Details	
Product Status	Active
Core Processor	e200z0h
Core Size	32-Bit Single-Core
Speed	64MHz
Connectivity	CANbus, LINbus, SCI, SPI
Peripherals	DMA, POR, PWM, WDT
Number of I/O	79
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 16
RAM Size	20K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/spc5602pef0vll6

1.2 Target applications

The MPC5602P belongs to an expanding range of automotive-focused products designed to address and target the following chassis and safety market segments:

- Electric hydraulic power steering (EHPS)
- Lower end of electric power steering (EPS)
- Airbag applications
- Anti-lock braking systems (ABS)
- Motor control applications

EHPS and EPS systems typically feature sophisticated and advanced electrical motor control periphery with special enhancements in the area of pulse width modulation, highly flexible timers, and functional safety.

1.2.1 Application examples

1.2.1.1 Electric power steering

Figure 1-1 outlines a typical electric power steering application built around the MPC5602P microcontroller.

3.2 64-pin LQFP pinout

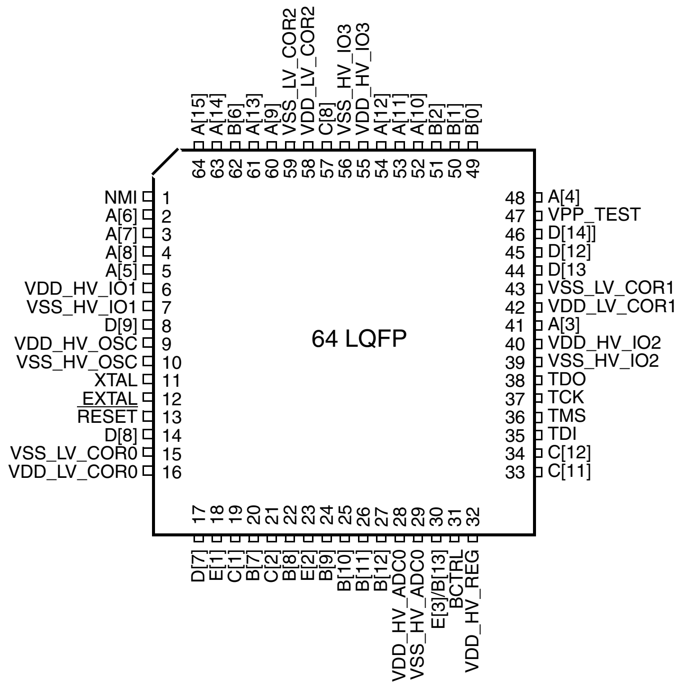

Figure 3-2. 64-pin LQFP pinout(top view)

3.3 Pin description

The following sections provide signal descriptions and related information about the functionality and configuration of the MPC5602P devices.

3.3.1 Power supply and reference voltage pins

Table 3-1 lists the power supply and reference voltage for the MPC5602P devices.

5.5.3 System Clock Select Status Register (CGM_SC_SS)

Address 0xC3FE_0378 Access: User read, Supervisor read, Test read

R	0	0	0	0	SELSTAT				0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5-4. System Clock Select Status Register (CGM_SC_SS)

This register provides the current system clock source selection.

Table 5-5. System Clock Select Status Register (CGM_SC_SS) Field Descriptions

Field	Description
SELSTAT	**System Clock Source Selection Status** — This value indicates the current source for the system clock. 0000 16 MHz int. RC osc. 0001 reserved 0010 4 MHz crystal osc. 0011 reserved 0100 system PLL 0101 reserved 0110 reserved 0111 reserved 1000 reserved 1001 reserved 1010 reserved 1011 reserved 1100 reserved 1101 reserved 1110 reserved 1111 system clock is disabled

5.5.4 System Clock Divider Configuration Register (CGM_SC_DC0)

Address 0xC3FE_037C Access: User read, Supervisor read/write, Test read/write

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DE0	0	0	0	DIV0				0	0	0	0	0	0	0	0
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

Figure 5-5. System Clock Divider Configuration Register (CGM_SC_DC0)

This register controls the system clock divider.

