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	HCS12X
Core Size	16-Bit
Speed	80MHz
Connectivity	EBI/EMI, I²C, IrDA, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	91
Program Memory Size	256KB (256K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	16K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	112-LQFP
Supplier Device Package	112-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s12xa256val

Table 1-2. Device Internal Resources (see Figure 1-3)

Device	RAMSIZE/ RAM_LOW	EEPROMSIZE/ EEPROM_LOW	FLASHSIZE/ FLASH_LOW
9S12XDP512	32K 0x0F_8000	4K 0x13_F000	512K 0x78_0000
9S12XDT512	20K 0x0F_B000	4K 0x13_F000	512K 0x78_0000
9S12XA512	32K 0x0F_8000	4K 0x13_F000	512K 0x78_0000
9S12XDG128	12K 0x0F_D000	2K 0x13_F800	128K 7E_0000
3S12XDG128	12K 0x0F_D000	2K 0x13_F800	128K 7E_0000
9S12XD128	8K 0x0F_E000	2K 0x13_F800	128K 7E_0000
9S12XD64	4K 0x0F_F000	1K 0x13_FC00	64K 7F_0000
9S12XB128	6K 0x0F_E800	1K 0x13_FC00	128K 7E_0000
9S12XA128	12K 0x0F_D000	2K 0x13_F800	128K 7E_0000

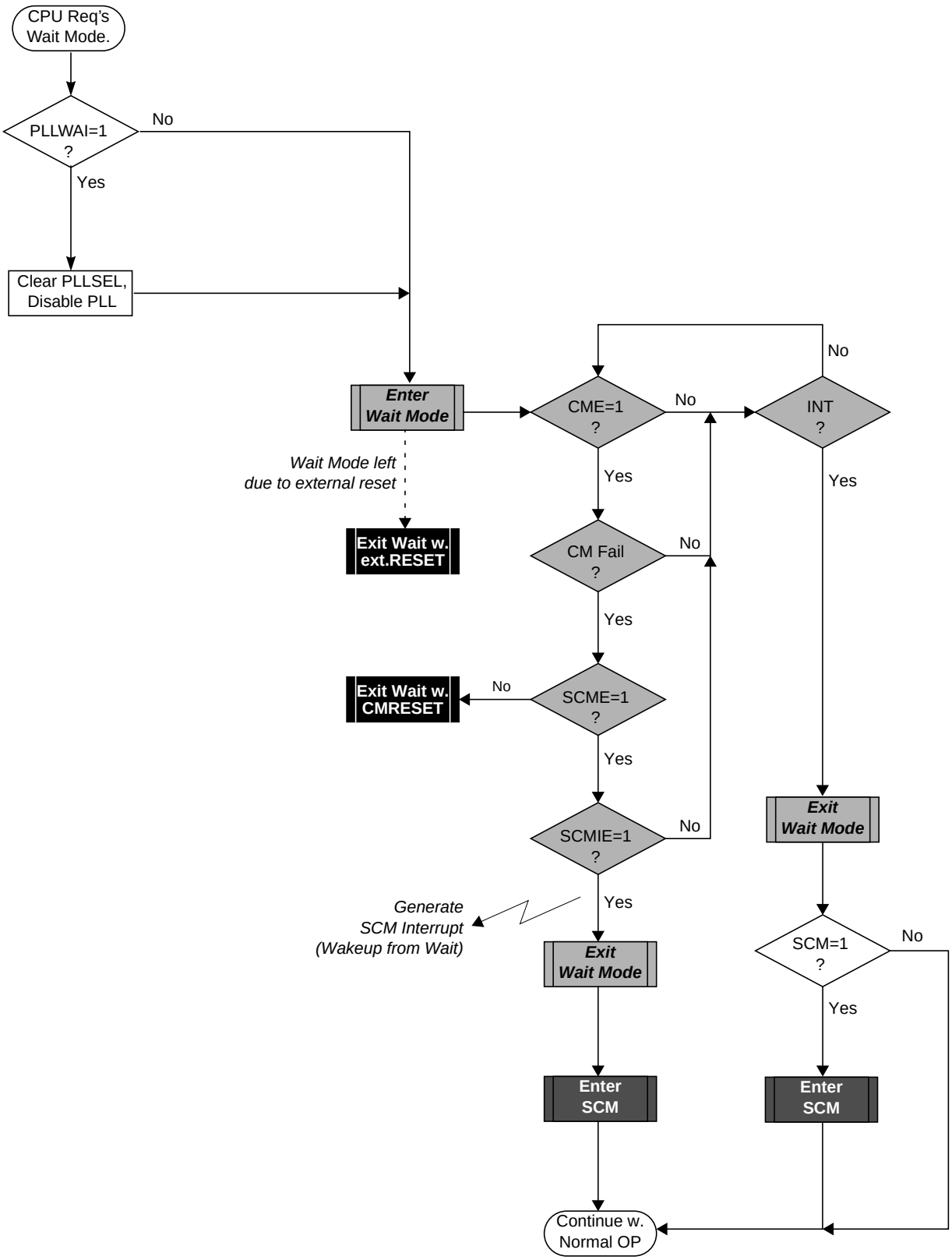


Figure 2-21. Wait Mode Entry/Exit Sequence

6.4.4 Semaphores

The XGATE module offers a set of eight hardware semaphores. These semaphores provide a mechanism to protect system resources that are shared between two concurrent threads of program execution; one thread running on the S12X_CPU and one running on the XGATE RISC core.

