LGT8FX8P Series – Programming Manual 1.0.4J
Errata Page 17-19
Jake 2019 Oct 19 / Oct 20th
Discrepancy My edited text ATMEL text Original+Google Translate
---

The LGT8FX8P series of microcontrollers have 3 blocks of register addresses in the first 256 bytes of RAM data memory. The first block section is the Register File. This is the 32 Special Function Register, and is addressed from 0x00 to 0x1F. The next 64 addresses 0x20 to 0x5F are the Direct I/O Register. The third section is the Extended I/O Register. This occupies from 0x60 to 0xFF. In decimal this breaks down to addresses 0 to 31, 32 through 95, and 96 through 255. The 32x8 Register File has already been described in the CPU Architecture section beginning on page 11. The 64 Direct I/O Registers can be accessed via the faster IN/OUT instructions. The following 160 Extended I/O Registers from 0x60-0xFF are mapped to the data storage space. These registers are only accessible through the slower data addressing commands such as ST/STS/STD and LD/LDS/LDD.

NOTE: major rewrite here. I was initially confused while trying to figure out what either datasheet was trying to communicate when I read this for the first time a couple of weeks ago. If we're going to call this an "overview" or "introduction" at least answer the most basic questions of what are we talking about, where does it begin and end, and what the total size is from the start.
The ATmega48P/88P/168P/328P is a complex microcontroller with more peripheral units than can be supported within the 64 locations reserved in the Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The LGT8FX8P family of microcontrollers is a relatively complex microcontroller that supports many different types of peripherals whose controllers are allocated in 64 I/O register spaces. It can be accessed directly via the IN/OUT instruction. The control registers of other peripherals are allocated in the 0x60~0xFF area. Since this part of the space is mapped into the data storage space, it can only be accessed by commands such as ST/STS/STD and LD/LDS/LDD.
LGT8FX8P 系列微控制器是一种相对复杂的微控制器,它支持多种不同类型的外设,这些外设的控制器被分配在 64 个 I/O 寄存器空间内。可以直接通过 IN/OUT 指令访问。另一些外设的控制寄存器分配在 0x60~0xFF 区域内,由于这部分空间是映射到数据存储空间内,只能通过 ST/STS/STD 以及 LD/LDS/LDD 等指令访问。

The first 256 addresses are directly followed by a maximum of 2K bytes of data SRAM. The portion of the space from 0x4000 to the end of 0xBFFF maps the FLASH program memory location.

(this is the area they were replacing but it is completely rewritten)
The lower 768/1280/1280/2303 data memory locations address both the Register File, the I/O memory, Extended I/O memory, and the internal data SRAM. The first 32 locations address the Register File, the next 64 location the standard I/O memory, then 160 locations of Extended I/O memory, and the next 512/1024/1024/2048 locations address the internal data SRAM.

The system data storage space of LGT8FX8P starts from 0 address and maps common working register file, I/O space, extended I/O space and internal data SRAM space. The first 32 byte addresses correspond to the 32 general working registers of the LGT8XM core. The next 64 addresses are standard I/O spaces that can be accessed directly by the IN/OUT instruction. The next 160 addresses are the extended I/O space, followed by a maximum of 2K bytes of data SRAM. The portion of the space from 0x4000 to the end of 0xBFFF maps the FLASH program memory location.
LGT8FX8P 的系统数据存储空间从 0 地址开始,分别映射了通用工作寄存器文件,I/O 空间,扩展 I/O 空间以及内部数据 SRAM 空间。最开始的 32 个字节地址对应 LGT8XM 内核 32 个通用工作寄存器。接下来的 64 个地址是可以通过 IN/OUT 指令直接访问的标准 I/O 空间。然后的 160 个地址是扩展 I/O 空间,在接下来就是最多 2K 字节的数据 SRAM。从 0x4000 开始到 0xBFFF 结束的这部分空间,映射了 FLASH 程序存储单元。

The 1KB (LGT8F88P/LGT8F168P) or 2KB (LGT8F328P) of SRAM is mapped to two separate locations. The first space from 0x0100 to 0x0900 is read and written by the CPU in 8-bit bytes. The second SRAM space starting from 0x2100 to 0x2900 is a 16-bit wide access space. The system RAM is mapped to the high-order address starting from 0x2100 and is designed to work with the uDSU module to achieve efficient 16-bit data storage. When programming, the normal 8-bit address is offset by 0x2000 to switch to 16-bit access mode.

