycles
1	0	3 Tq clock cycles
1	1	4 Tq clock cycles

BRP5–BRP0 — Baud Rate Prescaler Bits

These bits determine the time quanta (Tq) clock, which is used to build up the individual bit timing, according to Table 16-6.

Table 16-6. Baud Rate Prescaler

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Prescaler Value (P)
0	0	0	0	0	0	1
0	0	0	0	0	1	2
0	0	0	0	1	0	3
0	0	0	0	1	1	4
:	:	:	:	:	:	:
1	1	1	1	1	1	64

NOTE

The CBTR0 register can be written only if the SFTRES bit in CMCR0 is set.

Downloaded from Arrow.com.

16.12.4 msCAN12 Bus Timing Register 1

Address: $0103

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
Reset:	0	0	0	0	0	0	0	0

Figure 16-19. msCAN12 Bus Timing Register 1 (CBTR1)

SAMP — Sampling Bit

This bit determines the number of samples of the serial bus to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

0 = One sample per bit

1 = Three samples per bit.(1)

TSEG22–TSEG10 — Time Segment Bits

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. See Figure 16-7.

Table 16-7. Time Segment Syntax

SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit point	A node in transmit mode will transfer a new value to the CAN bus at this point.
Sample point	A node in receive mode will sample the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table 16-8.

Table 16-8. Time Segment Values

TSEG13	TSEG12	TSEG11	TSEG10	Time Segment 1
0	0	0	0	1 Tq clock cycle
0	0	0	1	2 Tq clock cycles
0	0	1	0	3 Tq clock cycles
0	0	1	1	4 Tq clock cycles
.	.	.	.	.
1	1	1	1	16 Tq clock cycles

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 Tq clock cycle
0	0	1	2 Tq clock cycles
.	.	.	.
1	1	1	8 Tq clock cycles

1. In this case, PHASE_SEG1 must be at least two times quanta.

Downloaded from Arrow.com.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta (Tq) clock cycles per bit (as shown in Table 16-8).

$$BitTime = \frac{Presc\ \ value}{f_{CGMCANCLK}} \bullet number\ \ of\ \ TimeQuanta$$

NOTE

The CBTR1 register can be written only if the SFTRES bit in CMCR0 is set.

16.12.5 msCAN12 Receiver Flag Register

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no longer. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

Address: $0104

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
Reset:	0	0	0	0	0	0	0	0

Figure 16-20. msCAN12 Receiver Flag Register (CRFLG)

WUPIF — Wakeup Interrupt Flag

If the msCAN12 detects bus activity while in sleep mode, it sets the WUPIF flag. If not masked, a wakeup interrupt is pending while this flag is set.

0 = No wakeup activity has been observed while in sleep mode.
1 = msCAN12 has detected activity on the bus and requested wakeup.

RWRNIF — Receiver Warning Interrupt Flag

This flag is set when the msCAN12 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the error interrupt flags nor the bus-off interrupt flag is set(1). If not masked, an error interrupt is pending while this flag is set.

0 = No receiver warning status has been reached.
1 = msCAN12 went into receiver warning status.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the msCAN12 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags nor the bus-off interrupt flag is set(2). If not masked, an error interrupt is pending while this flag is set.

0 = No transmitter warning status has been reached.
1 = msCAN12 went into transmitter warning status.

1. Condition to set the flag: RWRNIF = (96 ≤ REC ≤ 127) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TWRNIF = (96 ≤ TEC ≤ 127) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$

Downloaded from Arrow.com.

RERRIF — Receiver Error Passive Interrupt Flag

This flag is set when the msCAN12 goes into error passive status due to the receive error counter (REC) exceeding 127 and the bus-off interrupt flag is not set[1]. If not masked, an error interrupt is pending while this flag is set.

0 = No receiver error passive status has been reached.
1 = msCAN12 went into receiver error passive status.

TERRIF — Transmitter Error Passive Interrupt Flag

This flag is set when the msCAN12 goes into error passive status due to the transmit error counter (TEC) exceeding 127 and the bus-off interrupt flag is not set[2]. If not masked, an error interrupt is pending while this flag is set.

0 = No transmitter error passive status has been reached.
1 = msCAN12 went into transmitter error passive status.

BOFFIF — Bus-Off Interrupt Flag

This flag is set when the msCAN12 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the msCAN12 has monitored 128 times 11 consecutive recessive bits on the bus. If not masked, an error interrupt is pending while this flag is set.

0 = No bus-off status has been reached.
1 = msCAN12 went into bus-off status.

OVRIF — Overrun Interrupt Flag

This flag is set when a data overrun condition occurs. If not masked, an Error interrupt is pending while this flag is set.

0 = No data overrun has occurred.
1 = A data overrun has been detected.

RXF — Receive Buffer Full Flag

The RXF flag is set by the msCAN12 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer, the RXF flag must be handshaken (cleared) to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

0 = Receive buffer is released (not full).
1 = Receive buffer is full. A new message is available.

NOTE

To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.

1. Condition to set the flag: RERRIF = (128 ≤ REC ≤ 255) & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TERRIF = (128 ≤ TEC ≤ 255) & $\overline{\text{BOFFIF}}$

Downloaded from Arrow.com.

16.12.6 msCAN12 Receiver Interrupt Enable Register

Address: $0105

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
Reset:	0	0	0	0	0	0	0	0

Figure 16-21. msCAN12 Receiver Interrupt Enable Register (CRIER)

WUPIE — Wakeup Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A wakeup event results in a wakeup interrupt.

RWRNIE — Receiver Warning Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A receiver warning status event results in an error interrupt.

TWRNIE — Transmitter Warning Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A transmitter warning status event results in an error interrupt.

RERRIE — Receiver Error Passive Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A receiver error passive status event results in an error interrupt.

TERRIE — Transmitter Error Passive Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A transmitter error passive status event results in an error interrupt.

BOFFIE — Bus-Off Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A bus-off event results in an error interrupt.

OVRIE — Overrun Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = An overrun event results in an error interrupt.

RXFIE — Receiver Full Interrupt Enable Bit

0 = No interrupt is generated from this event.
1 = A receive buffer full (successful message reception) event results in a receive interrupt.

NOTE

The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.

Downloaded from Arrow.com.

16.12.7 msCAN12 Transmitter Flag Register

The abort acknowledge flags are read only. The transmitter buffer empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. Each transmitter buffer empty flag has an associated interrupt enable bit in the CTCR register. A hard or soft reset resets the register.

Address: $0106

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
Write:								
Reset:	0	0	0	0	0	1	1	1

= Unimplemented

Figure 16-22. msCAN12 Transmitter Flag Register (CTFLG)

ABTAK2–ABTAK0 — Abort Acknowledge Flag

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent in the meantime. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

0 = The message has not been aborted; it has been sent.
1 = The message has been aborted.

TXE2–TXE0 —Transmitter Buffer Empty Flag

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The msCAN12 sets the flag after the message has been sent successfully. The flag is also set by the msCAN12 when the transmission request was aborted successfully due to a pending abort request. See **16.12.8 msCAN12 Transmitter Control Register**. If not masked, a transmit interrupt is pending while this flag is set.

Clearing a TXEx flag also clears the corresponding ABTAKx flag. When a TXEx flag is set, the corresponding ABTRQx bit is cleared. See 16.12.8 msCAN12 Transmitter Control Register.

0 = The associated message buffer is full (loaded with a message due for transmission).
1 = The associated message buffer is empty (not scheduled).

NOTE

To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.The CTFLG register is held in the reset state if the SFTRES bit CMCR0 is set.

Downloaded from Arrow.com.

16.12.8 msCAN12 Transmitter Control Register

Address: $0107

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-23. msCAN12 Transmitter Control Register (CTCR)

ABTRQ2–ABTRQ0 — Abort Request Bits

The CPU sets an ABTRQx bit to request that a scheduled message buffer (TXEx = 0) shall be aborted. The msCAN12 grants the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted, the associated TXE and the abort acknowledge flag (ABTAK) (see **16.12.7 msCAN12 Transmitter Flag Register**) are set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

0 = No abort request
1 = Abort request pending

NOTE

The software must not clear one or more of the TXE flags in CTFGL and simultaneously set the respective ABTRQ bit(s).

TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable Bits

0 = No interrupt will be generated from this event.
1 = A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.

NOTE

The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.

16.12.9 msCAN12 Identifier Acceptance Control Register

Address: $0108

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-24. msCAN12 Identifier Acceptance Control Register (CIDAC)

Downloaded from Arrow.com.

IDAM1 and IDAM0— Identifier Acceptance Mode Flags

The CPU sets these flags to define the identifier acceptance filter organization. See **16.4 Identifier Acceptance Filter**. Table 16-8 summarizes the different settings. In filter closed mode, no messages are accepted so that the foreground buffer is never reloaded.

Table 16-9. Identifier Acceptance Mode Settings

IDAM1	IDAM0	Identifier Acceptance Mode
0	0	Two 32-bit acceptance filters
0	1	Four 16-bit acceptance filters
1	0	Eight 8-bit acceptance filters
1	1	Filter closed

IDHIT2–IDHIT0— Identifier Acceptance Hit Indicator Flags

The msCAN12 sets these flags to indicate an identifier acceptance hit. See **16.4 Identifier Acceptance Filter**. Table 16-8 summarizes the different settings.

Table 16-10. Identifier Acceptance Hit Indication

IDHIT2	IDHIT1	IDHIT0	Identifier Acceptance Hit
0	0	0	Filter 0 hit
0	0	1	Filter 1 hit
0	1	0	Filter 2 hit
0	1	1	Filter 3 hit
1	0	0	Filter 4 hit
1	0	1	Filter 5 hit
1	1	0	Filter 6 hit
1	1	1	Filter 7 hit

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

NOTE

The CIDAC register can be written only if the SFTRES bit in CMCR0 is set.

16.12.10 msCAN12 Receive Error Counter

Address: $010E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-25. msCAN12 Receive Error Counter (CRXERR)

This register reflects the status of the msCAN12 receive error counter. The register is read only.

Downloaded from Arrow.com.

16.12.11 msCAN12 Transmit Error Counter

Address: $010F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-26. msCAN12 Transmit Error Counter (CTXERR)

This register reflects the status of the msCAN12 transmit error counter. The register is read only.

NOTE

Both error counters may be read only when in sleep or soft-reset mode.

16.12.12 msCAN12 Identifier Acceptance Registers

On reception, each message is written into the background receive buffer. The CPU is only signalled to read the message if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the msCAN12 are applied on the IDR0 to IDR3 registers of incoming messages in a bit-by-bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/CIDMR1 and CIDAR0/CIDAR1) are applied.

Address: $0110

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

Address: $0111

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

Address: $0112

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

Address: $0113

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
Reset:	Unaffected by reset							

Figure 16-27. First Bank msCAN12 Identifier Acceptance Registers (CIDAR0–CIDAR3)

Downloaded from Arrow.com.

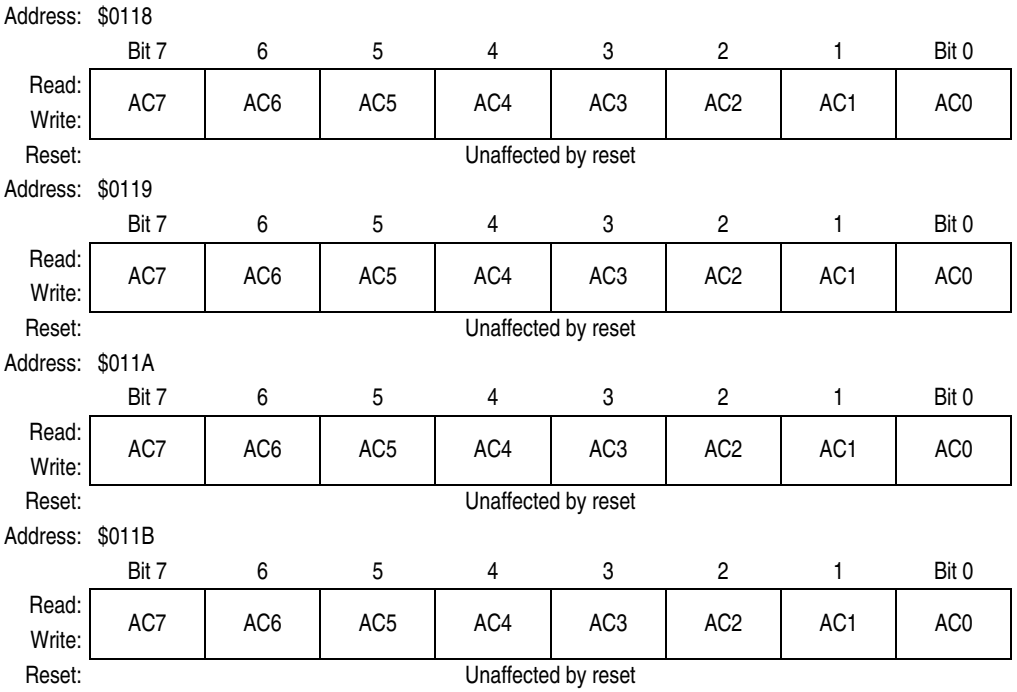


Figure 16-28. Second Bank msCAN12 Identifier Acceptance Registers (CIDAR4–CIDAR7)

AC7–AC0 — Acceptance Code Bits

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE

The CIDAR0-CIDAR7 registers can be written only if the SFTRES bit in CMCR0 is set.

16.12.13 msCAN12 Identifier Mask Registers

The identifier mask register specifies which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. To receive standard identifiers in 32-bit filter mode, the last three bits (AM2–AM0) in the mask registers CIDMR1 and CIDMR5 must be programmed to don't care. To receive standard identifiers in 16 bit filter mode the last three bits (AM2–AM0) in the mask registers CIDMR1, CIDMR3, CIDMR5, and CIDMR7 must be programmed to don't care.

Downloaded from Arrow.com.

Address: $0114	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $0115	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $0116	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $0117	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Figure 16-29. First Bank msCAN12 Identifier Mask Registers (CIDMR0–CIDMR3)

Address: $011C	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $011D	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $011E	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Address: $011F	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Figure 16-30. Second Bank msCAN12 Identifier Mask Registers (CIDMR4–CIDMR7)

Downloaded from Arrow.com.

AM7–AM0 — Acceptance Mask Bits

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.
0 = Match corresponding acceptance code register and identifier bits.

NOTE

The CIDMR0–CIDMR7 registers can be written only if the SFTRES bit in CMCR0 is set.

16.12.14 msCAN12 Port CAN Control Register

Address: $013D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	PUECAN	RDPCAN
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-31. msCAN12 Port CAN Control Register (PCTLCAN)

These bits control pins 7–2 of port CAN control register. Pins 1 and 0 are reserved for the RxCAN (input only) and TxCAN (output only) pins.

PUECAN — Pullup Enable Port CAN Bit

0 = Pull mode disabled for port CAN
1 = Pull mode enabled for port CAN

RDPCAN — Reduced Drive Port CAN

0 = Reduced drive disabled for port CAN
1 = Reduced drive enabled for port CAN

16.12.15 msCAN12 Port CAN Data Register

Address: $013E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PCAN7	PCAN6	PCAN5	PCAN4	PCAN2	PCAN2	TxCAN	RxCAN
Write:								
Reset:	Unaffected by reset							

= Unimplemented

Figure 16-32. msCAN12 Port CAN Data Register (PORTCAN)

PCAN7–PCAN2 — Port CAN Data Bits

Writing to PCANx stores the bit value in an internal bit memory. This value is driven to the respective pin only if DDRCANx = 1.

Reading PCANx returns:

- Value of the internal bit memory driven to the pin, if DDRCANx = 1
- Value of the respective pin, if DDRCANx = 0

Reading bits 1 and 0 returns the value of the TxCAN and RxCAN pins, respectively.

Downloaded from Arrow.com.

16.12.16 msCAN12 Port CAN Data Direction Register

Address: $013F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DDRCAN7	DDRCAN6	DDRCAN5	DDRCAN4	DDRCAN3	DDRCAN2	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-33. msCAN12 Port CAN Data Direction Register (DDRCAN)

DDRCAN7–DDRCAN2 — Data Direction Port CAN Bits

0 = Respective input/output (I/O) pin is configured for input.
1 = Respective I/O pin is configured for output.

Downloaded from Arrow.com.

Chapter 17
Analog-to-Digital Converter (ATD)

17.1 Introduction

The ATD is an 8-channel, 10-bit, multiplexed-input, successive-approximation, analog-to-digital converter, accurate to ± 2 least significant bit (LSB). It does not require external sample and hold circuits because of the type of charge redistribution technique used. The ATD converter timing is synchronized to the system P clock. The ATD module consists of a 16-word (32-byte) memory-mapped control register block used for control, testing, and configuration.

17.2 Functional Description

A single conversion sequence consists of four or eight conversions, depending on the state of the select 8-channel mode (S8CM) bit when ATDCTL5 is written. There are eight basic conversion modes. In the non-scan modes, the SCF bit is set after the sequence of four or eight conversions has been performed and the ATD module halts. In the scan modes, the SCF bit is set after the first sequence of four or eight conversions has been performed, and the ATD module continues to restart the sequence. In both modes, the CCF bit associated with each register is set when that register is loaded with the appropriate conversion result. That flag is cleared automatically when that result register is read. The conversions are started by writing to the control registers.

Downloaded from Arrow.com.