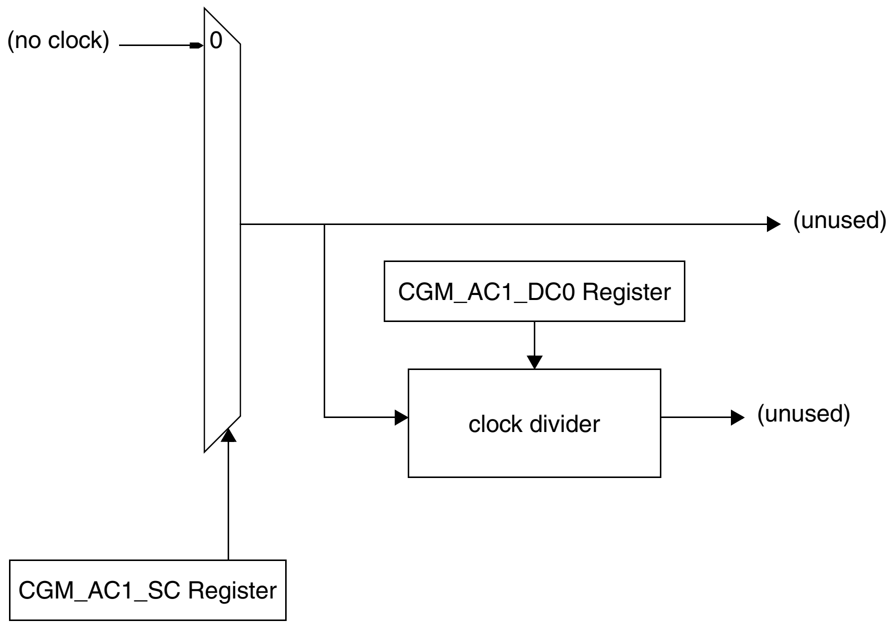

Figure 5-14. MC_CGM Auxiliary Clock 1 Generation Overview

6.1.2 Features

The MC_PCU includes the following features:

- maps the PMU registers to the MC_PCU address space

6.2 External Signal Description

The MC_PCU has no connections to any external pins.

6.3 Memory Map and Register Definition

6.3.1 Memory Map

Table 6-1. MC_PCU Register Description

Address	Name	Description	Size	Access			Location
				User	Supervisor	Test	
0xC3FE _8040	PCU_PSTAT	Power Domain Status Register	word	read	read	read	on page 127

NOTE

Any access to unused registers as well as write accesses to read-only registers will:

- not change register content
- cause a transfer error

Table 6-2. MC_PCU Memory Map

0xC3FE _80004 … 0xC3FE _803C	reserved																	
0xC3FE _8040	PCU_PSTAT	R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		W																
		R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PD0
		W																
0x044 … 0x07C	reserved																	

Chapter 8 Reset Generation Module (MC_RGM)

8.1 Introduction

8.1.1 Overview

The reset generation module (MC_RGM) centralizes the different reset sources and manages the reset sequence of the device. It provides a register interface and the reset sequencer. Various registers are available to monitor and control the device reset sequence. The reset sequencer is a state machine which controls the different phases (PHASE0, PHASE1, PHASE2, PHASE3, and IDLE) of the reset sequence and controls the reset signals generated in the system.

Figure 8-1 depicts the MC_RGM block diagram.

It should be noted that while the EEGR is associated with the RAM, similar capabilities exist for the flash, that is, the ability to program the non-volatile memory with single- or double-bit errors is supported for the same two reasons previously identified.

For both types of memories (RAM and flash), the intent is to generate errors during data write cycles, such that subsequent reads of the corrupted address locations generate ECC events, either single-bit corrections or double-bit non-correctable errors that are terminated with an error response.