(obviously this ain’t in grandpa’s Atmel datasheet)

The 1K/2K byte SRAM in the system is mapped to two spaces, respectively. This space from 0x0100 to 0x0900 is read and written by the kernel in 8-bit bytes. Starting from 0x2100 to 0x2900, this area is a 16-bit wide access space. The system RAM is mapped to the high-order address starting from 0x2100 and is mainly used to work with the uDSU module to achieve efficient 16-bit data storage. When programming, the normal 8-bit addressing variable address is offset by 0x2000 to switch to 16-bit access mode.
系统内 1K/2K 字节 SRAM 被分别映射到两个空间。从 0x0100 开始到 0x0900 结束的这个空间被内核以 8 位字节的宽度读写。从 0x2100 开始到 0x2900 结束这个区域为 16 位宽度的访问空间。系统 RAM 被映射到 0x2100 开始的高位地址主要用于配合 uDSU 模块工作,实现高效的 16 位数据存储。在编程时,将普通的 8 位寻址变量地址加上 0x2000 的偏移量,即可切换到 16 位访问模式。

{page 18 for continuity}

The five different addressing modes for the data memory are: Direct, Indirect, Indirect with Displacement, Indirect with Pre-decrement, and Indirect with Post-increment. In the Register File, registers R26 to R31 are the indirect addressing pointer registers. *Indirect* access can address the entire data storage space.

The Indirect with Displacement addressing mode incorporates a 6 bit address as part of the the instruction word. These six bits are added to the the address given by the Y or Z Registers to access up to 63 address locations from the Y or Z Register base address.

When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented. Please refer to the instruction set description section for details.

The five different addressing modes for the data memory cover: 1Direct, 2Indirect with Displacement, 3Indirect, 4Indirect with Pre-decrement, and 5Indirect with Post-increment. In the Register File, registers R26 to R31 feature the indirect addressing pointer registers.
The direct addressing reaches the entire data space.
The Indirect with Displacement mode reaches 63 address locations from the base address given by the Y- or Z-register.
When using register indirect addressing modes with automatic pre-decrement and post-increment, the address registers X, Y, and Z are decremented or incremented.
The 32 general purpose working registers, 64 I/O Registers, 160 Extended I/O Registers, and the 512/1024/1024/2048 bytes of internal data SRAM in the ATmega48P/88P/168P/328P are all accessible through all these addressing modes. The Register File is described in ”General Purpose Register File” on page 11.

Notes: I could certainly be wrong here but I think this (direct/indirect) may be a mistake in the Atmel datasheet that was caught and corrected in the LGT version or this is an intentional difference. I’m not sure. I’m going with the Indirect version though.
The system supports five different addressing modes that can cover the entire data space: 1direct access, 2indirect access with offset, 3indirect access, 4indirect access to the decremented address before access, and 5indirect access to the incremental address after access. The general working registers R26 to R31 are used for indirect access to the address pointer. Indirect access can address the entire data storage space. Indirect access with offset address can be addressed to 63 address spaces in the vicinity of the Y/Z register.
When using the register indirect access mode that supports address auto-increment/decrement, the address register X/Y/Z is automatically decremented/incremented by hardware before/after the access occurs. Please refer to the instruction set description section for details.
系统支持 5 种不同的寻址模式可以覆盖到整个数据空间:直接访问,带偏移的间接访问,间接访问,访问前递减地址的间接访问,访问后递增地址的间接访问。通用工作寄存器 R26 到 R31 用于间接访问的地址指针。间接访问可以寻址整个数据存储空间。带偏移地址的间接访问能够寻址到以 Y/Z 寄存器为基地址的附近 63 个地址空间。
当使用支持地址自动递增/递减的寄存器间接访问模式,地址寄存器 X/Y/Z 会在访问发生前/后自动由硬件递减/递增。具体请参考指令集描述部分。

The 16-bit register X/Y/Z and its associated automatic addressing mode (increment, decrement) also play a very important role in the 16-bit extended mode. The 16-bit extended mode can use the LD/ST increment/decrement mode to implement auto-increment and decrement addressing with variables. This mode is very effective when performing operations on arrays. For details, please refer to the relevant chapter, "Digital Operational Accelerator (uDSU)".

