F18A Technology Reference

This manual fully documents the instruction set and the generic I/O facilities incorporated in the F18A computer and, therefore, in chips comprised of them.

This manual is supplemented by lectures at the arrayForth Institute and by examples in the Application Notes and other documentation available from our company.

After understanding the F18A technology, the next step is to study the Data Book for the particular chip of interest. The Chip Data Book will show the configuration of the node array, the I/O facilities and interconnections in that particular chip, and the ROM content for each node. In addition the chips are characterized physically, electrically and thermally.

For people accustomed to reading hardware manuals this should be refreshingly brief, as befits an uncommonly simple architecture. We have made an effort to avoid redundancy in the interest of brevity; as a result, the information density is high, warranting a careful reading. More than one reader has glossed over sections that did not seem important only to find later that they were very important indeed!

Your satisfaction is very important to us! Please familiarize yourself with our website at http://www.greenarraychips.com/. This will lead you to the latest software and documentation as well as resources for solving problems and contact information for obtaining help or information in real time.

Contents

1. Introduction 3

1.1 Status of Data Given 3
1.2 Documentation Conventions 3
1.2.1 Numbers 3
1.2.2 Node coordinates 3
1.2.3 Register names 3
1.2.4 Bit Numbering 3

2. The F18A Computer 4

2.1 After Reset 5
2.2 Memory Addressing 5
2.3 Instruction Set 5
2.3.1 Instruction Word Format 6
2.3.2 Stack Operation 6
2.3.3 Arithmetic, Logic and Register Manipulation 7
2.3.4 Memory Read and Write 8
2.3.5 Transfer of Program Control 8
2.4 Fine Points 9
2.4.1 Execution Time 9
2.4.2 Prefetch considerations 9
2.4.3 Best Practices 9
2.4.4 Compiling Instructions Yourself 9

3. F18A I/O Facilities 10

3.1 The io Register 10
3.2 Suspension 11
3.3 Communication Ports 11
3.3.1 Multiport Operations 11
3.3.2 Port Execution 11
3.4 Nodes with GPIO Pins 12
3.4.1 Wakeup Pins 12
3.4.2 Shared (Read-only) Pins 12
3.4.3 Phantom Wakeup Signals 12
3.5 Parallel Bus Support 13
3.6 SERDES Nodes 13
3.7 Analog Nodes 14

4. Revision History 15

1. Introduction

The F18 technology consists of small, simple computers and several types of versatile, software defined I/O. Our chips are configured as arrays of the F18 computers with appropriate selections of I/O types.

An F18 computer is a physically small, very fast, energy-efficient and inexpensive 18-bit machine. Each F18 is self-contained as a complete system, with its own RAM, ROM, stacks, registers, and where appropriate external I/O. The F18 computers in an array operate independently and asynchronously, communicating with each other using efficient internal communication ports.

This book will familiarize you with the architecture, internal structure, and use of the computers and I/O of chips built from F18A technology. Please refer to the data books for specific chips using this technology to learn their exact configurations, pin-outs, ROM content and electrical characteristics. The current editions of these, along with many other relevant documents and application notes as well as the current edition of this document, may be found on our website at http://www.greenarraychips.com . It is always advisable to ensure that you are using the latest documents before starting work.

1.1 Status of Data Given

The data given herein are PRODUCTION. Electrical and timing values are illustrative only. Otherwise, the subject matter of this book is the architecture, instruction set, and programming of the F18A computers and I/O, which are fixed by design. Future revisions of this book may document upward-compatible enhancements of the F18A programming model.

1.2 Documentation Conventions

1.2.1 Numbers

Numbers are written in decimal unless otherwise indicated. Hexadecimal values are indicated by explicitly writing “hex” or by preceding the number with the lowercase letter “x”. In colorForth coding examples, hexadecimal values are italicized and darkened.

1.2.2 Node coordinates

Each GreenArrays chip is a rectangular array of ***nodes,*** each of which is an F18 computer. By convention these arrays are visualized as seen from the top of the silicon die, which is normally the top of the chip package, oriented such that pin 1 is in the upper left corner. Within the array, each node is identified by a three or four digit number denoting its Cartesian coordinates within the array as *yxx* or *yyxx* with the lower left corner node always being designated as node 000. Thus, for a GA144 chip whose computers are configured in an array of 18 columns and 8 rows, the numbers of the nodes in the lower right, upper left, and upper right corners are 017, 700, and 717 respectively.

1.2.3 Register names

Register names in prose may be used with or without the word "register" and are usually shown in a bold font and capitalized where necessary to avoid ambiguity, such as for example the registers **T S R I A B** and **IO** or **io** .

1.2.4 Bit Numbering

Binary numbers are visualized as a horizontal row of bits, numbered consecutively right to left in ascending significance with the least significant bit numbered zero. Thus bit *n* has the binary value 2^n. The notation P9 means bit 9 of register **P**, whose binary value is x200, and T17 means the sign (high order) bit of 18-bit register **T**.

2. The F18A Computer

Here is an introduction to the resources within each F18A computer. All elements are 18 bits wide unless otherwise noted.