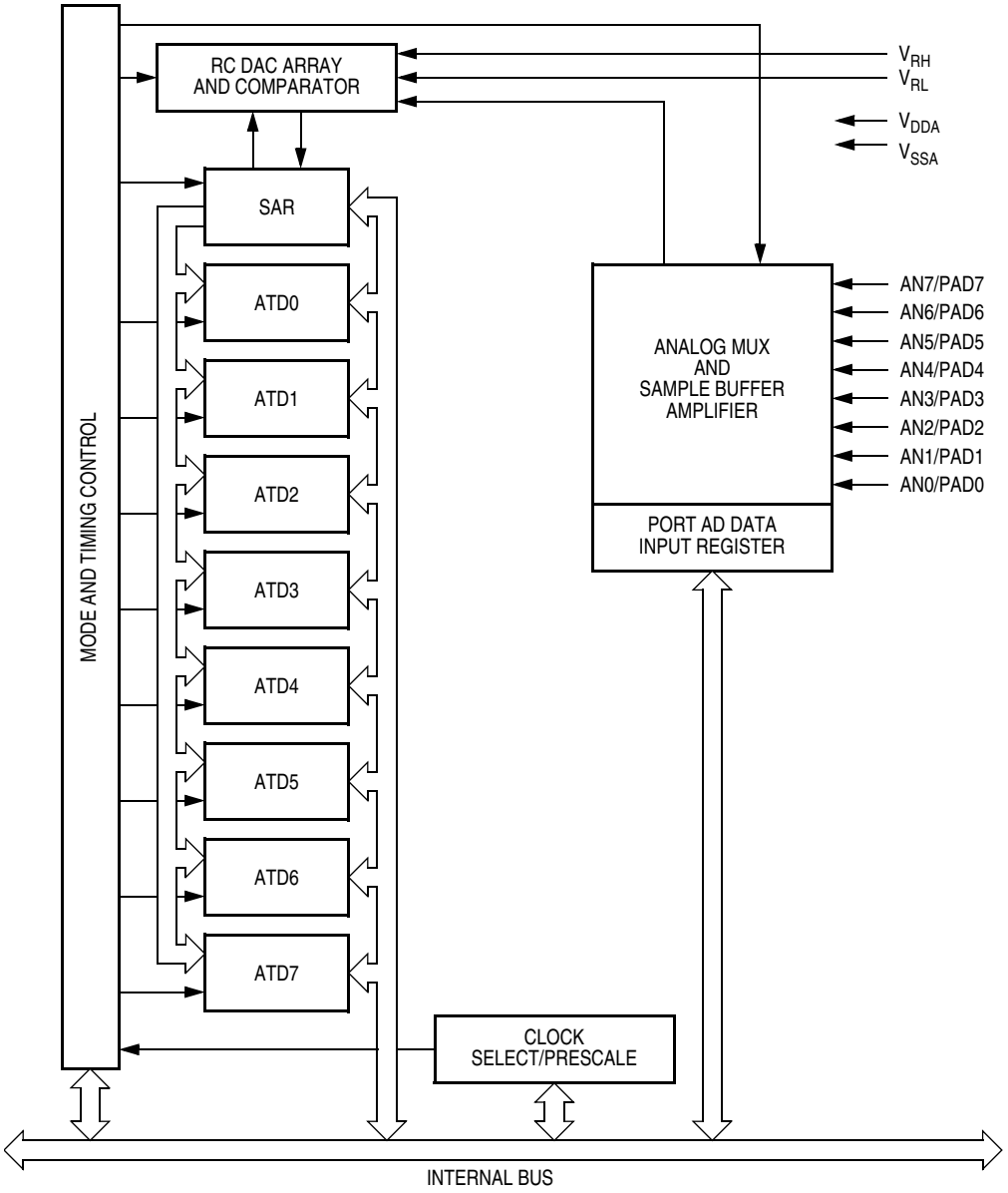


Figure 17-1. ATD Block Diagram

Downloaded from Arrow.com.

17.3 ATD Registers

This section describes the ATD registers.

17.3.1 ATD Control Register 0

Address: $0060

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 17-2. ATD Control Register 0 (ATDCTL0)

Read: Anytime

Write: Anytime

NOTE

Writes to this register abort the current conversion sequence.

17.3.2 ATD Control Register 1

Address: $0061

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	0	0	0	0	0	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 17-3. Reserved (ATDCTL1)

Read: Special mode only

Write: Has no meaning

17.3.3 ATD Control Register 2

Address: $0062

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADPU	AFFC	AWAI	0	0	0	ASCIE	ASCIF
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-4. ATD Control Register 2 (ATDCTL2)

Read: Anytime

Write: Anytime except ASCIF bit, which cannot be written

The ATD control register 2 (ATDCTL2) is used to select the power-up mode, interrupt control, and freeze control.

NOTE

Writing to this register aborts the current conversion sequence.

Downloaded from Arrow.com.

ADPU — ATD Disable Bit

Software can disable the clock signal to the ATD and power down the analog circuits to reduce power consumption. When reset to 0, the ADPU bit aborts any conversion sequence in progress. Because the bias currents to the analog circuits are turned off, the ATD requires a period of recovery time to stabilize the analog circuits after setting the ADPU bit.

0 = Disables the ATD, including the analog section for reduction in power consumption
1 = Allows the ATD to function normally

AFFC — ATD Fast Flag Clear Bit

0 = ATD flag clearing operates normally (read the status register before reading the result register to clear the associated CCF bit).
1 = Changes all ATD conversion complete flags to a fast clear sequence. Any access to a result register (ATD0–ATD7) will cause the associated CCF flag to clear automatically if it was set at the time.

AWAI — ATD Stop in Wait Mode Bit

0 = ATD continues to run when the MCU is in wait mode.
1 = ATD stops to save power when the MCU is in wait mode.

ASCIE — ATD Sequence Complete Interrupt Enable Bit

0 = Disables ATD interrupt
1 = Enables ATD interrupt on sequence complete

ASCIF — ATD Sequence Complete Interrupt Flag

Cannot be written in any mode.

0 = No ATD interrupt occurred
1 = ATD sequence complete

17.3.4 ADT Control Register 3

Address: $0063

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	FRZ1	FRZ0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-5. ATD Control Register 3 (ATDCTL3)

Read: Anytime

Write: Anytime

The ATD control register 3 (ATDCTL3) is used to select the power-up mode, interrupt control, and freeze control.

NOTE

Writing to this register aborts any current conversion sequence and suspends module operation at breakpoint.

Downloaded from Arrow.com.

FRZ1 and FRZ0 — Background Debug (Freeze) Enable Bits

When debugging an application, it is useful in many cases to have the ATD pause when a breakpoint is encountered. These two bits determine how the ATD will respond when background debug mode becomes active. See Table 17-1.

Table 17-1. ATD Response to Background Debug Enable

FRZ1	FRZ0	ATD Response
0	0	Continue conversions in active background mode
0	1	Reserved
1	0	Finish current conversion, then freeze
1	1	Freeze when BDM is active

17.3.5 ATD Control Register 4

Address: $0064

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	S10BM	SMP1	SMP0	PRS4	PRS3	PRS2	PRS1	PRS0
Reset:	0	0	0	0	0	0	0	1

Figure 17-6. ATD Control Register 4 (ATDCTL4)

The ATD control register 4 (ATDCTL4) selects the clock source and sets up the prescaler. Writes to the ATD control registers initiate a new conversion sequence. If a write occurs while a conversion is in progress, the conversion is aborted and ATD activity halts until a write to ATDCTL5 occurs.

S10BM — ATD 10-Bit Mode Control Bit

0 = 8-bit operation
1 = 10-bit operation

SMP1 and SMP0 — Select Sample Time Bits

These bits are used to select one of four sample times after the buffered sample and transfer has occurred. See Table 17-2.

Table 17-2. Final Sample Time Selection

SMP1	SMP0	Final Sample Time	Total 8-Bit Conversion Time	Total 10-Bit Conversion Time
0	0	2 ATD clock periods	18 ATD clock periods	20 ATD clock periods
0	1	4 ATD clock periods	20 ATD clock periods	22 ATD clock periods
1	0	8 ATD clock periods	24 ATD clock periods	26 ATD clock periods
1	1	16 ATD clock periods	32 ATD clock periods	34 ATD clock periods

PRS4–PRS0 — Select Divide-By Factor for ATD P-Clock Prescaler Bits

The binary value written to these bits (1 to 31) selects the divide-by factor for the modulo counter-based prescaler. The P clock is divided by this value plus one, and then fed into a divide-by-two circuit to generate the ATD module clock. The divide-by-two circuit ensures symmetry of the output clock signal.

Downloaded from Arrow.com.

Clearing these bits causes the prescale value to default to 1 which results in a divide-by-two prescale factor. This signal is then fed into the divide-by-two logic. The reset state divides the P clock by a total of four and is appropriate for nominal operation at a bus rate of between 2 MHz and 8 MHz. Table 17-3 shows the divide-by operation and the appropriate range of system clock frequencies.

Table 17-3. Clock Prescaler Values

Prescale Value	Total Divisor	Max P Clock(1)	Min P Clock(2)
00000	2	4 MHz	1 MHz
00001	4	8 MHz	2 MHz
00010	6	8 MHz	3 MHz
00011	8	8 MHz	4 MHz
00100	10	8 MHz	5 MHz
00101	12	8 MHz	6 MHz
00110	14	8 MHz	7 MHz
00111	16	8 MHz	8 MHz
01xxx	Do not use		
1xxxx			

1. Maximum conversion frequency is 2 MHz. Maximum P clock divisor value becomes maximum conversion rate that can be used on this ATD module.
2. Minimum conversion frequency is 500 kHz. Minimum P clock divisor value becomes minimum conversion rate that this ATD can perform.

17.3.6 ATD Control Register 5

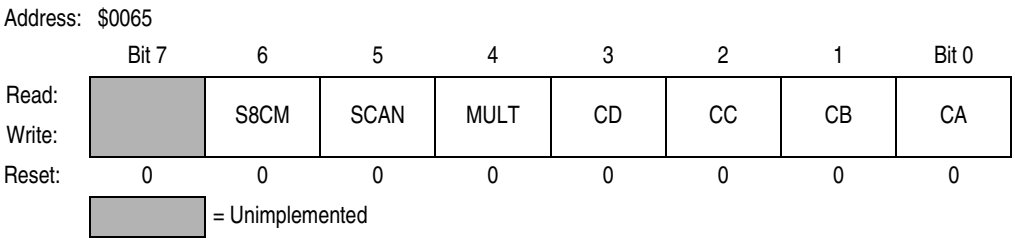


Figure 17-7. ATD Control Register 5 (ATDCTL5)

Read: Anytime

Write: Anytime

The ATD control register 5 is used to select the conversion modes, the conversion channel(s), and initiate conversions.

A write to ATDCTL5 initiates a new conversion sequence. If a conversion sequence is in progress when a write occurs, that sequence is aborted and the SCF and CCF bits are reset.

S8CM — Select 8 Channel Mode Bit

0 = Conversion sequence consists of four conversions.
1 = Conversion sequence consists of eight conversions.

Downloaded from Arrow.com.

SCAN — Enable Continuous Channel Scan Bit

When a conversion sequence is initiated by a write to the ATDCTL register, the user has a choice of performing a sequence of four (or eight, depending on the S8CM bit) conversions or continuously performing four (or eight) conversion sequences.

0 = Single conversion sequence
1 = Continuous conversion sequences (scan mode)

MULT — Enable Multichannel Conversion Bit

0 = ATD sequencer runs all four or eight conversions on a **single** input channel selected via the CD, CC, CB, and CA bits.
1 = ATD sequencer runs each of the four or eight conversions on **sequential** channels in a specific group. Refer to Table 17-4.

CD, CC, CB, and CA — Channel Select for Conversion Bits

Table 17-4. Multichannel Mode Result Register Assignment

S8CM	CD	CC	CB	CA	Channel Signal	Result in ADRx if MULT = 1
0	0	0	0	0	AN0	ADR0
			0	1	AN1	ADR1
			1	0	AN2	ADR2
			1	1	AN3	ADR3
0	0	1	0	0	AN4	ADR0
			0	1	AN5	ADR1
			1	0	AN6	ADR2
			1	1	AN7	ADR3
0	1	0	0	0	Reserved	ADR0
			0	1	Reserved	ADR1
			1	0	Reserved	ADR2
			1	1	Reserved	ADR3
0	1	1	0	0	V_{RH}	ADR0
			0	1	V_{RL}	ADR1
			1	0	$(V_{RH} + V_{RL})/2$	ADR2
			1	1	Test/reserved	ADR3
1	0	0	0	0	AN0	ADR0
		0	0	1	AN1	ADR1
		0	1	0	AN2	ADR2
		0	1	1	AN3	ADR3
		1	0	0	AN4	ADR4
		1	0	1	AN5	ADR5
		1	1	0	AN6	ADR6
		1	1	1	AN7	ADR7

Downloaded from Arrow.com.

Table 17-4. Multichannel Mode Result Register Assignment (Continued) (Continued)

S8CM	CD	CC	CB	CA	Channel Signal	Result in ADRx if MULT = 1
1	1	0	0	0	Reserved	ADR0
		0	0	1	Reserved	ADR1
		0	1	0	Reserved	ADR2
		0	1	1	Reserved	ADR3
		1	0	0	V_{RH}	ADR4
		1	0	1	V_{RL}	ADR5
		1	1	0	$(V_{RH} + V_{RL})/2$	ADR6
		1	1	1	Test/reserved	ADR7

Shaded bits are “don’t care” if MULT = 1 and the entire block of four or eight channels makes up a conversion sequence. When MULT = 0, all four bits (CD, CC, CB, and CA) must be specified and a conversion sequence consists of four or eight consecutive conversions of the single specified channel.

17.3.7 ATD Status Registers

Address: $0066

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	SCF	0	0	0	0	CC2	CC1	CC0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 17-8

Address: $0067

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CCF7	CCF6	CCF5	CCF4	CCF3	CCF2	CCF1	CCF0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-9. ATD Status Register (ATDSTAT)

Read: Normally anytime

Write: In special mode, the SCF bit and the CCF bits may also be written.

The ATD status registers contain the flags indicating the completion of ATD conversions.

SCF — Sequence Complete Flag

This bit is set at the end of the conversion sequence when in the single conversion sequence mode (SCAN = 0 in ATDCTL5) and is set at the end of the first conversion sequence when in the continuous conversion mode (SCAN = 1 in ATDCTL5). When AFFC = 0, SCF is cleared when a write is performed to ATDCTL5 to initiate a new conversion sequence. When AFFC = 1, SCF is cleared after the first result register is read.

Downloaded from Arrow.com.

CC2–CC0 — Conversion Counter Bits for Current 4 or 8 Conversions

This 3-bit value reflects the contents of the conversion counter pointer in a four or eight count sequence. This value also reflects which result register is written next, indicating which channel is currently being converted.

CCF7–CCF0 — Conversion Complete Flags

Each CCF bit is associated with an individual ATD result register. For each register, this bit is set at the end of conversion for the associated ATD channel and remains set until that ATD result register is read. It is cleared at that time if AFFC bit is set, regardless of whether a status register read has been performed (for example, a status register read is not a pre-qualifier for the clearing mechanism when AFFC = 1). Otherwise, the status register must be read to clear the flag.

17.3.8 ATD Test Registers

Address: $0068

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
Reset:	0	0	0	0	0	0	0	0

Figure 17-10

Address: $0069

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SAR1	SAR0	RST	TSTOUT	TST3	TST2	TST1	TST0
Reset:	0	0	0	0	0	0	0	0

Figure 17-11. ATD Test Register (ATDSTAT)

Read: Special modes only

Write: Special modes only

The test registers control various special modes which are used during manufacturing. In the normal modes, reads of the test register return 0 and writes have no effect.

SAR9–SAR0 — SAR Data Bits

Reads of this byte return the current value in the SAR. Writes to this byte change the SAR to the value written. Bits SAR9–SAR0 reflect the 10 SAR bits used during the resolution process for an 10-bit result.

RST — Module Reset Bit

When set, this bit causes all registers and activity in the module to assume the same state as out of power-on reset (except for ADPU bit in ATDCTL2, which remains set, allowing the ATD module to remain enabled).

TSTOUT — Multiplex Output of TST3–TST0 (Factory Use)

TST3–TST0 — Test Bits 3 to 0 (Reserved)

Selects one of 16 reserved factory testing modes

Downloaded from Arrow.com.

17.3.9 Port AD Data Input Register

Address: $006F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PAD7	PAD6	PAD5	PAD4	PAD3	PAD2	PAD1	PAD0
Write:								
Reset:	After reset, reflect the state of the inputs pins							

= Unimplemented

Figure 17-12. Port AD Data Input Register (PORTAD)

Read: Anytime

Write: Has no meaning or effect

PAD7–PAD0 — Port AD Data Input Bits

After reset, these bits reflect the state of the input pins.

PAD7–PAD0 may be used for general-purpose digital input. When the software reads PORTAD, it obtains the digital levels that appear on the corresponding port AD pins. Pins with signals not meeting V_{IL} or V_{IH} specifications will have an indeterminate value.

17.3.10 ATD Result Registers

ADRx0H: $0070
ADRx1H: $0072
ADRx2H: $0074
ADRx3H: $0076
ADRx4H: $0078
ADRx5H: $007A
ADRx6H: $007C
ADRx7H: $007E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Write:								
Reset:	Undefined							

= Unimplemented

Figure 17-13. ATD Result Registers High

ADRx0L: $0071
ADRx1L: $0073
ADRx2L: $0075
ADRx3L: $0077
ADRx4L: $0079
ADRx5L: $007B
ADRx6L: $007D
ADRx7L: $007F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Write:								
Reset:	Undefined							

= Unimplemented

Figure 17-14. ATD Result Registers Low

Downloaded from Arrow.com.

Read: Anytime

Write: Has no meaning or effect

ADRxH[15:8]–ADRxH[7:0] — ATD Conversion Result Bits

The reset condition for these registers is undefined.

These bits contain the left-justified, unsigned result from the ATD conversion. The channel from which this result was obtained is dependant on the conversion mode selected. These registers are always read-only in normal mode.

17.4 ATD Mode Operation

Stop

Causes all clocks to halt (if the S bit in the CCR is 0). The system is placed in a minimum-power standby mode. This aborts any conversion sequence in progress. During STOP recovery, the ATD must delay for the STOP recovery time (t_{SR}) before initiating a new ATD conversion sequence.

Wait

ATD conversion continues unless the AWAI bit in ATDCTL2 register is set.

BDM

Debug options available as set in register ATDCTL3.

User

ATD continues running unless ADPU is cleared.

ADPU

ATD operations are stopped if ADPU = 0, but registers are accessible.