(obviously this 16 bit extended mode and uDSU ain’t in grandpa’s Atmel datasheet)

The 16-bit register X/Y/Z and its associated automatic addressing mode (increment, decrement) also play a very important role in the 16-bit extended mode. The 16-bit extended mode can use the LD/ST increment/decrement mode to implement auto-increment and decrement addressing with variables. This mode is very effective when performing operations on arrays. For details, please refer to the relevant chapter of "Digital Operational Accelerator (uDSU)".
16 位寄存器 X/Y/Z 以及与之相关的自动寻址模式(递增、递减),在 16 位扩展模式下也有着非常重要的作用。16 位扩展模式可以使用 LD/ST 的递增/递减模式,实现带变量的自动递增、递减寻址。这种模式在对数组进行运算操作时,将非常有效。具体实现请参考“数字运算加速器(uDSU)”相关章节。

The following sections are omitted entirely from the LGT DS but are in the Atmel version:

Data Memory Access Times

This section describes the general access timing concepts for internal memory access. The internal data SRAM access is performed in two clkCPU cycles as described in Figure 5-4.
Figure 5-4.
On-chip Data SRAM Access Cycles

EEPROM Data Memory

The ATmega48P/88P/168P/328P contains 256/512/512/1K bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described in the following, specifying the EEPROM Address Registers, the EEPROM Data Register, and the EEPROM Control Register.
”Memory Programming” on page 294 contains a detailed description on EEPROM Programming in SPI or Parallel Programming mode.

EEPROM Read/Write Access

The EEPROM Access Registers are accessible in the I/O space.
The write access time for the EEPROM is given in Table 5-2. A self-timing function, however, lets the user software detect when the next byte can be written. If the user code contains instructions that write the EEPROM, some precautions must be taken. In heavily filtered power supplies, VCC is likely to rise or fall slowly on power-up/down. This causes the device for some period of time to run at a voltage lower than specified as minimum for the clock frequency used. See ”Preventing EEPROM Corruption” on page 20 for details on how to avoid problems in these situations.
In order to prevent unintentional EEPROM writes, a specific write procedure must be followed. Refer to the description of the EEPROM Control Register for details on this.
When the EEPROM is read, the CPU is halted for four clock cycles before the next instruction is executed. When the EEPROM is written, the CPU is halted for two clock cycles before the next instruction is executed.

Preventing EEPROM Corruption

During periods of low VCC, the EEPROM data can be corrupted because the supply voltage is too low for the CPU and the EEPROM to operate properly. These issues are the same as for board level systems using EEPROM, and the same design solutions should be applied.
An EEPROM data corruption can be caused by two situations when the voltage is too low. First, a regular write sequence to the EEPROM requires a minimum voltage to operate correctly. Secondly, the CPU itself can execute instructions incorrectly, if the supply voltage is too low.
EEPROM data corruption can easily be avoided by following this design recommendation:
Keep the AVR RESET active (low) during periods of insufficient power supply voltage. This can be done by enabling the internal Brown-out Detector (BOD). If the detection level of the internal BOD does not match the needed detection level, an external low VCC reset Protection circuit can be used. If a reset occurs while a write operation is in progress, the write operation will be completed provided that the power supply voltage is sufficient.

General Purpose I/O Registers
General purpose I/O register
通用 I/O 寄存器

The I/O space of the LGT8FX8P has three general-purpose I/O registers GPIOR2/1/0. These three registers can be accessed using the IN/OUT instruction to store user-defined data.

The ATmega48P/88P/168P/328P contains three General Purpose I/O Registers. These registers can be used for storing any information, and they are particularly useful for storing global variables and Status Flags. General Purpose I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI, CBI, SBIS, and SBIC instructions.
Note: this section is in reverse order on the LGT datasheet. Atmel has this little excerpt after the I/O Memory section instead of before, but is otherwise similar. I think the Atmel version is just over specified here and the LGT version is simplified so I'm not going to highlight it in my final text.

The I/O space of the LGT8FX8P has three general-purpose I/O registers GPIOR2/1/0. These three registers can be accessed using the IN/OUT instruction to store user-defined data.
LGT8FX8P 的 I/O 空间有三个通用 I/O 寄存器 GPIOR2/1/0,这三个寄存器可以使用 IN/OUT 指令访问,用于存放用户自定义数据。

I/O Memory
I/O Memory
Peripheral Register Space
外设寄存器空间
For more details about the I/O space, please refer to the "Register Summary" section.

