LGT8FX8P Series – Programming Manual 1.0.4J
Errata Page 10
Jake 2019 Oct 9 11:30am
Discrepancy My edited text ATMEL text Original+Google Translate Question or note Answer/Final Version

---

1.) Six of the 32 extra working registers can be used to form a combination of three 16-bit registers. <d1>These 16-bit registers can be used to access external memory and FLASH programs by indirectly addressing the address pointer. <q1>The LGT8XM supports single-cycle 16-bit arithmetic, greatly improving the efficiency of indirect addressing. The three special 16-bit registers in the LGT8XM core are named X,Y, Z registers, which are described in more detail later.

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing – enabling efficient address calculations. One of the these address pointers can also be used as an address pointer for look up tables in Flash program memory. These added function registers are the 16-bit X-, Y-, and Z-register, described later in this section.

32 个通过工作寄存器中的 6 个用于两两结合构成三个 16 位寄存器,可用于间接寻址地址指针,用于访问外部存储空间以及 FLASH 程序空间。LGT8XM 支持单周期的 16 位算术运算,极大的提高了间接寻址的效率。LGT8XM 内核中这三个特殊的 16 位寄存器被命名为 X,Y,Z 寄存器,将在后面详细介绍。
32 of the 32 working registers are used to form a combination of three 16-bit registers that can be used to indirectly address the address pointer for accessing external memory and FLASH program space. The LGT8XM supports single-cycle 16-bit arithmetic operations, greatly improving the efficiency of indirect addressing. These three special 16-bit registers in the LGT8XM core are named X, Y, and Z registers, which are described in detail later.

<d1>Can, “all three” 16b registers, or (like the ATMEL DS) can only the one access FLASH?
<q1>Is the 16 bit single cycle ALU really a unique feature?

2.) The ALU supports arithmetic logic operations between registers and between constants and registers. The operation of a single register can also be performed in the ALU. After the ALU operation is completed, the effect of the operation on the <q1>state of the kernal is updated to the status register (SREG).

The ALU supports arithmetic and logic operations between registers or between a constant and a register. Single register operations can also be executed in the ALU. After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

ALU 支持寄存器之间以及常数与寄存器之间的算术逻辑运算,单个寄存器的运算也可以在 ALU 中执行。ALU 运算完成后,运算结果对内核状态的影响更新到状态寄存器中(SREG)。
The ALU supports arithmetic logic operations between registers and between constants and registers. The operation of a single register can also be performed in the ALU. After the ALU operation is completed, the effect of the operation result on the kernel state is updated to the status register (SREG).
<d1> Is this just semantics?
<q1> Do these references to the “Kernal” have significance or are they just translation anomalies.

<a>Properly, those should be singular, since it is a two-operand machine and any instruction that phrase describes will specify one of each. "Kernel" is an approximation of "core".

2.) The ALU supports arithmetic logic operations between registers and between a constant and a register. The operation of a single register can also be performed in the ALU. After the ALU operation is completed, the effect of the operation on the state of the kernal is updated to the status register (SREG).

3.) Program flow is provided by conditional and unconditional jump and call instructions, able to directly address the whole address space. Most LGT8XM instructions are 16 bits. Each program address space corresponds to a 16-bit or 32-bit LGT8XM instruction.

Program flow is provided by conditional and unconditional jump and call instructions, able to

directly address the whole address space. Most AVR instructions have a single 16-bit word format. Every program memory address contains a 16- or 32-bit instruction.

程序流程控制通过条件和无条件跳转/调用实现,可以寻址到所以的程序区域。大部分 LGT8XM 指令为 16 位。每个程序地址空间对应一个 16 位或者 32 位的 LGT8XM 指令。
Program flow control can be addressed to the program area by conditional and unconditional jump/call implementations. Most LGT8XM instructions are 16 bits. Each program address space corresponds to a 16-bit or 32-bit LGT8XM instruction.

