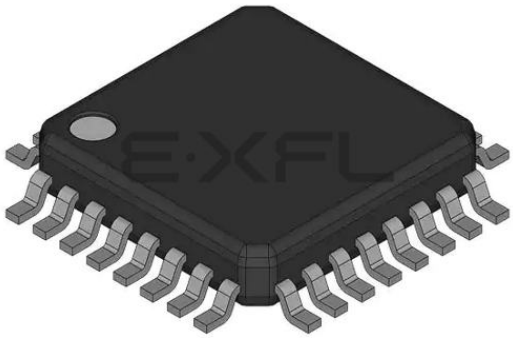


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

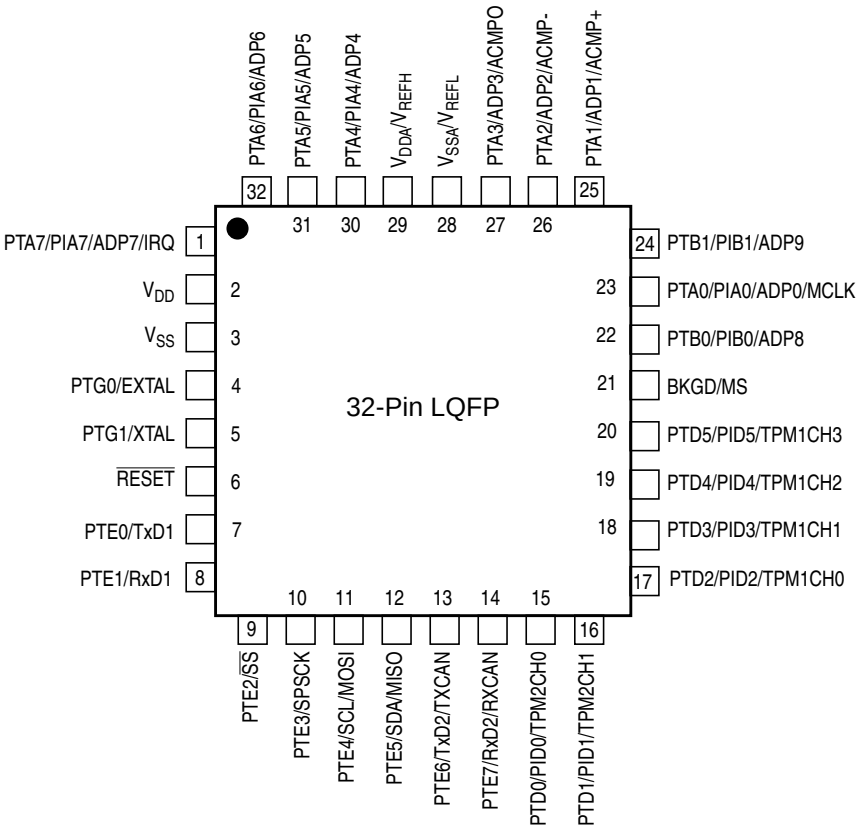
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, I²C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	48KB (48K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 10x12b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s08dv48aclc

Chapter 14
Serial Communications Interface (S08SCIV4)

Chapter 15
Real-Time Counter (S08RTCV1)



V_{REFH} and V_{REFL} are internally connected to V_{DDA} and V_{SSA}, respectively.

Figure 2-3. 32-Pin LQFP

5 ERCLKEN and EREFSTEN set in MCGC2 for, else in standby. For high frequency range (RANGE in MCGC2 set) requires the LVD to also be enabled in stop3.