Each semaphore can only be in one of the three states: “Unlocked”, “Locked by S12X_CPU”, and “Locked by XGATE”. The S12X_CPU can check and change a semaphore’s state through the XGATE semaphore register (XGSEM, see Section 6.3.1.6, “XGATE Semaphore Register (XGSEM)”). The RISC core does this through its SSEM and CSEM instructions.

Figure 6-21 illustrates the valid state transitions.

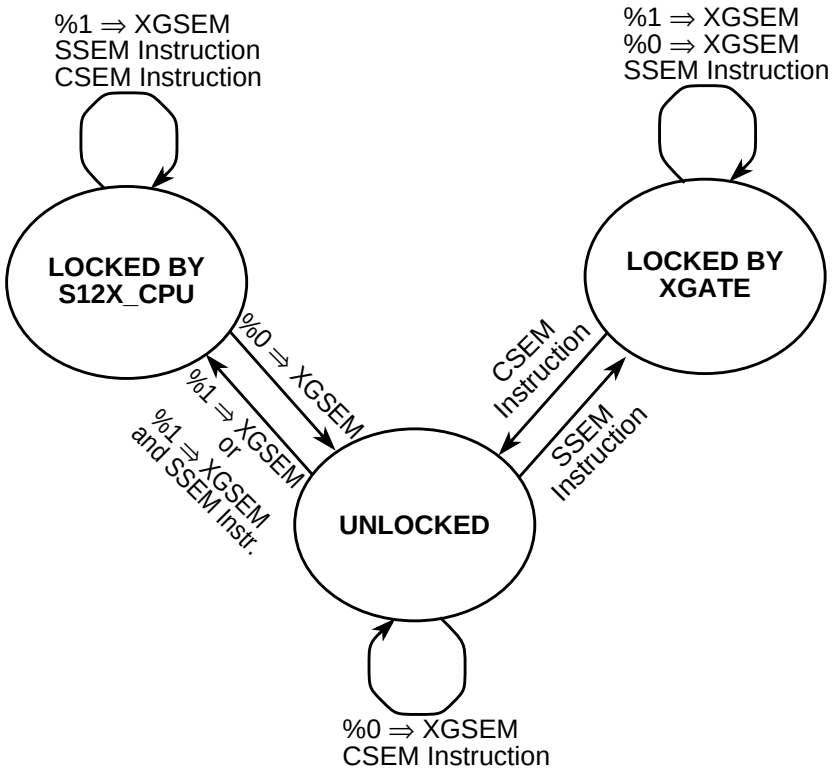


Figure 6-21. Semaphore State Transitions

NOTE

Register bits PCLK0 to PCLK7 can be written anytime. If a clock select is changed while a PWM signal is being generated, a truncated or stretched pulse can occur during the transition.

Table 8-3. PWMCLK Field Descriptions

Field	Description
7 PCLK7	**Pulse Width Channel 7 Clock Select** 0 Clock B is the clock source for PWM channel 7. 1 Clock SB is the clock source for PWM channel 7.
6 PCLK6	**Pulse Width Channel 6 Clock Select** 0 Clock B is the clock source for PWM channel 6. 1 Clock SB is the clock source for PWM channel 6.
5 PCLK5	**Pulse Width Channel 5 Clock Select** 0 Clock A is the clock source for PWM channel 5. 1 Clock SA is the clock source for PWM channel 5.
4 PCLK4	**Pulse Width Channel 4 Clock Select** 0 Clock A is the clock source for PWM channel 4. 1 Clock SA is the clock source for PWM channel 4.
3 PCLK3	**Pulse Width Channel 3 Clock Select** 0 Clock B is the clock source for PWM channel 3. 1 Clock SB is the clock source for PWM channel 3.
2 PCLK2	**Pulse Width Channel 2 Clock Select** 0 Clock B is the clock source for PWM channel 2. 1 Clock SB is the clock source for PWM channel 2.
1 PCLK1	**Pulse Width Channel 1 Clock Select** 0 Clock A is the clock source for PWM channel 1. 1 Clock SA is the clock source for PWM channel 1.
0 PCLK0	**Pulse Width Channel 0 Clock Select** 0 Clock A is the clock source for PWM channel 0. 1 Clock SA is the clock source for PWM channel 0.

8.3.2.4 PWM Prescale Clock Select Register (PWMPRCLK)

This register selects the prescale clock source for clocks A and B independently.

	7	6	5	4	3	2	1	0
R	0	PCKB2	PCKB1	PCKB0	0	PCKA2	PCKA1	PCKA0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 8-6. PWM Prescale Clock Select Register (PWMPRCLK)

Read: Anytime

Write: Anytime

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

Table 10-22. CANIDMR0–CANIDMR3 Register Field Descriptions

Field	Description
7:0 AM[7:0]	**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 is detected. The message is 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 does not affect whether or not the message is accepted. 0 Match corresponding acceptance code register and identifier bits 1 Ignore corresponding acceptance code register bit

Module Base + 0x001C (CANIDMR4)
0x001D (CANIDMR5)
0x001E (CANIDMR6)
0x001F (CANIDMR7)

	7	6	5	4	3	2	1	0
R/W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R/W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R/W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
R/W	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset	0	0	0	0	0	0	0	0

Figure 10-23. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7

Read: Anytime
Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

12.4.1 Master Mode

The SPI operates in master mode when the MSTR bit is set. Only a master SPI module can initiate transmissions. A transmission begins by writing to the master SPI data register. If the shift register is empty, the byte immediately transfers to the shift register. The byte begins shifting out on the MOSI pin under the control of the serial clock.