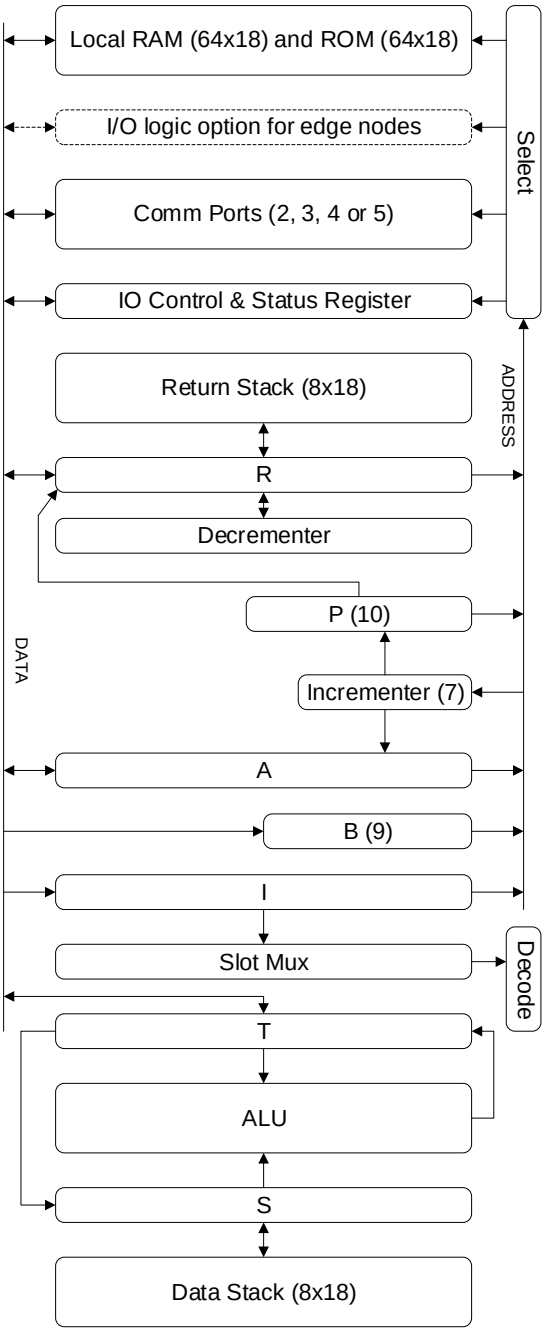

Figure 1 F18A Block Diagram

- **RAM**: 18 bits wide, 64 words in each node.
- **ROM**: 18 bits wide, 64 words in each node.
- **I/O logic** may be present on any edge node of a chip.
- **Communication Ports** are configured based on position within the array (four for inner; three ports array edges; two for corners.) One additional port is optional.
- **IO** control and status for communication ports and I/O logic.
- **Return Stack** consists of 8 registers addressed circularly (no fixed top or bottom.) Pushing a word onto this stack makes it the new "top" item, overwriting the former "bottom" item. After popping 8 items, those 8 items are repeated.
- **R** extends this stack to 9 entries but only 8 are circular. When used as a loop counter, **R** decrements toward zero.
- **P** serves as a "program counter", holding the address of the next word in the instruction stream. Within RAM or ROM, it is incremented after each word is fetched, circularly within whichever storage class it currently points to. When **P** addresses I/O space, it is not incremented. **Bit P9** enables Extended Arithmetic Mode when set.
- **A** is a general purpose read/write address or data register.
- **B** is a write only address register, containing on reset the address of **io** register.
- **I** is the instruction register into which instruction words containing 1, 2, 3 or 4 opcodes are fetched for execution.
- **T** is the top word of a 10-element data stack.
- **The ALU** (Arithmetic/Logic unit) performs unary operations on **T**, binary operations on **S** and **T**, or a multiply step involving **S**, **T**, and **A** as directed by opcodes. A carry latch is involved in some Extended Arithmetic mode instructions.
- **S** is the second word of a 10-element data stack.
- **Data Stack** is 8 circularly addressed registers constituting the third through 10th elements of a 10-element stack.

2.1 After Reset

On chip reset, the following registers are initialized:

P is set to the address configured in the chip layout. For F18A, this will always be either a multiport execute or will be ROM address ***x0aa***.

io is set to the state it would have after a program wrote ***x15555*** into the register.

B is set to the address of **io**.

Stack pointers are set to the same initial condition on every reset.

Other registers, stack contents, and RAM are not directly affected by reset.

2.2 Memory Addressing

The F18A decodes a nine-bit memory address as shown in Figure 2.

Start	End	Address Space
x000	x03F	RAM, 64 words
x040	x07F	RAM x000-x03F repeated
x080	x0BF	ROM, 64 words
x0C0	x0FF	ROM x080-x0BF repeated
x100	x1FF	I/O ports and registers

Figure 2 Address Map

Address incrementing, when applied to A or P, continues from x03F to x040 but wraps from x07F to x000. The same process occurs with addresses in the ROM space. Incrementing does not occur at all when the address lies in the region of I/O ports and registers. Incrementing never affects bits P8 or P9. P9 has no effect on memory decoding; it simply enables the Extended Arithmetic Mode when set.