The enabling of these error generation modes requires the same input enable signal (as that used to enable single-bit correction reporting) be asserted. This signal is tied to 1 at SoC level and hence reporting of single-bit memory corrections is always enabled.

Address Base + 0x004A Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	FRC 1BI	FR11 BI	0	0	FRC NCI	FR1 NCI	0	ERRBIT[6:0]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 15-11. ECC Error Generation register (EEGR)

Table 15-12. EEGR field descriptions

Field	Description
2 FRC1BI	Force RAM Continuous 1-Bit Data Inversions 0 No RAM continuous 1-bit data inversions generated 1 1-bit data inversions in the RAM continuously generated The assertion of this bit forces the RAM controller to create 1-bit data inversions, as defined by the bit position specified in ERRBIT[6:0], continuously on every write operation. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM. After this bit has been enabled to generate another continuous 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic. This bit can only be set if the same input enable signal (as that used to enable single-bit correction reporting) is asserted. This signal is tied to 1 at SoC level and hence reporting of single-bit memory corrections is always enabled.
3 FR11BI	Force RAM One 1-bit Data Inversion 0 No RAM single 1-bit data inversion generated 1 One 1-bit data inversion in the RAM generated The assertion of this bit forces the RAM controller to create one 1-bit data inversion, as defined by the bit position specified in ERRBIT[6:0], on the first write operation after this bit is set. The normal ECC generation takes place in the RAM controller, but then the polarity of the bit position defined by ERRBIT is inverted to introduce a 1-bit ECC event in the RAM. After this bit has been enabled to generate a single 1-bit data inversion, it must be cleared before being set again to properly re-enable the error generation logic. This bit can only be set if the same input enable signal (as that used to enable single-bit correction reporting) is asserted. This signal is tied to 1 at SoC level and hence reporting of single-bit memory corrections is always enabled.

Table 22-5. Message Buffer structure field description (continued)

Field	Description
TIME STAMP	Free-Running Counter Time Stamp This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.
PRIO	Local priority This 3-bit field is only used when LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Section 22.4.3, “Arbitration process.
ID	Frame Identifier In Standard Frame format, only the 11 most significant bits (3 to 13) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.
DATA	Data Field As many as 8 bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 22-6. Message buffer code for Rx buffers

Rx Code BEFORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0000	INACTIVE: MB is not active.	–	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Section 22.4.5, “Matching process for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Section 22.4.5, “Matching process for details about overrun behavior.

Table 22-11. MCR field descriptions

Field	Description
0 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management submodules. This is the only bit in MCR not affected by soft reset. See Section 22.4.9.2, "Module disable mode for more information. 0 Enable the FlexCAN module. 1 Disable the FlexCAN module.
1 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when the HALT bit in the MCR is set or when Debug Mode is requested at MCU level. When FRZ is asserted, FlexCAN is able to enter Freeze Mode. Negation of this bit field causes FlexCAN to exit from Freeze Mode. 0 Not able to enter Freeze Mode. 1 Able to enter Freeze Mode.
2 FEN	FIFO Enable This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region (0x80–0xFF) is used by the FIFO engine. See Section 22.3.3, "Rx FIFO structure and Section 22.4.7, "Rx FIFO for more information. 0 FIFO disabled. 1 FIFO enabled.
3 HALT	Halt FlexCAN Assertion of this bit puts the FlexCAN module into Freeze Mode. The CPU should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FlexCAN before this bit is cleared. While in Freeze Mode, the CPU has write access to the Error Counter Register, that is otherwise read-only. Freeze Mode can not be entered while FlexCAN is in any of the low power modes. See Section 22.4.9.1, "Freeze mode for more information. 0 No Freeze Mode request. 1 Enters Freeze Mode if the FRZ bit is asserted.
4 NOT_RDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable Mode, Stop Mode or Freeze Mode. It is negated once FlexCAN has exited these modes. 0 FlexCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode. 1 FlexCAN module is either in Disable Mode, Stop Mode or Freeze Mode.
5 WAK_MSK	Wake Up Interrupt Mask This bit enables the Wake Up Interrupt generation. 0 Wake Up Interrupt is disabled. 1 Wake Up Interrupt is enabled.