- Serial clock

  The SPR2, SPR1, and SPR0 baud rate selection bits, in conjunction with the SPPR2, SPPR1, and SPPR0 baud rate preselection bits in the SPI baud rate register, control the baud rate generator and determine the speed of the transmission. The SCK pin is the SPI clock output. Through the SCK pin, the baud rate generator of the master controls the shift register of the slave peripheral.
- MOSI, MISO pin

  In master mode, the function of the serial data output pin (MOSI) and the serial data input pin (MISO) is determined by the SPC0 and BIDIROE control bits.
- $\overline{SS}$ pin

  If MODFEN and SSOE are set, the $\overline{SS}$ pin is configured as slave select output. The $\overline{SS}$ output becomes low during each transmission and is high when the SPI is in idle state.

  If MODFEN is set and SSOE is cleared, the $\overline{SS}$ pin is configured as input for detecting mode fault error. If the $\overline{SS}$ input becomes low this indicates a mode fault error where another master tries to drive the MOSI and SCK lines. In this case, the SPI immediately switches to slave mode, by clearing the MSTR bit and also disables the slave output buffer MISO (or SISO in bidirectional mode). So the result is that all outputs are disabled and SCK, MOSI, and MISO are inputs. If a transmission is in progress when the mode fault occurs, the transmission is aborted and the SPI is forced into idle state.

  This mode fault error also sets the mode fault (MODF) flag in the SPI status register (SPISR). If the SPI interrupt enable bit (SPIE) is set when the MODF flag becomes set, then an SPI interrupt sequence is also requested.

  When a write to the SPI data register in the master occurs, there is a half SCK-cycle delay. After the delay, SCK is started within the master. The rest of the transfer operation differs slightly, depending on the clock format specified by the SPI clock phase bit, CPHA, in SPI control register 1 (see Section 12.4.3, “Transmission Formats”).

NOTE

A change of the bits CPOL, CPHA, SSOE, LSBFE, MODFEN, SPC0, or BIDIROE with SPC0 set, SPPR2-SPPR0 and SPR2-SPR0 in master mode will abort a transmission in progress and force the SPI into idle state. The remote slave cannot detect this, therefore the master must ensure that the remote slave is returned to idle state.

14.3.2.3 Autonomous Periodical Interrupt Control Register (VREGAPICL)

The VREGAPICL register allows the configuration of the VREG_3V3 autonomous periodical interrupt features.

	7	6	5	4	3	2	1	0
R	APICLK	0	0	0	0	APIFE	APIE	APIF
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 14-4. Autonomous Periodical Interrupt Control Register (VREGAPICL)

Table 14-4. VREGAPICL Field Descriptions

Field	Description
7 APICLK	**Autonomous Periodical Interrupt Clock Select Bit** — Selects the clock source for the API. Writable only if APIFE = 0; APICLK cannot be changed if APIFE is set by the same write operation. 0 Autonomous periodical interrupt clock used as source. 1 Bus clock used as source.
2 APIFE	**Autonomous Periodical Interrupt Feature Enable Bit** — Enables the API feature and starts the API timer when set. 0 Autonomous periodical interrupt is disabled. 1 Autonomous periodical interrupt is enabled and timer starts running.
1 APIE	**Autonomous Periodical Interrupt Enable Bit** 0 API interrupt request is disabled. 1 API interrupt will be requested whenever APIF is set.
0 APIF	**Autonomous Periodical Interrupt Flag** — APIF is set to 1 when the in the API configured time has elapsed. This flag can only be cleared by writing a 1 to it. Clearing of the flag has precedence over setting. Writing a 0 has no effect. If enabled (APIE = 1), APIF causes an interrupt request. 0 API timeout has not yet occurred. 1 API timeout has occurred.

Table 15-5. Hardware Commands

Command	Opcode (hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware is enabled. If enabled, an ACK will be issued when the part enters active background mode.
ACK_ENABLE	D5	None	Enable Handshake. Issues an ACK pulse after the command is executed.
ACK_DISABLE	D6	None	Disable Handshake. This command does not issue an ACK pulse.
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Must be aligned access.

NOTE:
If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

15.4.4 Standard BDM Firmware Commands

Firmware commands are used to access and manipulate CPU resources. The system must be in active BDM to execute standard BDM firmware commands, see Section 15.4.2, “Enabling and Activating BDM”. Normal instruction execution is suspended while the CPU executes the firmware located in the standard BDM firmware lookup table. The hardware command BACKGROUND is the usual way to activate BDM.

As the system enters active BDM, the standard BDM firmware lookup table and BDM registers become visible in the on-chip memory map at 0x7FFF00–0x7FFFFF, and the CPU begins executing the standard BDM firmware. The standard BDM firmware watches for serial commands and executes them as they are received.

The firmware commands are shown in Table 15-6.

Differently from the normal bit transfer (where the host initiates the transmission), the serial interface ACK handshake pulse is initiated by the target MCU by issuing a negative edge in the BKGD pin. The hardware handshake protocol in Figure 15-11 specifies the timing when the BKGD pin is being driven, so the host should follow this timing constraint in order to avoid the risk of an electrical conflict in the BKGD pin.

NOTE

The only place the BKGD pin can have an electrical conflict is when one side is driving low and the other side is issuing a speedup pulse (high). Other "highs" are pulled rather than driven. However, at low rates the time of the speedup pulse can become lengthy and so the potential conflict time becomes longer as well.