2.3 Instruction Set

The F18 instruction set consists of 5-bit opcodes as summarized in Figure 3:

Opcode	Notes	Opcode	Notes
;	return	**+***	multiply step
ex	execute (swap P and R)	**2***	left shift
name ;	jump to *name*	**2/**	right shift (signed)
name	call to *name*	**–**	invert all bits
unext	loop within I (decrement R)	**+**	add (or add with carry)
next	loop to address (decrement R)	**and**	
if	jump if T=0	**or**	exclusive or
-if	jump if T≥0	**drop**	
@p	literal (autoincrement P)	**dup**	
@+	fetch via A (autoincrement A)	**pop**	from R to T
@b	fetch via B	**over**	
@	fetch via A	**a**	fetch *from* register A
!p	store via P (autoincrement P)	**.**	nop
!+	store via A (autoincrement A)	**push**	from T to R
!b	store via B	**b!**	store *into* register B
!	store via A	**a!**	store *into* register A

Figure 3 Table of Opcodes

The shaded opcodes may be used in slot 3 of an instruction word (see 2.3.1 below). Add (+) and multiply step (+*) behave differently in Extended Arithmetic Mode.

2.3.1 Instruction Word Format

Each F18A instruction word is subdivided into four possible slots for opcodes that are normally executed sequentially left to right when viewing the word as conventionally oriented with most significant bit to the left. Jump instructions may be written in slots 0, 1, or 2, and when they are encountered the remainder of the instruction word is interpreted as an address field. This results in four possible instruction word formats as shown in Figure 4:

<table>
<tr><th colspan="5">Slot 0</th><th colspan="3">Slot 1</th><th colspan="4">Slot 2</th><th colspan="4">Slot 3</th></tr>
<tr><td colspan="5">Instruction 0</td><td colspan="3">Instruction 1</td><td colspan="4">Instruction 2</td><td colspan="4">Instruction 3</td></tr>
<tr><td colspan="5">Instruction 0</td><td colspan="3">Instruction 1</td><td colspan="4">Jump instruction</td><td colspan="4">Destination</td></tr>
<tr><td colspan="5">Instruction 0</td><td colspan="3">Jump instruction</td><td colspan="8">Destination</td></tr>
<tr><td colspan="5">Jump instruction</td><td colspan="3">Unused</td><td colspan="8">Destination</td></tr>
<tr><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

Figure 4 Instruction word formats

The n-bit destination address field simply replaces the n low order bits of the current (incremented) value of P at the time the jump instruction executes. Thus the scope of a slot 2 jump is very limited while a slot 0 jump can reach any addressable destination, and can control P9 to explicitly enable or disable Extended Arithmetic Mode. Slot 1 and 2 jumps have no effect upon P9, so its selection is unchanged as these jumps are executed. In the F18A, slot 1 and 2 jumps additionally force P8 to zero. This means the destination of any slot 1 or 2 jump may never be in I/O space.

The instruction selected for slot 3 must be one of those shaded in Figure 3 above. When the desired opcode cannot be fit into the current instruction word, either because it may not be used in slot 3 or because it is a jump whose destination requires a longer address field, the remaining slots of the current word may be filled with **nop** instructions.

In addition to jumps, three other instructions affect the consecutive execution of opcodes within an instruction word:

; (return) and **ex** (execute) discontinue execution of the current instruction word.

unext (micronext) conditionally causes execution to loop back to slot 0 (and continue with consecutive slots). When the loop is complete, execution continues with the slot following unext .

The arrayForth compiler formats instruction words correctly and automatically. If you wish to compile your own instruction words, to modify code in memory, or to write your own compiler for the F18A, please refer to section 2.4.4 for details about instruction encoding.

2.3.2 Stack Operation

In all cases of pushing and popping stacks mentioned above, the mechanism is the same:

Pushing data stack: S is pushed onto the circular stack, replacing its bottom entry, and T is copied into S.

Popping data stack: The top entry of the circular stack is copied into S, and S is copied into T unless the instruction states that it sets T to some other value.

Pushing return stack: R is pushed onto the circular stack, replacing its bottom entry.

Popping return stack: The top entry of the circular stack is copied into R.

Pushing and popping of the circular stacks is accomplished by adjusting pointers. When popping a circular stack, the previous top item becomes the new bottom item. When pushing onto a circular stack, the new top value replaces the previous bottom value.

2.3.3 Arithmetic, Logic and Register Manipulation

There are sixteen instructions in this class. Approximate execution times (see section 2.4.1) are 1.5 nanoseconds.