17.5 Using the ATD to Measure a Potentiometer Signal

This exercise allows the student to utilize the ATD on the HC12 to measure a potentiometer signal output routed from the UDLP1 board to the HC12 ATD pin PAD6. First the ATDCTL registers are initialized. A delay loop of 100 µs is then executed. The resolution is set up followed by a conversion set up on channel 6. After waiting for the status bit to set, the result goes to the D accumulator. If the program is working properly, a different value should be found in the D accumulator as the left potentiometer is varied for each execution of the program.

17.5.1 Equipment

For this exercise, use the M68HC912B32EVB emulation board.

17.5.2 Code Listing

NOTE

A comment line is deliminted by a semi-colon. If there is no code before comment, an ";" must be placed in the first column to avoid assembly errors.

Downloaded from Arrow.com.

```
; ---------------------------------------------------------------------
;           MAIN PROGRAM
; ---------------------------------------------------------------------

          ORG      $7000                ; 16K On-Board RAM, User code data area,
;                                        start main program at $7000

MAIN:
          BSR      INIT                 ; Branch to INIT subroutine to Initialize ATD
          BSR      CONVERT              ; Branch to CONVERT Subroutine for conversion
DONE:     BRA      DONE                 ; Branch to Self, Convenient place for
breakpoint

; ---------------------------------------------
;           Subroutine INIT: Initialize ATD    ;
; ---------------------------------------------

INIT:
          LDAA     #$80                 ; Allow ATD to function normally,
          STAA     ATDCTL2              ; ATD Flags clear normally & disable
interrupts

          BSR      DELAY                ; Delay (100 uS) for WAIT delay time.

          LDAA     #$00                 ; Select continue conversion in BGND Mode
          STAA     ATDCTL3              ; Ignore FREEZE in ATDCTL3

          LDAA     #$01                 ; Select Final Sample time = 2 A/D clocks
          STAA     ATDCTL4              ; Prescaler = Div by 4 (PRS4:0 = 1)

          RTS                           ; Return from subroutine

; ---------------------------------------------
;              Subroutine CONVERT:             ;
; ---------------------------------------------
; Set-up ATD, make single conversion and store the result to a memory location.
; Configure and start A/D conversion
; Analog Input Signal: On PORT AD6
; Convert: using single channel, non-continuous
; The result will be located in ADR2H

CONVERT:
          LDAA     #$06                 ; Initializes ATD SCAN=0,MULT=0, PAD6,
;                                        ; Write Clears Flag
          STAA     ATDCTL5              ; 4 conversions on a Single Conversion
;                                        ; sequence,

WTCONV:   BRCLR    ATDSTATH,#$80,WTCONV ; Wait for Sequence Complete Flag

          LDD      ADR2H                ; Loads conversion result(ADR2H)
;                                        ; into Accumulator

          BRA      CONVERT              ; Continuously updates results

          RTS                           ; Return from subroutine

;* ------------------------------
;*    Subroutine DELAY 100 uS    *
;* ------------------------------
; Delay Required for ATD converter to Stabilize (100 uSec)

          LDAA     #$C8                 ; Load Accumulator with "100 uSec delay value"
DELAY:    DECA                          ; Decrement ACC
          BNE      DELAY                ; Branch if not equal to Zero
          RTS                           ; Return from subroutine

          END                           ; End of program
```


Downloaded from Arrow.com.

Chapter 18
Development Support

18.1 Introduction

Development support involves complex interactions between MCU resources and external development systems. This section concerns instruction queue and queue tracking signals, background debug mode, breakpoints, and instruction tagging.

18.2 Instruction Queue

It is possible to monitor CPU activity on a cycle-by-cycle basis for debugging.The CPU12 instruction queue provides at least three bytes of program information to the CPU when instruction execution begins. The CPU12 always completely finishes executing an instruction before beginning to execute the next instruction. Status signals IPIPE1 and IPIPE0 provide information about data movement in the queue and indicate when the CPU begins to execute instructions. Information available on the IPIPE1 and IPIPE0 pins is time multiplexed. External circuitry can latch data movement information on rising edges of the E-clock signal; execution start information can be latched on falling edges. Table 18-1 shows the meaning of data on the pins.

Table 18-1. IPIPE Decoding

Data Movement — IPIPE[1:0] Captured at Rising Edge of E Clock(1)		
IPIPE[1:0]	**Mnemonic**	**Meaning**
0:0	—	No movement
0:1	LAT	Latch data from bus
1:0	ALD	Advance queue and load from bus
1:1	ALL	Advance queue and load from latch
Execution Start — IPIPE[1:0] Captured at Falling Edge of E Clock(2)		
IPIPE[1:0]	**Mnemonic**	**Meaning**
0:0	—	No start
0:1	INT	Start interrupt sequence
1:0	SEV	Start even instruction
1:1	SOD	Start odd instruction

1. Refers to data that was on the bus at the previous E falling edge.
2. Refers to bus cycle starting at this E falling edge.

Downloaded from Arrow.com.

Program information is fetched a few cycles before it is used by the CPU. To monitor cycle-by-cycle CPU activity, it is necessary to externally reconstruct what is happening in the instruction queue. Internally, the MCU only needs to buffer the data from program fetches. For system debug it is necessary to keep the data and its associated address in the reconstructed instruction queue. The raw signals required for reconstruction of the queue are ADDR, DATA, R/$\overline{W}$, ECLK, and status signals IPIPE1 and IPIPE0.

The instruction queue consists of two 16-bit queue stages and a holding latch on the input of the first stage. To advance the queue means to move the word in the first stage to the second stage and move the word from either the holding latch or the data bus input buffer into the first stage. To start even (or odd) instruction means to execute the opcode in the high-order (or low-order) byte of the second stage of the instruction queue.

18.3 Background Debug Mode (BDM)

Background debug mode (BDM) is used for system development, in-circuit testing, field testing, and programming. BDM is implemented in on-chip hardware and provides a full set of debug options.

Because BDM control logic does not reside in the CPU, BDM hardware commands can be executed while the CPU is operating normally. The control logic generally uses CPU dead cycles to execute these commands, but can steal cycles from the CPU when necessary. Other BDM commands are firmware based and require the CPU to be in active background mode for execution. While BDM is active, the CPU executes a firmware program located in a small on-chip ROM that is available in the standard 64-Kbyte memory map only while BDM is active.

The BDM control logic communicates serially with an external host development system, via the BKGD pin. This single-wire approach minimizes the number of pins needed for development support.

18.3.1 BDM Serial Interface

The BDM serial interface requires the external controller to generate a falling edge on the BKGD pin to indicate the start of each bit time. The external controller provides this falling edge whether data is transmitted or received.

BKGD is a pseudo-open-drain pin that can be driven either by an external controller or by the MCU. Data is transferred MSB first at 16 E-clock cycles per bit (nominal speed). The interface times out if 512 E-clock cycles occur between falling edges from the host. The hardware clears the command register when this timeout occurs.

The BKGD pin can receive a high or low level or transmit a high or low level. Figure 18-1, Figure 18-2, and Figure 18-3 show timing for each of these cases. Interface timing is synchronous to MCU clocks but asynchronous to the external host. The internal clock signal is shown for reference in counting cycles.

Downloaded from Arrow.com.

Figure 18-1 shows an external host transmitting a logic 1 or 0 to the BKGD pin of a target M68HC12 MCU. The host is asynchronous to the target so there is a 0-to-1 cycle delay from the host-generated falling edge to where the target perceives the beginning of the bit time. Ten target E cycles later, the target senses the bit level on the BKGD pin. Typically, the host actively drives the pseudo-open-drain BKGD pin during host-to-target transmissions to speed up rising edges. Since the target does not drive the BKGD pin during this period, there is no need to treat the line as an open-drain signal during host-to-target transmissions.

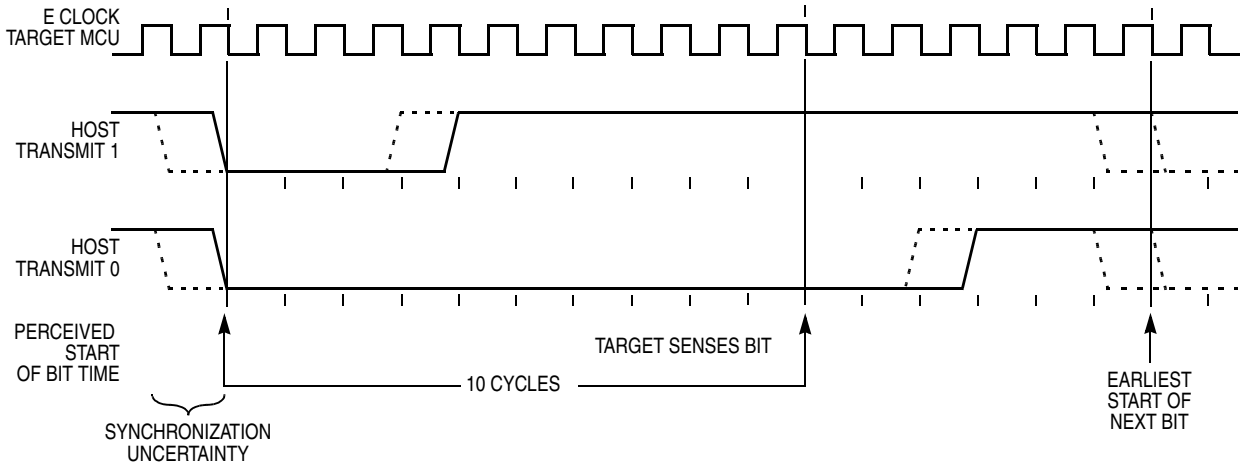


Figure 18-1. BDM Host to Target Serial Bit Timing

Figure 18-2 shows the host receiving a logic 1 from the target MCU. Since the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the perceived start of the bit time in the target MCU. The host holds the BKGD pin low long enough for the target to recognize it (at least two target E cycles). The host must release the low drive before the target MCU drives a brief active-high speed-up pulse seven cycles after the perceived start of the bit time. The host should sample the bit level about 10 cycles after it started the bit time.

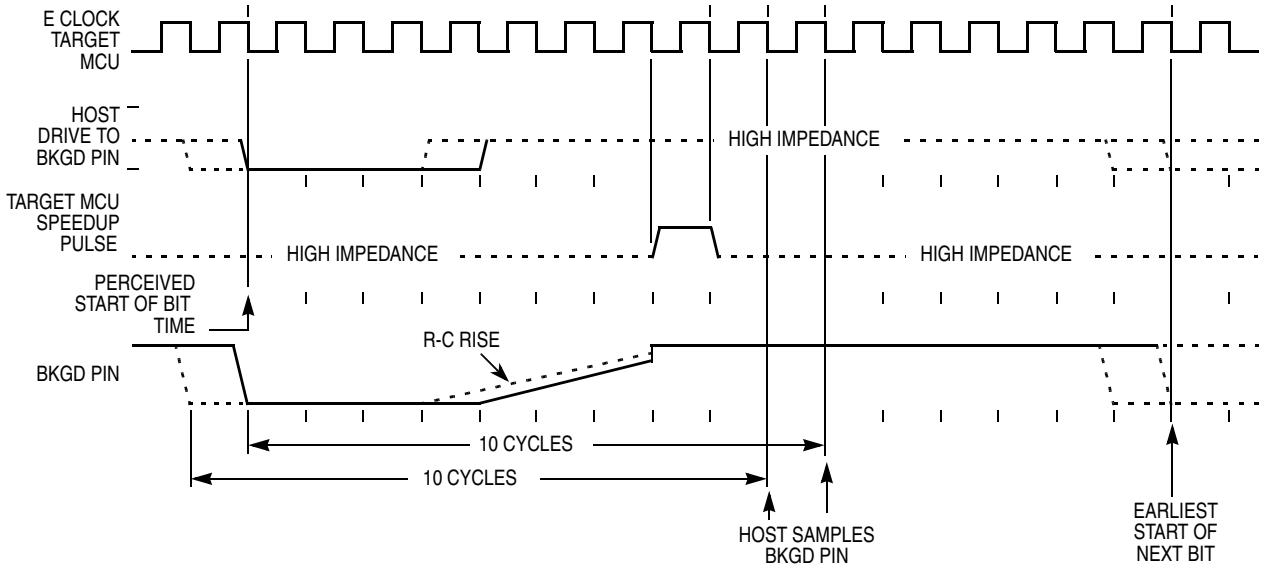


Figure 18-2. BDM Target to Host Serial Bit Timing (Logic 1)

Downloaded from Arrow.com.

Figure 18-3 shows the host receiving a logic 0 from the target MCU. Since the host is asynchronous to the target MCU, there is a 0-to-1 cycle delay from the host-generated falling edge on BKGD to the start of the bit time as perceived by the target MCU. The host initiates the bit time but the target MCU finishes it. Since the target wants the host to receive a logic 0, it drives the BKGD pin low for 13 E-clock cycles, then briefly drives it high to speed up the rising edge. The host samples the bit level about 10 cycles after starting the bit time.

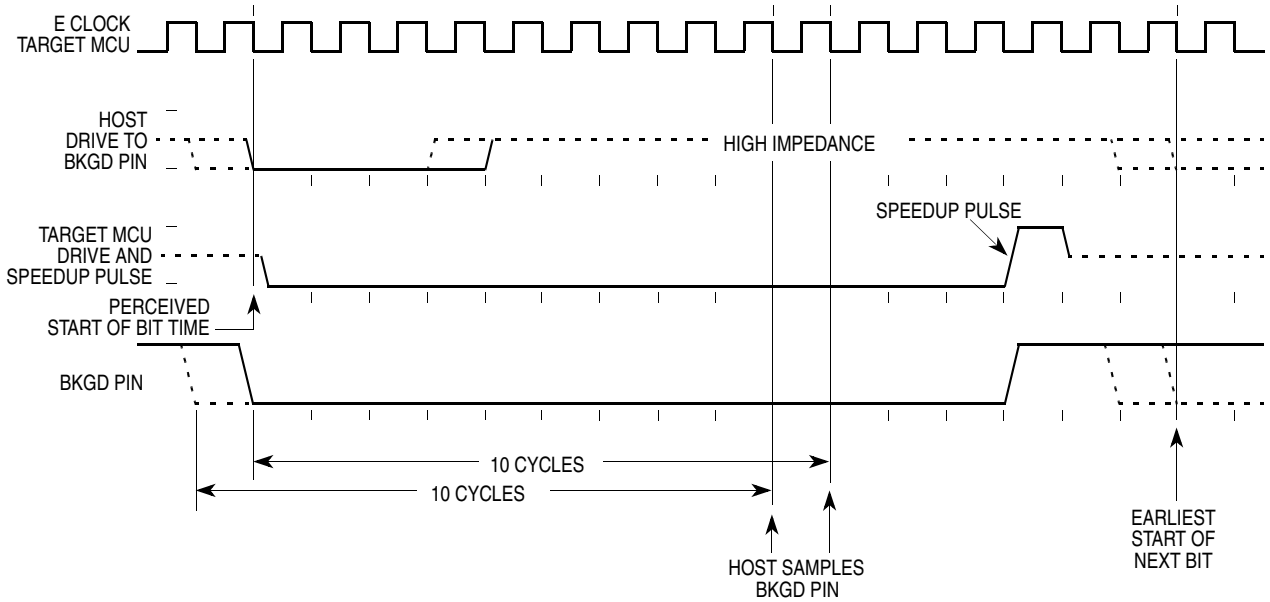


Figure 18-3. BDM Target to Host Serial Bit Timing (Logic 0)

18.3.2 Enabling BDM Firmware Commands

BDM is available in all operating modes, but must be made active before firmware commands can be executed. BDM is enabled by setting the ENBDM bit in the BDM STATUS register via the single-wire interface (using a hardware command; WRITE_BD_BYTE at $FF01). BDM must then be activated to map BDM registers and ROM to addresses $FF00 to $FFFF and to put the MCU in active background mode.

After the firmware is enabled, BDM can be activated by the hardware BACKGROUND command, by the BDM tagging mechanism, or by the CPU BGND instruction. An attempt to activate BDM before firmware has been enabled causes the MCU to resume normal instruction execution after a brief delay.

BDM becomes active at the next instruction boundary following execution of the BDM BACKGROUND command, but tags activate BDM before a tagged instruction is executed.

In special single-chip mode, background operation is enabled and active immediately out of reset. This active case replaces the M68HC11 boot function and allows programming a system with blank memory.

While BDM is active, a set of BDM control registers is mapped to addresses $FF00 to $FF06. The BDM control logic uses these registers which can be read anytime by BDM logic, not user programs. Refer to 18.3.4 BDM Registers for detailed descriptions.

Some on-chip peripherals have a BDM control bit which allows suspending the peripheral function during BDM. For example, if the timer control is enabled, the timer counter is stopped while in BDM. Once normal program flow is continued, the timer counter is re-enabled to simulate real-time operations.

Downloaded from Arrow.com.

18.3.3 BDM Commands

All BDM command opcodes are eight bits long and can be followed by an address and/or data, as indicated by the instruction. These commands do not require the CPU to be in active BDM for execution.

The host controller must wait 150 cycles for a non-intrusive BDM command to execute before another command can be sent. This delay includes 128 cycles for the maximum delay for a dead cycle. For data read commands, the host must insert this delay between sending the address and attempting to read the data.

BDM logic retains control of the internal buses until a read or write is completed. If an operation can be completed in a single cycle, it does not intrude on normal CPU operation. However, if an operation requires multiple cycles, CPU clocks are frozen until the operation is complete.

The two types of BDM commands are:

- Hardware
- Firmware

Hardware commands allow target system memory to be read or written. Target system memory includes all memory that is accessible by the CPU12 including on-chip RAM, EEPROM, on-chip I/O and control registers, and external memory connected to the target HC12 MCU. Hardware commands are implemented in hardware logic and do not require the HC12 MCU to be in BDM mode for execution. The control logic watches the CPU12 buses to find a free bus cycle to execute the command so that the background access does not disturb the running application programs. If a free cycle is not found within 128 E-clock cycles, the CPU12 is momentarily frozen so the control logic can steal a cycle. Refer to Table 18-2 for commands implemented in BDM control logic.