The ACK handshake protocol does not support nested ACK pulses. If a BDM command is not acknowledge by an ACK pulse, the host needs to abort the pending command first in order to be able to issue a new BDM command. When the CPU enters wait or stop while the host issues a hardware command (e.g., WRITE_BYTE), the target discards the incoming command due to the wait or stop being detected. Therefore, the command is not acknowledged by the target, which means that the ACK pulse will not be issued in this case. After a certain time the host (not aware of stop or wait) should decide to abort any possible pending ACK pulse in order to be sure a new command can be issued. Therefore, the protocol provides a mechanism in which a command, and its corresponding ACK, can be aborted.

NOTE

The ACK pulse does not provide a time out. This means for the GO_UNTIL command that it can not be distinguished if a stop or wait has been executed (command discarded and ACK not issued) or if the "UNTIL" condition (BDM active) is just not reached yet. Hence in any case where the ACK pulse of a command is not issued the possible pending command should be aborted before issuing a new command. See the handshake abort procedure described in Section 15.4.8, "Hardware Handshake Abort Procedure".

15.4.8 Hardware Handshake Abort Procedure

The abort procedure is based on the SYNC command. In order to abort a command, which had not issued the corresponding ACK pulse, the host controller should generate a low pulse in the BKGD pin by driving it low for at least 128 serial clock cycles and then driving it high for one serial clock cycle, providing a speedup pulse. By detecting this long low pulse in the BKGD pin, the target executes the SYNC protocol, see Section 15.4.9, "SYNC — Request Timed Reference Pulse", and assumes that the pending command and therefore the related ACK pulse, are being aborted. Therefore, after the SYNC protocol has been completed the host is free to issue new BDM commands. For Firmware READ or WRITE commands it can not be guaranteed that the pending command is aborted when issuing a SYNC before the corresponding ACK pulse. There is a short latency time from the time the READ or WRITE access begins until it is finished and the corresponding ACK pulse is issued. The latency time depends on the firmware READ or WRITE command that is issued and if the serial interface is running on a different clock rate than the bus. When the SYNC command starts during this latency time the READ or WRITE command will not be aborted, but the corresponding ACK pulse will be aborted. A pending GO, TRACE1 or

18.1.1 Terminology

Table 18-1. Acronyms and Abbreviations

Logic level “1”	Voltage that corresponds to Boolean true state
Logic level “0”	Voltage that corresponds to Boolean false state
0x	Represents hexadecimal number
x	Represents logic level ’don’t care’
byte	8-bit data
word	16-bit data
local address	based on the 64 KBytes Memory Space (16-bit address)
global address	based on the 8 MBytes Memory Space (23-bit address)
Aligned address	Address on even boundary
Mis-aligned address	Address on odd boundary
Bus Clock	System Clock. Refer to CRG Block Guide.
expanded modes	Normal Expanded Mode Emulation Single-Chip Mode Emulation Expanded Mode Special Test Mode
single-chip modes	Normal Single-Chip Mode Special Single-Chip Mode
emulation modes	Emulation Single-Chip Mode Emulation Expanded Mode
normal modes	Normal Single-Chip Mode Normal Expanded Mode
special modes	Special Single-Chip Mode Special Test Mode
NS	Normal Single-Chip Mode
SS	Special Single-Chip Mode
NX	Normal Expanded Mode
ES	Emulation Single-Chip Mode
EX	Emulation Expanded Mode
ST	Special Test Mode
Unimplemented areas	Areas which are accessible by the pages (RPAGE,PPAGE,EPAGE) and not implemented
External Space	Area which is accessible in the global address range 14_0000 to 3F_FFFF
external resource	Resources (Emulator, Application) connected to the MCU via the external bus on expanded modes (Unimplemented areas and External Space)
PRR	Port Replacement Registers
PRU	Port Replacement Unit located on the emulator side
MCU	MicroController Unit
NVM	Non-volatile Memory; Flash EEPROM or ROM

18.1.2 Features

The main features of this block are:

- Paging capability to support a global 8 Mbytes memory address space
- Bus arbitration between the masters CPU, BDM and XGATE

19.4.6.1 External Tagging using $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$

External tagging using the external $\overline{\text{TAGHI}}$ and $\overline{\text{TAGLO}}$ pins can only be used to tag CPU opcodes; tagging of XGATE code using these pins is not possible. An external tag triggers the state sequencer into State0 when the tagged opcode reaches the execution stage of the instruction queue.

The pins operate independently, thus the state of one pin does not affect the function of the other. External tagging is possible in emulation modes only. The presence of logic level 0 on either pin at the rising edge of the external clock (ECLK) performs the function indicated in the Table 19-43. It is possible to tag both bytes of an instruction word. If a taghit comes from the low or high byte, a breakpoint generated according to the DBGBRK and BDM bits in DBGC1. Each time $\overline{\text{TAGHI}}$ or $\overline{\text{TAGLO}}$ are low on the rising edge of ECLK, the old tag is replaced by a new one

Table 19-43. Tag Pin Function

$\overline{\text{TAGHI}}$	$\overline{\text{TAGLO}}$	Tag
1	1	No tag
1	0	Low byte
0	1	High byte
0	0	Both bytes

19.4.7 Breakpoints

It is possible to select breakpoints to the XGATE and let the CPU continue operation, setting DBGBRK[0], or breakpoints to the CPU and let the XGATE continue operation setting, DBGBRK[1], or a breakpoint to both CPU and XGATE, setting both bits DBGBRK[1:0].

There are several ways to generate breakpoints to the XGATE and CPU modules.

- Through XGATE software breakpoint requests.
- From comparator channel triggers to final state.
- Using software to write to the TRIG bit in the DBGC1 register.
- From taghits generated using the external TAGHI and TAGLO pins.