Table 22-11. MCR field descriptions (continued)

Field	Description
6 SOFT_RST	Soft Reset When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR, IMASK1, IFLAG1. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: • CTRL • RXIMR0–RXIMR31 • RXGMASK, RX14MASK, RX15MASK • all Message Buffers The SOFT_RST bit can be asserted directly by the CPU when it writes to the MCR, but it is also asserted when global soft reset is requested at MCU level. Since soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied. 0 No reset request. 1 Resets the registers marked as “affected by soft reset” in Table 22-2.
7 FRZ_ACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FlexCAN has actually entered Freeze Mode. If Freeze mode request is negated, then this bit is negated once the FlexCAN prescaler is running again. If Freeze mode is requested while FlexCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Section 22.4.9.1, “Freeze mode for more information. 0 FlexCAN not in Freeze mode, prescaler running. 1 FlexCAN in Freeze mode, prescaler stopped.
8 SUPV	Supervisor Mode This bit configures some of the FlexCAN registers to be either in Supervisor or Unrestricted memory space. The registers affected by this bit are marked as S/U in the Access Type column of Table 22-2. The reset value of this bit is 1, so the affected registers start with Supervisor access restrictions. 0 Affected registers are in Unrestricted memory space. 1 Affected registers are in Supervisor memory space. Any access without supervisor permission behaves as though the access was done to an unimplemented register location.
10 WRN_EN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters. 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96.

Table 22-23. RXIMR0–RXIMR31 addresses (continued)

Address	Register	Address	Register
Base + 0x08A0	RXIMR8	Base + 0x08E0	RXIMR24
Base + 0x08A4	RXIMR9	Base + 0x08E4	RXIMR25
Base + 0x08A8	RXIMR10	Base + 0x08E8	RXIMR26
Base + 0x08AC	RXIMR11	Base + 0x08EC	RXIMR27
Base + 0x08B0	RXIMR12	Base + 0x08F0	RXIMR28
Base + 0x08B4	RXIMR13	Base + 0x08F4	RXIMR29
Base + 0x08B8	RXIMR14	Base + 0x08F8	RXIMR30
Base + 0x08BC	RXIMR15	Base + 0x08FC	RXIMR31

  - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count “rolls over” to zero.

The external inputs to each counter/timer are shareable among each of the six channels within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be “captured”
- They can be used to generate interrupt requests

The polarity of the external inputs is selectable.

The primary output of each channel is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is programmable.

Any channel can be assigned as a Master. A master’s compare signal can be broadcast to the other channels within the module. The other channels can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a Master channel’s compare event occurs.

26.7.2 Counting modes

The selected external signals are sampled at the eTimer’s base clock rate and then run through a transition detector. The maximum count rate is one-half of the eTimer’s base clock rate when using an external signal. Internal clock sources can be used to clock the counters at the eTimer’s base clock rate.

If a counter is programmed to count to a specific value and then stop, the CNTMODE field in the CTRL1 register is cleared when the count terminates.

26.7.2.1 STOP mode

When the CNTMODE field is set to 000, the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

26.7.2.2 COUNT mode

When the CNTMODE field is set to 001, the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as “widgets” on a conveyor belt passing a sensor. If the selected input is inverted by setting the PIPS bit, then the negative edge of the selected external input signal is counted.