6 If ENBDM is set when entering stop2, the MCU will actually enter stop3.

7 If LVDSE is set when entering stop2, the MCU will actually enter stop3.

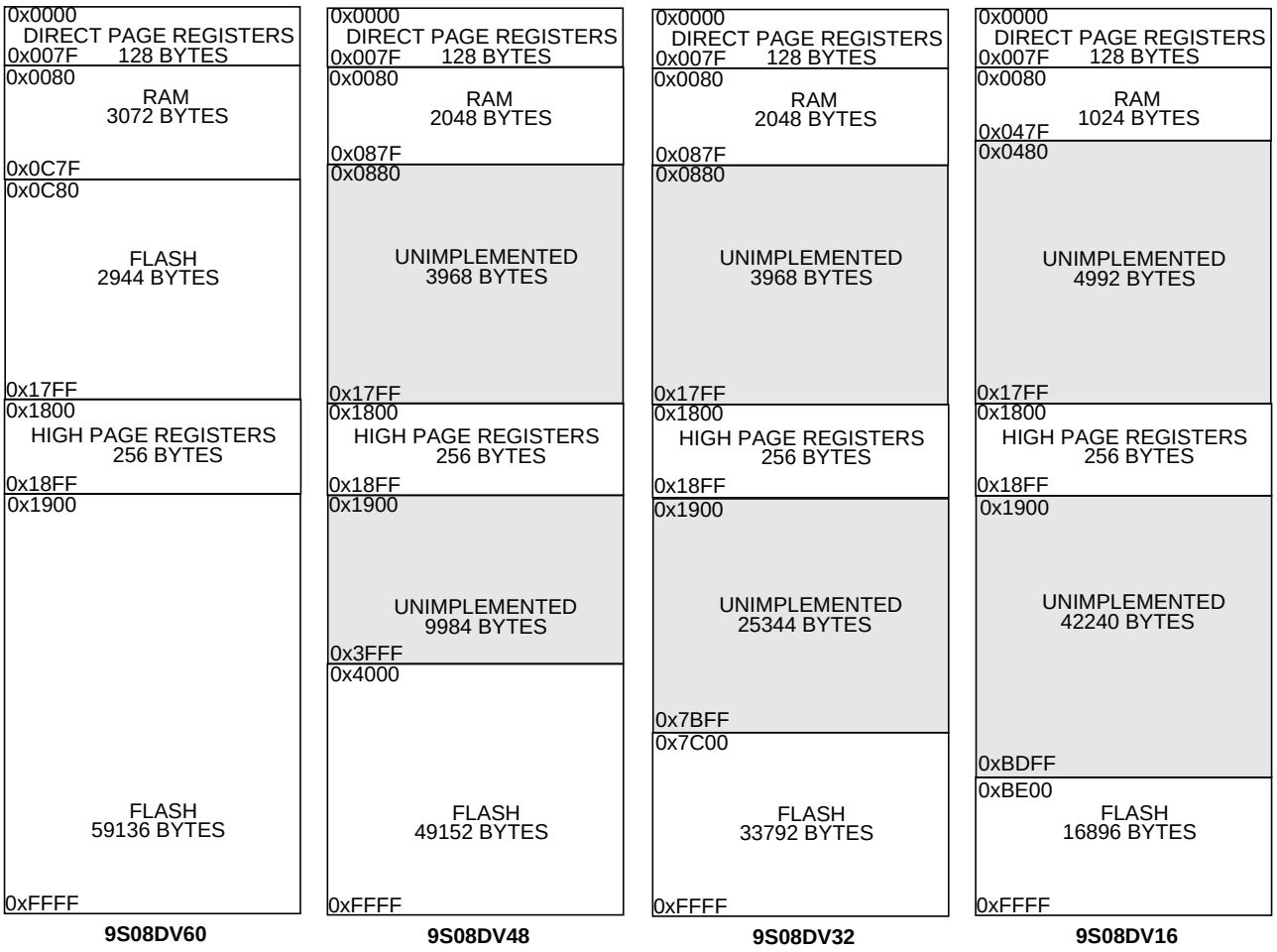


Figure 4-1. MC9S08DV60 Series Memory Map

4.2 Reset and Interrupt Vector Assignments

Table 4-1 shows address assignments for reset and interrupt vectors. The vector names shown in this table are the labels used in the MC9S08DV60 Series equate file provided by Freescale Semiconductor.

Table 4-1. Reset and Interrupt Vectors

Address (High/Low)	Vector	Vector Name
0xFFC0:0xFFC1	ACMP2	Vacmp2
0xFFC2:0xFFC3	ACMP1	Vacmp1
0xFFC4:0xFFC5	MSCAN Transmit	Vcantx
0xFFC6:0xFFC7	MSCAN Receive	Vcanrx
0xFFC8:0xFFC9	MSCAN errors	Vcanerr
0xFFCA:0xFFCB	MSCAN wake up	Vcanwu
0xFFCC:0xFFCD	RTC	Vrtc

4.5.2 Program and Erase Times

Before any program or erase command can be accepted, the Flash clock divider register (FCDIV) must be written to set the internal clock for the Flash module to a frequency (f_{FCLK}) between 150 kHz and 200 kHz (see Section 4.5.10.1, “Flash Clock Divider Register (FCDIV)”). This register can be written only once, so normally this write is performed during reset initialization. The user must ensure that FACCERR is not set before writing to the FCDIV register. One period of the resulting clock ($1/f_{FCLK}$) is used by the command processor to time program and erase pulses. An integer number of these timing pulses is used by the command processor to complete a program or erase command.

Table 4-6 shows program and erase times. The bus clock frequency and FCDIV determine the frequency of FCLK (f_{FCLK}). The time for one cycle of FCLK is $t_{FCLK} = 1/f_{FCLK}$. The times are shown as a number of cycles of FCLK and as an absolute time for the case where $t_{FCLK} = 5$ µs. Program and erase times shown include overhead for the command state machine and enabling and disabling of program and erase voltages.

Table 4-6. Program and Erase Times

Parameter	Cycles of FCLK	Time if FCLK = 200 kHz
Byte program	9	45 µs
Burst program	4	20 µs[1]
Sector erase	4000	20 ms
Mass erase	20,000	100 ms
Sector erase abort	4	20 µs[1]

1 Excluding start/end overhead

4.5.3 Program and Erase Command Execution

The FCDIV register must be initialized after any reset and any error flag is cleared before beginning command execution. The command execution steps are:

1. Write a data value to an address in the Flash array. The address and data information from this write is latched into the Flash interface. This write is a required first step in any command sequence. For erase and blank check commands, the value of the data is not important. For sector erase commands, the address can be any address in the sector of Flash to be erased. For mass erase and blank check commands, the address can be any address in the Flash .