19.4.7.1 XGATE Software Breakpoints

The XGATE software breakpoint instruction BRK can request an CPU breakpoint, via the DBG module. In this case, if the XGSBPE bit is set, the DBG module immediately generates a forced breakpoint request to the CPU, the state sequencer is returned to state0 and tracing, if active, is terminated. If configured for begin-trigger and tracing has not yet been triggered from another source, the trace buffer contains no new information. Breakpoint requests from the XGATE module do not depend upon the state of the DBGBRK or ARM bits in DBGC1. They depend solely on the state of the XGSBPE and BDM bits. Thus it is not necessary to ARM the DBG module to use XGATE software breakpoints to generate breakpoints in the CPU program flow, but it is necessary to set XGSBPE. Furthermore if a breakpoint to BDM is required, the BDM bit must also be set. When the XGATE requests an CPU breakpoint, the XGATE program flow stops by default, independent of the DBG module. The user can thus determine if an XGATE breakpoint has occurred by reading out the XGATE program counter over the BDM interface.

20.3.2.7.1 Debug State Control Register 1 (DBGSCR1)

Address: 0x0027

	7	6	5	4	3	2	1	0
R	0	0	0	0	SC3	SC2	SC1	SC0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 20-9. Debug State Control Register 1 (DBGSCR1)

Read: Anytime

Write: Anytime when S12XDBG not armed.

This register is visible at 0x0027 only with COMRV[1:0] = 00. The state control register 1 selects the targeted next state whilst in State1. The matches refer to the match channels of the comparator match control logic as depicted in Figure 20-1 and described in Section 20.3.2.8.1". Comparators must be enabled by setting the comparator enable bit in the associated DBGXCTL control register.

Table 20-20. DBGSCR1 Field Descriptions

Field	Description
3–0 SC[3:0]	These bits select the targeted next state whilst in State1, based upon the match event.

Table 20-21. State1 Sequencer Next State Selection

SC[3:0]	Description
0000	Any match triggers to state2
0001	Any match triggers to state3
0010	Any match triggers to Final State
0011	Match2 triggers to State2..... Other matches have no effect
0100	Match2 triggers to State3..... Other matches have no effect
0101	Match2 triggers to Final State..... Other matches have no effect
0110	Match0 triggers to State2..... Match1 triggers to State3..... Other matches have no effect
0111	Match1 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1000	Match0 triggers to State2..... Match2 triggers to State3..... Other matches have no effect
1001	Match2 triggers to State3..... Match0 triggers Final State..... Other matches have no effect
1010	Match1 triggers to State2..... Match3 triggers to State3..... Other matches have no effect
1011	Match3 triggers to State3..... Match1 triggers to Final State..... Other matches have no effect
1100	Match3 has no effect..... All other matches (M0,M1,M2) trigger to State2
1101	Reserved
1110	Reserved
1111	Reserved

The trigger priorities described in Table 20-38 dictate that in the case of simultaneous matches, the match on the lower channel number (0,1,2,3) has priority. The SC[3:0] encoding ensures that a match leading to final state has priority over all other matches.

21.4.2 Internal Visibility

Internal visibility allows the observation of the internal MCU address and data bus as well as the determination of the access source and the CPU pipe (queue) status through the external bus interface.

Internal visibility is always enabled in emulation single chip mode and emulation expanded mode. Internal CPU and BDM accesses are made visible on the external bus interface, except those to BDM firmware and BDM registers.

Internal reads are made visible on ADDRx/IVDx (address and read data multiplexed, see Table 21-9 to Table 21-11), internal writes on ADDRx and DATAx (see Table 21-12 to Table 21-14). R/$\overline{W}$ and $\overline{LSTRB}$ show the type of access. External read data are also visible on IVDx.

21.4.2.1 Access Source and Instruction Queue Status Signals

The access source (bus master) can be determined from the external bus control signals ACC[2:0] as shown in Table 21-8.

Table 21-8. Determining Access Source from Control Signals

ACC[2:0]	Access Description
000	Repetition of previous access cycle
001	CPU access
010	BDM access
011	XGATE PRR access[1]
100	No access[2]
101, 110, 111	Reserved

1 Invalid IVD brought out in read cycles

2 Denotes also accesses to BDM firmware and BDM registers (IQSTATx are 'XXXX' and R/$\overline{W}$ = 1 in these cases)

The CPU instruction queue status (execution-start and data-movement information) is brought out as IQSTAT[3:0] signals. For decoding of the IQSTAT values, refer to the S12X_CPU section.

21.4.2.2 Emulation Modes Timing

A bus access lasts 1 ECLK cycle. In case of a stretched external access (emulation expanded mode), up to an infinite amount of ECLK cycles may be added. ADDRx values will only be shown in ECLK high phases, while ACCx, IQSTATx, and IVDx values will only be presented in ECLK low phases.

Based on this multiplex timing, ACCx are only shown in the current (first) access cycle. IQSTATx and (for read accesses) IVDx follow in the next cycle. If the access takes more than one bus cycle, ACCx display NULL (0x000) in the second and all following cycles of the access. IQSTATx display NULL (0x0000) from the third until one cycle after the access to indicate continuation.

The resulting timing pattern of the external bus signals is outlined in the following tables for read, write and interleaved read/write accesses. Three examples represent different access lengths of 1, 2, and n–1 bus cycles. Non-shaded bold entries denote all values related to Access #0.

21.5.2.2 Example 2b: Emulation Expanded Mode

This mode is used for emulation systems in which the target application is operating in normal expanded mode.