Table 18-2. BDM Hardware Commands

Command	Opcode (Hex)	Data	Description
BACKGROUND	90	None	Enter background mode (if firmware enabled).
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
STATUS(1)	E4	FF01, 0000 0000 (out)	READ_BD_BYTE $FF01. Running user code. (BGND instruction is not allowed.)
		FF01, 1000 0000 (out)	READ_BD_BYTE $FF01. BGND instruction is allowed.
		FF01, 1100 0000 (out)	READ_BD_BYTE $FF01. Background mode active (waiting for single wire serial command).
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access) must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access) must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
ENABLE_ FIRMWARE(2)	C4	FF01, 1xxx xxxx (in)	Write byte $FF01, set the ENBDM bit. This allows execution of commands which are implemented in firmware. Typically, read STATUS, OR in the MSB, write the result back to STATUS.

Downloaded from Arrow.com.

Table 18-2. BDM Hardware Commands (Continued)

Command	Opcode (Hex)	Data	Description
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access) must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access) must be aligned access.

1. STATUS command is a specific case of the READ_BD_BYTE command.
2. ENABLE_FIRMWARE is a specific case of the WRITE_BD_BYTE command.

The second type of BDM commands are called firmware commands because they are implemented in a small ROM within the HC12 MCU. The CPU must be in background mode to execute firmware commands. The usual way to get to background mode is by the hardware command BACKGROUND. The BDM ROM is located at $FF20 to $FFFF while BDM is active. There are also seven bytes of BDM registers which are located at $FF00 to $FF06 while BDM is active. The CPU executes code from this ROM to perform the requested operation. The BDM firmware watches for serial commands and executes them as they are received. The firmware commands are shown in Table 18-3.

Table 18-3. BDM Firmware Commands

Command	Opcode (Hex)	Data	Description
READ_NEXT	62	16-bit data out	X = X + 2; Read next word pointed to by X
READ_PC	63	16-bit data out	Read program counter
READ_D	64	16-bit data out	Read D accumulator
READ_X	65	16-bit data out	Read X index register
READ_Y	66	16-bit data out	Read Y index register
READ_SP	67	16-bit data out	Read stack pointer
WRITE_NEXT	42	16-bit data in	X = X + 2; Write next word pointed to by X
WRITE_PC	43	16-bit data in	Write program counter
WRITE_D	44	16-bit data in	Write D accumulator
WRITE_X	45	16-bit data in	Write X index register
WRITE_Y	46	16-bit data in	Write Y index register
WRITE_SP	47	16-bit data in	Write stack pointer
GO	08	None	Go to user program
TRACE1	10	None	Execute one user instruction then return to BDM
TAGGO	18	None	Enable tagging and go to user program

Each of the hardware and firmware BDM commands starts with an 8-bit command code (opcode). Depending upon the commands, a 16-bit address and/or a 16-bit data word is required as indicated in the tables by the command. All the read commands output 16 bits of data despite the byte/word implication in the command name.

Downloaded from Arrow.com.

The external host should wait 150 E-clock cycles for a non-intrusive BDM command to execute before another command is sent. This delay includes 128 E-clock cycles for the maximum delay for a free cycle. For data read commands, the host must insert this delay between sending the address and attempting to read the data. In the case of a write command, the host must delay after the data portion, before sending a new command, to be sure the write has finished.

The external host should delay about 32 target E-clock cycles between a firmware read command and the data portion of these commands. This allows the BDM firmware to execute the instructions needed to get the requested data into the BDM shifter register.

The external host should delay about 32 target E-clock cycles after the data portion of firmware write commands to allow BDM firmware to complete the requested write operation before a new serial command disturbs the BDM shifter register.

The external host should delay about 64 target E-clock cycles after a TRACE1 or GO command before starting any new serial command. This delay is needed because the BDM shifter register is used as a temporary data holding register during the exit sequence to user code.

BDM logic retains control of the internal buses until a read or write is completed. If an operation can be completed in a single cycle, it does not intrude on normal CPU12 operation. However, if an operation requires multiple cycles, CPU12 clocks are frozen until the operation is complete.

18.3.4 BDM Registers

Seven BDM registers are mapped into the standard 64-Kbyte address space when BDM is active. Mapping is shown in Table 18-4.

Table 18-4. BDM Registers

Address	Register	Mnemonic
$FF00	BDM instruction register	INSTRUCTION
$FF01	BDM status register	STATUS
$FF02–$FF03	BDM shift register	SHIFTER
$FF04–$FF05	BDM address register	ADDRESS
$FF06	BDM CCR holding register	CCRSAV

The content of the instruction register is determined by the type of background command being executed. The status register indicates BDM operating conditions. The shift register contains data being received or transmitted via the serial interface. The address register is temporary storage for BDM commands. The CCR holding register preserves the content of the CPU12 condition code register while BDM is active.

The only registers of interest to users are the status register and the CCR holding register. The other BDM registers are used only by the BDM firmware to execute commands. The registers are accessed by means of the hardware READ_BD and WRITE_BD commands, but should not be written during BDM operation (except the CCRSAV register which could be written to modify the CCR value).

The instruction register is written by the BDM hardware as a result of serial data shifted in on the BKGD pin. It is readable and writable in special peripheral mode on the parallel bus. It is discussed here for two conditions: when a **hardware** command is executed and when a **firmware** command is executed.

The instruction register can be read or written in all modes. The hardware clears the instruction register if 512 E-clock cycles occur between falling edges from the host.

Downloaded from Arrow.com.

18.3.5 BDM Instruction Register

This section describes the BDM instruction register under hardware command and firmware command.

18.3.5.1 Hardware Command

Address: $FF00

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	H/F	DATA	R/W	BKGND	W/B	BD/U	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 18-4. BDM Instruction Register (INSTRUCTION)

The bits in the BDM instruction register have the following meanings when a **hardware** command is executed.

H/F — Hardware/Firmware Flag

0 = Firmware instruction
1 = Hardware instruction

DATA — Data Flag

0 = No data
1 = Data included in command

R/W — Read/Write Flag

0 = Write
1 = Read

BKGND — Hardware Request to Enter Active Background Mode

0 = Not a hardware background command
1 = Hardware background command (INSTRUCTION = $90)

W/B — Word/Byte Tansfer Flag

0 = Byte transfer
1 = Word transfer

BD/U — BDM Map/User Map Flag

Indicates whether BDM registers and ROM are mapped to addresses $FF00 to $FFFF in the standard 64-Kbyte address space. Used only by hardware read/write commands.

0 = BDM resources not in map
1 = BDM resources in map

Downloaded from Arrow.com.

18.3.5.2 Firmware Command

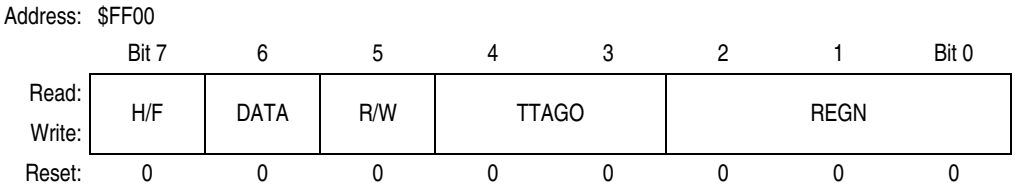


Figure 18-5. BDM Instruction Register (INSTRUCTION)

The bits in the BDM instruction register have these meanings when a **firmware** command is executed.

H/F — Hardware/Firmware Flag

0 = Firmware control logic
1 = Hardware control logic

DATA — Data Flag

0 = No data
1 = Data included in command

R/W — Read/Write Flag

0 = Write
1 = Read

TTAGO — Trace, Tag, Go Field

Table 18-5. TTAGO Decoding

TTAGO Value	Instruction
00	—
01	GO
10	TRACE1
11	TAGGO

REGN — Register/Next Field

Indicates which register is being affected by a command. In the case of a READ_NEXT or WRITE_NEXT command, index register X is pre-incremented by two and the word pointed to by X is then read or written.

Table 18-6. REGN Decoding

TTAGO Value	Instruction
000	—
001	—
010	READ/WRITE NEXT
011	PC

Downloaded from Arrow.com.

18.3.6 BDM Status Register

Address: $FF01

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ENBDM	EDMACT	ENTAG	SDV	TRACE	0	0	0
Reset:	0	0	0	0	0	0	0	0
Single-Chip Peripheral:	1	0	0	0	0	0	0	0

Figure 18-6. BDM Status Register (STATUS)

This register can be read or written by BDM commands or firmware.

ENBDM — Enable BDM Bit (permit active background debug mode)

0 = BDM cannot be made active (hardware commands still allowed).
1 = BDM can be made active to allow firmware commands.

BDMACT — Background Mode Active Status Bit

0 = BDM not active
1 = BDM active and waiting for serial commands

ENTAG — Instruction Tagging Enable Bit

Set by the TAGGO instruction and cleared when BDM is entered.

0 = Tagging not enabled, or BDM active
1 = Tagging active (BDM cannot process serial commands while tagging is active.)

SDV — Shifter Data Valid Bit

Shows that valid data is in the serial interface shift register. Used by firmware-based instructions.

0 = No valid data
1 = Valid data

TRACE

Asserted by the TRACE1 instruction

18.3.7 BDM Shifter Register

Address: $FF02

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	S15	S14	S13	S12	S11	S10	S9	S8
Reset:	0	0	0	0	0	0	0	0

Address: $FF03

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	S7	S6	S5	S4	S3	S2	S1	S0
Reset:	0	0	0	0	0	0	0	0

Figure 18-7. BDM Shifter Register (SHIFTER)

The 16-bit SHIFTER register contains data being received or transmitted via the serial interface. It is also used by the BDM firmware for temporary storage. The register can be read or written in all modes but is not normally accessed by users.

Downloaded from Arrow.com.

18.3.8 BDM Address Register

Address: $FF04

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	A15	A14	A13	A12	A11	A10	A9	A8
Reset:	0	0	0	0	0	0	0	0

Address: $FF05

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	A7	A6	A5	A4	A3	A2	A1	A0
Reset:	0	0	0	0	0	0	0	0

Figure 18-8. BDM Address Register (ADDRESS)

The 16-bit ADDRESS register is temporary storage for BDM hardware and firmware commands. The register can be read in all modes but is not normally accessed by users. It is written only by BDM hardware.

18.3.9 BDM CCR Holding Register

Address: $FF06

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	CCR7	CCR6	CCR5	CCR4	CCR3	CCR2	CCR1	CCR0
Reset:	0	0	0	0	0	0	0	0

Figure 18-9. BDM CCR Holding Register (CCRSAV)

The CCRSAV register is used to save the state of the condition code register (CCR) of the user's program when entering BDM. It is also used for temporary storage in the BDM firmware. The register is initialized by the firmware to equal the CPU CCR register.

18.4 Breakpoints

Hardware breakpoints are used to debug software on the MCU by comparing actual address and data values to predetermined data in setup registers. A successful comparison places the CPU in background debug mode (BDM) or initiates a software interrupt (SWI).

Breakpoint features designed into the MCU include:

- Mode selection for BDM or SWI generation
- Program fetch tagging for cycle of execution breakpoint
- Second address compare in dual address modes
- Range compare by disable of low byte address
- Data compare in full feature mode for non-tagged breakpoint
- Byte masking for high/low byte data compares
- R/$\overline{W}$ compare for non-tagged compares
- Tag inhibit on BDM TRACE

Downloaded from Arrow.com.

18.4.1 Breakpoint Modes

Three modes of operation determine the type of breakpoint in effect.

1. Dual address-only breakpoints, each of which causes a software interrupt (SWI)
2. Single full-feature breakpoint which causes the part to enter background debug mode (BDM)
3. Dual address-only breakpoints, each of which causes the part to enter BDM

Breakpoints do not occur when BDM is active.

18.4.1.1 SWI Dual Address Mode

In this mode, dual address-only breakpoints can be set, each of which causes a software interrupt. This is the only breakpoint mode which can force the CPU to execute an SWI. Program fetch tagging is the default in this mode; data breakpoints are not possible. In dual mode each address breakpoint is affected by the respective BKALE bit. The BKxRW, BKxRWE, BKMBH, and BKMBL bits are ignored. In dual address mode, the BKDBE becomes an enable for the second address breakpoint.

18.4.1.2 BDM Full Breakpoint Mode

This is a single full-featured breakpoint which causes the part to enter background debug mode. BK1ALE, BK1RW, and BK1RWE have no meaning in full breakpoint mode.

BKDBE enables data compare but has no meaning if BKPM = 1. BKMBH and BKMBL allow masking of high and low byte compares but has no meaning if BKPM = 1. BK0ALE enables compare of low address byte.

- Breakpoints are not allowed if the BDM mode is already active. Active mode means the CPU is executing out of the BDM ROM.
- BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. This is important because even if the ENABLE bit in the BDM is negated, the CPU actually does execute the BDM ROM code. It checks the ENABLE and returns if not set. If the BDM is not serviced by the monitor, then the breakpoint would be re-asserted when the BDM returns to normal CPU flow. There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

18.4.1.3 BDM Dual Address Mode

This mode has dual address-only breakpoints, each of which causes the part to enter background debug mode. In dual mode, each address breakpoint is affected by the BKPM bit, the BKxALE bits, and the BKxRW and BKxRWE bits. In dual address mode, the BKDBE becomes an enable for the second address breakpoint. The BKMBH and BKMBL bits have no effect when in a dual address mode. BDM may be entered by a breakpoint only if an internal signal from the BDM indicates background debug mode is enabled. If BKPM = 1, then BKxRW, BKxRWE, BKMBH, and BKMBL have no meaning.

- Breakpoints are not allowed if the BDM is already active. Active mode means the CPU is executing out of the BDM ROM.
- BDM should not be entered from a breakpoint unless the ENABLE bit is set in the BDM. This is important because even if the ENABLE bit in the BDM is negated, the CPU actually does execute the BDM ROM code. It checks the ENABLE and returns if not set. If the BDM is not serviced by the monitor, then the breakpoint would be re-asserted when the BDM returns to normal CPU flow. There is no hardware to enforce restriction of breakpoint operation if the BDM is not enabled.

Downloaded from Arrow.com.

18.4.2 Breakpoint Registers

Breakpoint operation consists of comparing data in the breakpoint address registers (BRKAH/BRKAL) to the address bus and comparing data in the breakpoint data registers (BRKDH/BRKDL) to the data bus. The breakpoint data registers also can be compared to the address bus. The scope of comparison can be expanded by ignoring the least significant byte of address or data matches.

The scope of comparison can be limited to program data only by setting the BKPM bit in breakpoint control register 0.

To trace program flow, setting the BKPM bit causes address comparison of program data only. Control bits are also available that allow checking read/write matches.

18.4.2.1 Breakpoint Control Register 0

Address: $0020

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BKEN1	BKEN0	BKPM	0	BK1ALE	BK0ALE	0	0
Reset:	0	0	0	0	0	0	0	0

Figure 18-10. Breakpoint Control Register 0 (BRKCT0)

Read and write anytime.
This register is used to control the breakpoint logic.

BKEN1 and BKEN0 — Breakpoint Mode Enable Bits

See Table 18-7.

Table 18-7. Breakpoint Mode Control

BKEN1	BKEN0	Mode Selected	BRKAH/L Usage	BRKDH/L Usage	R/$\overline{W}$	Range
0	0	Breakpoints off	—	—	—	—
0	1	SWI — dual address mode	Address match	Address match	No	Yes
1	0	BDM — full breakpoint mode	Address match	Data match	Yes	Yes
1	1	BDM — dual address mode	Address match	Address match	Yes	Yes

BKPM — Break on Program Addresses

This bit controls whether the breakpoint causes an immediate data breakpoint (next instruction boundary) or a delayed program breakpoint related to an executable opcode. Data and unexecuted opcodes cannot cause a break if this bit is set. This bit has no meaning in SWI dual address mode. The SWI mode only performs program breakpoints.

0 = On match, break at the next instruction boundary
1 = On match, break if the match is an instruction to be executed. This uses tagging as its breakpoint mechanism.

BK1ALE — Breakpoint 1 Range Control Bit

Only valid in dual address mode

0 = BRKDL is not used to compare to the address bus.
1 = BRKDL is used to compare to the address bus.

Downloaded from Arrow.com.

BK0ALE — Breakpoint 0 Range Control Bit

Valid in all modes

0 = BRKAL is not used to compare to the address bus.

1 = BRKAL is used to compare to the address bus.

Table 18-8. Breakpoint Address Range Control

BK1ALE	BK0ALE	Address Range Selected
—	0	Upper 8-bit address only for full mode or dual mode BKP0
—	1	Full 16-bit address for full mode or dual mode BKP0
0	—	Upper 8-bit address only for dual mode BKP1
1	—	Full 16-bit address for dual mode BKP1

18.4.2.2 Breakpoint Control Register 1

Address: $0021

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	0	BKDBE	BKMBH	BKMBL	BK1RWE	BK1RW	BK0RWE	BK0RW
Reset:	0	0	0	0	0	0	0	0

Figure 18-11. Breakpoint Control Register 1 (BRKCT1)

This register is read/write in all modes.

BKDBE — Enable Data Bus Bit

Enables comparing of address or data bus values using the BRKDH/L registers.

0 = BRKDH/L registers are not used in any comparison.

1 = BRKDH/L registers are used to compare address or data (depending upon the mode selections BKEN1 and BKEN0).

BKMBH — Breakpoint Mask High Bit

Disables the comparing of the high byte of data when in full breakpoint mode. Used in conjunction with the BKDBE bit (which should be set)

0 = High byte of data bus (bits 15:8) are compared to BRKDH.

1 = High byte is not used in comparisons.

BKMBL — Breakpoint Mask Low Bit

Disables the matching of the low byte of data when in full breakpoint mode. Used in conjunction with the BKDBE bit (which should be set)

0 = Low byte of data bus (bits 7–0) are compared to BRKDL.

1 = Low byte is not used to in comparisons.