NOTE: The following is from the ATMEL DS but is omitted from LGT as is the SPM instruction.
Program Flash memory space is divided in two sections, the Boot Program section and the Application Program section. Both sections have dedicated Lock bits for write and read/write protection. The SPM instruction that writes into the Application Flash memory section must reside in the Boot Program section.

4.) After the kernel responds to an interrupt or a subroutine call, the return address Program Counter (PC) is stored on the stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. <d1>All applications that support interrupt or subroutine calls must first initialize the Stack Pointer Register (SP), which can be accessed through the IO space. Data SRAM can be accessed in five different addressing modes.

The internal memory of the LGT8XM is linearly mapped to a uniform address space. Please refer to the the storage memory chapter for further details.

During interrupts and subroutine calls, the return address Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the Reset routine (before subroutines or interrupts are executed). The Stack Pointer (SP) is read/write accessible in the I/O space. The data SRAM can easily be accessed through the five different addressing modes supported in the AVR architecture.

The memory spaces in the AVR architecture are all linear and regular memory maps.

内核响应中断或子程序调用后,返回地址(PC)被存储在堆栈中。堆栈被分配在系统的一般数据 SRAM 中,因此堆栈的大小仅受限于系统中 SRAM 的大小和用法。所有的支持中断或子程序调用的应用,必须首先初始化堆栈指针寄存器(SP),SP 可以通过 IO 空间访问。数据 SRAM 可以通过 5 种不同的寻址模式访问。LGT8XM 的内部存储空间都被线性的映射到一个统一的地址空间。具体请参考存储章节的介绍。
After the kernel responds to an interrupt or a subroutine call, the return address (PC) is stored on the stack. The stack is allocated in the system's general data SRAM, so the size of the stack is limited only by the size and usage of the SRAM in the system. All applications that support interrupt or subroutine calls must first initialize the Stack Pointer Register (SP), which can be accessed through the IO space. Data SRAM can be accessed in five different addressing modes. The internal memory of the LGT8XM is linearly mapped to a uniform address space. Please refer to the introduction of the storage chapter for details.

<d1> Is there a difference between just “calling the stack pointer and calling the stack pointer “in the Reset routine?”

<a>"First" is correct for all intents and purposes. "in the Reset routine" is also correct but possibly overspecified. As long as it's set before you use it, the core doesn't care.
4.) After the kernel responds to an interrupt or a subroutine call, the return address Program Counter (PC) is stored on the stack. The Stack is effectively allocated in the general data SRAM, and consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM. <d1>All applications that support interrupt or subroutine calls must first initialize the Stack Pointer Register (SP), which can be accessed through the IO space. Data SRAM can be accessed in five different addressing modes.

The internal memory of the LGT8XM is linearly mapped to a uniform address space. Please refer to the the storage memory chapter for further details.

5.) The LGT8XM includes a flexible interrupt controller. It can be controlled by a global interrupt enable bit in the Status Register. Every interrupt has a separate interrupt vector. The priority of the interrupt has a corresponding relationship with the interrupt vector address. The smaller the interrupt address, the higher the priority of the interrupt.

What is I/O space

where is the Status Register

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All interrupts have a separate Interrupt Vector in the Interrupt Vector table. The interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

LGT8XM 内核包含了一个灵活的中断控制器,中断功能可以通过状态寄存器中的一个全局中断使能位控制。所有的中断都有一个独立的中断向量。中断的优先级与中断向量地址有对应关系,中断地址越小,中断的优先级就越高。

The LGT8XM core includes a flexible interrupt controller that can be controlled by a global interrupt enable bit in the status register. All interrupts have a separate interrupt vector. The priority of the interrupt has a corresponding relationship with the interrupt vector address. The smaller the interrupt address, the higher the priority of the interrupt.

<d1> Is there a significant difference between specifying that an interrupt controller is “in the I/O space and not?

<a>The I/O space can be accessed by IN and OUT instructions that are shorter and faster than most other addressing forms. While it would be incredibly odd, the LGT's interrupt controller may have its control registers accessible only in the memory space. Confirm with the memory map.