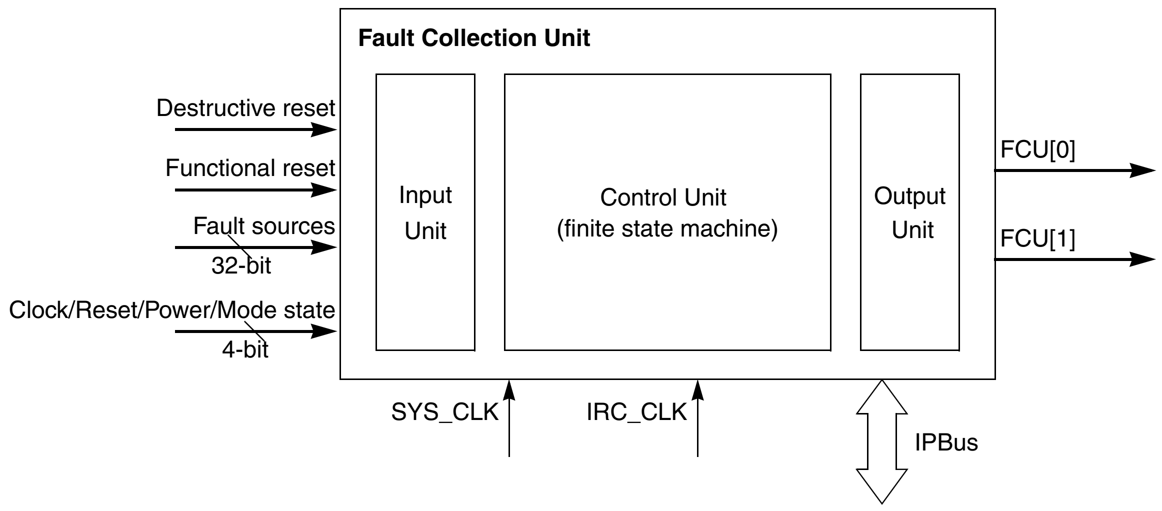

Figure 28-1. Fault Collection Unit (FCU) block diagram

Figure 28-2 shows the flow chart of FCU fault handling.

The FCU module and its state machine run on the system clock, making the two modules synchronous. The high speed RC clock (16 MHz) is used only in the Alarm state, in order to compute a deterministic timeout.

- Timers can generate DMA trigger pulses to initiate DMA transfers with other peripherals (ex: initiate a SPI message transfer sequence)
- Timers can generate interrupts
- All interrupts are maskable
- Independent timeout periods for each timer

30.2 Signal description

The PIT module has no external pins.

30.3 Memory map and registers description

This section provides a detailed description of all registers accessible in the PIT module.

30.3.1 Memory map

Table 30-1 gives an overview on all PIT registers.

Table 30-1. PIT memory map

Offset from PIT_BASE (0xC3FF_0000)	Register	Location
0x0000	PITMCR—PIT Module Control Register	on page 791
0x0004–0x00FF	Reserved	
Timer Channel 0		
0x0100	LDVAL0—Timer 0 Load Value Register	on page 792
0x0104	CVAL0—Timer 0 Current Value Register	on page 792
0x0108	TCTRL0—Timer 0 Control Register	on page 793
0x010C	TFLG0—Timer 0 Flag Register	on page 794
Timer Channel 1		
0x0110	LDVAL1—Timer 1 Load Value Register	on page 792
0x0114	CVAL1—Timer 1 Current Value Register	on page 792
0x0118	TCTRL1—Timer 1 Control Register	on page 793
0x011C	TFLG1—Timer 1 Flag Register	on page 794
Timer Channel 2		
0x0120	LDVAL2—Timer 2 Load Value Register	on page 792
0x0124	CVAL2—Timer 2 Current Value Register	on page 792
0x0128	TCTRL2—Timer 2 Control Register	on page 793
0x012C	TFLG2—Timer 2 Flag Register	on page 794

Table 33-6. Fields of SSCM STATUS register used by BAM

Field	Description
BMODE [2:0]	Device Boot Mode 000 Test Flash/autobaud_scan 001 CAN Serial Boot Loader 010 SCI Serial Boot Loader 011 Single Chip 100–111Reserved This field is updated only during reset.

Then, the initial device configuration is restored and the code jumps to the address of downloaded code. At this point BAM has just finished its task.