BK1RWE — R/$\overline{W}$ Compare Enable Bit

Enables the comparison of the R/$\overline{W}$ signal to further specify what causes a match. This bit is NOT useful in program breakpoints or in full breakpoint mode. This bit is used in conjunction with a second address in dual address mode when BKDBE = 1.

0 = R/$\overline{W}$ is not used in comparisons.

1 = R/$\overline{W}$ is used in comparisons.

Downloaded from Arrow.com.

BK1RW — R/$\overline{W}$ Compare Value Bit

When BK1RWE = 1, this bit determines the type of bus cycle to match.

0 = A write cycle is matched.

1 = A read cycle is matched.

BK0RWE — R/$\overline{W}$ Compare Enable Bit

Enables the comparison of the R/$\overline{W}$ signal to further specify what causes a match. This bit is not useful in program breakpoints.

0 = R/$\overline{W}$ is not used in the comparisons.

1 = R/$\overline{W}$ is used in comparisons.

BK0RW — R/$\overline{W}$ Compare Value Bit

When BK0RWE = 1, this bit determines the type of bus cycle to match.

0 = Write cycle is matched.

1 = Read cycle is matched.

Table 18-9. Breakpoint Read/Write Control

BK1RWE	BK1RW	BK0RWE	BK0RW	Read/Write Selected
—	—	0	X	R/$\overline{W}$ is don't care for full mode or dual mode BKP0
—	—	1	0	R/$\overline{W}$ is write for full mode or dual mode BKP0
—	—	1	1	R/$\overline{W}$ is read for full mode or dual mode BKP0
0	X	—	—	R/$\overline{W}$ is don't care for dual mode BKP1
1	0	—	—	R/$\overline{W}$ is write for dual mode BKP1
1	1	—	—	R/$\overline{W}$ is read for dual mode BKP1

18.4.2.3 Breakpoint Address Register High

Address: $0022

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Power on reset:	0	0	0	0	0	0	0	0

Figure 18-12. Breakpoint Address Register High (BRKAH)

These bits are used to compare against the most significant byte of the address bus.

18.4.2.4 Breakpoint Address Register Low

Address: $0023

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Power on reset:	0	0	0	0	0	0	0	0

Figure 18-13. Breakpoint Address Register Low (BRKAL)

These bits are used to compare against the least significant byte of the address bus. These bits may be excluded from being used in the match if BK0ALE = 0.

Downloaded from Arrow.com.

18.4.2.5 Breakpoint Data Register High

Address: $0024

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
Power on reset:	0	0	0	0	0	0	0	0

Figure 18-14. Breakpoint Data Register High (BRKDH)

These bits are compared to the most significant byte of the data bus in full breakpoint mode or the most significant byte of the address bus in dual address modes. BKE1, BKE0, BKDBE, and BKMBH control how this byte is used in the breakpoint comparison.

18.4.2.6 Breakpoint Data Register Low Byte

Address: $0025

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
Power on reset:	0	0	0	0	0	0	0	0

Figure 18-15. Breakpoint Data Register Low (BRKDL)

These bits are compared to the least significant byte of the data bus in full breakpoint mode or the least significant byte of the address bus in dual address modes. BKEN1, BKEN0, BKDBE, BK1ALE, and BKMBL control how this byte is used in the breakpoint comparison.

NOTE

After a power-on reset, registers BRKAH, BRKAL, BRKDH, and BRKDL are cleared but these registers are not affected by normal resets.

18.5 Instruction Tagging

The instruction queue and cycle-by-cycle CPU activity can be reconstructed in real time or from trace history that was captured by a logic analyzer. However, the reconstructed queue cannot be used to stop the CPU at a specific instruction, because execution has already begun by the time an operation is visible outside the MCU. A separate instruction tagging mechanism is provided for this purpose.

Executing the BDM TAGGO command configures two MCU pins for tagging. Tagging information is latched on the falling edge of ECLK along with program information as it is fetched. Tagging is allowed in all modes. Tagging is disabled when BDM becomes active and BDM serial commands cannot be processed while tagging is active.

$\overline{\text{TAGHI}}$ is a shared function of the BKGD pin.

$\overline{\text{TAGLO}}$ is a shared function of the PE3/$\overline{\text{LSTRB}}$ pin, a multiplexed I/O pin. For 1/4 cycle before and after the rising edge of the E clock, this pin is the $\overline{\text{LSTRB}}$ driven output.

$\overline{\text{TAGLO}}$ and $\overline{\text{TAGHI}}$ inputs are captured at the falling edge of the E clock. A logic 0 on $\overline{\text{TAGHI}}$ and/or $\overline{\text{TAGLO}}$ marks (tags) the instruction on the high and/or low byte of the program word that was on the data bus at the same falling edge of the E clock.

Downloaded from Arrow.com.

Table 18-10 shows the functions of the two tagging pins. The pins operate independently; the state of one pin does not affect the function of the other. The presence of logic level 0 on either pin at the fall of ECLK performs the indicated function. Tagging is allowed in all modes. Tagging is disabled when BDM becomes active and BDM serial commands are not processed while tagging is active.

Table 18-10. Tag Pin Function

TAGHI	TAGLO	Tag
1	1	No tag
1	0	Low byte
0	1	High byte
0	0	Both bytes

The tag follows the information in the queue as the queue is advanced. When a tagged instruction reaches the head of the queue, the CPU enters active background debug mode rather than executing the instruction. This is the mechanism by which a development system initiates hardware breakpoints.

Currently, the tool configuration shown in Figure 18-16 is used.

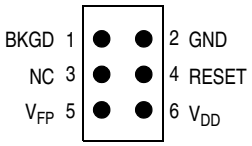


Figure 18-16. BDM Tool Connector

Downloaded from Arrow.com.

Downloaded from Arrow.com.

Chapter 19
Electrical Specifications

19.1 Introduction

This section contains electrical and timing specifications.

19.2 Maximum Ratings

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

NOTE

This device is not guaranteed to operate properly at the maximum ratings. Refer to 19.5 5.0 Volt DC Electrical Characteristics for guaranteed operating conditions.

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}, V_{DDA}, V_{DDX}	–0.3 to +6.5	V
Input voltage	V_{In}	–0.3 to +6.5	V
Maximum current per pin excluding V_{DD} and V_{SS}	I_{In}	± 25	mA
Storage temperature	T_{STG}	–55 to +150	°C
V_{DD} differential voltage	V_{DD}–V_{DDX}	6.5	V

NOTE

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{In} and V_{Out} be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).

Downloaded from Arrow.com.

19.3 Functional Operating Range

Rating	Symbol	Value	Unit
All devices in this document meet these operating temperature ranges: “C” temperature range “V” temperature range “M” temperature range	T_A	T_L to T_H –40 to +85 –40 to +105 –40 to +125	°C
Operating voltage range	V_{DD}	5.0 ± 10%	V

19.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Average junction temperature	T_J	$T_A + (P_D \times \Theta_{JA})$	°C
Ambient temperature	T_A	User-determined	°C
Package thermal resistance (junction-to-ambient) 80-pin quad flat pack (QFP)	Θ_{JA}	76	°C/W
Total power dissipation(1)	P_D	$P_{INT} + P_{I/O}$ or $\frac{K}{T_J + 273°C}$	W
Device internal power dissipation	P_{INT}	$I_{DD} \times V_{DD}$	W
I/O pin power dissipation(2)	$P_{I/O}$	User-determined	W
A constant(3)	K	$P_D \times (T_A + 273\ °C) + \Theta_{JA} \times P_D^2$	W/°C

1. This is an approximate value, neglecting $P_{I/O}$.
2. For most applications, $P_{I/O} \ll P_{INT}$ and can be neglected.
3. K is a constant pertaining to the device. Solve for K with a known T_A and a measured P_D (at equilibrium). Use this value of K to solve for P_D and T_J iteratively for any value of T_A.

Downloaded from Arrow.com.

19.5 5.0 Volt DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Max	Unit
Input high voltage, all inputs	V_{IH}	$0.7 \times V_{DD}$	$V_{DD} + 0.3$	V
Input low voltage, all inputs	V_{IL}	$V_{SS} - 0.3$	$0.2 \times V_{DD}$	V
Output high voltage, all I/O and output pins except XTAL Normal drive strength I_{OH} = –10.0 µA I_{OH} = –0.8 mA Reduced drive strength I_{OH} = –4.0 µA I_{OH} = –0.3 mA	V_{OH}	 $V_{DD} - 0.2$ $V_{DD} - 0.8$ $V_{DD} - 0.2$ $V_{DD} - 0.8$	 — — — —	V
Output low voltage, all I/O and output pins except XTAL Normal drive strength I_{OL} = 10.0 µA I_{OL} = 1.6 mA Reduced drive strength I_{OL} = 3.6 µA I_{OL} = 0.6 mA	V_{OL}	 — — — —	 $V_{SS} + 0.2$ $V_{SS} + 0.4$ $V_{SS} + 0.2$ $V_{SS} + 0.4$	V
Input leakage current[2] $V_{In} = V_{DD}$ or V_{SS} All input-only pins except ATD[3] and V_{FP}	I_{In}	—	± 5	µA
Three-state leakage, I/O ports, BKGD, and $\overline{\text{RESET}}$	I_{OZ}	—	± 2.5	µA
Input capacitance All input pins and ATD pins (non-sampling) ATD pins (sampling) All I/O pins	C_{In}	 — — —	 10 15 20	pF
Output load capacitance All outputs except PS7–PS4 PS7–PS4 when configured as SPI	C_L	 — —	 90 200	pF
Programmable active pullup current $\overline{\text{XIRQ}}$, $\overline{\text{DBE}}$, $\overline{\text{LSTRB}}$, R/$\overline{\text{W}}$, ports A, B, DLC, P, S, T MODA, MODB active pulldown during reset BKGD passive pullup	I_{APU}	 50 50 50	 500 500 500	µA

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. Specification is for parts in the –40 to +85 °C range. Higher temperature ranges will result in increased current leakage.
3. See 19.8 ATD DC Electrical Characteristics.

Downloaded from Arrow.com.

19.6 Supply Current

Characteristic[1]	Symbol	2 MHz	4 MHz	8 MHz	Unit
Maximum total supply current					
RUN:	I_{DD}				
Single-chip mode		15	25	45	mA
Expanded mode		25	45	70	mA
WAIT: (All peripheral functions shut down)					
Single-chip mode	W_{IDD}	1.5	3	5	mA
Expanded mode		4	7	10	mA
STOP:					
Single-chip mode, no clocks					
–40 to +85	S_{IDD}	10	10	10	µA
+85 to +105		25	25	25	µA
+105 to +125		50	50	50	µA
Maximum power dissipation[2]					
Single-chip mode	P_D	75	125	225	mW
Expanded mode		125	225	350	

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. Includes I_{DD} and I_{DDA}

19.7 ATD Maximum Ratings

Characteristic	Symbol	Value	Units
ATD reference voltage			
$V_{RH} \le V_{DDA}$	V_{RH}	–0.3 to +6.5	V
$V_{RL} \ge V_{SSA}$	V_{RL}	–0.3 to +6.5	
V_{SS} differential voltage	$\lvert V_{SS}-V_{SSA}\rvert$	0.1	V
V_{DD} differential voltage	$V_{DD}-V_{DDA}$	6.5	V
	$V_{DDA}-V_{DD}$	0.3	
V_{REF} differential voltage	$\lvert V_{RH}-V_{RL}\rvert$	6.5	V
Reference to supply differential voltage	$\lvert V_{RH}-V_{DDA}\rvert$	6.5	V

Downloaded from Arrow.com.

19.8 ATD DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Max	Unit
Analog supply voltage	V_{DDA}	4.5	5.5	V
Analog supply current, normal operation	I_{DDA}	—	1.0	mA
Reference voltage, low	V_{RL}	V_{SSA}	$V_{DDA}/2$	V
Reference voltage, high	V_{RH}	$V_{DDA}/2$	V_{DDA}	V
V_{REF} differential reference voltage[2]	$V_{RH}-V_{RL}$	4.5	5.5	V
Input voltage[3]	V_{INDC}	V_{SSA}	V_{DDA}	V
Input current, off channel[4]	I_{OFF}	—	100	nA
Reference supply current	I_{REF}	—	250	µA
Input capacitance Not Sampling Sampling	C_{INN} C_{INS}	— —	10 15	pF

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD clock = 2 MHz, unless otherwise noted
2. Accuracy is guaranteed at $V_{RH} - V_{RL}$ = 5.0 V ± 10%.
3. To obtain full-scale, full-range results, $V_{SSA} \le V_{RL} \le V_{INDC} \le V_{RH} \le V_{DDA}$.
4. Maximum leakage occurs at maximum operating temperature. Current decreases by approximately one-half for each 10 °C decrease from maximum temperature.

Downloaded from Arrow.com.

19.9 Analog Converter Operating Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit
8-bit resolution[2]	1 count	—	20	—	mV
8-bit differential non-linearity[3]	DNL	–0.5	—	+0.5	Count
8-bit integral non-linearity[3]	INL	–1	—	+1	Count
8-bit absolute error[3], [4] 2, 4, 8, and 16 ATD sample clocks	AE	–1	—	+1	Count
10-bit resolution[2]	1 count	—	5	—	mV
10-bit differential non-linearity[3]	DNL	–2	—	2	Count
10-bit integral non-linearity[3]	INL	–2	—	2	Count
10-bit absolute error[3] 2, 4, 8, and 16 ATD sample clocks	AE	–2.5	—	2.5	Count
Maximum source impedance	R_S	—	20	See [5]	kΩ

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD clock = 2 MHz, unless otherwise noted
2. $V_{RH}–V_{RL} \geq 5.12$ V; $V_{DDA}–V_{SSA} = 5.12$ V
3. At V_{REF} = 5.12 V, one 8-bit count = 20 mV, and one 10-bit count = 5 mV. INL and DNL are characterized using the process window parameters affecting the ATD accuracy, but they are not tested.
4. Eight-bit absolute error of 1 count (20 mV) includes 1/2 count (10 mV) inherent quantization error and 1/2 count (10 mV) circuit (differential, integral, and offset) error.
5. Maximum source impedance is application-dependent. Error resulting from pin leakage depends on junction leakage into the pin and on leakage due to charge-sharing with internal capacitance.
   Error from junction leakage is a function of external source impedance and input leakage current. Expected error in result value due to junction leakage is expressed in voltage (V_{ERRJ}):
   $$V_{ERRJ} = R_S \times I_{OFF}$$
   Where
   I_{OFF} is a function of operating temperature. Charge-sharing effects with internal capacitors are a function of ATD clock speed, the number of channels being scanned, and source impedance. For 8-bit conversions, charge pump leakage is computed as:
   $$V_{ERRJ} = 0.25\ pF \times V_{DDA} \times R_S \times ATDCLK/(8 \times \text{number of channels})$$

Downloaded from Arrow.com.

19.10 ATD AC Operating Characteristics (Operating)

Characteristic[1]	Symbol	Min	Max	Unit
MCU clock frequency (p-clock)	f_{PCLK}	2.0	8.0	MHz
ATD operating clock frequency	f_{ATDCLK}	0.5	2.0	MHz
ATD 8-bit conversion period ATD clock cycles[2] ATD conversion time[3]	 n_{CONV8} t_{CONV8}	 18 9	 32 16	 Cycles µs
ATD 10-bit conversion period ATD clock cycles[2] ATD conversion time[3]	 n_{CONV10} t_{CONV10}	 20 10.0	 34 17	 Cycles µs
Stop and ATD power-up recovery time[4] V_{DDA} = 5.0 V	t_{SR}	—	10	µs

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, ATD clock = 2 MHz, unless otherwise noted
2. The minimum time assumes a final sample period of 2 ATD clock cycles while the maximum time assumes a final sample period of 16 ATD clocks.
3. This assumes an ATD clock frequency of 2.0 MHz.
4. From the time ADPU is asserted until the time an ATD conversion can begin

19.11 EEPROM Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit
Minimum programming clock frequency[2]	f_{PROG}	1.0	—	—	MHz
Programming time	t_{PROG}	10.0	—	10.5	ms
Clock recovery time, following STOP, to continue programming	t_{CRSTOP}	—	—	t_{PROG}+ 1	ms
Erase time	t_{ERASE}	10.0	—	10.5	ms
Write/erase endurance	—	10,000	—	—	Cycles
Data retention	—	10	—	—	Years

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. RC oscillator must be enabled if programming is desired and $f_{SYS} < f_{PROG}$.

Downloaded from Arrow.com.

19.12 FLASH EEPROM Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Units
Program/erase supply voltage Read only Program/erase/verify	V_{FP}	V_{DD} –0.35 11.4	V_{DD} 12.0	V_{DD} +0.5 12.6	V
Program/erase supply current Word program (V_{FP} = 12 V) Erase (V_{FP} = 12 V)	I_{FP}	— —	— —	30 4	mA
Number of programming pulses	n_{PP}	—	—	50	Pulses
Programming pulse	t_{PPULSE}	20	—	25	µs
Program to verify time	t_{VPROG}	10	—	—	µs
Program margin	p_m	100[2]	—	—	%
Number of erase pulses	n_{EP}	—	—	5	Pulses
Erase pulse	t_{EPULSE}	5	—	10	ms
Erase to verify time	t_{VERASE}	1	—	—	ms
Erase margin	e_m	100[3]	—	—	%
Program/erase endurance		100	—	—	Cycles
Data retention		10	—	—	Years

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted
2. The number of margin pulses required is the same as the number of pulses used to program or erase.

Downloaded from Arrow.com.

19.12.1 Programming Voltage Supply Envelope

The key to preventing damage to the FLASH array or corruption of the data contained in the memory is the programming voltage envelope shown in Figure 19-1. Many of the problems that customers experience with FLASH devices are due to a failure to ensure that their voltage sources always meet these requirements.

The most important single thing to remember from this diagram is that V_{FP} and V_{DD} should always be at the same level, except during an actual program or erase cycle. Corruption of FLASH data is often encountered when V_{FP} is allowed to exceed V_{DD} during the power-up and power-down phases.