Op	Source	Function
x10	**+***	**Multiply step.** Signed Multiplicand is in S, unsigned multiplier in A, T=0 at start of a step sequence. Uses T:A as a 36-bit shift register with multiplier in A. Does the following: 1. If bit A0 is zero, shifts the 36-bit register T:A right one bit arithmetically (T17 is not changed and is copied into T16. T0 is copied to A17 and A0 is discarded.) 2. If bit A0 is one, S and T are added as though they had both been extended to be 19 bit signed numbers, and the 37-bit concatenation of this sum and A is shifted right one bit to replace T:A. *Overflow may occur if S and T are both nonzero and their signs differ; this can only occur through improper initialization of T*. This instruction is affected in Extended Arithmetic Mode by including the latched carry in the sum in case (2) above, and by latching the carry out from bit 17 of the sum, but this is not a particularly useful effect and may be changed in later F18 versions. *You must arrange that the previously executed instruction has not changed the values of S, T or P9. Use **nop** preceding **Multiply Step** if necessary to meet this condition.*
x11	**2***	**Two-Star.** Shifts T left one bit logically (shifts zero into T0, discards T17) thus multiplying a signed or unsigned value by two.
x12	**2/**	**Two-Slash.** Shifts T right one bit arithmetically (propagates T17 by leaving it unchanged; discards T0) thus dividing a signed value by two and discarding the positive remainder.
x13	**—**	**Not.** Inverts each bit of T, replacing T with its ones complement.
x14	**+**	**Plus.** Replaces T with the twos complement sum of S and T. Pops data stack into S. This instruction is affected in Extended Arithmetic Mode, becoming **Add with carry.** Includes the latched carry in the sum, and latches the carry out from bit 17. *You must arrange that the previously executed instruction has not changed the values of S, T or P9. Use **nop** preceding **Plus** if necessary to meet this condition.*
x15	**and**	Replaces T with the Boolean AND of S and T. Pops data stack.
x16	**or**	**Exclusive Or.** Replaces T with the Boolean XOR of S and T. Pops data stack.
x17	**drop**	Drops the top item from the data stack by copying S into T and popping the data stack.
x18	**dup**	Duplicates the top item on the data stack by pushing the data stack and copying T into S.
x19	**pop**	Moves R into T, popping the return stack and pushing the data stack.
x1A	**over**	Makes a copy of S on top of the data stack by pushing S onto the stack, moving T into S, and replacing T by the previous value of S.
x1B	**a**	Fetches the contents of register A into T, pushing the data stack.
x1C	**.**	**Nop.** Spends time while making no explicit state changes in registers or stacks.
x1D	**push**	Moves T into R, pushing the return stack and popping the data stack.
x1E	**b!**	**B-Store.** Stores T into register B, popping the data stack.
x1F	**a!**	**A-Store.** Stores T into register A, popping the data stack.

Depending on their arguments, any of these may alter T and/or S, with the sole exception of nop .

The carry latch is not affected by reset and its state on power-up is unpredictable.

2.3.4 Memory Read and Write

There are eight instructions in this class. Approximate execution times (see section 2.4.1) are 5.1 nanoseconds when reading or writing internal memory. The addresses used may be in memory or I/O spaces; some I/O addresses may suspend execution until the conditions necessary for their completion are met. The notation [*X*] should be read in the descriptions below as "the memory or I/O addressed by *X*".

Op	Source	Function
x08	**@p**	**Fetch-P.** Pushes data stack, reads [P] into T, and increments P (see 2.2)
x09	**@+**	**Fetch-plus.** Pushes data stack, reads [A] into T, and increments A (see 2.2)
x0A	**@b**	**Fetch-B.** Pushes data stack and reads [B] into T.
x0B	**@**	**Fetch.** Pushes data stack and reads [A] into T.
x0C	**!p**	**Store-P.** Writes T into [P], pops the data stack, and increments P (see 2.2)
x0D	**!+**	**Store-plus.** Writes T into [A], pops the data stack, and increments A (see 2.2)
x0E	**!b**	**Store-B.** Writes T into [B] and pops the data stack.
x0F	**!**	**Store.** Writes T into [A] and pops the data stack.

Figure 6 Memory Read and Write Instructions

2.3.5 Transfer of Program Control

There are eight instructions in this class. Approximate execution times (see section 2.4.1) are 5.1 nanoseconds when the destination is in internal memory. **unext** takes approximately 2.0 nanoseconds. When P is changed by one of these instructions, the new destination may be in memory or I/O space. Some I/O addresses may suspend execution until the conditions necessary for their completion are met.

Op	Source	Function
x00	**;**	**Return.** Moves R into P, popping the return stack. Skips any remaining slots and fetches next instruction word.
x01	**ex**	**Execute.** Exchanges R and P, skips any remaining slots and fetches next instruction word.
x02	***name ;***	**Jump.** Sets P to destination address (see 2.3.1) and fetches next instruction word.
x03	***name***	**Call.** Moves P into R, pushing an item onto the return stack, sets P to destination address (see 2.3.1) and fetches next instruction word.
x04	**unext**	**Micronext.** If R is zero, pops the return stack and continues with the next opcode. If R is nonzero, decrements R by 1 and causes execution to continue with slot 0 of the current instruction word without re-fetching the word.
x05	**next**	**Next.** If R is zero, pops the return stack and continues with the next instruction word addressed by P. If R is nonzero, decrements R by 1 and jumps (see 2.3.1).
x06	**if**	**If.** If T is nonzero, continues with the next instruction word addressed by P. If T is zero, jumps (see 2.3.1).
x07	**-if**	**Minus-if.** If T is negative (T17 set), continues with the next instruction word addressed by P. If T is positive, jumps (see 2.3.1).