The I/O space definition of the ATmega48P/88P/168P/328P is shown in "Register Summary" on page 425.
For a detailed definition of the I/O space, please refer to the "Register Overview" chapter in the LGT8FX8P data sheet.
I/O 空间的详细定义,请参考 LGT8FX8P 数据手册中"寄存器概述"章节。

All LGT8FX8P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Please refer to the instruction set description section for details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. Peripheral registers allocated in the Extended I/O space (0x60~0xFF) can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

All ATmega48P/88P/168P/328P I/Os and peripherals are placed in the I/O space. All I/O locations may be accessed by the LD/LDS/LDD and ST/STS/STD instructions, transferring data between the 32 general purpose working registers and the I/O space. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions. Refer to the instruction set section for more details. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48P/88P/168P/328P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

The peripherals of the LGT8FX8P are all allocated to the I/O space. All I/O space addresses are accessible by LD/LDS/LDD and ST/STS/STD instructions. The accessed data is passed through 32 general working registers. I/O registers between 0x00 and 0x1F are accessible through the bit addressing instructions SBI and CBI. In these registers, the value of a bit can be detected using the SBIS and SBIC instructions to control the execution flow of the program. Please refer to the instruction set description section for details.

When accessing an I/O register using an IN/OUT instruction, the address between 0x00 and 0x3F must be addressed. When accessing the I/O space using LD or ST instructions, the mapped address of the unified mapped space in the system data memory must be accessed through the I/O space (plus an offset of 0x20). Other peripheral registers (0x60~0xFF) allocated in the extended I/O space can only be accessed using ST/STS/STD and LD/LDS/LDD instructions.

LGT8FX8P 所以的外设都被分配到 I/O 空间。所有的 I/O 空间地址都可以被 LD/LDS/LDDD 以及 ST/STS/STD 指令访问。访问的数据都是通过 32 个通用工作寄存器传递。在 0x00~0x1F 之间的 I/O 寄存器可以通过位寻址指令 SBI 和 CBI 访问。在这些寄存器中,某一个位的值可以使用 SBIS 和 SBIC 指令检测,用以控制程序的执行流程。具体请参考指令集描述部分。

当使用 IN/OUT 指令访问 I/O 寄存器时,必须寻址 0x00~0x3F 之间的地址。当使用 LD 或 ST 指令访问 I/O 空间时,必须通过 I/O 空间在系统数据存储器统一映射空间的映射地址访问(加上 0x20 的偏移)。其他一些分配在扩展 I/O 空间的外设寄存器(0x60~0xFF), 只能够使用 ST/STS/STD 和 LD/LDS/LDD 指令访问。

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written. Some of the Status Flags are cleared by writing a logical one to them. Note that, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.

For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
The I/O and peripherals control registers are explained in later sections.

In order to be compatible with future devices, the reserved bits must be written with a 0 when writing. You cannot perform a write operation on a reserved I/O space.
Some registers include a status flag that needs to be written to 1 to be cleared. It should be noted that CBI and SBI instructions only support specific bits, so CBI/SBI can only work on registers that contain these status flags. In addition, the CBI/SBI instruction can only operate on registers in the address range 0x00 to 0x1F.
为了与未来的设备兼容,保留位在写操作时必须写 0。不能在保留的 I/O 空间上执行写操作。
一些寄存器中包括了状态标志,需要被写 1 才能清零。需要注意的是,CBI 和 SBI 指令仅仅支持特定的位,因此 CBI/SBI 也只能工作在包含这些状态标志的寄存器上。除此之外,CBI/SBI 指令只能工作在 0x00 到 0x1F 这个地址范围内的寄存器。

FLASH Controller (E2PCTL)
The LGT8FX8P has a special internal memory access controller, the E2PCTL. This controller has access to the entire EFLASH memory space at the byte level. The E2PCTL handles the emulated E2PROM read and write access within the FLASH Program Memory space. This E2PROM management includes an algorithm for erasure and equalization that can increase the life cycle of the data FLASH by an order of magnitude, and can guarantee more than 100,000 erase cycles.
In addition to managing the emulated E2PROM, the E2PCTL controller also manages the FLASH Program Memory. This includes the in circuit programming features for upgrading software inside the microcontroller. When the E2PCTL is in the FLASH Program Memory access mode, it supports page erase only. Each page is 1024 bytes. The E2PCTL FLASH Program Memory read and write access is 32-bits wide.