Conversely, if V_{FP} is allowed to fall below 0.35 volts lower than V_{DD} at any time, damage to the FLASH array can occur.

NOTE

Although Figure 19-1 shows a lower boundary of 4.15 volts on V_{FP} during the normal read phase, V_{FP} always must be no more than .35 volts below V_{DD}. For example, If the operating voltage of V_{DD} in the system is 5.2 volts, V_{FP} can be no lower than 4.85 volts.

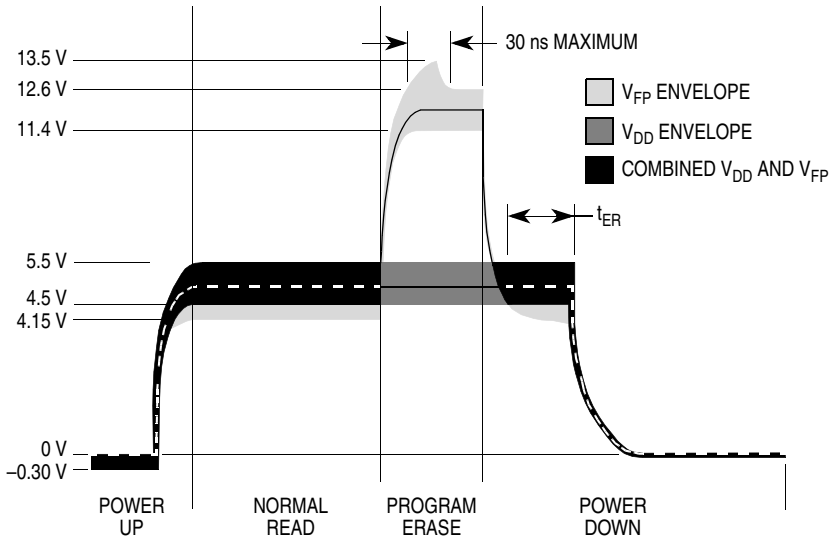


Figure 19-1. Programming Voltage Envelope

Downloaded from Arrow.com.

19.12.2 Example V_{FP} Protection Circuitry

Figure 19-2 shows an example of a circuit which, if properly implemented, can maintain the appropriate voltage levels on the V_{FP} pin. This section outlines the design for this circuit, what each component is intended to do, and some design considerations when designing V_{FP} pin protection.

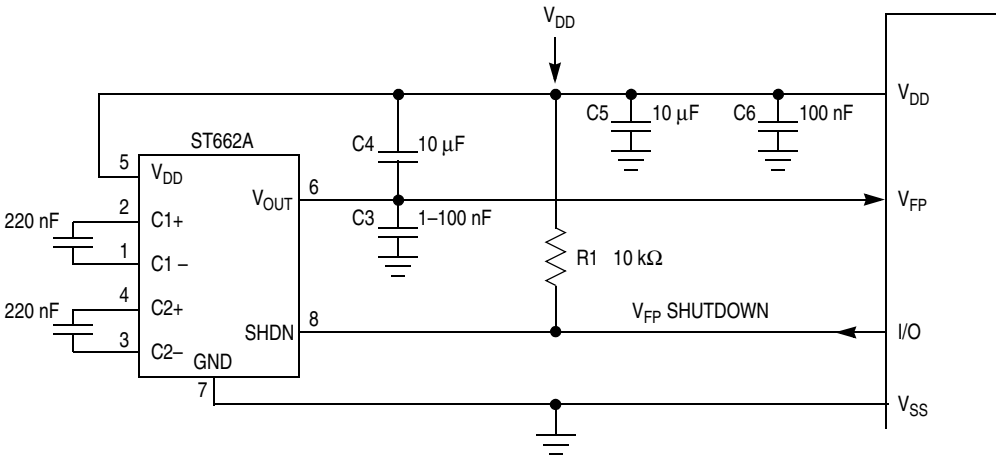


Figure 19-2. V_{FP} Supply Circuit

The general idea of this circuit implementation is to supply V_{FP} from a dc-dc converter. This dc-dc converter, like most, provides a shutdown feature which allows the converter's output to be shut off. When the SHDN pin on the converter is pulled high, as the 10-kΩ pullup resistor (R1) does, the output V_{Out} is shorted to the V_{DD} supply. This requires that the programming and erasing routines assert a port pin on the MCU to turn on the converter and supply the 12-volt programming voltage during the programming or erasing cycle. Simple programming and erasing routines, such as those shown earlier in this application note, will no longer suffice.

By implementing this solution, V_{FP} is tied to V_{DD} on power-up and power-down, ensuring that they rise and fall together. Capacitors C5 and C6 are the normal decoupling capacitors on the V_{DD} supply lines. C3 is used to reduce electromagnetic interference (EMI) in the circuit. If C3 is too large, V_{FP} will not be allowed to fall with V_{DD}, potentially causing data corruption in the FLASH array. (Refer to Figure 19-3.) C4 is where the dc-dc converter stores charge to supply V_{Out} to the target device. The supply must be able to source approximately 30 mA of current for at least 20 µs (based on programming cycle requirements) and 4 mA of current for at least 10 ms (based on erase cycle requirements).

A certain degree of experimentation might be required when selecting C4 and C3. When trying different capacitor values, always monitor the effects on V_{FP} decay during power-down and current supplied to the V_{FP} pin.

R1 must be no larger than 10 kΩ, to make certain that the SHDN pin on the dc-dc converter is never allowed to fall below V_{DD} unless the output pin of the microcontroller is driven low. The external pullup ensures this behavior, no matter what port pin is used on the microcontroller or what the internal structure of that pin looks like. Without a strong enough pullup resistor on R1, the voltage on the SHDN pin might drop during a reset event, causing the dc-dc converter to activate and begin driving the voltage on V_{Out} to begin to rise to 12 volts. This would result in data corruption in the FLASH.

Downloaded from Arrow.com.

NOTE

Figure 19-2 is different from the recommended circuit shown in information about ST662A from ST Microelectronics, but it is correct. The change is in the location of the capacitor C4, which is now placed between V_{DD} and V_{FP}. This change was implemented with the cooperation of ST Microelectronics to aid in tracking a rapidly falling V_{DD} voltage level, such as in Figure 19-3 and Figure 19-4. This circuit also has been verified with the Maxim Integrated Products device (MAX662).

Be certain that V_{FP} decays with V_{DD}, as shown in Figure 19-4, as new capacitance values are tested. The rate of decay of the V_{DD} supply powering down will help define how large the C3 capacitance can be made.

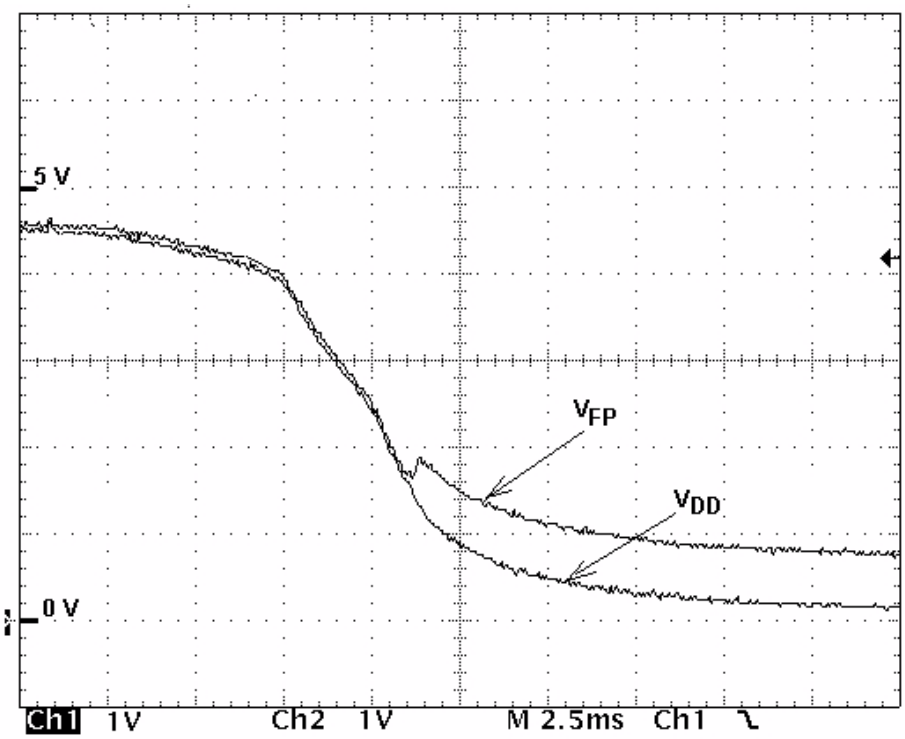


Figure 19-3. V_{FP} Exceeding V_{DD} during Power-Down

Downloaded from Arrow.com.

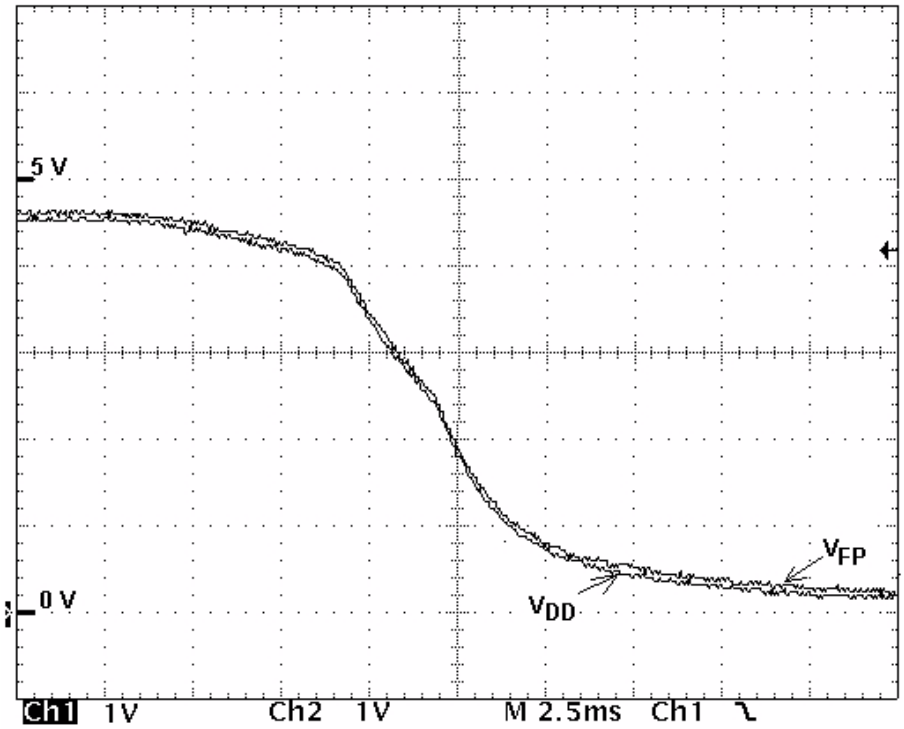


Figure 19-4. V_{FP} Tracking V_{DD} during Power-Down

When checking to ensure that the reservoir capacitance value of C4 is not too low, the voltage level of V_{FP} can be monitored during an initial erase and a write pulse. Remember that the largest current draw on erasing is when all of the bits of the FLASH are programmed to 0. Conversely, the highest programming current is seen when programming all the bits to 0 from the erased state of 1. The user should look at this on an oscilloscope, due to the brevity of the pulses. Using a port pin or the SHDN signal may be useful to trigger the scope when the pulses are fired. If the voltage dips below 11.4 volts, the capacitance used can be increased, but be sure to verify that decay rates of V_{DD} and V_{FP} are still the same. If V_{FP} is declining with each successive pulse, try inserting some delays between each pulse to allow the charge pump to recharge.

The solution shown here uses the ST662A dc-dc converter, but any similar device will work. Some other options are the LTC1262C from Linear Technology Corporation or the MAX662 from Maxim Integrated Products, Inc.

Downloaded from Arrow.com.

19.13 Pulse-Width Modulator Characteristics

Characteristic[1]	Symbol	Min	Max	Unit
E-clock frequency	f_{ECLK}	—	8.0	MHz
A-clock frequency Selectable	f_{ACLK}	$f_{ECLK}/128$	f_{ECLK}	Hz
B-clock frequency Selectable	f_{BCLK}	$f_{ECLK}/128$	f_{ECLK}	Hz
Left-aligned PWM frequency 8-bit 16-bit	f_{LPWM}	$f_{ECLK}/1$ M $f_{ECLK}/256$ M	$f_{ECLK}/2$ $f_{ECLK}/2$	Hz
Left-aligned PWM resolution	r_{LPWM}	$f_{ECLK}/4$ K	f_{ECLK}	Hz
Center-aligned PWM frequency 8-bit 16-bit	f_{CPWM}	$f_{ECLK}/2$ M $f_{ECLK}/512$ M	f_{ECLK} f_{ECLK}	Hz
Center-aligned PWM resolution	r_{CPWM}	$f_{ECLK}/4$ K	f_{ECLK}	Hz

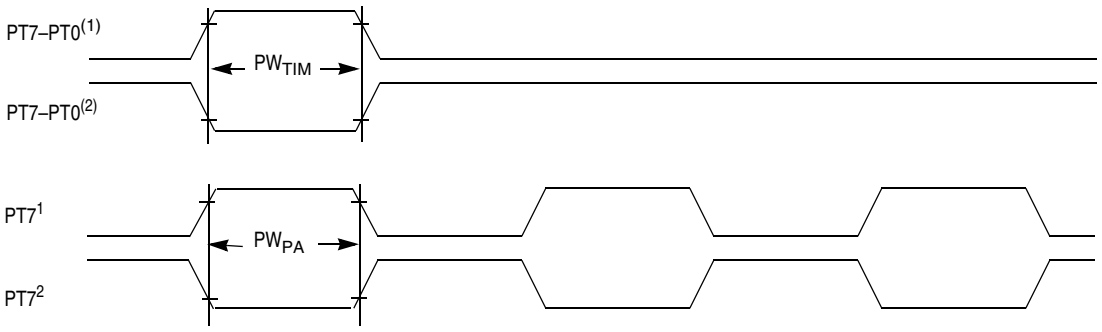
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted

Downloaded from Arrow.com.

19.14 Control Timing

Characteristic	Symbol	8.0 MHz		Unit
		Min	Max	
Frequency of operation	f_o	dc	8.0	MHz
E-clock period	t_{cyc}	125	—	ns
Crystal frequency	f_{XTAL}	—	16.0	MHz
External oscillator frequency	$2\ f_o$	dc	16.0	MHz
Processor control setup time $t_{PCSU} = t_{cyc}/2+ 20$	t_{PCSU}	82	—	ns
Reset input pulse width(1) To guarantee external reset vector Minimum input time (can be pre-empted by internal reset)	PW_{RSTL}	 32 2	 — —	t_{cyc}
Mode programming setup time	t_{MPS}	4	—	t_{cyc}
Mode programming hold time	t_{MPH}	10	—	ns
Interrupt pulse width, $\overline{IRQ}$ edge-sensitive mode $PW_{IRQ} = 2\ t_{cyc} + 20$	PW_{IRQ}	270	—	ns
Wait recovery startup time $t_{WRS} = 4\ t_{cyc}$	t_{WRS}	—	4	t_{cyc}
Timer input capture pulse width $PW_{TIM} = 2\ t_{cyc} + 20$	PW_{TIM}	270	—	ns
Pulse accumulator pulse width	PW_{PA}	TBD	—	ns

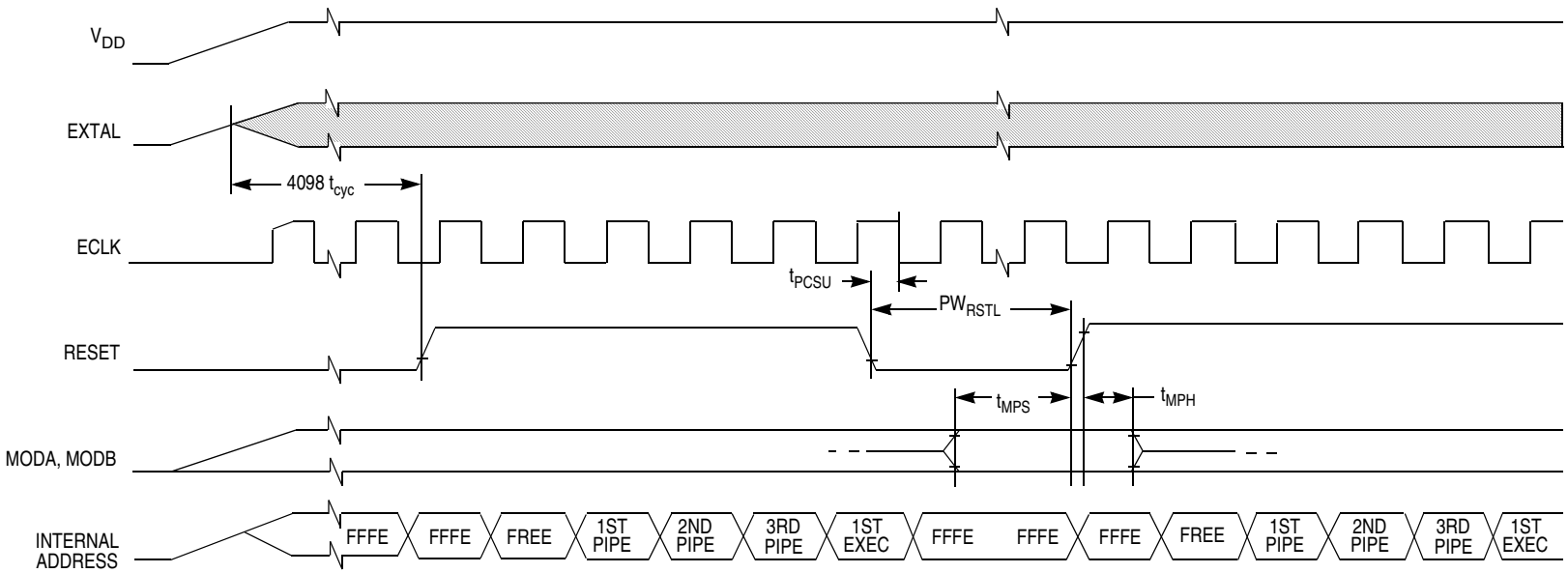
1. $\overline{RESET}$ is recognized during the first clock cycle it is held low. Internal circuitry then drives the pin low for 16 clock cycles, releases the pin, and samples the pin level eight cycles later to determine the source of the interrupt.