6.) The I/O space contains 64 register spaces that can be directly addressed by IN/OUT instructions. These registers are used for kernel control as well as control functions for status registers, SPI and other I/O peripherals. <d1>This space can be accessed directly or by their address mapped to the data memory space (0x20 - 0x5F). In addition, the LGT8FX8P also includes extended I/O space, which is mapped to data memory space 0x60 — OxFF, which can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

The I/O memory space contains 64 addresses for CPU peripheral functions as Control Registers, SPI, and other I/O functions. The I/O Memory can be accessed directly, or as the Data Space locations following those of the Register File, 0x20 - 0x5F. In addition, the ATmega48P/88P/168P/328P has Extended I/O space from 0x60 - 0xFF in SRAM where only the ST/STS/STD and LD/LDS/LDD instructions can be used.

I/O 空间包含了 64 个可以通过 IN/OUT 指令直接寻址的寄存器空间。这些寄存器现实对内核控制以及状态寄存器,SPI 以及其他 I/O 外设的控制功能。这部分空间可以通过 IN/OUT 指令直接访问,也可以通过他们映射到数据存储器空间的地址访问(0x20 – 0x5F)。另外,LGT8FX8P 也包含扩展的 I/O 空间,他们被映射到数据存储空间 0x60 – 0xFF,这里只能使用 ST/STS/STD 以及 LD/LDS/LDD 指令访问。

The I/O space contains 64 register spaces that can be directly addressed by IN/OUT instructions. These registers are realistic for kernel control as well as control functions for status registers, SPI and other I/O peripherals. This space can be accessed directly by the IN/OUT instruction or by their address mapped to the data memory space (0x20 – 0x5F). In addition, the LGT8FX8P also contains extended I/O space, which is mapped to data memory space 0x60 – 0xFF, which can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

<d1> Is the I/O memory in or after 0x20-0x5F?

7.) <q1>To enhance the computing power of the LGT8XM core, a 16-bit LD/ST extension has been added to the instruction set. This 16-bit LD/ST expansion works with the <q2>16-bit Dimensional Operation Acceleration Unit (uDSU) For efficient 16-bit data operations. At the same time, the kernel also increases the 16-bit access capability to the RAM space. The 16-bit LD/ST extension can pass 16 bits of data between the uDSU, RAM, and working registers. Please refer to the "Digital Operation Accelerator" section for details.

<Not in ATMEL DS>

为增强 LGT8XM 内核的运算能力 ,指令流行线中增加了 16 位的 LD/ST 扩展。此 16 位 LD/ST 扩展配合 16 数字运算加速单元(uDSU)工作,实现高效的 16 位数据运算。同时内核也增加对 RAM 空间的 16 位访问能力。因此 16 位 LD/ST 扩展可以在 uDSU,RAM,以及工作寄存器之间传递 16 位的数据。具体细节请参考”数字运算加速器”章节。
To enhance the computing power of the LGT8XM core, a 16-bit LD/ST extension has been added to the instruction line. This 16-bit LD/ST expansion works with the 16-Dimensional Operation Acceleration Unit (uDSU) for efficient 16-bit data operations. At the same time, the kernel also increases the 16-bit access capability to the RAM space. So the 16-bit LD/ST extension can pass 16 bits of data between the uDSU, RAM, and working registers. Please refer to the "Digital Operation Accelerator" section for details.

<q1> Does this look correct? (I have no second source reference for this)
<q2> What is the meaning behind the “16-” in “16-Dimensional Operation Acceleration Unit (uDSU)….?” Is this some kind of error that should be “16-bit” or is it some kind of feature? (I assume it’s “16bit” but must ask to avoid making an ASS=U+ME)

ARITHMETIC LOGIC UNIT (ALU)

8.) <q1>The LGT8XM internally contains a 16-bit arithmetic logic unit that can perform 16 bit arithmetic operations on data in one cycle. The highly efficient ALU is connected to 32 general purpose working registers. The ALU has the ability to perform two arithmetic operations between registers or registers and immediate data in one cycle. There are three types of ALU operations: arithmetic, logic, and bit operations. <q2>The ALU also includes a single-cycle hardware multiplier that implements direct signed or unsigned operations on two 8-bit registers in a single cycle. Please refer to the detailed description in the instruction set section.