If an error occurs, (e.g., communication error, wrong boot selected, etc.), the BAM restores the default configuration and puts the device into static mode. Static mode means the device enters the low power mode SAFE and the processor executes a wait instruction. This is needed if the device cannot boot in the selected mode. During BAM execution and after, the mode reported by the field S_CURRENT_MODE of the register ME_GS in the module ME Module is “DRUN”.

33.5.5.3 BAM resources

BAM uses/initializes the following MCU resources:

- ME and CGM modules to initialize mode and clock sources
- CAN_0, LINFlex_0, and their pads when performing serial boot mode
- SSCM to check the boot mode and during password check (see Table 33-6 and Figure 33-5)
- External oscillator

The following hardware resources are used only when autobaud feature is selected:

- STM to measure the baud rate
- CMU to measure the external clock frequency related to the internal RC clock source
- FMPLL to work with system clock near the maximum allowed frequency (this to have higher resolution during baud rate measurement).

As already mentioned, the initial configuration is restored before executing the downloaded code.

When the autobaud feature is disabled, the system clock is selected directly from the external oscillator. Thus the oscillator frequency defines baud rates for serial interfaces used to download the user application (see Table 33-7).

Table 33-7. Serial boot mode without autobaud—baud rates

Crystal frequency (MHz)	LINFlex baud rate (baud)	FlexCAN bit rate (bit/s)
f_{extal}	f_{extal} / 833	f_{extal} / 40
8	9600	200 K
12	14400	300 K

with MSR_{DE}. Hardware-owned resources which set DBSR bits when $DBCR0_{EDM}$=1 will cause an entry into debug mode. DBERC0 is read-only by software. When resource sharing is enabled, ($DBCR0_{EDM}$=1 and $DBERC0_{IDM}$=1), only software-owned resources may be modified by software, and all status bits associated with hardware-owned resources will be forced to ‘0’ in DBSR when read by software via a **mfspr** instruction. Hardware always has full access to all registers and all register fields through the OnCE register access mechanism, and it is up to the debug firmware to properly implement modifications to these registers with read-modify-write operations to implement any control sharing with software. Settings in DBERC0 should be considered by the debug firmware in order to preserve software settings of control and status registers as appropriate when hardware modifications to the debug registers is performed.

36.9.4.1 Simultaneous Hardware and Software Debug Event Handing

Since it is possible that a hardware “owned” resource can produce a debug event in conjunction with a software-owned resource producing a different debug event simultaneously, a priority ordering mechanism is implemented which guarantees that the hardware event is handled as soon as possible, while preserving the recognition of the software event. The CPU will give highest priority to the software event initially in order to reach a recoverable boundary, and then will give highest priority to the hardware event in order to enter debug mode as near the point of event occurrence as possible. This is implemented by allowing software exception handing to begin internal to the CPU and to reach the point where the current program counter and MSR values have been saved into DSRR0/1, and the new PC pointing to the debug interrupt handler, along with the new MSR updates. At this point, hardware priority takes over, and the CPU enters debug mode.

Figure 36-2 shows the e200z0h debug resources.

Table 36-6 provides bit definitions for the Debug Status Register.