Notes:
1. Rising edge sensitive input
2. Falling edge sensitive input

Figure 19-5. Timer Inputs

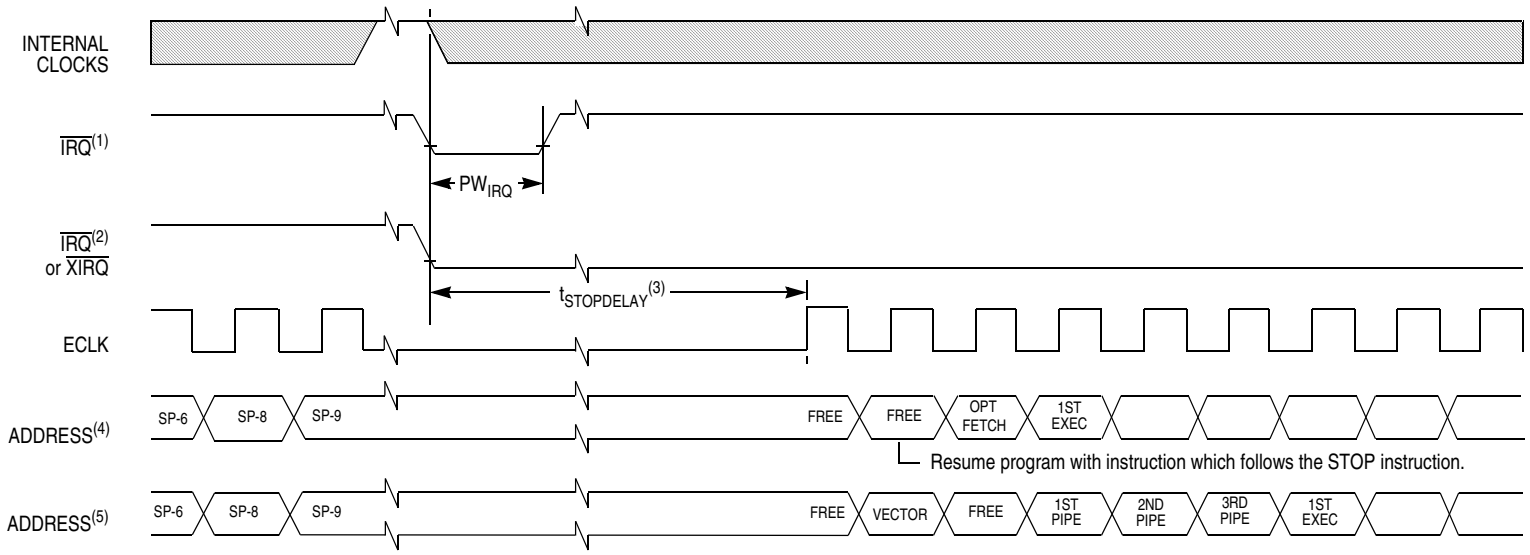
Downloaded from Arrow.com.



Note: Reset timing is subject to change.

Figure 19-6. Power-On and External Reset Timing Diagram

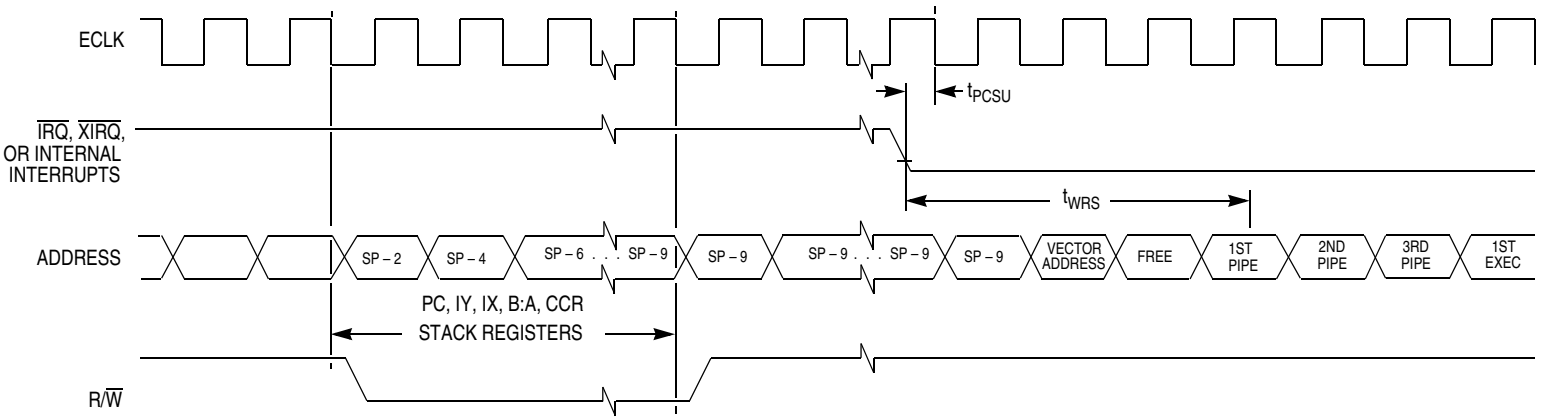
Downloaded from Arrow.com.



Notes:
1. Edge-sensitive $\overline{\text{IRQ}}$ pin (IRQE bit = 1)
2. Level-sensitive $\overline{\text{IRQ}}$ pin (IRQE bit = 0)
3. $t_{\overline{\text{STOP}}\text{DELAY}}$ = 4098 t_{cyc} if DLY bit = 1 or 2 t_{cyc} if DLY = 0.
4. $\overline{\text{XIRQ}}$ with X bit in CCR = 1.
5. $\overline{\text{IRQ}}$ or ($\overline{\text{XIRQ}}$ with X bit in CCR = 0)

Figure 19-7. Stop Recovery Timing Diagram

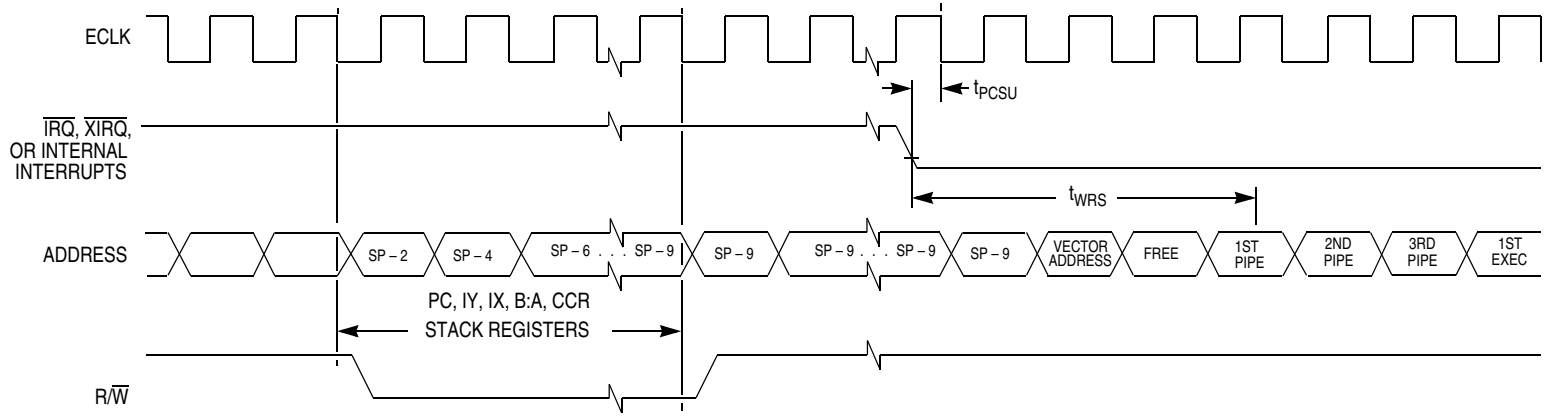
Downloaded from Arrow.com.



Note: $\overline{\text{RESET}}$ also causes recovery from WAIT.

Figure 19-8. Wait Recovery Timing Diagram

Downloaded from Arrow.com.



Note: $\overline{\text{RESET}}$ also causes recovery from WAIT.

Figure 19-9. Interrupt Timing Diagram

Downloaded from Arrow.com.

19.15 Peripheral Port Timing

Characteristic	Symbol	8.0 MHz		Unit
		Min	Max	
Frequency of operation E-clock frequency	f_o	dc	8.0	MHz
E-clock period	t_{cyc}	125	—	ns
Peripheral data setup time MCU read of ports $t_{PDSU} = t_{cyc}/2 + 40$	t_{PDSU}	102	—	ns
Peripheral data hold time MCU read of ports	t_{PDH}	0	—	ns
Delay time, peripheral data write MCU write to ports	t_{PWD}	—	40	ns
Delay time, peripheral data write MCU write to port CAN	t_{PWD}	—	71	ns

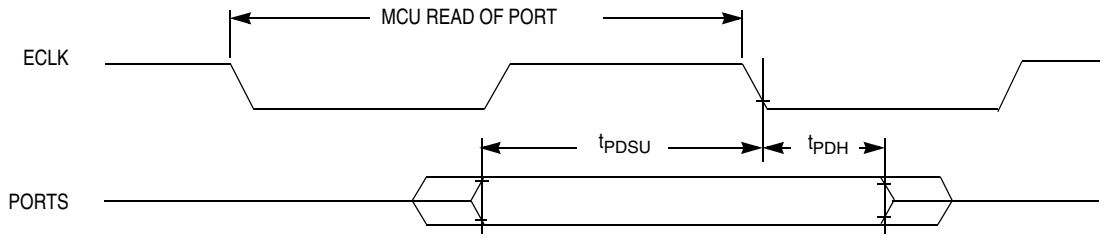


Figure 19-10. Port Read Timing Diagram

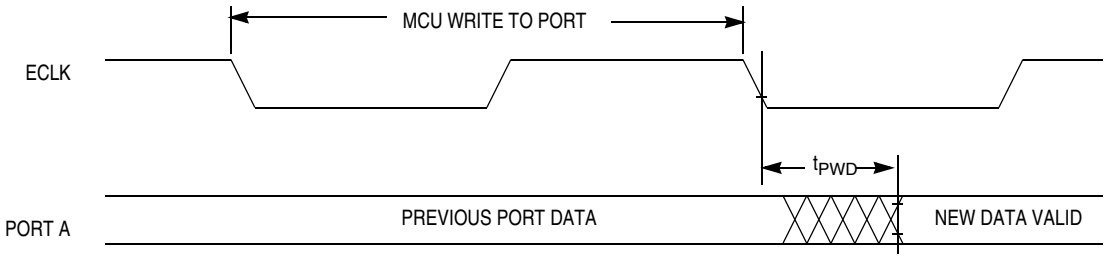


Figure 19-11. Port Write Timing Diagram

Downloaded from Arrow.com.

19.16 Multiplexed Expansion Bus Timing

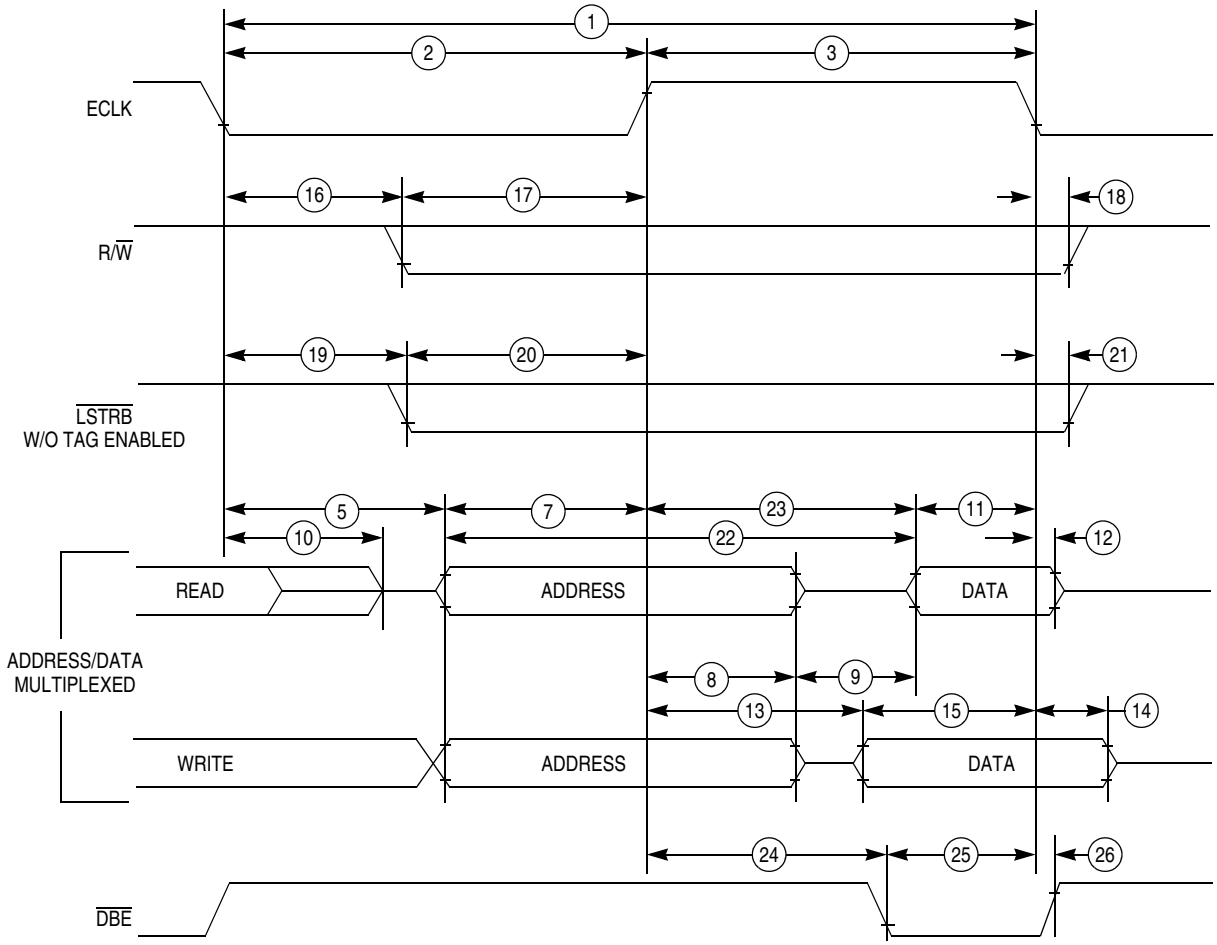
NOTE

Use of the multiplexed expansion bus at 8 MHz is discouraged due to TAD delay factors.

Num	Characteristic[1], [2], [3], [4], [5]	Delay	Symbol	8 MHz		2 MHz		Unit
				Min	Max	Min	Max	
—	Frequency of operation (E-clock frequency)	—	f_o	dc	8.0	dc	8.0	MHz
1	Cycle time $t_{cyc} = 1/f_o$	—	t_{cyc}	125	—	500	—	ns
2	Pulse width, E low $PW_{EL} = t_{cyc}/2 + delay$	–4	PW_{EL}	59	—	246	—	ns
3	Pulse width, E high[6] $PW_{EH} = t_{cyc}/2 + delay$	–2	PW_{EH}	59	—	248	—	ns
5	Address delay time $t_{AD} = t_{cyc}/4 + delay$	27	t_{AD}	—	67.5	—	152	ns
7	Address valid time to ECLK rise $t_{AV} = PW_{EL} – t_{AD}$	—	t_{AV}	–6.2	—	94	—	ns
8	Multiplexed address hold time $t_{MAH} = t_{cyc}/4 + delay$	–18	t_{MAH}	13	—	107	—	ns
9	Address hold to data valid	—	t_{AHDS}	30	—	20	—	ns
10	Data hold to high impedance $t_{DHZ} = t_{AD} – 20$	—	t_{DHZ}	—	45.2	—	132	ns
11	Read data setup time	—	t_{DSR}	31.2	—	25	—	ns
12	Read data hold time	—	t_{DHR}	0	—	0	—	ns
13	Write data delay time	—	t_{DDW}	—	62.5	—	165	ns
14	Write data hold time	—	t_{DHW}	25	—	20	—	ns
15	Write data setup time[6] $t_{DSW} = PW_{EH} – t_{DDW}$	—	t_{DSW}	5.8	—	83	—	ns
16	Read/write delay time $t_{RWD} = t_{cyc}/4 + delay$	18	t_{RWD}	—	57.5	—	143	ns
17	Read/write valid time to E rise $t_{RWV} = PW_{EL} – t_{RWD}$	—	t_{RWV}	3.8	—	103	—	ns
18	Read/write hold time	—	t_{RWH}	25	—	20	—	ns
19	Low strobe[7] delay time $t_{LSD} = t_{cyc}/4 + delay$	18	t_{LSD}	—	57.5	—	143	ns
20	Low strobe[7] valid time to E rise $t_{LSV} = PW_{EL} – t_{LSD}$	—	t_{LSV}	3.8	—	103	—	ns
21	Low strobe[7] hold time	—	t_{LSH}	25	—	20	—	ns
22	Address access time[6] $t_{ACCA} = t_{cyc} – t_{AD} – t_{DSR}$	—	t_{ACCA}	—	27.6	—	323	ns
23	Access time from E rise[6] $t_{ACCE} = PW_{EH} – t_{DSR}$	—	t_{ACCE}	—	27.8	—	223	ns
24	$\overline{DBE}$ delay from ECLK rise[6] $t_{DBED} = t_{cyc}/4 + delay$	8	t_{DBED}	—	57.5	—	133	ns
25	$\overline{DBE}$ valid time $t_{DBE} = PW_{EH} – t_{DBED}$	—	t_{DBE}	11.8	—	115	—	ns
26	$\overline{DBE}$ hold time from ECLK fall	—	t_{DBEH}	–3	10	–3	10	ns

1. $V_{DD} = 5.0$ Vdc ± 10%, $V_{SS} = 0$ Vdc, $T_A = T_L$ to T_H, unless otherwise noted
2. All timings are calculated for normal port drives.
3. Crystal input is required to be within 45% to 55% duty.
4. Reduced drive must be off to meet these timings.
5. Unequalled loading of pins will affect relative timing numbers.
6. This characteristic is affected by clock stretch.
   Add $N \times t_{cyc}$ where N = 0, 1, 2, or 3, depending on the number of clock stretches.
7. Without TAG enabled

Downloaded from Arrow.com.