NOTE

Before programming a particular byte in the Flash , the sector in which that particular byte resides must be erased by a mass or sector erase operation. Reprogramming bits in an already programmed byte without first performing an erase operation may disturb data stored in the Flash memory.

2. Write the command code for the desired command to FCMD. The six valid commands are blank check (0x05), byte program (0x20), burst program (0x25), sector erase (0x40), mass erase[1] (0x41), and sector erase abort (0x47). The command code is latched into the command buffer.

1. A mass erase is possible only when the Flash block is fully unprotected.

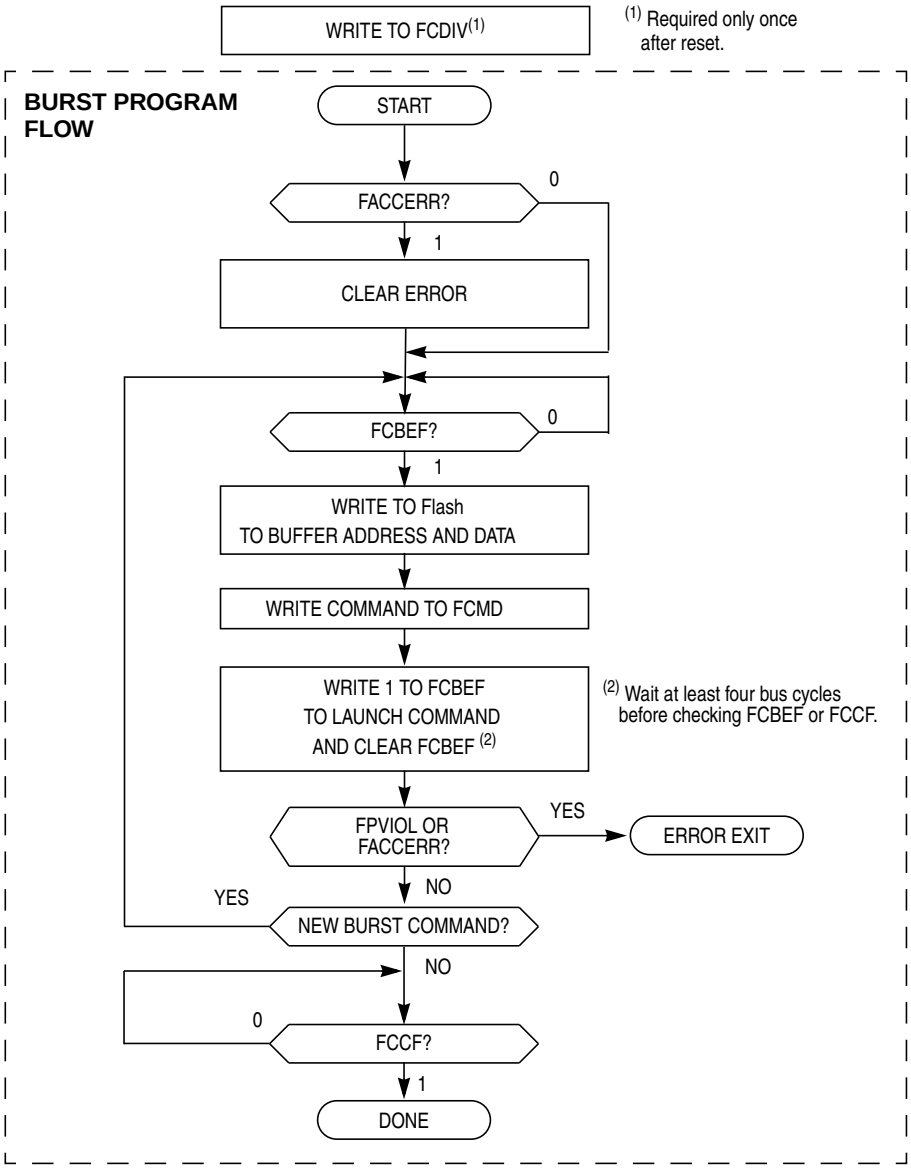


Figure 4-3. Burst Program Flowchart

6.5.6.3 Port F Pull Enable Register (PTFPE)

	7	6	5	4	3	2	1	0
R W	PTFPE7	PTFPE6	PTFPE5	PTFPE4	PTFPE3	PTFPE2	PTFPE1	PTFPE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-39. Internal Pull Enable for Port F Register (PTFPE)

Table 6-37. PTFPE Register Field Descriptions

Field	Description
7:0 PTFPE[7:0]	**Internal Pull Enable for Port F Bits** — Each of these control bits determines if the internal pull-up device is enabled for the associated PTF pin. For port F pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up device disabled for port F bit n. 1 Internal pull-up device enabled for port F bit n.

NOTE

Pull-down devices only apply when using pin interrupt functions, when corresponding edge select and pin select functions are configured.