If the external bus is used with a PRU, the external device rebuilds the data select and data direction signals $\overline{UDS}$, $\overline{LDS}$, $\overline{RE}$, and $\overline{WE}$ from the ADDR0, $\overline{LSTRB}$, and R/$\overline{W}$ signals.

Figure 21-6 shows the PRU connection with the available external bus signals in an emulator application.

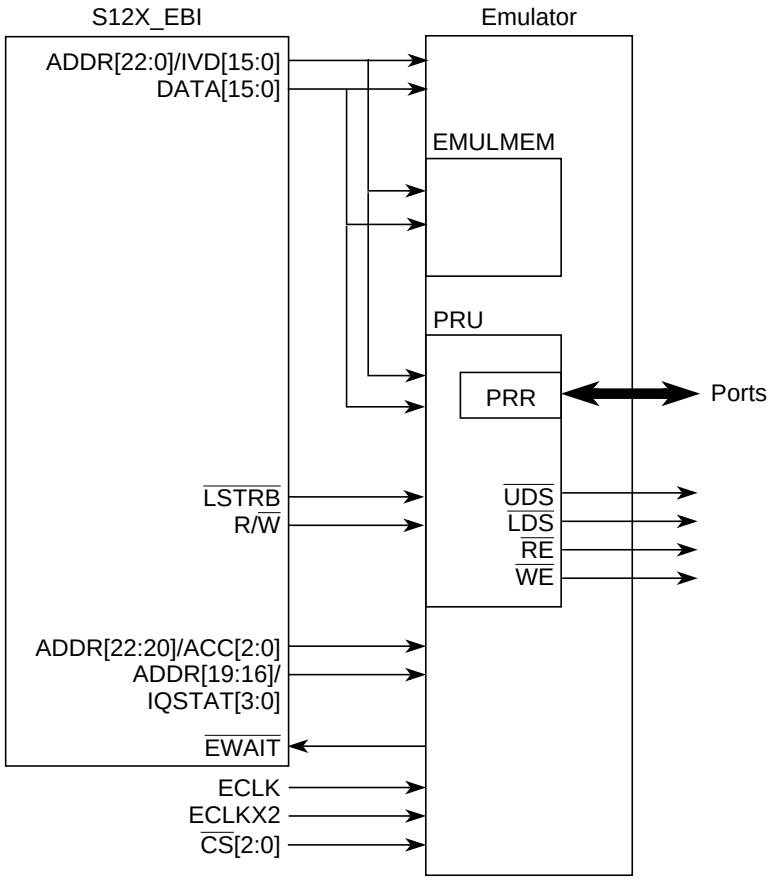


Figure 21-6. Application in Emulation Expanded Mode

The timings of accesses with 1 stretch cycle are shown in

- Figure ‘Example 2b: Emulation Expanded Mode — Read with 1 Stretch Cycle’
- Figure ‘Example 2b: Emulation Expanded Mode — Write with 1 Stretch Cycle’

The associated timing numbers are given in

- Table ‘Example 2b: Emulation Expanded Mode Timing V_{DD5} = 5.0 V (EWAITE = 0)’ (this also includes examples for alternative settings of 2 and 3 additional stretch cycles)

Timing considerations:

- If no stretch cycle is added, the timing is the same as in Emulation Single-Chip Mode.

Table 24-52. PIEJ Field Descriptions

Field	Description
7–0 PIEJ[7:6] PIEJ[1:0]	**Interrupt Enable Port J** 0 Interrupt is disabled (interrupt flag masked). 1 Interrupt is enabled.

24.0.5.57 Port J Interrupt Flag Register (PIFJ)

	7	6	5	4	3	2	1	0
R	PIFJ7	PIFJ6	0	0	0	0	PIFJ1	PIFJ0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 24-59. Port J Interrupt Flag Register (PIFJ)

Read: Anytime.

Write: Anytime.

Each flag is set by an active edge on the associated input pin. This could be a rising or a falling edge based on the state of the PPSJ register. To clear this flag, write logic level “1” to the corresponding bit in the PIFJ register. Writing a “0” has no effect.

Table 24-53. PIEJ Field Descriptions

Field	Description
7–0 PIFJ[7:6] PIFJ[1:0]	**Interrupt Flags Port J** 0 No active edge pending. Writing a “0” has no effect. 1 Active edge on the associated bit has occurred (an interrupt will occur if the associated enable bit is set). Writing a logic level “1” clears the associated flag.

24.0.5.58 Port AD1 Data Register 0 (PT0AD1)

	7	6	5	4	3	2	1	0
R	PT0AD115	PT0AD114	PT0AD113	PT0AD112	PT0AD111	PT0AD110	PT0AD19	PT0AD18
W								
Reset	0	0	0	0	0	0	0	0

Figure 24-60. Port AD1 Data Register 0 (PT0AD1)

Read: Anytime.

Write: Anytime.

This register is associated with AD1 pins PAD[15:8]. These pins can also be used as general purpose I/O.

27.4.2.2 Data Compress Command

The data compress operation will check Flash code integrity by compressing data from a selected portion of the Flash memory into a signature analyzer.

An example flow to execute the data compress operation is shown in Figure 27-26. The data compress command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the data compress command. The address written determines the starting address for the data compress operation and the data written determines the number of consecutive words to compress. If the data value written is 0x0000, 64K addresses or 128 Kbytes will be compressed. Multiple Flash blocks can be simultaneously compressed by writing to the same relative address in each Flash block. If more than one Flash block is written to in this step, the first data written will determine the number of consecutive words to compress in each selected Flash block.
2. Write the data compress command, 0x06, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the data compress command.