Note: Measurement points shown are 20% and 70% of V_{DD}.

Figure 19-12. Multiplexed Expansion Bus Timing Diagram

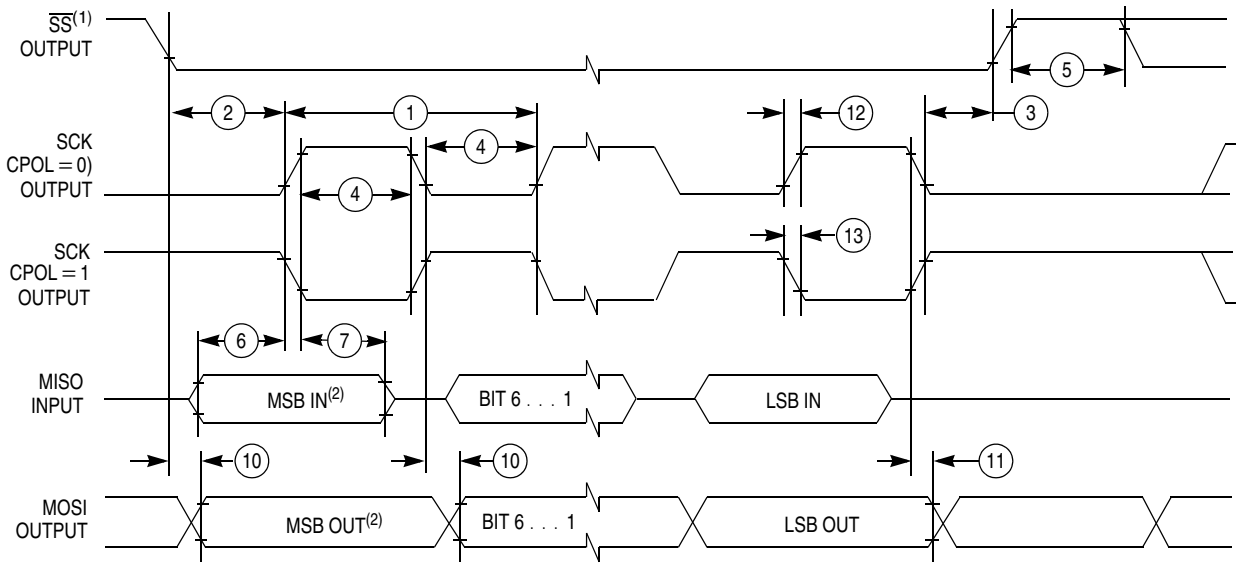
Downloaded from Arrow.com.

19.17 Serial Peripheral Interface (SPI) Timing

Num	Function[1]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	f_{OP}	 DC DC	 1/2 1/2	E-clock frequency
1	SCK period Master Slave	t_{SCK}	 2 2	 256 —	 t_{cyc} t_{cyc}
2	Enable lead time Master Slave	t_{Lead}	 1/2 1	 — —	 t_{sck} t_{cyc}
3	Enable lag time Master Slave	t_{LAG}	 1/2 1	 — —	 t_{sck} t_{cyc}
4	Clock (SCK) high or low time Master Slave	t_{WSCK}	 t_{cyc} – 30 t_{cyc} – 30	 128 t_{cyc} —	 ns ns
5	Sequential transfer delay Master Slave	t_{TD}	 1/2 1	 — —	 t_{sck} t_{cyc}
6	Data setup time (inputs) Master Slave	t_{SU}	 30 30	 — —	 ns ns
7	Data hold time (inputs) Master Slave	t_{HI}	 0 30	 — —	 ns ns
8	Slave access time	t_A	—	1	t_{cyc}
9	Slave MISO disable time	t_{DIS}	—	1	t_{cyc}
10	Data valid (after SCK edge) Master Slave	t_V	 — —	 50 50	 ns ns
11	Data hold time (outputs) Master Slave	t_{HO}	 0 0	 — —	 ns ns
12	Rise Time Input Output	 t_{RI} t_{RO}	 — —	 t_{cyc} –30 30	 ns ns
13	Fall Time Input Output	 t_{FI} t_{FO}	 — —	 t_{cyc} –30 30	 ns ns

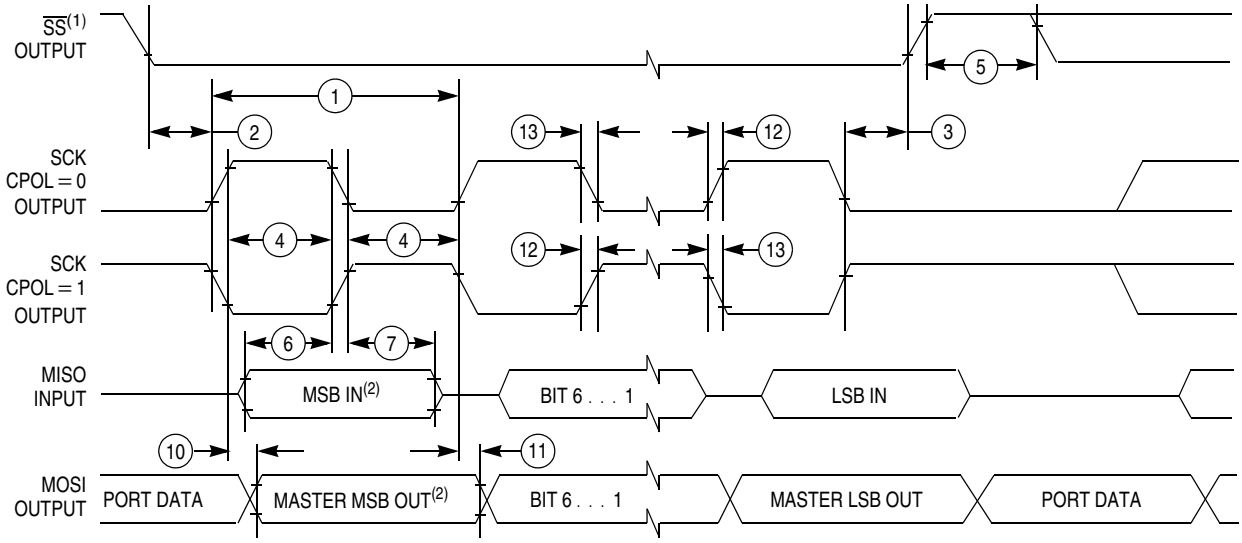
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, 200 pF load on all SPI pins, AC timing is shown with respect to 20% V_{DD} and 70% V_{DD} levels, unless otherwise noted.

Downloaded from Arrow.com.



Notes:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

A) SPI Master Timing (CPHA = 0)

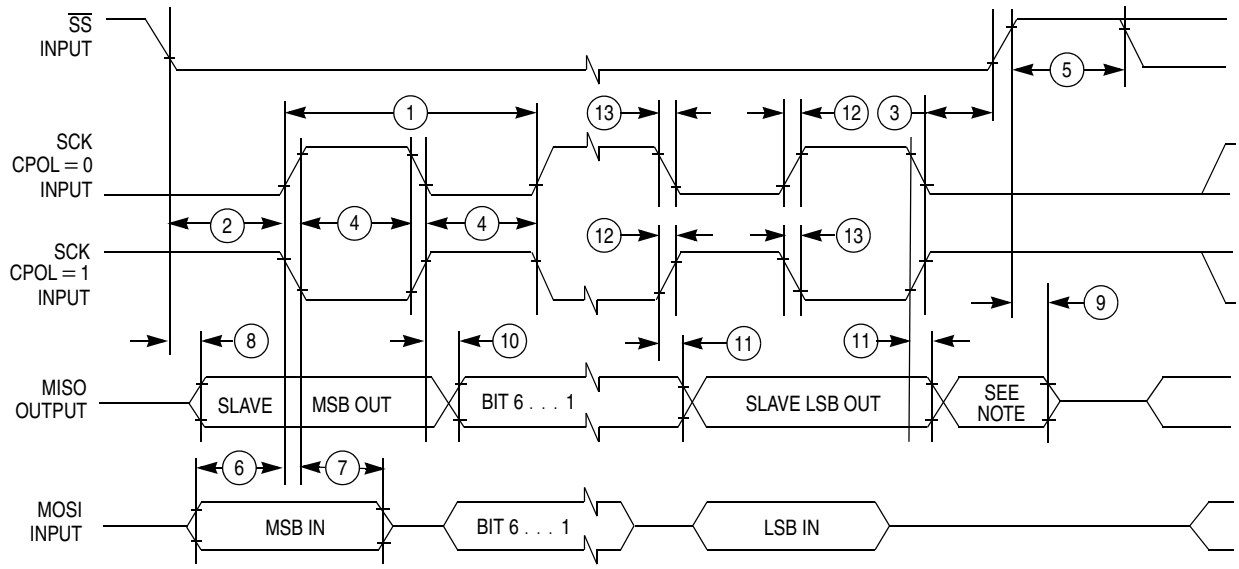


Notes:
1. $\overline{SS}$ output mode (DDS7 = 1, SSOE = 1).
2. LSBF = 0. For LSBF = 1, bit order is LSB, bit 1, ..., bit 6, MSB.

B) SPI Master Timing (CPHA = 1)

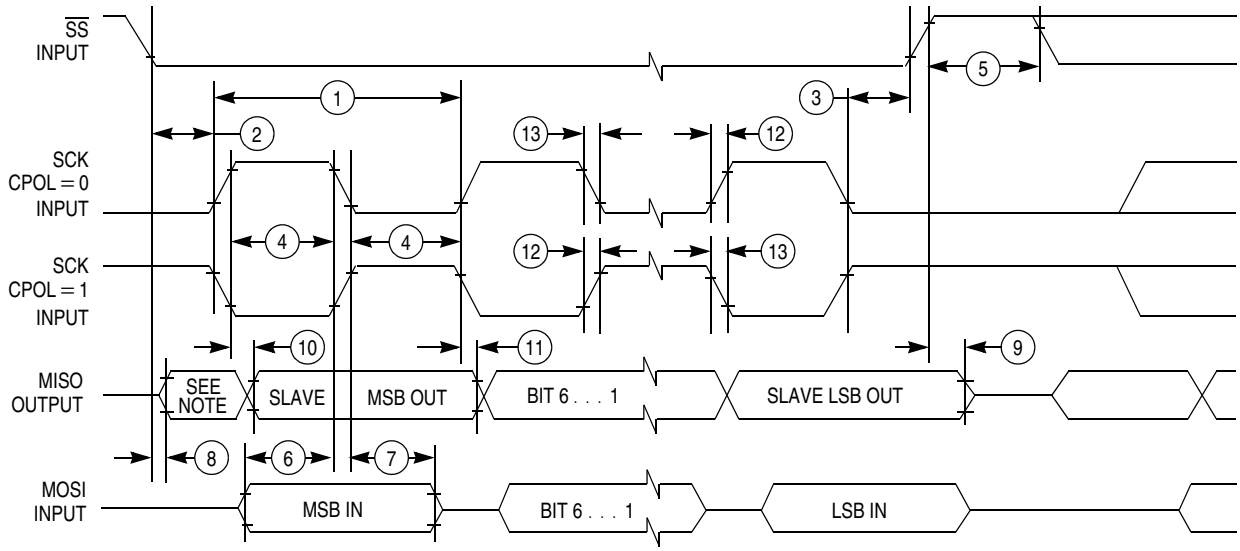
Figure 19-13. SPI Master Timing Diagram

Downloaded from Arrow.com.



Note: Not defined but normally MSB of character just received

A) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character just received

B) SPI Slave Timing (CPHA = 1)

Figure 19-14. SPI Slave Timing Diagram

Downloaded from Arrow.com.

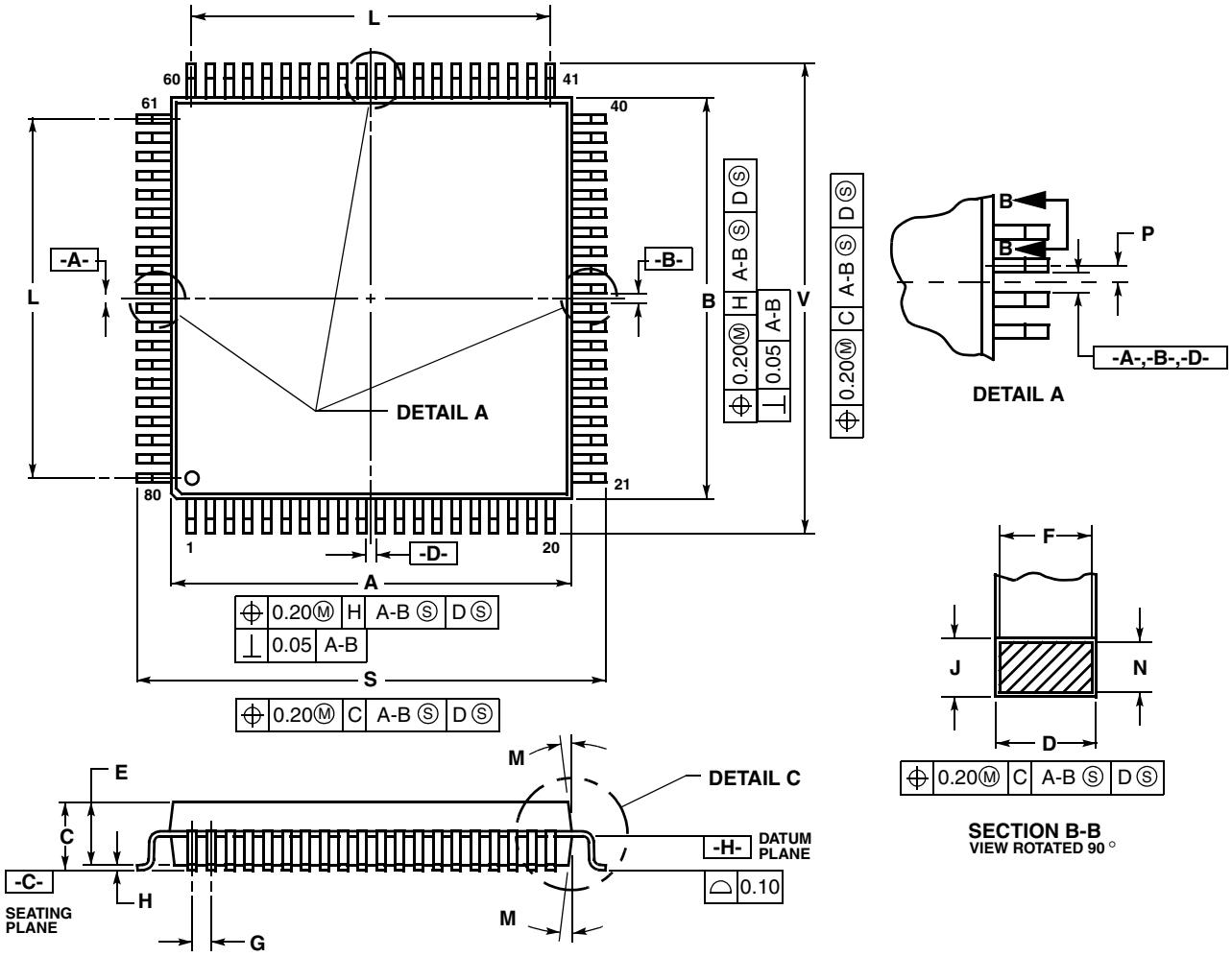
Chapter 20
Mechanical Specifications

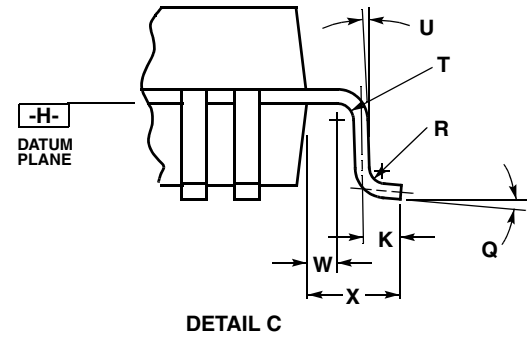
20.1 Introduction

This section provides dimensions for the 80-pin quad flat pack (QFP).

Downloaded from Arrow.com.

20.2 80-Pin Quad Flat Pack (Case 841B-02)





NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE -H- IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS -A-, -B- AND -D- TO BE DETERMINED AT DATUM PLANE -H-.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -C-.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

	MILLIMETERS	
DIM	MIN	MAX
A	13.90	14.10
B	13.90	14.10
C	2.15	2.45
D	0.22	0.38
E	2.00	2.40
F	0.22	0.33
G	0.65 BSC	
H	---	0.25
J	0.13	0.23
K	0.65	0.95
L	12.35 REF	
M	5°	10°
N	0.13	0.17
P	0.325 BSC	
Q	0°	7°
R	0.13	0.30
S	16.95	17.45
T	0.13	---
U	0°	---
V	16.95	17.45
W	0.35	0.45
X	1.6 REF	

Downloaded from Arrow.com.

Downloaded from Arrow.com.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

RoHS-compliant and/or Pb- free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb- free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale.s Environmental Products program, go to http://www.freescale.com/epp.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2004. All rights reserved.

M68HC12B
Rev. 9.1, 07/2005

freescale™ semiconductor

Downloaded from Arrow.com.