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, arithmetic operations between general purpose registers or between a register and an immediate are executed. The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Some implementations of the architecture also provide a powerful multiplier supporting both signed/unsigned multiplication and fractional format. See the “Instruction Set” section for a detailed description.

LGT8XM 内部包含了一个 16 位的算术逻辑运算单元,能够在一个周期内完成 16 为数据的算术运算。高效的 ALU 与 32 个通用工作寄存器相连。能够在一个周期内完成两个寄存器或者寄存器与立即数之间的算术逻辑运算。ALU 的运算分为三种:算术,逻辑以及位运算。同时 ALU 部分也包含了一个单周期的硬件乘法器,能够在一个周期内实现两个 8 位寄存器直接的有符号或者无符号运算。请参考指令集部分的详细介绍。
The LGT8XM internally contains a 16-bit arithmetic logic unit that can perform 16 arithmetic operations on data in one cycle. The highly efficient ALU is connected to 32 general purpose working registers. The ability to perform two arithmetic operations between registers or registers and immediate data in one cycle. There are three types of ALU operations: arithmetic, logic, and bit operations. At the same time, the ALU part also contains a single-cycle hardware multiplier that can directly implement signed or unsigned operations on two 8-bit registers in one cycle. Please refer to the detailed description of the instruction set section.

<q1>This has become more of a personal question for me at this point, but... does this seem significant? I don't believe ATMEL ever disclosed the architecture they use for the AVR's ALU. Someone on AVRfreaks took an educated guess in a thread a couple of years ago concluding that he thought it had to be an 8 bit unit. This is all well beyond my paygrade though. The questions on my mind are, is this an upgraded feature or marketing spin, like "it works with two 8 bit operands, therefore it's "16bit" (wink wink)?" Or, is this the functional source of the claimed faster instruction executions? <<this is the rabbit hole that has been really bugging me>>

<q2>Does this look like a correct translation? This is a unique line/feature added that is not in the ATMEL DS.

STATUS REGISTER (SREG)

9.) The status register mainly stores the result information generated by the execution of the most recent ALU operation. This information is used to control the program execution flow. The status register is updated after the ALU operation has completely ended. This usually eliminates the need for separate compare instructions, resulting in a more compact and efficient code implementation. The value of the status register is not automatically saved and restored in response to an interrupt and exit from the interrupt, which requires software to implement.

The Status Register contains information about the result of the most recently executed arithmetic instruction. This information can be used for altering program flow in order to perform conditional operations. Note that the Status Register is updated after all ALU operations, as specified in the Instruction Set Reference. This will in many cases remove the need for using the dedicated compare instructions, resulting in faster and more compact code.
The Status Register is not automatically stored when entering an interrupt routine and restored when returning from an interrupt. This must be handled by software

状态寄存器中主要保存了因执行最近一次 ALU 运算而产生的结果信息。这些信息用于控制程序执行流程。状态寄存器是在 ALU 操作完全结束后更新,这样就可以省去了使用单独的比较指令,可以带来更加紧凑高效的代码实现。状态寄存器的值在响应中断和从中断中退出时并不会自动保存和恢复,这需要软件去实现。
The status register mainly stores the result information generated by the execution of the most recent ALU operation. This information is used to control the program execution flow. The status register is updated after the ALU operation has completely ended, thus eliminating the need for separate compare instructions, resulting in a more compact and efficient code implementation. The value of the status register is not automatically saved and restored in response to an interrupt and exit from the interrupt, which requires software to implement.