Figure 7 Control Transfer Instructions

2.4 Fine Points

For full mastery of the F18A computer, there are several additional things you should understand.

2.4.1 Execution Time

The F18A is built from asynchronous logic, therefore its instruction times are naturally approximate. The time required for all activity varies directly with temperature, inversely with power supply voltage (V_{DD}), and randomly within a statistical distribution due to variations in the process of fabricating the chips themselves. Additional variations may occur with minor revisions in the version of F18 technology used by a particular chip. Applications should not be designed to depend upon the speed of activities within F18 computers for timing purposes without taking all of these variables into consideration.

The time required for interaction with entities outside an F18 computer (other nodes, or I/O) naturally depends on the behavior of those outside devices.

The approximate execution times shown in the text above are typical of the F18A. For any particular chip, see its Product Briefs and Data Books for pertinent preliminary information from prototype chips and, later, statistical data from measurements on production chips.

2.4.2 Prefetch considerations

Any F18A computer may, when it is logically possible to do so, be designed to fetch instructions before they are needed. We reserve the right to make changes or improvements in this process with each chip we design, and therefore you are advised, in the interest of software portability, *not* to depend on any particular prefetching strategy for the timing of your program or for avoiding the need to insert a **nop** between an instruction that alters S, T or P9 and an **add** or **multiply step** instruction. If your requirements justify sacrificing upward compatibility in favor of taking such steps to achieve maximum performance, contact our support staff (hotline mailbox) for recommendations pertinent to specific chips.

2.4.3 Best Practices

Within I/O space, read and write only those addresses defined to be readable or writable to avoid unexpected results.

Reading or writing inappropriate I/O addresses, or writing ROM, may result in a hung computer.

Upward compatibility to future versions of the F18 is not guaranteed. Think in terms of source, not binary compatibility. Be aware of the reset state of the **io** register since that will be the default setting for any bits assigned to control future I/O facilities.

2.4.4 Compiling Instructions Yourself

If you are writing a compiler, or wish to compose F18A instructions by hand or to read or modify instruction words in memory, you must be aware that instruction words are *encoded* as follows:

Compose instruction words as shown in 2.3.1 with opcode values as shown in the “Op” columns of the instruction tables above and exclusive OR the result with the value x15555. If the word requires a destination address field, mask that field to zero and insert an un-encoded address value consistent with those shown in 2.2 and the specific I/O addresses defined in section 3. In the particular case of slot zero jumps, it is permissible to set unused bits 10, 11 and 12 to any convenient states.

3. F18A I/O Facilities

Input-output capabilities of F18A nodes consist of internode communication ports (2, 3, 4, or 5 depending on array configuration and the node's position in the array) and, for nodes on array edges, optional GPIO pins and/or other more specialized circuitry such as parallel buses, analog I/O, or serializer-deserializer (SERDES).

Name	Addr	Description	Notes
io	x15D	I/O control and status (see below). Bit assignments depend on the facilities present.	
data	x141	Up port with no handshake.	3
---u	x145	Up	1
ldata	x171	Left port with no handshake.	3
--l-	x175	Left	2
--lu	x165	Left or Up	1,2
-d--	x115	Down	
-d-u	x105	Down or Up	1
-dl-	x135	Down or Left	2
-dlu	x125	Down, Left or Up	1,2
r---	x1D5	Right	
r--u	x1C5	Right or Up	1
r-l-	x1F5	Right or Left	2
r-lu	x1E5	Right, Left or Up	1,2
rd--	x195	Right or Down	
rd-u	x185	Right, Down or Up	1
rdl-	x1B5	Right, Down or Left	2
rdlu	x1A5	Right, Down, Left or Up	1,2

Figure 8 I/O Address Table

Figure 8 shows all of the defined I/O addresses in the F18A. Shaded addresses are not *present* in all nodes; see the numbered notes for more information.

Addresses should only be used in a given node if they are *present* according to these notes; right, down, left and up ports are present for read and write in all nodes except as noted:

Blank: *Present* for read/write in all nodes.

1 *Absent* on top and bottom nodes unless supported by particular I/O circuitry

2 *Absent* on side nodes unless supported by particular I/O circuitry

3 *Present* only when documented as such for specific I/O circuitry; read, write, and handshaking facilities are defined by each class of circuitry.

3.1 The io Register

This 18-bit control and status register is each F18A computer's interface with its I/O circuitry, if any, and with handshake lines for any communication ports that are present. The bit assignments are shown for reference here. Their semantics are defined in detail in later sections. By default, **io** is populated with latches that read as the inverse of what was last written to them. Any **io** bits that are undefined for a given node because the associated I/O facilities are absent implement this behavior. A second general principle is that on reset each node's **io** register is set as though the value x15555 had been written to it under program control.