After launching the data compress command, the CCIF flag in the FSTAT register will set after the data compress operation has completed. The number of bus cycles required to execute the data compress operation is equal to two times the number of consecutive words to compress plus the number of Flash blocks simultaneously compressed plus 18 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. Once the CCIF flag is set, the signature generated by the data compress operation is available in the FDATA registers. The signature in the FDATA registers can be compared to the expected signature to determine the integrity of the selected data stored in the selected Flash memory. If the last address of a Flash block is reached during the data compress operation, data compression will continue with the starting address of the same Flash block. The MRDS bits in the FTSTMOD register will determine the sense-amp margin setting during the data compress operation.

NOTE

Since the FDATA registers (or data buffer) are written to as part of the data compress operation, a command write sequence is not allowed to be buffered behind a data compress command write sequence. The CBEIF flag will not set after launching the data compress command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a data compress operation active, the ACCERR flag in the FSTAT register will be set. A new command write sequence should only be started after reading the signature stored in the FDATA registers.

In order to take corrective action, it is recommended that the data compress command be executed on a Flash sector or subset of a Flash sector. If the data compress operation on a Flash sector returns an invalid signature, the Flash sector should be erased using the sector erase command and then reprogrammed using the program command.

The data compress command can be used to verify that a sector or sequential set of sectors are erased.

28.4.2.5 Mass Erase Command

The mass erase operation will erase all addresses in a Flash block using an embedded algorithm.

An example flow to execute the mass erase operation is shown in Figure 28-30. The mass erase command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the mass erase command. The address and data written will be ignored. Multiple Flash blocks can be simultaneously mass erased by writing to the same relative address in each Flash block.
2. Write the mass erase command, 0x41, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the mass erase command.

If a Flash block to be erased contains any protected area, the PVIOL flag in the FSTAT register will set and the mass erase command will not launch. Once the mass erase command has successfully launched, the CCIF flag in the FSTAT register will set after the mass erase operation has completed unless a new command write sequence has been buffered.

Table A-5. Thermal Package Characteristics[1]

Num	C	Rating	Symbol	Min	Typ	Max	Unit
LQFP144							
1	T	Thermal resistance LQFP144, single sided PCB[2]	θ_{JA}	—	—	41	°C/W
2	T	Thermal resistance LQFP144, double sided PCB with 2 internal planes[3]	θ_{JA}	—	—	32	°C/W
3		Junction to Board LQFP 144	θ_{JB}	—	—	22	°C/W
4		Junction to Case LQFP 144[4]	θ_{JC}	—	—	7/4	°C/W
5		Junction to Package Top LQFP144[5]	Ψ_{JT}	—	—	3	°C/W
LQFP112							
6	T	Thermal resistance LQFP112, single sided PCB[2]	θ_{JA}	—	—	43[3]/49[4]	°C/W
7	T	Thermal resistance LQFP112, double sided PCB with 2 internal planes[5]	θ_{JA}	—	—	32[3]/39[4]	°C/W
8		Junction to Board LQFP112	θ_{JB}	—	—	22[3]/27[4]	°C/W
9		Junction to Case LQFP112[4]	θ_{JC}	—	—	7[3]/11[4]	°C/W
10		Junction to Package Top LQFP112[5]	Ψ_{JT}	—	—	3/2	°C/W
QFP80							
11	T	Thermal resistance QFP 80, single sided PCB[2]	θ_{JA}	—	—	45[3]/49[4]	°C/W
12	T	Thermal resistance QFP 80, double sided PCB with 2 internal planes[3]	θ_{JA}	—	—	33[3]/36[4]	°C/W
13	T	Junction to Board QFP 80	θ_{JB}	—	—	19[3]/20[4]	°C/W
14	T	Junction to Case QFP 80[6]	θ_{JC}	—	—	11[3]/14[4]	°C/W
15	T	Junction to Package Top QFP 80[7]	Ψ_{JT}	—	—	3	°C/W

1 The values for thermal resistance are achieved by package simulations

2 Junction to ambient thermal resistance, θ_{JA} was simulated to be equivalent to the JEDEC specification JESD51-2 in a horizontal configuration in natural convection.