6.5.6.4 Port F Slew Rate Enable Register (PTFSE)

	7	6	5	4	3	2	1	0
R W	PTFSE7	PTFSE6	PTFSE5	PTFSE4	PTFSE3	PTFSE2	PTFSE1	PTFSE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-40. Slew Rate Enable for Port F Register (PTFSE)

Table 6-38. PTFSE Register Field Descriptions

Field	Description
7:0 PTFSE[7:0]	**Output Slew Rate Enable for Port F Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTF pin. For port F pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port F bit n. 1 Output slew rate control enabled for port F bit n.

Note: Slew rate reset default values may differ between engineering samples and final production parts. Always initialize slew rate control to the desired value to ensure correct operation.

6.5.7 Port G Registers

Port G is controlled by the registers listed below.

6.5.7.1 Port G Data Register (PTGD)

	7	6	5	4	3	2	1	0
R	0	0	PTGD5	PTGD4	PTGD3	PTGD2	PTGD1	PTGD0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 6-42. Port G Data Register (PTGD)

Table 6-40. PTGD Register Field Descriptions

Field	Description
5:0 PTGD[5:0]	**Port G Data Register Bits** — For port G pins that are inputs, reads return the logic level on the pin. For port G pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port G pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTGD to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pull-ups disabled.

6.5.7.2 Port G Data Direction Register (PTGDD)

	7	6	5	4	3	2	1	0
R	0	0	PTGDD5	PTGDD4	PTGDD3	PTGDD2	PTGDD1	PTGDD0
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 6-43. Port G Data Direction Register (PTGDD)

Table 6-41. PTGDD Register Field Descriptions

Field	Description
5:0 PTGDD[5:0]	**Data Direction for Port G Bits** — These read/write bits control the direction of port G pins and what is read for PTGD reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port G bit n and PTGD reads return the contents of PTGDn.

— If entering FEE, set RDIV appropriately, clear the IREFS bit to switch to the external reference, and leave the CLKS bits at %00 so that the output of the FLL is selected as the system clock source.

— If entering FBE, clear the IREFS bit to switch to the external reference and change the CLKS bits to %10 so that the external reference clock is selected as the system clock source. The RDIV bits should also be set appropriately here according to the external reference frequency because although the FLL is bypassed, it is still on in FBE mode.

— The internal reference can optionally be kept running by setting the IRCLKEN bit. This is useful if the application will switch back and forth between internal and external modes. For minimum power consumption, leave the internal reference disabled while in an external clock mode.

3. After the proper configuration bits have been set, wait for the affected bits in the MCGSC register to be changed appropriately, reflecting that the MCG has moved into the proper mode.

— If ERCLKEN was set in step 1 or the MCG is in FEE, FBE, PEE, PBE, or BLPE mode, and EREFS was also set in step 1, wait here for the OSCINIT bit to become set indicating that the external clock source has finished its initialization cycles and stabilized. Typical crystal startup times are given in Appendix A, “Electrical Characteristics”.

— If in FEE mode, check to make sure the IREFST bit is cleared and the LOCK bit is set before moving on.

— If in FBE mode, check to make sure the IREFST bit is cleared, the LOCK bit is set, and the CLKST bits have changed to %10 indicating the external reference clock has been appropriately selected. Although the FLL is bypassed in FBE mode, it is still on and will lock in FBE mode.

To change from FEI clock mode to FBI clock mode, follow this procedure:

1. Change the CLKS bits to %01 so that the internal reference clock is selected as the system clock source.
2. Wait for the CLKST bits in the MCGSC register to change to %01, indicating that the internal reference clock has been appropriately selected.

8.5.2 MCG Mode Switching

When switching between operational modes of the MCG, certain configuration bits must be changed in order to properly move from one mode to another. Each time any of these bits are changed (PLLS, IREFS, CLKS, or EREFS), the corresponding bits in the MCGSC register (PLLST, IREFST, CLKST, or OSCINIT) must be checked before moving on in the application software.

Additionally, care must be taken to ensure that the reference clock divider (RDIV) is set properly for the mode being switched to. For instance, in PEE mode, if using a 4 MHz crystal, RDIV must be set to %001 (divide-by-2) or %010 (divide -by-4) in order to divide the external reference down to the required frequency between 1 and 2 MHz.

The RDIV and IREFS bits should always be set properly before changing the PLLS bit so that the FLL or PLL clock has an appropriate reference clock frequency to switch to.

10.5 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. You can configure the module for 8-, 10-, or 12-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 10-7, Table 10-8, and Table 10-9 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

10.5.1 ADC Module Initialization Example

10.5.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
2. Update status and control register 2 (ADCSC2) to select the conversion trigger (hardware or software) and compare function options, if enabled.
3. Update status and control register 1 (ADCSC1) to select whether conversions will be continuous or completed only once, and to enable or disable conversion complete interrupts. The input channel on which conversions will be performed is also selected here.