Bit	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reset	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
WRITE	GPIO pin 17 control						WD		phan 9		phan 7		GPIO pin 5 control		GPIO pin 3 control		GPIO pin 1 control	
Alt write	SR			A/D mode		DB				9-bit D/A value XORed with x155								
READ	pin 17	Rr-	Rw	Dr-	Dw	Lr-	Lw	Ur-	Uw				pin 5		pin 3		pin 1	

Figure 9 IO Register Bit Assignments

3.2 Suspension

A node is either executing instructions at full speed or it is *suspended* in mid-instruction, waiting for a "wake up" condition that depends on the state of another node or of other I/O circuitry. Since an F18 has no clock, the acts of suspending and waking up consume no energy or time in and of themselves. Perception of changing conditions takes picoseconds within the chip and, typically, nanoseconds for events external to the chip. When a node is suspended the power used by its computer is nil, on the order of 100 nanowatts. When executing, an F18 computer uses power on the order of less than 6 milliwatts depending on the instructions used and the data on which they are operating.

3.3 Communication Ports

Communication (comm) ports provide for direct, point to point bidirectional information transfer within a chip. Each F18A node may have from two to five ports connected. Comm ports are named **Right**, **Down**, **Left**, **Up**; the fifth port is not implemented in F18A based chips.

Each comm port is an independent, bidirectional, half duplex 18-bit channel connecting two nodes. Communication occurs when a sending node writes to a comm port address and a receiving node to which that port is connected reads the corresponding comm port address at the same time. When a node operates on a port the data transfer occurs at approximately memory speed unless the other node connected to the port is not yet performing the complementary operation; in this case the operating node suspends, waking up when the other node connected to that port is performing the complementary operation.

The status of each port is visible through the bits in **io** labeled **R/D/L/Ur-** and **R/D/L/Uw** above. **Dr**- is zero if the node at the other end of **Down** is reading; **Lw** is one if the node at the other end of **Left** is writing.

3.3.1 Multiport Operations

As shown in Figure 8 above, the I/O addressing scheme of the F18A permits comm ports to be combined. Reading or writing ***Multiport Addresses*** such as **rd-u** is permitted and suspends until one or more of the nodes to which the selected ports connect is performing the complementary operation. The design rules for communication flow using multiport operations are straightforward: When a node is doing a multiport read, only one of the selected ports may be written by another node at any given time. When a node is doing a multiport write, every node that intends to read the value written must already be doing so and suspended. It is not possible to directly determine which port(s) transferred data after a multiport operation by inspection of **io**; if it is necessary for a node to make this determination it must be accomplished via protocols.

3.3.2 Port Execution

An F18 may execute instruction streams directly from a comm port. When the F18 fetches an instruction word from a comm port, P is not incremented and unless the instruction stream transfers control elsewhere subsequent words are fetched from the same port. **@p** reads a word from the port and **!p** writes a word into the port. Multiport addresses may be used so long as the design rules in the preceding section are followed.

3.4 Nodes with GPIO Pins

Any edge node may have up to four fully featured General Purpose I/O (GPIO) pins. A schematic is shown in Figure 10. These pins have useful electrical characteristics and are designed to facilitate software defined I/O. GPIO pins are designated by their bit positions as shown in Figure 9 above (pins 17, 5, 3 and 1.)

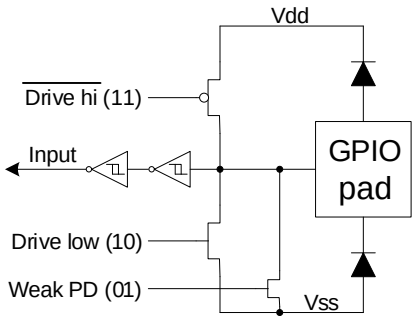

Figure 10 F18A GPIO Pin Circuit

Under program control, you may place each pin in one of four states by writing a value into its 2-bit GPIO field of **io** as follows:

Value	State of pin drive circuitry
00	High Impedance (≈2.8 pF, ≈250 MΩ)
01	Weak pulldown (≈38 µA in saturation). This is the reset state of all GPIO pins.
10	Drive to V_{SS} (≈41 mA in saturation)
11	Drive to V_{DD} (≈41 mA in saturation)

Regardless of the drive state selected for a GPIO pin, you may always read the **io** register to measure the actual voltage on the pin as a logic state (1=pin voltage high) using bits 17, 5, 3 or 1

Input hysteresis, provided by two Schmitt triggers, protects against noise and other classical problems with signals whose rise and fall times are long relative to computer speeds. The F18A computers and I/O circuitry are fast and sensitive enough that parasitic resistance, inductance and capacitance effects may be directly observed by reading the pins as external circuits are being driven, and are fast enough that a 1 MHz sine wave is a slow-moving signal. Refer to the documentation on specific chips for electrical characteristics of their GPIO pins.

3.4.1 Wakeup Pins

When an edge node is configured with a GPIO pin 17 it may be configured and documented for wakeup on that pin. When this is the case, the pin may be "read" using a comm port address which becomes *present for reading*: **Up** for nodes on the top and bottom edges of the array, or **Left** for nodes on the lateral sides. The node suspends until the pin is in the state specified by the **WD** bit in the **io** register. When **WD** is written as zero, which is the reset state of that bit, a pin read suspends until the pin is logically high. To suspend until the pin is low, write 1 to **WD**. *When the read completes, it is required that you must immediately **drop** the value that was read into T.*

F18A multiport read operations may include the wakeup pin as **Up** or **Left**. *In this case you must still immediately **drop** the value read*. To deduce that the pin was the source of the wakeup it is necessary to read **io** and check the current state of bit 17.