3 Maskset L15Y / M84E in LQFP112 or QFP80

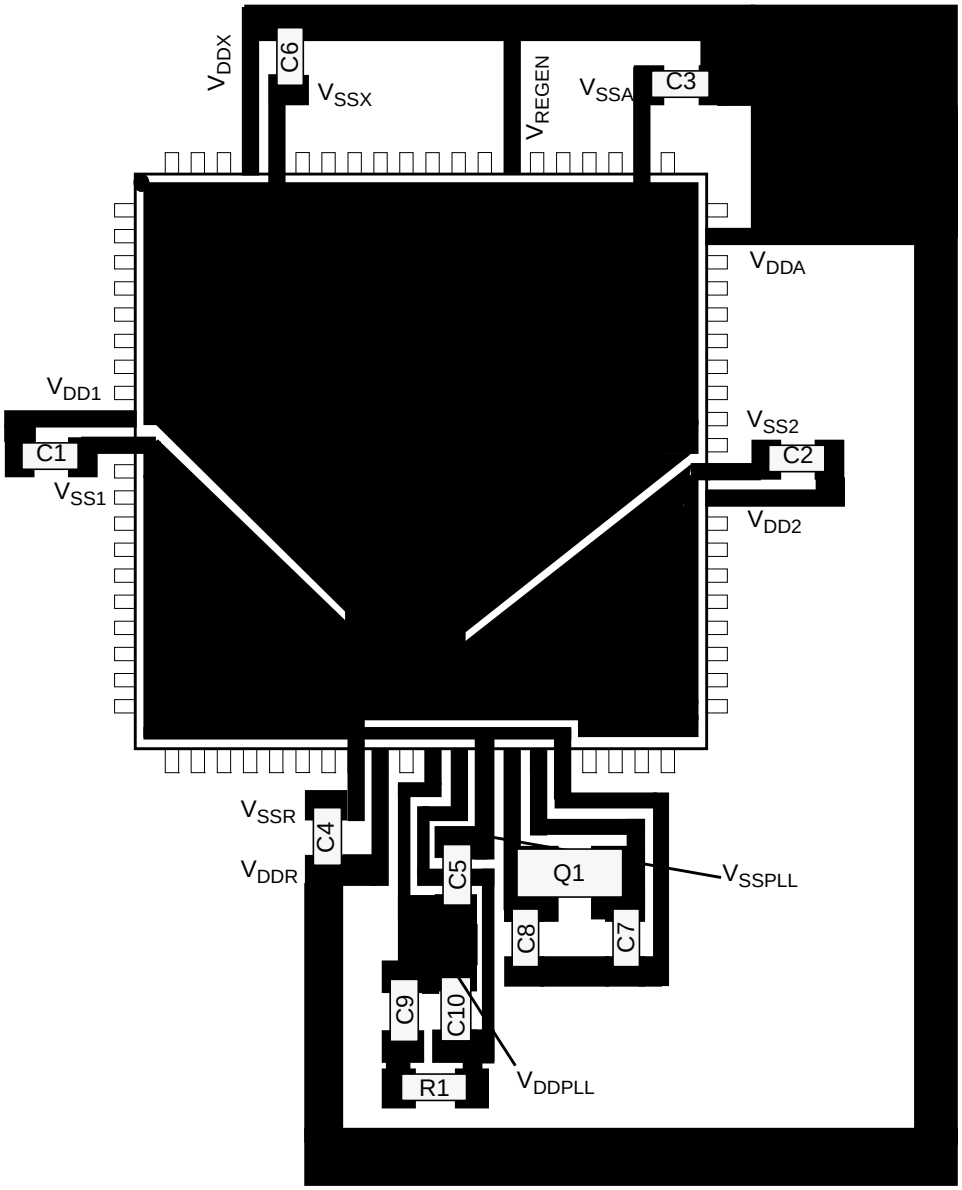
4 Maskset M42E in LQFP112 or QFP80

5 Junction to ambient thermal resistance, θ_{JA} was simulated to be equivalent to the JEDEC specification JESD51-7 in a horizontal configuration in natural convection.

6 Junction to case thermal resistance was simulated to be equivalent to the measured values using the cold plate technique with the cold plate temperature used as the “case” temperature. This basic cold plate measurement technique is described by MIL-STD 883D, Method 1012.1. This is the correct thermal metric to use to calculate thermal performance when the package is being used with a heat sink.

7 Thermal characterization parameter Ψ_{JT} is the “resistance” from junction to reference point thermocouple on top center of the case as defined in JESD51-2. Ψ_{JT} is a useful value to use to estimate junction temperature in a steady state customer enviroment.

Figure C-3. 80-Pin QFP Recommended PCB Layout



E.4 MC9S12XD/A/B -Family SRAM & EEPROM Configuration

Figure E-2. Available RAM Pages on S12XD-Family[1]

<table>
<tr><th>RAM Page
RP[7:0]</th><th>DP512
A512</th><th>DT512
DT384</th><th>DQ256
A256</th><th>DG128
A128</th><th>D128</th><th>D64</th></tr>
<tr><td>0xF6</td><td rowspan="2"></td><td rowspan="5"></td><td rowspan="8"></td><td rowspan="10"></td><td rowspan="9"></td><td rowspan="8"></td></tr>
<tr><td>0xF7</td></tr>
<tr><td>0xF8</td><td rowspan="8">32K Byte</td></tr>
<tr><td>0xF9</td></tr>
<tr><td>0xFA</td></tr>
<tr><td>0xFB</td><td rowspan="6">20K Byte</td></tr>
<tr><td>0xFC</td><td rowspan="5">16K Byte</td></tr>
<tr><td>0xFD</td></tr>
<tr><td>0xFE</td><td rowspan="2">12K Byte</td><td rowspan="2">8K Byte</td></tr>
<tr><td>0xFF</td><td>4K Byte</td></tr>
</table>

[1] On 9S12XD256 14K byte RAM available pages FF,FE,FD and upper half of page FC
On 9S12XB256 10K byte RAM available pages FF, FE upper half of page FD
On 9S12XB218 6K byte RAM available pages FF and upper half of page FE

Table E-4. Available EEPROM Pages on MC9S12XD-Family

<table>
<tr><th>EEPROM
Page
EP[7:0]</th><th>DP512
DT512
DT384
DQ256
A256
A512</th><th>DG128
D128
A128
B256</th><th>D64
B128</th></tr>
<tr><td>0xFA</td><td rowspan="2"></td><td rowspan="4"></td><td rowspan="5"></td></tr>
<tr><td>0xFB</td></tr>
<tr><td>0xFC</td><td rowspan="4">4K Byte</td></tr>
<tr><td>0xFD</td></tr>
<tr><td>0xFE</td><td rowspan="2">2K Byte</td></tr>
<tr><td>0xFF</td><td>1K Byte</td></tr>
</table>