10.5.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADCCFG = 0x98 (%10011000)

```
Bit 7    ADLPC    1        Configures for low power (lowers maximum clock speed)
Bit 6:5  ADIV     00       Sets the ADCK to the input clock ÷ 1
Bit 4    ADLSMP   1        Configures for long sample time
Bit 3:2  MODE     10       Sets mode at 10-bit conversions
Bit 1:0  ADICLK   00       Selects bus clock as input clock source
```

ADCSC2 = 0x00 (%00000000)

```
Bit 7    ADACT    0        Flag indicates if a conversion is in progress
Bit 6    ADTRG    0        Software trigger selected
Bit 5    ACFE     0        Compare function disabled
Bit 4    ACFGT    0        Not used in this example
Bit 3:2           00       Reserved, always reads zero
Bit 1:0           00       Reserved for Freescale's internal use; always write zero
```

10.6.2.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 µF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

10.6.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\ lsb = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 10-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

Table 12-30. IDR1 Register Field Descriptions

Field	Description
7:5 ID[2:0]	**Standard Format Identifier —** The identifiers consist of 11 bits (ID[10:0]) for the standard format. ID10 is the most significant bit and is transmitted first on the CAN bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number. See also ID bits in Table 12-29.
4 RTR	**Remote Transmission Request** — This flag reflects the status of the Remote Transmission Request bit in the CAN frame. In the case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In the case of a transmit buffer, this flag defines the setting of the RTR bit to be sent. 0 Data frame 1 Remote frame
3 IDE	**ID Extended** — This flag indicates whether the extended or standard identifier format is applied in this buffer. In the case of a receive buffer, the flag is set as received and indicates to the CPU how to process the buffer identifier registers. In the case of a transmit buffer, the flag indicates to the MSCAN what type of identifier to send. 0 Standard format (11 bit) 1 Extended format (29 bit)

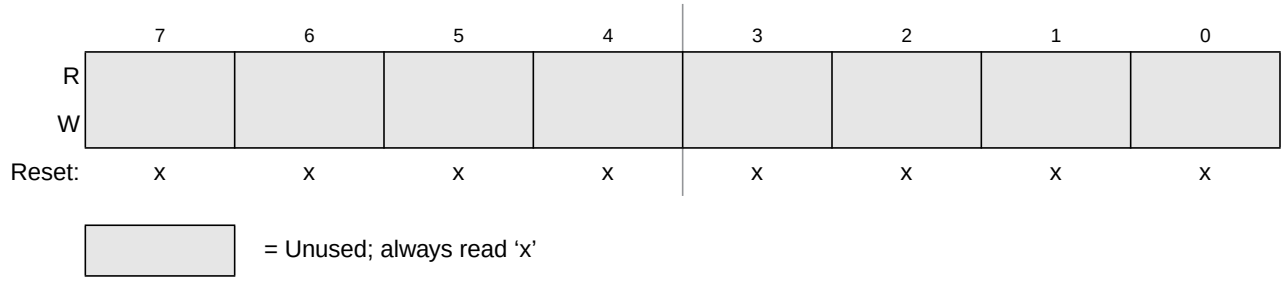


Figure 12-31. Identifier Register 2 — Standard Mapping

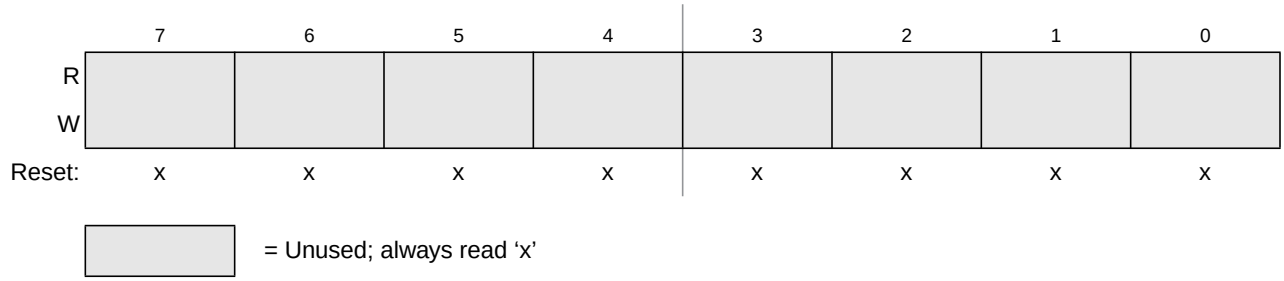


Figure 12-32. Identifier Register 3 — Standard Mapping

12.4.3 Data Segment Registers (DSR0-7)

The eight data segment registers, each with bits DB[7:0], contain the data to be transmitted or received. The number of bytes to be transmitted or received is determined by the data length code in the corresponding DLR register.

12.5.2.1 Message Transmit Background

Modern application layer software is built upon two fundamental assumptions:

- Any CAN node is able to send out a stream of scheduled messages without releasing the CAN bus between the two messages. Such nodes arbitrate for the CAN bus immediately after sending the previous message and only release the CAN bus in case of lost arbitration.
- The internal message queue within any CAN node is organized such that the highest priority message is sent out first, if more than one message is ready to be sent.