3.4.2 Shared (Read-only) Pins

Some nodes may be configured to *observe* the voltage on a GPIO pin that they do not directly control. This is typically done to provide versatility in timing analog to digital conversion. In these cases, the shared pin may be read as bit 17 of **io**. Wakeup may be enabled for reading, writing, or both on a shared pin, as will be documented for the type of I/O node observing the pin. Wakeup polarity is controlled by **WD** as with GPIO pins. See the chip Data Book for details on each node's wakeup configuration as the choice of this configuration may depend on their intended uses.

3.4.3 Phantom Wakeup Signals

A *sending* node may be configured to drive a "phantom" signal that a *receiving* node treats as a shared pin. In this configuration the receiving node operates exactly as described above in 3.4.2. The sending node controls the phantom signal by writing into bits **phan 7** or **phan 9**; writing 1 makes the signal look high to the receiver, 0 makes it look low, and the reset state of **io** is equivalent to writing zeroes (low) into these bits.

3.5 Parallel Bus Support

F18A Parallel Bus support consists of a cluster of three adjacent nodes, typically laid out on the "bottom" edge of a chip. The center node has 4 GPIO pins for control signals and each of its parallel port neighbors has I/O circuitry for an 18-bit bidirectional bus interface. The purposes of all these pins and the direction and timing of signals are defined by software.

The center node is configured to act as sender of a phantom wakeup signal to each of the parallel bus nodes to supplement the comm ports between them to provide versatility in timing.

In each node with an 18-bit bus, the addresses **Up** and **Data** are enabled for reading and writing. The difference between these two addresses is that operations on **Up** suspend when appropriate (see 3.4.3) while operations on **Data** do not depend upon the phantom wakeup signal.

Each of the 18 pins of a parallel port node has a subset of the capabilities of a GPIO pin. When **DB** in **io** is set to 1 (its reset state) all 18 pins are in output mode and are driven high or low depending on the state of an 18-bit Bus Register. When **DB** is set to zero, all 18 pins are set to high impedance input mode. Unlike GPIO pins, parallel bus pins *do not* have Schmitt triggers, nor do they have weak pull-down capabilities.

Writing to **Up** or **Data** always sets the Bus Register, regardless of the state of **DB**. Writing a 1 or 0 to a bit in this register will, in output mode, drive the corresponding pin to V_{DD} or V_{SS}, respectively. Reading from **Up** or **Data** in input mode reads the pin states with high reading as 1; in output mode, reading returns the current value of the Bus Register.

Note that parallel buses reset to output mode and that reset does not initialize the Bus Register. Its value after power up is unpredictable.

Refer to the documentation on specific chips for electrical characteristics of their parallel bus pins.

3.6 SERDES Nodes

The high speed Serializer-Deserializer (SERDES) node is equipped with a transceiver for 2-wire (clock and data) synchronous, half duplex communications at a rate of approximately 450 megabits per second. While a SERDES node might be capable of communicating with something other than another SERDES node, it was not designed with any other objective than communication between our chips.

To receive an 18-bit word, a SERDES node must place the value x3FFFE in T and read the **Up** port. The node suspends until a word has been received.

To begin transmitting, a SERDES node must write the first word to **Data** and must then set bit 17 of **io** to 1. Each subsequent word is written to **Up** ; the node will suspend until the preceding word has been finished and transmission of the new word may begin. If the node wishes to stop transmitting cleanly, it must wait long enough after writing the last word to **Up** (or after setting **io** bit 17 if only one word is to be transmitted) for a full word to have been sent.

Refer to the documentation on specific chips for electrical and timing characteristics of their SERDES nodes and pins.

3.7 Analog Nodes

Any F18A edge node may be configured with an analog input pin and an analog output pin along with the associated circuitry. A block diagram is shown in Figure 11.

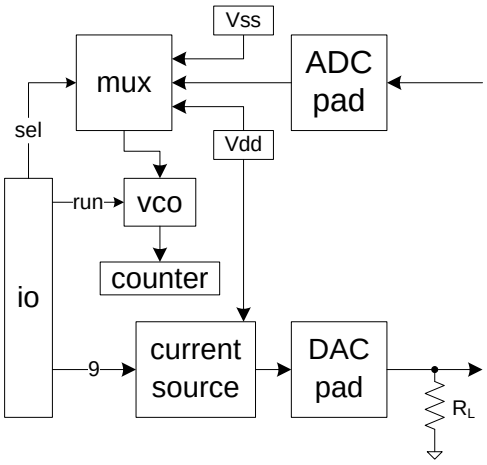

Figure 11 Analog I/O Circuitry

The Analog to Digital converter (ADC) consists of a high speed, free running down-counter whose frequency varies as a function of input voltage between ≈3.6 GHz for V_{DD} input and ≈5.6 GHz for V_{SS}, as shown in the typical curve in Figure 13. The **A/D Mode** field in **io** controls the counter as follows:

Value	Mode and input of ADC counter
00	High Impedance ADC input pad (≈2.8 pF, ≈250 MΩ)
01	V_{DD} Calibration
10	Counter disabled to save power (the reset state)
11	V_{SS} Calibration

A voltage is measured within the operating range (≈750 mV to ≈1.3v) by calculating the difference between two readings separated by a known time interval.