Table 36-6. DBSR Bit Definitions

Bit(s)	Name	Description
0	IDE	Imprecise Debug Event Set to ‘1’ if MSR$_{DE}$=0, DBCR0$_{IDM}$=1 and a debug event causes its respective Debug Status Register bit to be set to ‘1’. If DBERC0$_{IDM}$=0, it may also be set to ‘1’ if DBCR0$_{EDM}$=1 and an imprecise debug event occurs due to a DAC event on a load or store which is terminated with errorIf DBCR0$_{EDM}$=1 and DBERC0$_{IDM}$=1, this bit is “owned” by software. The hardware debugger will not be informed directly of an imprecise event.
1	UDE	Unconditional Debug Event Set to ‘1’ if an Unconditional debug event occurred.
2:3	MRR	Most Recent Reset. 00 – No reset occurred since these bits were last cleared by software 01 – A hard reset occurred since these bits were last cleared by software 10 – Reserved 11 – Reserved
4	ICMP	Instruction Complete Debug Event Set to ‘1’ if an Instruction Complete debug event occurred.
5	BRT	Branch Taken Debug Event Set to ‘1’ if an Branch Taken debug event occurred.
6	IRPT	Interrupt Taken Debug Event Set to ‘1’ if an Interrupt Taken debug event occurred.
7	TRAP	Trap Taken Debug Event Set to ‘1’ if a Trap Taken debug event occurred.
8	IAC1	Instruction Address Compare 1 Debug Event Set to ‘1’ if an IAC1 debug event occurred.
9	IAC2	Instruction Address Compare 2 Debug Event Set to ‘1’ if an IAC2 debug event occurred.
10	IAC3	Instruction Address Compare 3 Debug Event Set to ‘1’ if an IAC3 debug event occurred.
11	IAC4	Instruction Address Compare 4 Debug Event Set to ‘1’ if an IAC4 debug event occurred.
12	DAC1R	Data Address Compare 1 Read Debug Event Set to ‘1’ if a read-type DAC1 debug event occurred while DBCR0$_{DAC1}$=0b10 or DBCR0$_{DAC1}$=0b11
13	DAC1W	Data Address Compare 1 Write Debug Event Set to ‘1’ if a write-type DAC1 debug event occurred while DBCR0$_{DAC1}$=0b01 or DBCR0$_{DAC1}$=0b11
14	DAC2R	Data Address Compare 2 Read Debug Event Set to ‘1’ if a read-type DAC2 debug event occurred while DBCR0$_{DAC2}$=0b10 or DBCR0$_{DAC2}$=0b11
15	DAC2W	Data Address Compare 2 Write Debug Event Set to ‘1’ if a write-type DAC2 debug event occurred while DBCR0$_{DAC2}$=0b01 or DBCR0$_{DAC2}$=0b11

Appendix A Registers Under Protection

For MPC5602P, the Register Protection module is operable on the registers listed in Table A-1.

Table A-1. Registers under protection

Module	Register	Register size (bits)	Register offset	Protected bitfields
Code Flash—Base address: 0xC3F8_8000 4 registers to protect				
Code Flash	MCR	32	0x0000	bits[0:31]
Code Flash	PFCR0	32	0x001C	bits[0:31]
Code Flash	PFCR1	32	0x0020	bits[0:31]
Code Flash	PFAPR	32	0x0024	bits[0:31]
Data Flash—Base address: 0xC3F8_C000 1 register to protect				
Data Flash	MCR	32	0x0000	bits[0:31]
SIU lite—Base address: 0xC3F9_0000 97 registers to protect				
SIUL	IRER	32	0x0018	bits[0:31]
SIUL	IREER	32	0x0028	bits[0:31]
SIUL	IFEER	32	0x002C	bits[0:31]
SIUL	IFER	32	0x0030	bits[0:31]
SIUL	PCR0	16	0x0040	bits[0:15]
SIUL	PCR1	16	0x0042	bits[0:15]
SIUL	PCR2	16	0x0044	bits[0:15]
SIUL	PCR3	16	0x0046	bits[0:15]
SIUL	PCR4	16	0x0048	bits[0:15]
SIUL	PCR5	16	0x004A	bits[0:15]
SIUL	PCR6	16	0x004C	bits[0:15]
SIUL	PCR7	16	0x004E	bits[0:15]
SIUL	PCR8	16	0x0050	bits[0:15]
SIUL	PCR9	16	0x0052	bits[0:15]
SIUL	PCR10	16	0x0054	bits[0:15]
SIUL	PCR11	16	0x0056	bits[0:15]
SIUL	PCR12	16	0x0058	bits[0:15]
SIUL	PCR13	16	0x005A	bits[0:15]