The behavior described in the bullets above cannot be achieved with a single transmit buffer. That buffer must be reloaded immediately after the previous message is sent. This loading process lasts a finite amount of time and must 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 CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme de-couples the reloading of the transmit buffer from the actual message sending and, therefore, reduces the reactiveness requirements of the CPU. Problems can arise if the sending of a message is finished while the CPU re-loads the second buffer. No buffer would then be ready for transmission, and the CAN bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN implements with the "local priority" concept described in Section 12.5.2.2, "Transmit Structures."

12.5.2.2 Transmit Structures

The MSCAN triple transmit buffer scheme optimizes real-time performance by allowing multiple messages to be set up in advance. The three buffers are arranged as shown in Figure 12-38.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see Section 12.4, "Programmer's Model of Message Storage"). An additional Section 12.4.5, "Transmit Buffer Priority Register (TBPR) contains an 8-bit local priority field (PRIO) (see Section 12.4.5, "Transmit Buffer Priority Register (TBPR)"). The remaining two bytes are used for time stamping of a message, if required (see Section 12.4.6, "Time Stamp Register (TSRH–TSRL)").

To transmit a message, the CPU must identify an available transmit buffer, which is indicated by a set transmitter buffer empty (TXEx) flag (see Section 12.3.6, "MSCAN Transmitter Flag Register (CANTFLG)"). If a transmit buffer is available, the CPU must set a pointer to this buffer by writing to the CANTBSEL register (see Section 12.3.10, "MSCAN Transmit Buffer Selection Register (CANTBSEL)"). This makes the respective buffer accessible within the CANTXFG address space (see Section 12.4, "Programmer's Model of Message Storage"). The algorithmic feature associated with the CANTBSEL register simplifies the transmit buffer selection. In addition, this scheme makes the handler software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

field of the CAN frame, is received into the next available RxBG. If the MSCAN receives an invalid message in its RxBG (wrong identifier, transmission errors, etc.) the actual contents of the buffer will be over-written by the next message. The buffer will then not be shifted into the FIFO.

When the MSCAN module is transmitting, the MSCAN receives its own transmitted messages into the background receive buffer, RxBG, but does not shift it into the receiver FIFO, generate a receive interrupt, or acknowledge its own messages on the CAN bus. The exception to this rule is in loopback mode (see Section 12.3.2, "MSCAN Control Register 1 (CANCTL1)") where the MSCAN treats its own messages exactly like all other incoming messages. The MSCAN receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN must be prepared to become a receiver.

An overrun condition occurs when all receive message buffers in the FIFO are filled with correctly received messages with accepted identifiers and another message is correctly received from the CAN bus with an accepted identifier. The latter message is discarded and an error interrupt with overrun indication is generated if enabled (see Section 12.5.7.5, "Error Interrupt"). The MSCAN remains able to transmit messages while the receiver FIFO is full, but all incoming messages are discarded. As soon as a receive buffer in the FIFO is available again, new valid messages will be accepted.

12.5.3 Identifier Acceptance Filter

The MSCAN identifier acceptance registers (see Section 12.3.11, "MSCAN Identifier Acceptance Control Register (CANIDAC)") define the acceptable patterns of the standard or extended identifier (ID[10:0] or ID[28:0]). Any of these bits can be marked 'don't care' in the MSCAN identifier mask registers (see Section 12.3.16, "MSCAN Identifier Mask Registers (CANIDMR0–CANIDMR7)").

A filter hit is indicated to the application software by a set receive buffer full flag (RXF = 1) and three bits in the CANIDAC register (see Section 12.3.11, "MSCAN Identifier Acceptance Control Register (CANIDAC)"). These identifier hit flags (IDHIT[2:0]) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. If more than one hit occurs (two or more filters match), the lower hit has priority.

A very 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 (see Bosch CAN 2.0A/B protocol specification):

- Two identifier acceptance filters, each to be applied to:
  - The full 29 bits of the extended identifier and to the following bits of the CAN 2.0B frame:
    - Remote transmission request (RTR)
    - Identifier extension (IDE)
    - Substitute remote request (SRR)
  - The 11 bits of the standard identifier plus the RTR and IDE bits of the CAN 2.0A/B messages[1]. This mode implements two filters for a full length CAN 2.0B compliant extended identifier. Figure 12-39 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

1. Although this mode can be used for standard identifiers, it is recommended to use the four or eight identifier acceptance filters for standard identifiers