For analog nodes on top and bottom edges, **Data** and **Up** are present for reading and writing; for those on sides of an array, **Ldata** and **Left** are present. The procedure for sampling the ADC requires two steps: First *write* to the **Up/Left** or **Data/Ldata** port as appropriate to signal the VCO to stop for reading, and then *read* from **Data/Ldata** or **Up/Left** as appropriate to obtain the value of the down-counter and restart the VCO. Reading without writing first will not yield useful values. An analog node is equipped with a shared wakeup pin (see 3.4.2) and configured to suspend when writing to **Up/Left**.

The Digital to Analog Converter (DAC) is a programmable current source designed to produce a voltage across a resistance to ground or to source an op-amp. By writing the xor of a desired *value* and x155 into the low order nine bits of **io** that *value* controls the DAC output. A value of x1FF (written as x0AA) sets the DAC for maximum current; a value of 0 (written as x155, the reset state) sets minimum current, high impedance output. Typical DAC transfer functions, in mV versus DAC *values*, into 75, 50, 37.5 and 8 Ohms are shown in Figure 12; as the resistance decreases, the voltage decreases and the function becomes more linear.

Refer to the documentation on specific chips for electrical characteristics of their analog nodes.

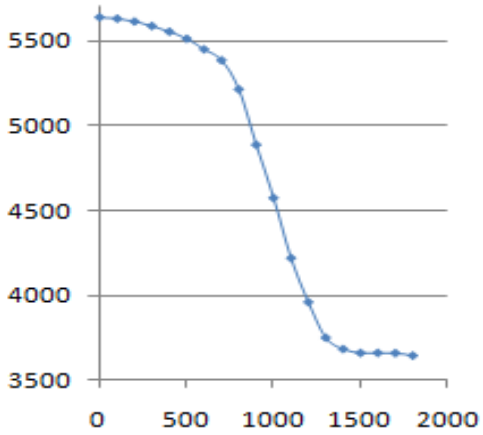

Figure 13 Typical ADC Transfer Function

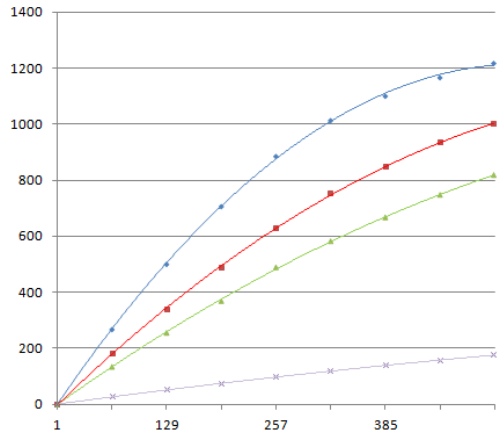

Figure 12 Typical DAC Transfer Function

4. Revision History

REVISION	DESCRIPTION
110412	Initial Release
171107	Resolve some comments received: Add Ldata for side Analog nodes. Clarify variable length destination address field. Emphasize conditions for overflow in multiply step. Clarify shared pin wake-up configuration is a chip design issue not an F18 issue.

GreenArrays®

Product Data Book DB001 *Revised 11/07/17*

IMPORTANT NOTICE

GreenArrays Incorporated (GAI) reserves the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to GAI's terms and conditions of sale supplied at the time of order acknowledgment.

GAI disclaims any express or implied warranty relating to the sale and/or use of GAI products, including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright, or other intellectual property right.

GAI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using GAI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

GAI does not warrant or represent that any license, either express or implied, is granted under any GAI patent right, copyright, mask work right, or other GAI intellectual property right relating to any combination, machine, or process in which GAI products or services are used. Information published by GAI regarding third-party products or services does not constitute a license from GAI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from GAI under the patents or other intellectual property of GAI.

Reproduction of GAI information in GAI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. GAI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of GAI products or services with statements different from or beyond the parameters stated by GAI for that product or service voids all express and any implied warranties for the associated GAI product or service and is an unfair and deceptive business practice. GAI is not responsible or liable for any such statements.

GAI products are not authorized for use in safety-critical applications (such as life support) where a failure of the GAI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of GAI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by GAI. Further, Buyers must fully indemnify GAI and its representatives against any damages arising out of the use of GAI products in such safety-critical applications.

GAI products are neither designed nor intended for use in military/aerospace applications or environments unless the GAI products are specifically designated by GAI as military-grade or "enhanced plastic." Only products designated by GAI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of GAI products which GAI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

GAI products are neither designed nor intended for use in automotive applications or environments unless the specific GAI products are designated by GAI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, GAI will not be responsible for any failure to meet such requirements.

For current information on GreenArrays products and application solutions, see www.GreenArrayChips.com

Mailing Address: GreenArrays, Inc., 821 East 17th Street, Cheyenne, Wyoming 82001
Printed in the United States of America
Phone (775) 298-4748 fax (775) 548-8547 email Sales@GreenArrayChips.com