LGT8FX8P internal integration integrates a flexible and reliable EFLASH read/write controller, which can utilize the existing data FLASH storage space in the system to realize the storage space of byte read and write access, realize the storage application similar to E2PROM; E2PROM interface simulation adopts erasing The equalization algorithm can increase the usage period of the data FLASH by about 1 time, and can guarantee more than 100,000 erasing cycles.
The E2PCTL controller also implements an online erasing operation for the FLASH program space, and the function of automatically upgrading the firmware online can be realized by software.
Access to the program FLASH program space through the FLASH controller, only support page erase (1024 bytes) and 32-bit wide read and write access.
LGT8FX8P 内部实现集成了一个灵活可靠的 EFLASH 读写控制器,可以利用系统中已有的数据 FLASH 存储空间,实现字节读写访问的存储空间,实现类似 E2PROM 的存储应用;E2PROM 接口模拟采用擦写均衡的算法,可以将数据 FLASH 的使用周期提高 1 倍左右,能够保证 100,000 次以上的擦写周期。

E2PCTL 控制器也实现了对 FLASH 程序空间的在线擦写操作,可以通过软件实现在线自动升级固件的功能。通过 FLASH 控制器访问程序 FLASH 程序空间,只支持页擦除(1024 字节)以及 32 位宽度的读写访问。
{page19}

LGT8F88D/168D E2PCTL Controller Block Diagram
LGT8F88D/168D E2PCTL 控制器结构图

When the E2PCTL is in the emulated E2PROM mode, it supports 8-bit and 32-bit read/write widths. It supports 32-bit read/write when accessing the FLASH Program Memory. The entire EFLASH Memory of the LGT8FX8P is mapped as 32 bits. Therefore it is recommended to use 32-bit access, especially for programming operations. The 32-bit read and write operation is more efficient, and also helps ensure the erasing cycle life of the FLASH memory.

When the E2PCTL analog E2PROM function accesses the data FLASH space, it can support 8-bit and 32-bit read/write widths. Supports page erase and 32-bit data read and write when accessing the program's FLASH space. Since the minimum memory location of the internal FLASH of the LGT8FX8P is 32 bits, it is recommended to use 32-bit access, especially for write operations. The 32-bit access read and write operation is not only efficient, but also helps protect the erasing life of the FLASH memory unit.
E2PCTL 模拟 E2PROM 功能访问数据 FLASH 空间时,可以支持 8 位、32 位读写宽度。访问程序 FLASH 空间时,支持页擦除和 32 位数据读写。由于 LGT8FX8P 内部 FLASH 的最小存储单元为 32 位,因此建议用 32 位访问方式,特别是对于写操作。32 位访问的读写操作不仅效率高,也有利于保护 FLASH 存储单元的擦写寿命。

LGT8F328P E2PCTL Controller Block Diagram
LGT8F328P E2PCTL 控制器结构图

The entire Program Memory is monolithically integrated inside the LGT8F328P. The LGT8XM architecture in the LGT8F328P shares the internal 32K bytes of EFLASH memory with the emulated E2PROM. Users can partition the 32K byte EFLASH Memory into separate Program Memory and emulated E2PROM Memory according to their needs. By configuring the E2PCTL settings, you can adjust the partition size of the emulated E2PROM. The emulated E2PROM is implemented using a 1K byte page swap mode. For instance, to simulate 1K bytes of E2PROM space, the E2PCTL controller requires 2K bytes of FLASH space. In this example a 1K byte page has programmed data. In order to erase or write changes an additional 1K byte page is required to make a page swap. Likewise, to achieve 4K bytes of E2PROM, 8K bytes of FLASH Program Memory are partitioned and reserved for E2PROM. For more information, please refer to the E2PCTL Algorithm Implementation description.

There is no extra data FLASH inside the LGT8F328P. Therefore, the LGT8XM core shares the internal 32K bytes of FLASH memory space with E2PCTL. Users can divide the 32K byte FLASH space into program space and data space as needed. By configuring the E2PCTL controller, you can set the size of the analog E2PROM. E2PCTL implements the analog E2PROM logic using the page swap mode, which is in pages (1K bytes). Therefore, to simulate 1K bytes of E2PROM space, it takes 2K bytes of FLASH space, and so on, to achieve 4K bytes of E2PROM, it takes 8K bytes of FLASH space. For specific implementation, please refer to the description of the implementation of the E2PCTL algorithm.

LGT8F328P 内部没有多余的数据 FLASH. 因此,LGT8XM 内核与 E2PCTL 共享内部 32K 字节 FLASH 存储空间。用户可以根据需要,将 32K 字节 FLASH 空间划分为程序空间和数据空间。通过配置 E2PCTL 控制器,可以设置模拟 E2PROM 的空间大小。E2PCTL 使用页交换模式实现模拟 E2PROM 逻辑,算法以页(1K 字节)为单位。因此模拟 1K 字节的 E2PROM 空间,需要占用 2K 字节的 FLASH 空间,以此类推,实现 4K 字节的 E2PROM,需要占用 8K 字节的 FLASH 空间。具体实现方式,请参考 E2PCTL 算法实现的描述。