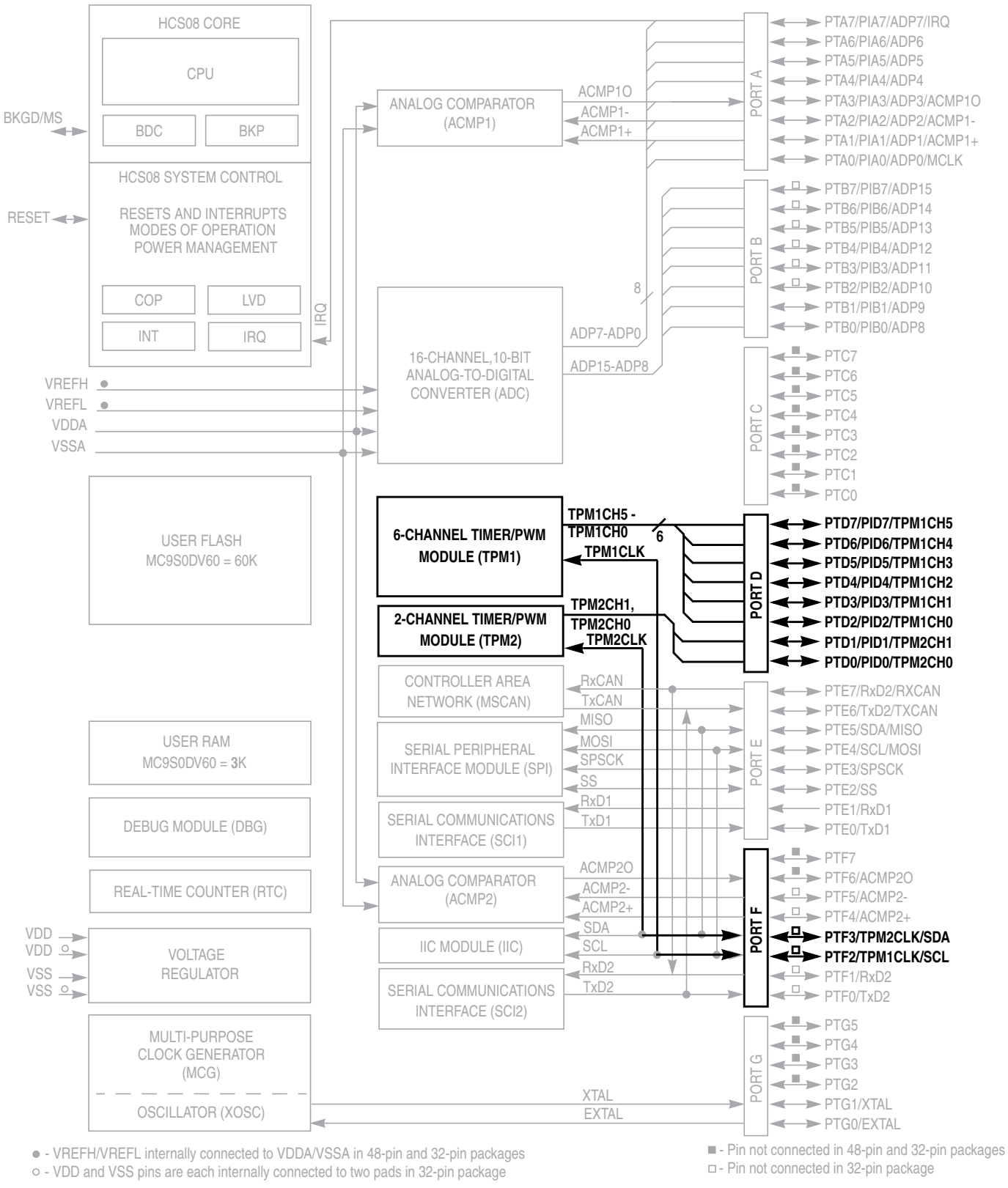


Figure 16-1. MC9S08DV60 Block Diagram

Table 17-1. BDC Command Summary

Command Mnemonic	Active BDM/ Non-intrusive	Coding Structure	Description
SYNC	Non-intrusive	n/a[1]	Request a timed reference pulse to determine target BDC communication speed
ACK_ENABLE	Non-intrusive	D5/d	Enable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
ACK_DISABLE	Non-intrusive	D6/d	Disable acknowledge protocol. Refer to Freescale document order no. HCS08RMv1/D.
BACKGROUND	Non-intrusive	90/d	Enter active background mode if enabled (ignore if ENBDM bit equals 0)
READ_STATUS	Non-intrusive	E4/SS	Read BDC status from BDCSCR
WRITE_CONTROL	Non-intrusive	C4/CC	Write BDC controls in BDCSCR
READ_BYTE	Non-intrusive	E0/AAAA/d/RD	Read a byte from target memory
READ_BYTE_WS	Non-intrusive	E1/AAAA/d/SS/RD	Read a byte and report status
READ_LAST	Non-intrusive	E8/SS/RD	Re-read byte from address just read and report status
WRITE_BYTE	Non-intrusive	C0/AAAA/WD/d	Write a byte to target memory
WRITE_BYTE_WS	Non-intrusive	C1/AAAA/WD/d/SS	Write a byte and report status
READ_BKPT	Non-intrusive	E2/RBKP	Read BDCBKPT breakpoint register
WRITE_BKPT	Non-intrusive	C2/WBKP	Write BDCBKPT breakpoint register
GO	Active BDM	08/d	Go to execute the user application program starting at the address currently in the PC
TRACE1	Active BDM	10/d	Trace 1 user instruction at the address in the PC, then return to active background mode
TAGGO	Active BDM	18/d	Same as GO but enable external tagging (HCS08 devices have no external tagging pin)
READ_A	Active BDM	68/d/RD	Read accumulator (A)
READ_CCR	Active BDM	69/d/RD	Read condition code register (CCR)
READ_PC	Active BDM	6B/d/RD16	Read program counter (PC)
READ_HX	Active BDM	6C/d/RD16	Read H and X register pair (H:X)
READ_SP	Active BDM	6F/d/RD16	Read stack pointer (SP)
READ_NEXT	Active BDM	70/d/RD	Increment H:X by one then read memory byte located at H:X
READ_NEXT_WS	Active BDM	71/d/SS/RD	Increment H:X by one then read memory byte located at H:X. Report status and data.
WRITE_A	Active BDM	48/WD/d	Write accumulator (A)
WRITE_CCR	Active BDM	49/WD/d	Write condition code register (CCR)
WRITE_PC	Active BDM	4B/WD16/d	Write program counter (PC)
WRITE_HX	Active BDM	4C/WD16/d	Write H and X register pair (H:X)
WRITE_SP	Active BDM	4F/WD16/d	Write stack pointer (SP)
WRITE_NEXT	Active BDM	50/WD/d	Increment H:X by one, then write memory byte located at H:X
WRITE_NEXT_WS	Active BDM	51/WD/d/SS	Increment H:X by one, then write memory byte located at H:X. Also report status.

[1] The SYNC command is a special operation that does not have a command code.

Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

B.2.1 Timer Status and Control Register (TPMxSC)

TPMxSC contains the overflow status flag and control bits that are used to configure the interrupt enable, TPM configuration, clock source, and prescale divisor. These controls relate to all channels within this timer module.

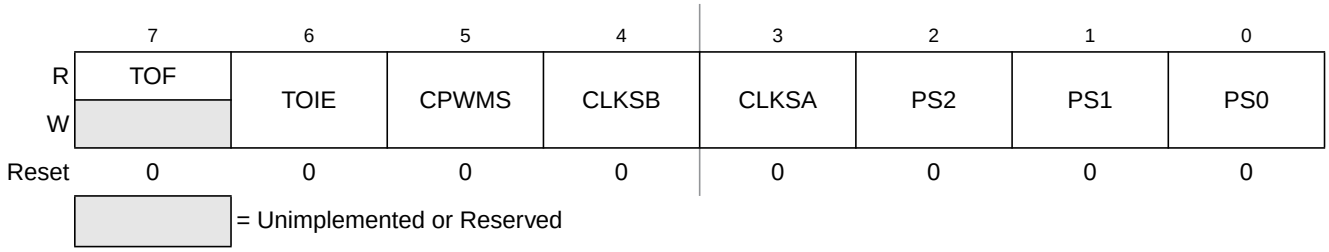


Figure B-2. Timer Status and Control Register (TPMxSC)

Table B-1. TPMxSC Register Field Descriptions

Field	Description
7 TOF	**Timer Overflow Flag** — This flag is set when the TPM counter changes to 0x0000 after reaching the modulo value programmed in the TPM counter modulo registers. When the TPM is configured for CPWM, TOF is set after the counter has reached the value in the modulo register, at the transition to the next lower count value. Clear TOF by reading the TPM status and control register when TOF is set and then writing a 0 to TOF. If another TPM overflow occurs before the clearing sequence is complete, the sequence is reset so TOF would remain set after the clear sequence was completed for the earlier TOF. Reset clears TOF. Writing a 1 to TOF has no effect. 0 TPM counter has not reached modulo value or overflow 1 TPM counter has overflowed
6 TOIE	**Timer Overflow Interrupt Enable** — This read/write bit enables TPM overflow interrupts. If TOIE is set, an interrupt is generated when TOF equals 1. Reset clears TOIE. 0 TOF interrupts inhibited (use software polling) 1 TOF interrupts enabled
5 CPWMS	**Center-Aligned PWM Select** — This read/write bit selects CPWM operating mode. Reset clears this bit so the TPM operates in up-counting mode for input capture, output compare, and edge-aligned PWM functions. Setting CPWMS reconfigures the TPM to operate in up-/down-counting mode for CPWM functions. Reset clears CPWMS. 0 All TPMx channels operate as input capture, output compare, or edge-aligned PWM mode as selected by the MSnB:MSnA control bits in each channel's status and control register 1 All TPMx channels operate in center-aligned PWM mode
4:3 CLKS[B:A]	**Clock Source Select** — As shown in Table B-2, this 2-bit field is used to disable the TPM system or select one of three clock sources to drive the counter prescaler. The external source and the XCLK are synchronized to the bus clock by an on-chip synchronization circuit.
2:0 PS[2:0]	**Prescale Divisor Select** — This 3-bit field selects one of eight divisors for the TPM clock input as shown in Table B-3. This prescaler is located after any clock source synchronization or clock source selection, so it affects whatever clock source is selected to drive the TPM system.